UNIVERSIDADE PRESBITERIANA MACKENZIE

ALEX BONIFÁCIO

EFICÁCIA NA GESTÃO DA QUALIDADE EM PROJETOS DE DESENVOLVIMENTO DE *SOFTWARE*

São Paulo

2013

ALEX BONIFÁCIO

EFICÁCIA NA GESTÃO DA QUALIDADE EM PROJETOS DE DESENVOLVIMENTO DE *SOFTWARE*

Trabalho de Conclusão de Curso apresentado ao Programa de Pós-graduação Lato Sensu da Escola de Engenharia da Universidade Presbiteriana Mackenzie, como requisito parcial para a obtenção do Título de Especialista em Gestão de Projetos.

São Paulo

2013

RESUMO

Qualidade é um fator que vem se tornando cada vez mais importante para as organizações prestadoras de serviços de TI. Os clientes de projetos de desenvolvimento de sistemas estão se tornando cada vez mais exigentes com o passar do tempo, sendo que hoje em dia pode-se notar que uma das principais fontes de descontentamento de grande parte deles é a baixa qualidade dos produtos e serviços a eles prestados. Nos últimos anos, cada vez mais empresas têm buscado a melhoria da qualidade de seus produtos e serviços, porém é necessário que a cultura da qualidade seja desenvolvida corretamente nas empresas. O presente trabalho se propõe a analisar os processos de planejamento da qualidade, controle da qualidade e garantia da qualidade em projetos de desenvolvimento de *software*, visando à melhoria contínua da qualidade em todos os produtos e serviços entregues e nos processos executados, desde as etapas iniciais do projeto até a sua conclusão, além de apresentar uma abordagem importante sobre a gestão do custo da qualidade.

Palavras-chave: Qualidade. *Software*. Gestão da qualidade.

ABSTRACT

Quality is a factor that has become increasingly important for IT service provider organizations. Customers for system development projects are becoming more demanding as time passes and these days one can realize that, for many of them, one of the main sources of dissatisfaction is the low quality of products and services provided to them. During the last few years, more and more companies have sought to improve the quality of their products and services, but a quality culture must be properly developed at a corporate level. The aim of this work is to analyze the processes related to quality planning, quality control and quality assurance in software development projects, aiming at a continuous improvement of quality in all the products and services delivered and processes executed, from their early stages to their completion, and it also presents an important approach to quality cost management.

Keywords: Quality. Software. Quality Management.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

Ad hoc	Expressão latina que significa “para esta finalidade”
ATAQS	Área de Tecnologia para Avaliação de Qualidade de *Software*
Baseline	Configuração inicial utilizada como base para o desenvolvimento do sistema
Checklist	Lista de verificação para itens requeridos
CMM	*Capability Maturity Model*
Help-desk	Setor de apoio a usuários para suporte e resolução de problemas técnicos
IEC	*International Eletrotechnical Comission*
ISO	*International Organization for Standardization*
NBR	Norma Brasileira
Peer-review	Técnica de revisão formada por pares de revisores
PMBOK	*Project Management Book Of Knowledge*
PMI	*Project Management Institute*
RTF	Revisão Técnica Formal
SCM	*Software Configuration Management*
Software	Sequência de instruções executada em um computador
SPICE	*Software Process Improvement and Capability dEtermination*
SQA	*Software Quality Assurance*
SW-CMM	*Software Capability Maturity Model*
SWEBOK	*Software Engineering Body of Knowledge*
TI	Tecnologia da Informação
TQM	*Total Quality Management*
Walkthrough	Técnica de revisão realizada em forma de reunião de revisão lógica

SUMÁRIO

1 INTRODUÇÃO

De acordo com a norma NBR ISO 9000:2005, a gestão da qualidade se dá quando existem atividades designadas para dirigir e controlar a qualidade dos produtos, serviços ou processos de uma organização. Estes direção e controle incluem o estabelecimento de uma política de qualidade, dos objetivos da qualidade, do planejamento da qualidade, do controle da qualidade, da garantia de qualidade e da melhoria da qualidade (FREITAS, 2012).

Uma das grandes queixas atualmente dos clientes de projetos de desenvolvimento de sistemas de TI é a falta de qualidade nos produtos e serviços entregues. Muitas vezes são aplicados processos de garantia e controle de qualidade a esses produtos e serviços, mas a gestão ideal de todo o processo de qualidade não é realizada ou, na maioria das vezes, realizada parcialmente devido a fatores, externos ou internos, ligados ao projeto. É necessário ressaltar que a gestão da qualidade deve ser realizada por todos os indivíduos envolvidos no projeto, e muitas vezes a baixa qualidade pode ser causada pela falta de atenção ou treinamento necessário aos critérios de controle e garantia da qualidade nos processos envolvidos nos projetos de desenvolvimento de TI. Este trabalho visa ilustrar os principais aspectos para a gestão da qualidade eficaz, de acordo com a estrutura mostrada pelo PMBOK (2008), que enfatiza que os principais pontos da gestão da qualidade são o planejamento, o controle e a garantia, e também de acordo com o equilíbrio entre escopo, custo e tempo, descritos no chamado “triângulo de ferro” da qualidade.

Para estruturar o assunto abordado, este trabalho será desenvolvido em cinco capítulos, onde nos quatro primeiros serão tratados os assuntos Planejamento da Qualidade, Controle de Qualidade, Garantia de Qualidade, Custo da Qualidade e no último, a Conclusão, será mostrado o resultado final do estudo realizado e as conclusões do autor.

Abaixo segue um breve resumo de cada capítulo:

Na seção 2 (Planejamento da Qualidade), será destacada a importância do planejamento da qualidade em projetos, bem como conceitos importantes sobre qualidade em geral e qualidade de *software* e alguns dos principais modelos de qualidade utilizados em desenvolvimento de sistemas.

Na seção 3 (Controle de Qualidade de *Software*) serão abordados um dos principais meios de controle de qualidade de *software*, o teste de *software*, as principais técnicas para teste de *software* e também o Gerenciamento de Configuração de *Software* (SCM).

Na seção 4 (Garantia de Qualidade de *Software*) serão tratados conceitos sobre garantia de qualidade e revisões de *software*. Serão analisados os principais tipos de revisão de *software* existentes.

A seção 5 (Custo da Qualidade de *Software*) faz uma rápida abordagem sobre os custos da qualidade em projetos de desenvolvimento de sistemas, suas causas e os tipos de custo da qualidade existentes.

Por fim, na seção 6 (Conclusão) serão apresentadas as conclusões do autor a partir do embasamento teórico adquirido pelo autor durante o trabalho e também da sua experiência em projetos de desenvolvimento de sistemas.

2 PLANEJAMENTO DA QUALIDADE

De acordo com o PMBOK (2008), a gerência da qualidade busca garantir que o projeto cumpra todos os requisitos e necessidades estipulados em sua contratação, e é formada pelo planejamento, controle e garantia da qualidade.

Na etapa de planejamento da qualidade, são identificados os padrões de qualidade necessários para o projeto e a forma como estes padrões serão cumpridos (PMBOK, 2008). Neste cenário podem entrar tanto os chamados padrões de qualidade "voluntários" quanto os padrões de qualidade definidos por normas e pela legislação corrente, que serão exigidos de acordo com o objetivo do projeto e também com questões internas e externas relacionadas às organizações envolvidas (OLIVEIRA, 2009). Uma organização que implementa um modelo de gestão com foco na qualidade gera credibilidade ao cliente, exigindo também menos trabalho e recursos adicionais (MATOS, 2009).

Atualmente as organizações buscam a melhoria contínua da qualidade, seja no produto final ou serviço, na qualidade do trabalho e processos executados ou na exploração do seu negócio, com o fim de se tornarem mais competitivas. Nesse caso, a existência de processos para agilizar e otimizar o fluxo de trabalho destas organizações é necessária e isso torna-se possível com uma gestão da qualidade adequada, onde essa gestão acaba determinando o sucesso de um projeto, agregando valor ao negócio da organização (MATOS, 2009).

A etapa de planejamento segue uma sequência de parâmetros básicos que têm como função a definição das atividades no projeto. Estes parâmetros baseiam-se na resposta às seguintes questões: o que será feito, por que será feito, quem vai fazer, onde será feito, como será feito, quanto deverá ser feito, qual o custo e quando será feito (FARIAS FILHO, 1996). A equipe do projeto também deve estar sempre atenta a um dos princípios fundamentais da gerência da qualidade nos dias atuais: a qualidade é planejada, não inspecionada (PMBOK, 2008).

2.1 A IMPORTÂNCIA DO PLANEJAMENTO DA QUALIDADE

Todo projeto que visa ser bem-sucedido em sua execução passa por um processo estruturado de planejamento com metas bem definidas ao longo do seu desenvolvimento. Apenas o bom planejamento não garante ganho de qualidade, mas aumenta bastante as chances de obtenção de um alto nível de qualidade, e esta qualidade envolve não apenas o

produto desenvolvido como também os processos executados em seu desenvolvimento, o fluxo de trabalho executado, a produção realizada, o nível de satisfação do cliente e o diferencial no mercado. O planejamento quantifica os prazos e custos relacionados, estabelece um escopo e permite uma visão geral do esforço e requisitos necessários ao desenvolvimento do projeto (MATOS, 2009).

Segundo Chermont (2001), alguns fatores tornam importante o planejamento da qualidade:

- As características dos produtos e índices de falhas são determinados em grande parte durante a etapa de planejamento da qualidade;

- Grande parte das atividades atuais em projetos consiste em repetir trabalhos anteriores.

Pelos fatores descritos acima, na execução do processo de gestão da qualidade, é importante que se tenha a máxima atenção e dedicação ao processo de planejamento da qualidade. A alta qualidade é obtida quando o produto final, seja ele um bem ou um serviço, satisfaz a necessidade do cliente pela obtenção das melhores características e pela ausência de deficiências (CHERMONT, 2001).

De acordo com o PMBOK (2008), as atividades de gerenciamento da qualidade são divididas dentro das etapas mostradas na figura abaixo:

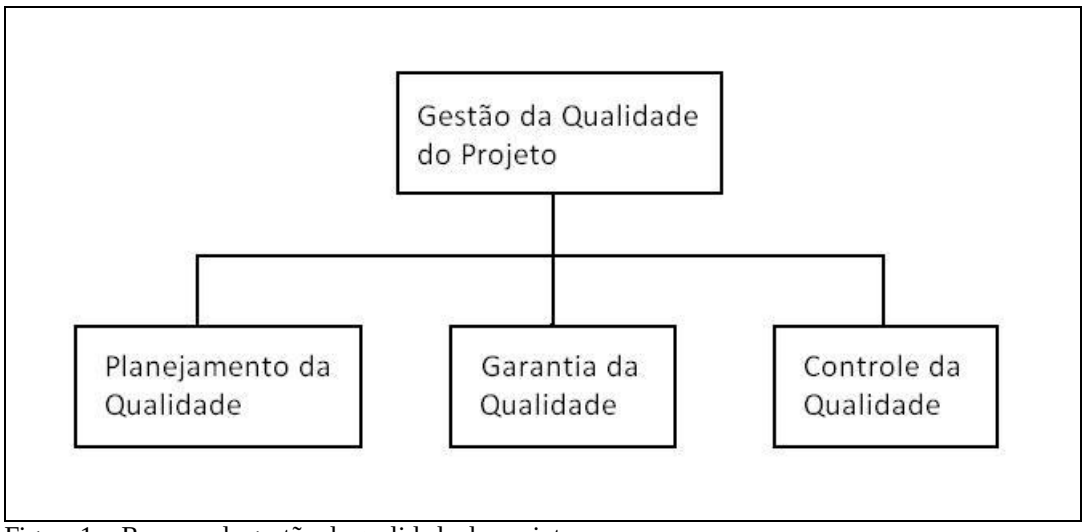

Figura 1 – Resumo da gestão da qualidade do projeto
Fonte: Adaptado de PMBOK, 2008.

O gerenciamento moderno da qualidade serve como complemento ao gerenciamento de projetos, onde as duas competências reconhecem a importância de se obter a satisfação do cliente; de se realizar o correto planejamento, projeção e incorporação da qualidade ao invés da sua inspeção; de se buscar sempre a melhoria contínua da qualidade; da participação de todos os membros da equipe do projeto e da responsabilidade da gerência em fornecer todos os recursos necessários ao bom desempenho da qualidade (PMBOK, 2008).

A boa qualidade do *software* necessita do equilíbrio entre escopo, custo e tempo, pois ela é diretamente afetada pelo balanceamento destas três variáveis. A figura 2 abaixo ilustra bem esta situação por meio do chamado "triângulo de ferro" da qualidade:

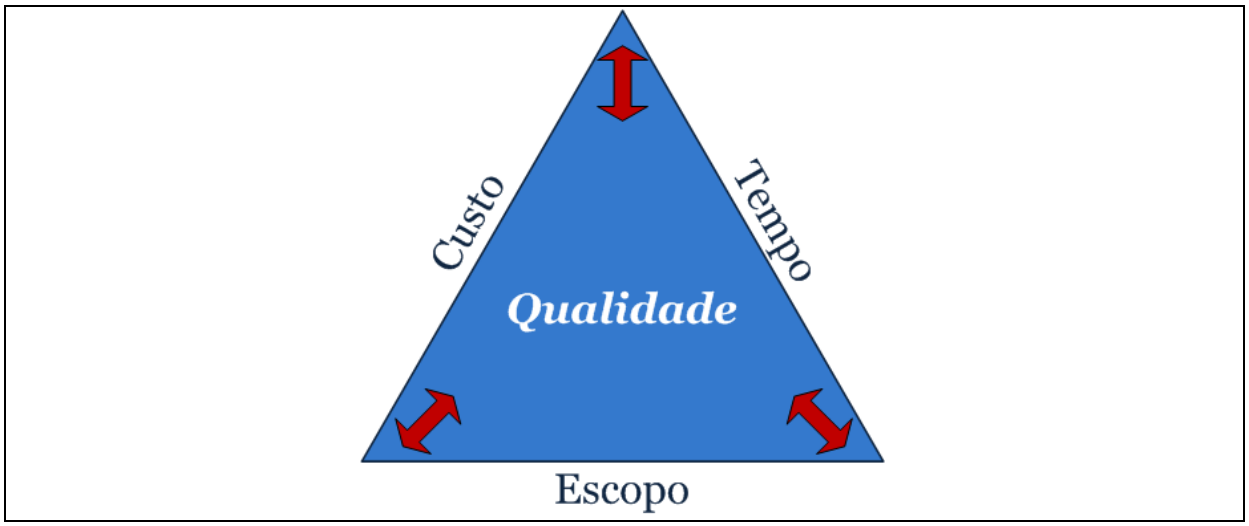

Figura 2 – O "triângulo de ferro" da Qualidade
Fonte: GP3, ([200-?])

Com as ferramentas atuais e processos de desenvolvimento em cascata, o tempo e o custo se tornaram fatores bastante importantes, pois são bastante requeridos pela qualidade de *software*. Com o objetivo de manter a competitividade, as organizações necessitam buscar meios para melhorar a qualidade e, ao mesmo tempo, diminuir o tempo de comercialização dos *software*s cruciais para os seus negócios (COHEN; LUNDBLAD, 2009).

Porém, não podemos afirmar que o sucesso do projeto esteja ligado apenas aos controles de prazo, custo e qualidade. O sucesso de um projeto não possui uma receita simples e direta, e por isso uma relação de dimensões e medidas de desempenho deve ser considerada. Entretanto, é imprescindível que o gerente do projeto obtenha claro entendimento sobre as variáveis relevantes para o desempenho de cada projeto e também sobre a relação entre estas variáveis, buscando assim o domínio sobre elas e o seu equilíbrio ideal (GP3, [200-?]).

2.2 CONCEITOS DE QUALIDADE

Conforme a norma NBR ISO 9000:2005, qualidade é o "grau no qual um conjunto de características inerentes satisfaz a requisitos". Nesse contexto, podemos nos referir ao termo "qualidade" com a utilização de adjetivos como má, boa ou excelente, ao termo "inerente" como a existência de uma característica permanente e ao termo "requisito" como "necessidade ou expectativa que é expressa, geralmente, de forma implícita ou obrigatória", onde qualificadores podem ser utilizados para distinguir tipos específicos de requisito (NBR ISO 9000:2005).

De acordo com Bonifácio et al. (2008), a qualidade pode ser medida a partir dos requisitos de *software*, que se não estiverem em conformidade resultarão em baixo nível de qualidade. É necessária também a especificação de padrões, sob os quais será realizado o desenvolvimento do *software*. Se estes padrões não forem levados em consideração no projeto de desenvolvimento, haverá falta de qualidade no produto (nesse caso, o *software* desenvolvido no projeto). Também deve-se ter atenção aos requisitos implícitos do projeto, que muitas vezes não são documentados mas devem ser seguidos no desenvolvimento de um sistema.

Uma definição interessante e que se aplica bem ao conceito de qualidade de *software* estudado neste trabalho é dada por Balparda (2008): "primeiro, o *software* precisa cumprir ao que foi concebido, ou seja, não pode ocorrer erro; segundo, deve ser executado rápido e com baixo custo de operação".

Atualmente, conforme citam Bonifácio et al. (2008), existe uma crescente demanda pela melhoria contínua na qualidade dos sistemas em geral, o que mostra a crescente importância da qualidade total não mais como diferencial mas sim como padrão. O sistema deve ser desenvolvido, instalado e executado com sucesso, cumprindo o objetivo para o qual ele foi construído, o que envolve um ciclo de desenvolvimento bastante abrangente, desafiando a todos os envolvidos no projeto de desenvolvimento de um sistema.

Fatores importantes para a melhoria da qualidade são:

- Projeto realizado dentro de um rigor científico que propicie qualidade ao produto de *software*;

- Controle do processo de desenvolvimento;

- Medição do processo de desenvolvimento;

- Técnicas de Garantia da Qualidade (SQA);

- Atividades auxiliares independentes de fase (gerenciamento de configuração, técnicas de melhoria contínua);

- Utilização de métodos, padrões e ferramentas adequadas.

De acordo com Campos (2008), atualmente as empresas utilizam com frequência o termo TQM (Total Quality Management), que descreve uma visão de melhoria da qualidade e pode adquirir diversos significados, dependendo de sua aplicação e interpretação. Podemos resumir os elementos-chave do TQM conforme a figura abaixo:

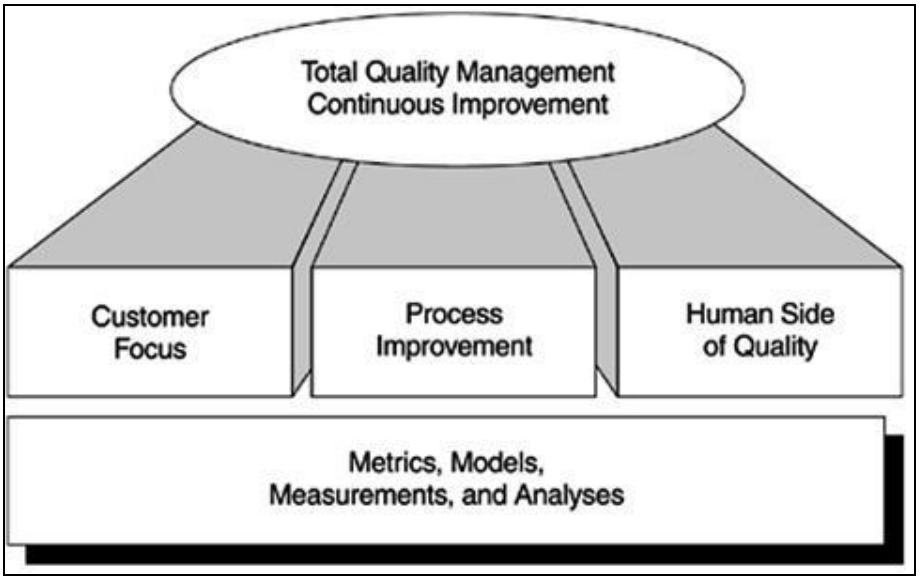

Figura 3: Elementos chave do TQM.
Fonte: CAMPOS, 2008

De acordo com a figura ilustrada por Campos (2008), o Foco do Cliente (*Customer Focus*) tem como objetivo atingir a satisfação total do cliente por meio do estudo de suas necessidades e vontades, coleta de seus requisitos e medição e gerenciamento do seu nível de satisfação. A Melhoria de Processo (*Process Improvement*) visa reduzir as variações de processo e atingir a melhoria contínua de qualidade. Aqui estão inclusos tanto os processos de negócio como o processo de desenvolvimento do produto, e a qualidade do produto é reforçada por meio da melhoria de processo. O Lado Humano da Qualidade (*Human Side of Quality*) visa criar e difundir a cultura de qualidade na empresa, tendo como áreas de foco a

liderança, o apoio da alta gerência, a participação total dos colaboradores da empresa e fatores sociais e psicológicos desta. Já as Métricas, Modelos, Medições e Análises (*Metrics, Models, Measurement and Analysis*) direcionam a melhoria contínua em todos os parâmetros da qualidade do sistema desenvolvido por um sistema de medição orientado a metas.

Para Campos (2008), o conceito da ISO é totalmente aplicável ao contexto de sistemas de informação e *software*. Os usuários finais sempre vão esperar pelo mais alto padrão de qualidade possível, pois querem que suas tarefas sejam desempenhadas sempre da maneira mais correta.

2.3 MODELOS DE QUALIDADE

Os principais modelos de qualidade de *software* atualmente são a ISO 9000-3, a ISO/IEC 12207-1, o *Software Capability Maturity Model* (SW-CMM) e o *Software Process Improvement and Capability dEtermination* (SPICE).

De acordo com as diretrizes propostas na NBR ISO 9000-3:1993, o foco em questões como entendimento entre o contratante e o contratado, requisitos funcionais e o uso de metodologias consistentes para o desenvolvimento do *software* e o gerenciamento total do projeto também é bastante importante em projetos de desenvolvimento de *software*. Na norma NBR ISO 9001:2008 são especificados os requisitos necessários para que as empresas possam demonstrar sua capacidade para assegurar a qualidade de seus produtos e serviços, atendendo assim aos requisitos do cliente e aumentando a sua satisfação.

A norma ISO/IEC apresenta uma definição abrangente em relação aos processos, atividades e tarefas a serem executados na aquisição, fornecimento, desenvolvimento, operação e manutenção do sistema. A importância deste modelo é a consolidação de uma estrutura de classificação de processos normalizando a terminologia, conforme Bonifácio et al. (2008).

O modelo SW-CMM avalia a capacidade de desenvolvimento dos fornecedores de produtos de *software*, indicando por meio de cinco níveis de maturidade o cumprimento de conjuntos de requisitos estruturais para as áreas-chave do processo de desenvolvimento de *software* (CMMI-2, 2002).

No modelo SPICE, seu propósito é entender o estado dos processos de uma organização para a sua melhoria e determinar a adequação dos processos de uma organização para um requisito particular ou um grupo de requisitos. No contexto de melhoria, seu objetivo

é caracterizar as práticas correntes do projeto em termos de capacidade dos processos selecionados (BONIFÁCIO et al., 2008).

A NBR ISO 9000-3:1993 aplica-se aos casos de desenvolvimento, fornecimento e manutenção de *software*, desde que específico para um cliente, onde existe um contrato formal entre fornecedor e cliente. A qualidade, nesse caso, significa conformidades com as especificações contratuais firmadas entre as partes. O guia ISO/IEC aplica-se a empresas de *software* que pretendem estruturar seus sistemas de gestão da qualidade com base nos requisitos da ISO 9001:2000. Já de acordo com Bonifácio et al. (2008), a aplicação do SW-CMM auxilia nos métodos quantitativos e recursos humanos para melhoria do material e serviços fornecidos, bem como todos os processos dentro de uma empresa e também o nível de atendimento das necessidades do cliente tanto no presente como no futuro.

De acordo com a Área de Tecnologia para Avaliação de Qualidade de *Software* – ATAQS (2008 apud BONIFÁCIO et al., 2008), segue abaixo um quadro de comparação com os modelos apresentados:

Aspectos Abordados	ISO 9000-3	ISO/IEC 12207-1	SW-CMM	SPICE
Objetivo	Certificar a organização com padrões estabelecidos em situações de contrato de fornecimento de *software*	Estabelecer uma terminologia e um entendimento comum ara os processos entre todos os envolvidos com *software*	Determinar a capacitação da organização e apoiar a sua evolução de acordo com os níveis estabelecidos	Conhecer e avaliar os processos da organização, determinar a capacitação e promover a melhoria.
Abordagem	Verificação de conformidade de processos a padrões documentados	Definição dos processos para aquisição, fornecimento, desenvolvimento, operação e manutenção do *software*	Avaliação dos processos e enquadramento da organização em um dos níveis de maturidade	Avaliação dos processos da organização em relação a níveis de capacitação
Organizações Alvo	Organizações que necessitam de uma certificação	Organizações em geral	Organizações que necessitam de comprovação formal de sua capacidade	Organizações em geral
Definições de Processos	Não estabelece processos, estabelece atividades a serem cumpridas, com visão de estrutura, ciclo de vida e suporte	Estabelece 17 processos, organizados em 3 categorias	Estabelece 18 áreas de processos organizados em 5 níveis crescentes de maturidade	Estabelece 29 processos organizados em 5 categorias
Flexibilidade nos Aspectos definidos pelo modelo	Não admite adaptação nos aspectos abordados	Classificação de processos pode ser utilizada conforme os objetivos da organização	Níveis e áreas chave de processo são a base do modelo e ao podem ser alterados	Permite a definição de perfis de processo e práticas de acordo com os objetivos da organização
Instrumento de Avaliação	Lista de Verificação	Não se aplica	Questionário e entrevistas	Fornece orientações para definição dos instrumentos
Inspiração e Influência	Normas militares americanas, canadenses, Sistema de qualidade do Reino Unido	TQM, PDCA	Princípios de Shewart, Deming, Juran, Crosby	TQM, PDCA, SW-CMM, STD, Trillium, Malcolm Baldrige, Bootstrap
Aspectos Positivos	Norma Internacional; Difusão extensa; Reconhecimento do valor da certificação	Norma Internacional; Definição de uma taxonomia para processos útil para qualquer organização	Estabelecimento de diretrizes para a melhoria contínua. Difusão extensa nos EUA	Norma Internacional em elaboração; Expansão e flexibilidade dos modelos citados
Limitações	Risco de colocar a Certificação como objetivo principal. Ausência de apoio à melhoria contínua. Falta abordagem de produto.	Apenas uma definição de taxonomia de processos	Pouca consideração à diversidade das organizações. Dificuldade de aplicação em pequenas organizações. Falta abordagem de produto.	Devido à grande quantidade de informações, exige treinamento para sua aplicação. Falta abordagem de produto.

Tabela 1: Quadro de comparação entre os modelos apresentados
Fonte: ATAQS (2008 apud BONIFÁCIO et al., 2008)

Os modelos descritos no quadro acima são os principais modelos de qualidade utilizados em projetos de desenvolvimento de *software*. Cada um possui características particulares, e cabe à gerência avaliar qual o modelo que melhor se adéqua às necessidades e objetivos da organização ou do projeto a ser realizado e como utilizá-lo corretamente de acordo com as expectativas.

3 CONTROLE DE QUALIDADE DE *SOFTWARE*

Quando falamos sobre controle da qualidade de *software*, nos referimos ao processo de validação do produto de *software* de acordo com normas, padrões e requisitos pré-estabelecidos. Quando são encontradas falhas e/ou inconformidades no produto, são tomadas medidas de correção e prevenção a futuros erros que possam ser encontrados. Neste processo realiza-se o monitoramento e registro dos dados resultantes da execução dos processos de qualidade para a avaliação do desempenho desta atividade e sua orientação a mudanças. É um processo essencial em projetos de desenvolvimento de *software* pelo fato de ter o seu foco na detecção de erros (GOMES, 2010a).

O controle da qualidade pode ser realizado em todos os processos de desenvolvimento do produto de *software*, facilitando a verificação de entregas intermediárias do projeto e, no caso de identificação de problemas, seja possível a sua correção antes que os mesmos se propaguem às fases seguintes do projeto de desenvolvimento de *software* (GOMES, 2010a). Contudo, é necessário que o gestor ou algum dos componentes do projeto seja capaz de identificar e definir processos de controle de qualidade para todos os processos relacionados ao desenvolvimento do *software* durante a execução do projeto. Por ser um processo de *feedback*, o controle de qualidade é essencial para minimizar os defeitos produzidos (BUENO; CAMPELO, [200-?]).

De acordo com Staa (2012):

- Aproximadamente 70% das falhas de *software* são geradas por especificações incorretas;

- Aproximadamente 80% do custo do *software* é gerado pela sua própria manutenção;

- Aproximadamente 50% do *software* entregue possui defeitos unusuais;

- Aproximadamente 40% do esforço dispensado é gasto em retrabalho inútil.

Caso o processo de controle seja realizado adequadamente, as técnicas utilizadas podem ser de grande importância para identificação e resolução de erros remanescentes, e se tornarão ferramenta fundamental na posterior prevenção destes erros.

Alguns dos métodos utilizados para controlar a qualidade de um produto de *software* são os testes de *software* e inspeções de *software*. Estudaremos estes métodos no item a seguir.

3.1 TESTE DE *SOFTWARE*

De acordo com Myers (1979), o teste se baseia em executar o programa ou sistema com a intenção de encontrar problemas.

Porém, "é de conhecimento geral entre os analistas de *software* que nunca se elimina o último bug de um programa. Os bugs são aceitos como uma triste realidade. Esperamos eliminá-los todos, um por um, mas nunca conseguiremos nos livrar deles" (DEMARCO, 1991) .

Morell (1987) enfatiza que o teste, por ser uma forma de verificar os itens de conformidade da qualidade do produto, não pode ser feito sem a sua comparação com especificações detalhadas, sendo esta a única forma de avaliar corretamente a qualidade do *software* desenvolvido.

De acordo com Beizer (1990), os principais objetivos do processo de teste são:

- Foco na prevenção de erros;

- Descoberta de sintomas causados por erros;

- Fornecimento de diagnósticos claros para facilitar a correção de erros.

Porém, de acordo com Macoratti ([200-?]), o tema não é demasiado simples porque:

- Erros nem sempre são óbvios;

- Erros de naturezas diferentes podem ter a mesma manifestação;

- Encontrar um erro não significa que já existe conhecimento para corrigí-lo.

Atualmente existem sete princípios atribuídos ao teste de *software* (2007 apud CARRÉRA, 2012):

- Os testes demonstram a presença de problemas: Os testes podem reduzir a permanência de erros desconhecidos no sistema, porém a ausência de erros não é prova de conformidade.

- O teste exaustivo é impossível: Pelo fato de o numero de combinações de cenários de teste ser gigantesco, não se pode afirmar que tudo foi testado.

- Os testes devem iniciar rapidamente e, quanto mais demorada for a verificação do erro, maior o custo para a sua correção: Podemos possibilitar que erros em estágios iniciais do projeto, caso existam, sejam encontrados no momento ideal com a aplicação de testes desde o início do projeto.

- Agrupamento de erros: A maior parte dos erros geralmente é causada em uma pequena parte do código. Se estas áreas sensíveis forem detectadas podem ser priorizadas pelos testes enquanto é feita a busca por erros nas outras regiões do *software*.

- Paradoxo do pesticida: A aplicação dos mesmos testes repetidamente resulta em sua inutilização, pois acabam não conseguindo encontrar novos erros. É necessária uma revisão frequente dos dados e processos utilizados nos testes.

- O teste sempre depende do contexto: Diferentes aplicações podem exigir diferentes técnicas ou processos de teste.

- Ausência de problemas é uma ilusão: Não adianta o sistema estar funcionalmente correto; é necessário atender à real necessidade do usuário.

Os princípios citados acima demonstram bem que, por melhor que seja o desempenho dos testes de qualidade do sistema, a extinção de erros pelos testes é algo praticamente impossível de acontecer, e é necessário o melhor planejamento de controle da qualidade possível para que o sistema seja utilizado pelo usuário sem quaisquer erros encontrados em seu desenvolvimento.

Para Macoratti ([200-?]), para que haja eficácia nos testes realizados é necessário que o processo de *software* seja definido por meio da definição, medição, controle e melhoria dos processos. O controle dos processos deve sempre garantir uma variabilidade estável e resultados previsíveis.

3.1.1 TIPOS DE TESTE DE *SOFTWARE*

De acordo com Laurindo e Moraes ([200-?]), os tipos de teste de *software* existentes são os seguintes:

- Testes de verificação: Conduzidos pela equipe de desenvolvimento, têm como objetivo verificar a conformidade entre os resultados obtidos até o momento e as especificações previamente realizadas.

- Testes de validação: Junto com o cliente, a equipe de desenvolvimento avalia o nível de conformidade do *software* às especificações de requisitos estabelecidas.

- Testes de aceitação: Realizados pelo cliente em conjunto com a equipe de desenvolvimento, avaliam o produto prestes a ser entregue. O método e os critérios a serem utilizados nestes testes são acordados entre ambas as partes nas fases iniciais do desenvolvimento do *software*.

Os de casos de teste podem ser criados utilizando-se as estratégias de caixa branca e caixa preta (1995 apud LAURINDO; MORAES, [200-?]).

3.1.1.1 TESTE DE CAIXA BRANCA

Neste caso, os casos de teste são gerados a partir da análise da estrutura de controle e estrutura de dados do programa a ser testado e garante que todos os caminhos independentes dentro de um mesmo programa tenham sido utilizados pelo menos uma vez. Dentre outras, podemos citar as seguintes técnicas:

- Teste de caminho básico: é gerado um conjunto mínimo de casos de testes que executa cada instrução do programa pelo menos uma vez, em busca principalmente de instruções incorretas ou inadequadas. Não está excluída a possibilidade de encontrar outros tipos de erros em sua execução.

- Teste das estruturas de controle – Condição: neste caso de teste são gerados dados para teste das condições lógicas de um módulo de programa;

- Teste das estruturas de controle – Laços: aqui os casos de teste gerados analisam as estruturas de repetição do programa. Num programa podem ser encontrados quatro tipos de

laços: laços simples, laços aninhados, laços concatenados e laços não estruturados. Estes laços são testados por meio de processos ligeiramente diferentes entre si.

3.1.1.2 TESTE DE CAIXA PRETA

Os casos de teste são gerados com base nos requisitos funcionais do programa a ser testado. Demonstram que as funções do *software* são operacionais, tendo sua entrada adequadamente aceita e a saída corretamente produzida, mantendo assim a integridade das informações externas (MACORATTI, [200-?]). De acordo com Laurindo e Moraes ([200-?]), podemos citar os seguintes tipos:

- Particionamento de equivalência: Neste tipo, particiona-se o conjunto dos dados de entrada em classes de equivalência. Presume-se aqui que, se as entradas estão divididas em partições de equivalência, o comportamento do sistema será o mesmo para qualquer entrada selecionada em uma mesma partição.

- Valor limite: Identifica os problemas que ocorrem nos limites dos intervalos de valores válidos.

- Gráfico de causa-efeito: Dividida em quatro etapas (identificação das causas e efeitos, desenvolvimento de gráfico de relação causa-efeito, conversão do gráfico em uma tabela de decisão, conversão das regras de decisão e casos de teste), estabelece a relação lógica entre as condições de entrada e saída possíveis do *software*.

- Testes de comparação: Esta técnica geralmente é utilizada em aplicações críticas, e consiste na criação de duas versões redundantes do *software*, desenvolvidas independentemente e testadas sob as mesmas condições. Os resultados nas duas versões deverão ser idênticos.

3.2 GERENCIAMENTO DE CONFIGURAÇÃO DE *SOFTWARE* (SCM)

Um processo importante no controle de qualidade de desenvolvimento de *software* é o processo de Gestão de Configuração de *Software* (SCM). O gerenciamento de configuração é o conjunto de atividades desenvolvidas com a finalidade de administração das alterações durante o ciclo de vida do *software*, possibilitando assim um ambiente de trabalho estável por meio do controle de toda e qualquer alteração ocorrida em qualquer etapa do projeto de desenvolvimento do *software*.

No SCM, quaisquer correções, adaptações e modificações que sejam aplicadas durante o ciclo de vida do *software* são controladas e notificadas aos elementos envolvidos. Isso assegura que o processo de desenvolvimento do *software* seja sistemático e rastreável, e é imprescindível em projetos de desenvolvimento onde duas equipes utilizem programas e artefatos em comum. Existe um grande apelo para o uso do SCM durante a etapa de manutenção do *software*, porém seu uso provém controle sobre todos os programas e documentos produzidos desde o planejamento e levantamento de requisitos até a construção e entrega do sistema (DANTAS, [201-?]).

De acordo com o SWEBOK (2005), as atividades que fazem parte dos processos relacionados ao SCM são as seguintes:

I – Planejamento do processo de SCM;

II – Identificação da configuração do *software*;

III – Controle da configuração do *software*;

IV – Relato da situação verificada;

V – Avaliação da configuração do *software*;

VI – Gerência de liberação e entrega.

A figura abaixo ilustra as atividades e sub-atividades do processo de SCM:

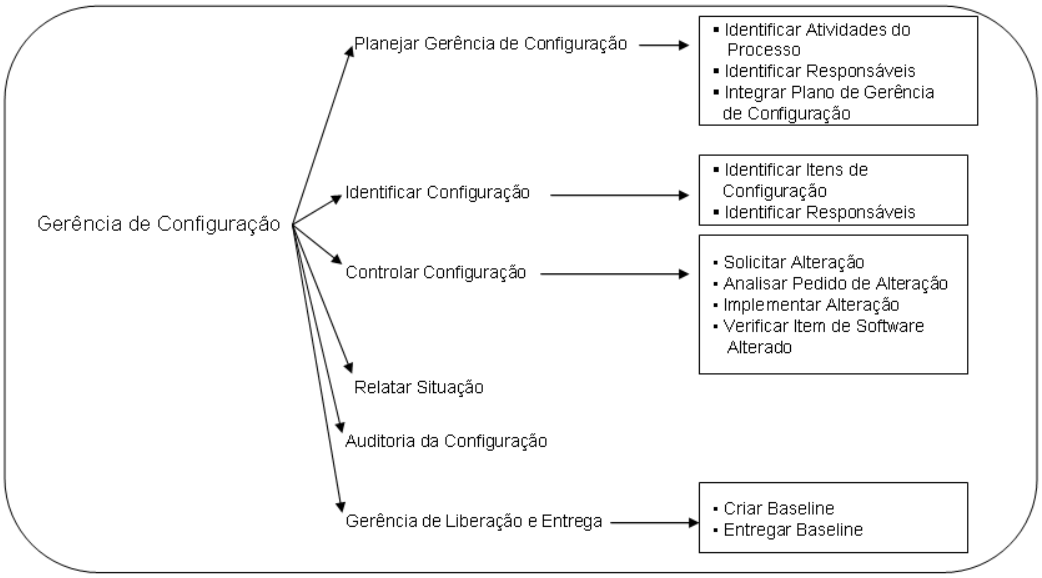

Figura 4 – Processo de Gerência de Configuração
Fonte: FIGUEIREDO; ROCHA; SANTOS, 2004.

Na figura acima, o termo *baseline* refere-se a uma configuração formalmente aprovada e que sirva como referência para o desenvolvimento posterior do sistema (DIAS, 2011).

Conforme Figueiredo, Rocha e Santos (2004) ilustram na figura acima, a primeira das seis atividades a ser executada é o planejamento do SCM, que pode ser dividido em:

- Identificar atividades: Aqui as atividades que farão parte do SCM são identificadas de acordo com a característica da organização. Os responsáveis por cada atividade, bem como o momento e o modo como as atividades serão executadas e os recursos necessários para sua execução deverão estar definidos.

- Identificar responsáveis: Aqui se identificam os componentes do projeto responsáveis por avaliar, aprovando ou não, as solicitações de alteração e alterações em si.

- Integrar plano de gerência de configuração: Aqui é produzido o plano de gerência de configuração, onde estão contidas as atividades referentes ao processo de gerência de configuração e seus responsáveis.

Após a implementação do processo de gerência de configuração, Figueiredo, Rocha e Santos (2004) afirmam que a atividade de identificação da configuração deve ser realizada. Esta atividade contém as seguintes sub-atividades:

- Identificar itens de configuração: Aqui o gerente de projeto deve identificar todos os itens de configuração produzidos ao longo do tempo, identificando os que deverão ter as suas modificações controladas, descrevendo as características importantes de cada um dos itens de configuração e identificando em que momento do ciclo de vida do projeto cada item passará a estar sob controle.

- Identificar Responsáveis: Nesta atividade o gerente do projeto pode identificar os responsáveis por cada item de configuração, que poderá possuir diferentes responsáveis de acordo com suas características.

O controle da configuração mantém o registro e o controle das mudanças nos itens de configuração no decorrer do processo de desenvolvimento (FIGUEIREDO; ROCHA; SANTOS, 2004):

- Solicitar alteração: Executada por qualquer membro da equipe de desenvolvimento, nela deve ser identificado o item de configuração a ser alterado dentre os itens liberados para sofrer modificações, e junto da identificação devem constar o motivo da alteração e o impacto causado pela alteração solicitada.

- Analisar pedido de alteração: Os responsáveis por esta atividade analisam a solicitação de alteração e emitem um parecer aprovando-a ou não e indicando o responsável pela alteração. Nesse caso, o solicitante e o responsável pela alteração não são a mesma pessoa.

- Implementar alteração: Tendo a última versão do item de configuração em mãos, o desenvolvedor realiza as mudanças necessárias e submete a alteração realizada à análise, enviando o item alterado com a descrição da alteração efetuada.

- Verificar item de *software* alterado: Os responsáveis por esta atividade analisam a alteração efetuada verificando a descrição da alteração e o item de *software* alterado e fornecendo um parecer de aprovação, não aprovação ou de revisão das alterações efetuadas. É necessário que o parecer tenha uma justificativa expressa.

A atividade de relatar situação mantém os componentes do projeto informados sobre as alterações ocorridas nos itens de configuração do *software*. O acesso a estas informações pelos membros autorizados deve ser rápido (FIGUEIREDO; ROCHA; SANTOS, 2004).

A auditoria da configuração é conduzida de acordo com processos bem definidos constituídos de vários papéis e responsabilidades de auditores. Seu objetivo é assegurar que as alterações tenham sido implementadas corretamente. Podem ser funcionais (onde a auditoria investiga os aspectos lógicos dos arquivos) ou físicas (onde a auditoria verifica se a configuração a ser congelada é composta pela versão mais recente dos itens de configuração determinados e também se os procedimentos e padrões foram realizados corretamente. Geralmente são executadas no fim de cada fase) (FIGUEIREDO; ROCHA; SANTOS, 2004).

Por fim, temos a gerência de liberação e entrega. Nesta atividade são controlados a liberação e entrega dos produtos e da documentação de *software* de acordo com a política da organização. O responsável por esta atividade é o gerente do projeto e ela é composta das seguintes sub-atividades (FIGUEIREDO; ROCHA; SANTOS, 2004):

- Criar *baseline*: Aqui o gerente do projeto identifica os itens de configuração e as versões destes itens que devem fazer parte do *baseline*, e descreve as diferenças entre esta versão do *baseline* e as anteriores.

- Entregar *baseline*: Nesta atividade o gerente do projeto seleciona, empacota e entrega o *baseline* modificado ao receptor.

A primeira atividade (planejamento do SCM) deve ser executada durante o planejamento do projeto, e as outras deve ser executadas várias vezes ao longo do desenvolvimento do projeto, sempre que necessário. A tabela abaixo mostra os responsáveis por cada atividade do processo:

Atividade do Processo de SCM	**Gerente do Projeto**	**Responsáveis pelo Item**	**Equipe do Projeto**
Identificar Atividades do Processo			
Identificar Responsáveis			
Integrar Plano de SCM			
Identificar Itens de Configuração			
Identificar Responsáveis			
Solicitar Alteração			
Analisar Pedido de Alteração			
Implementar Alteração			
Verificar Item de *Software* Alterado			
Relatar Situação			
Auditoria da Configuração			
Criar *Baseline*			
Entregar *Baseline*			

Tabela 2 – Responsáveis pelas atividades no processo de SCM.
Fonte: FIGUEIREDO; ROCHA; SANTOS, 2004.

A necessidade de melhoria no processo de desenvolvimento é impulsionada por crescentes exigências do mercado por mais qualidade e produtividade no desenvolvimento. A Gestão de Controle de Modificações é um processo que deve ser utilizado em todos os projetos de desenvolvimento de *software*. Porém, é necessário algum esforço para a adequação ao processo e treinamento específicos (DIAS, 2011).

4 GARANTIA DE QUALIDADE DE *SOFTWARE*

A garantia de qualidade de *software* é composta basicamente pelas funções de auditoria e registro, fornecendo à gestão os dados necessários sobre a qualidade do produto em desenvolvimento. Por meio destes dados pode-se analisar se as metas de qualidade do *software* estão sendo atingidas. É de responsabilidade da gerência a aplicação de medidas, técnicas e recursos para correção em caso de problemas.

Atualmente, a garantia de qualidade de *software* não é mais tratada como um diferencial pelo mercado de trabalho, mas sim como um pré-requisito para que seus produtos e serviços obtenham chance de colocação frente à necessidade cada vez maior por produtos de *software* de boa qualidade (CAMPOS, 2008).

A garantia de qualidade de *software* engloba todo o processo de desenvolvimento do *software* por meio da monitoração e melhoria de processos relacionados, assegurando a correta utilização de padrões e procedimentos e garantindo assim que todo problema encontrado tenha sua respectiva ação corretiva tomada. Não é associada apenas às atividades de desenvolvimento do *software*, mas sim a todo o seu ciclo de vida, por meio da monitoração de processos e produtos de *software*. Na monitoração de processos podemos compreender as atividades de auditoria e reporte para a alta gerência. Para a monitoração de produtos podemos considerar revisões, testes de *software* e inspeções formais, bem como revisão dos resultados dos testes de *software* aplicados, auditorias do produto de *software* e testes realizados pelo cliente (CAMPOS, 2008).

De acordo com Bueno e Campelo ([200-?]), as atividades de garantia de qualidade de *software* são realizadas por dois grupos diferentes:

- Engenheiros de *software*: Responsáveis pela parte técnica do trabalho, aplicam métodos e medidas técnicas sólidas, conduzem revisões técnicas formais e efetuam testes;

- Equipe de Garantia de Qualidade de *Software*: Responsável pelo planejamento, supervisão, registro, análise e relato da garantia de qualidade. Aplicam as medidas necessárias seguindo o ponto de vista do cliente, garantindo assim a manutenção da qualidade do *software* desenvolvido.

4.1 REVISÃO DE *SOFTWARE*

As revisões de *software* geralmente funcionam como um filtro para o processo de desenvolvimento do *software* pelo fato de serem aplicadas em diversos pontos deste processo. Sua função é descobrir erros e defeitos no produto de *software*, que podem ser posteriormente removidos.

De acordo com Pressman (1995), os processos de revisão de *software* podem fornecer subsídios aos processos de avaliação de qualidade de *software*. Olson (1999) também descreve outros benefícios proporcionados pela revisão de *software*:

- Aumentam a qualidade do produto de trabalho;

- Identificam e documentam defeitos;

- Identificam melhorias necessárias ao produto de trabalho;

- Verificam a conformidade do produto de trabalho com padrões, especificações e requerimentos adotados pela equipe do projeto;

- Visam o consenso em relação aos produtos de trabalho;

- Aumentam o conhecimento dos componentes do projeto sobre o produto;

- Servem como via gerencial para formalmente completar uma tarefa.

Os processos de revisão de *software* são constituídos pelas seguintes atividades (FALBO, 2008):

- Planejamento: Define quem será o responsável pela revisão, quando será realizada a revisão, quais técnicas serão utilizadas, quantas pessoas farão parte da equipe de revisão, quantas etapas terá a revisão, etc.

- Preparação (opcional): Realizada pelo autor do artefato revisado, realiza a sua descrição do produto de *software* e a perspectiva utilizada em sua construção.

- Revisão Individual: Visa detectar erros e realizar o seu registro adequadamente.

- Reunião (opcional): Geralmente produzem uma lista de erros, porém não contribuem significativamente para aumentar o número de erros encontrados.

- Correção: Aqui o desenvolvedor, com base na lista de erros, deve corrigir o erro ou explicar o motivo de o erro não representar um problema real.

4.1.1 REVISÃO TÉCNICA FORMAL

A revisão técnica formal (RTF) é uma atividade realizada por engenheiros de *software*, tendo como objetivo: a descoberta de erros na função, lógica ou implementação do sistema; a confirmação de que o *software* atende aos requisitos especificados; a correta aplicação dos padrões pré-definidos ao desenvolvimento do produto de *software*; a garantia de um desenvolvimento uniforme dos *software*s desenvolvidos; aumentar a facilidade no gerenciamento de projetos.

Podemos definir dois tipos de revisões: as revisões *"ad hoc"*, onde os revisores utilizam técnicas não sistemáticas e, portanto, possuem as mesmas responsabilidades; e as revisões com utilização de *checklist*, onde os revisores utilizam uma lista de checagem que sugere formas para identificação de falhas (AGUIAR, [200-?]).

Geralmente as RTFs são realizadas por meio de reuniões, que devem sempre envolver entre três e cinco pessoas, além de ser previamente preparadas por cada componente e possuírem duração de no máximo duas horas. Antes da reunião o desenvolvedor comunica a necessidade de revisão do programa ao líder do projeto, e em conjunto é analisada a facilidade de leitura do programa a ser revisado. Após isso, são geradas cópias do programa que serão distribuídas aos revisores, e o responsável pela revisão agenda a reunião de RTF. No decorrer da reunião, devem estar presentes o responsável pela revisão, os revisores e o desenvolvedor do programa revisado, e o desenvolvedor explica o produto desenvolvido, de modo que os revisores possam interromper quando necessário para fazer críticas e sugestões ao produto de *software*. Todos os erros detectados são anotados por um dos revisores. Ao final da reunião de RTF, a equipe de revisão decide se aceita o produto sem modificações, rejeita o produto (é sempre necessária uma nova revisão após a correção dos erros) ou se aceita o produto provisoriamente (os erros encontrados são corrigidos e não há necessidade de nova reunião) (AGUIAR, [200-?]).

De acordo com Pressman (1995), todos os assuntos levantados durante a reunião de RTF são resumidos em dois documentos: o relatório de revisão, contendo o que foi revisado, quem fez a revisão e quais as descobertas e conclusões; a lista de questões de revisão, que identifica áreas-foco de problemas e orienta o corretor dos erros conforme as correções são

feitas no programa. É necessário o acompanhamento do responsável pela revisão, para garantir a correção de todos os itens da lista.

4.1.2 TÉCNICAS DE REVISÃO TÉCNICA FORMAL

4.1.2.1 *WALKTHROUGH*

Nesta técnica de RTF, a revisão é realizada executando-se um procedimento ou programa passo a passo no papel, visando a descoberta de erros. Geralmente se concentra em um módulo ou parte do *software*. Baseada nas características gerais das RTFs, na reunião *walkthrough* cada revisor realiza uma simulação da execução do programa ou trecho de programa revisado, e essa simulação é controlada pelo testador, que disponibiliza durante a reunião um conjunto de casos de teste e avalia os resultados obtidos com os revisores. Nessas reuniões, o responsável pela revisão seleciona os revisores e não há papéis definidos e, ao invés da leitura do programa ou checagem por meio de *checklist*, a execução do programa testado é simulada conforme os casos de teste apresentados pelo testador (TRAVASSOS, 2007).

4.1.2.2 REVISÃO POR PARES (*PEER-REVIEW*)

Técnica de RTF executada por pares de desenvolvedores que possuam o mesmo nível de conhecimento, e o seu objetivo é encontrar problemas de qualidade como erros e desvios de padrões por meio de pontos de vista diferentes do desenvolvedor, sugerindo oportunidades de melhorias ao desenvolvedor do programa e possibilitando a troca de técnicas e experiências entre os revisores. A integração do grupo pode contribuir bastante para o sucesso da reunião, pois podem ocorrer disputas pessoais entre os pares. Aqui é revisado um programa ou um grupo de funcionalidades de cada vez e os resultados alcançados são registrados em um relatório para revisão, passando para o relatório final oficial caso sejam pertinentes ao assunto (MELO, [200-?]).

4.1.2.3 INSPEÇÃO DE *SOFTWARE*

A inspeção de *software* é um tipo particular de revisão aplicado a todos os artefatos produzidos durante o ciclo de desenvolvimento do sistema (FAGAN, 1976). Considerada parte das atividades de garantia de qualidade, devem ser adotadas durante todo o processo de desenvolvimento, garantindo assim maior qualidade aos artefatos desenvolvidos e liberados em cada fase do ciclo de desenvolvimento (BASILI *et al*, 1996).

Em inspeção de *software* utilizam-se bastante os termos discrepância e defeito. Na inspeção, quando é encontrado algum item que diverge da especificação original do código inspecionado, este é tratado como uma discrepância até que seja devidamente analisado, podendo vir a se tornar um defeito ou não. Por outro lado, caso seja encontrado um defeito real é realizado o seu registro como um defeito, e não discrepância (PORTO, 2009).

Fagan (1986) descreveu o processo de inspeção de *software* com cinco etapas: planejamento, apresentação, preparação, reunião, retrabalho e continuação. As características das etapas citadas são as seguintes:

- Planejamento: Um usuário pré-designado como moderador da inspeção define os parâmetros da inspeção (descrição, técnica utilizada, documentos inspecionados, etc.). Após a definição, o moderador seleciona os inspetores e realiza a distribuição do material que deverá ser submetido à inspeção.

- Apresentação: É feita a apresentação das características do artefato inspecionado pelos autores do mesmo e também da forma como ocorrerá a inspeção. Caso os inspetores designados possuam conhecimento sobre o projeto e os artefatos submetidos à inspeção, esta etapa pode ser descartada.

- Preparação: É realizada a análise individual do artefato por cada inspetor, e durante esta análise podem ser realizadas anotações relatando discrepâncias encontradas. Esta tarefa pode ter sua execução facilitada com o fornecimento de técnicas de leitura.

- Reunião: Ocorre envolvendo o moderador, os inspetores e os autores do artefato inspecionado. Todas as discrepâncias são discutidas e classificadas ou não como defeitos, sendo o moderador responsável por essa classificação. Não são discutidas soluções para os defeitos encontrados, e as reuniões não devem durar mais de duas horas pelo fato de haver uma redução da capacidade de análise dos inspetores após este tempo.

- Retrabalho: Ao receber os registros de defeitos encontrados pelos inspetores e confirmados pelo moderador, o autor do artefato realiza a correção destes defeitos.

- Continuação: Após as correções realizadas, a documentação do artefato corrigido é reenviada para o moderador, que por sua vez analisa a inspeção como um todo e faz uma reavaliação de qualidade do artefato inspecionado. Tendo feito esta reavaliação, ele toma a decisão de submeter o artefato novamente a uma inspeção ou não.

Este processo de inspeção de *software* possui uma estrutura rígida de colaboração, onde papéis, atividades e relacionamentos entre atividades estão sempre bem definidos. De acordo com Fagan (1976), a composição da equipe de inspeção, bem como seus respectivos papéis, é a seguinte:

- Moderador: É o responsável por garantir que a inspeção foi bem-sucedida, além de ser o líder da equipe. Sendo assim, torna-se componente chave no processo de inspeção. Pode ou não ser um especialista no artefato inspecionado, e é ele quem determina o local da reunião e realiza o acompanhamento na etapa de continuação. Deve ter sensibilidade pessoal e habilidade suficientes para obter harmonia e bom trabalho em equipe dos participantes.

- Autores: Profissionais responsáveis pela especificação e codificação do artefato inspecionado.

- Testador: Profissional responsável pela criação e/ou execução de casos de teste ou pela aplicação de outros tipos de teste ao artefato inspecionado.

A figura abaixo ilustra o processo de inspeção de *software*:

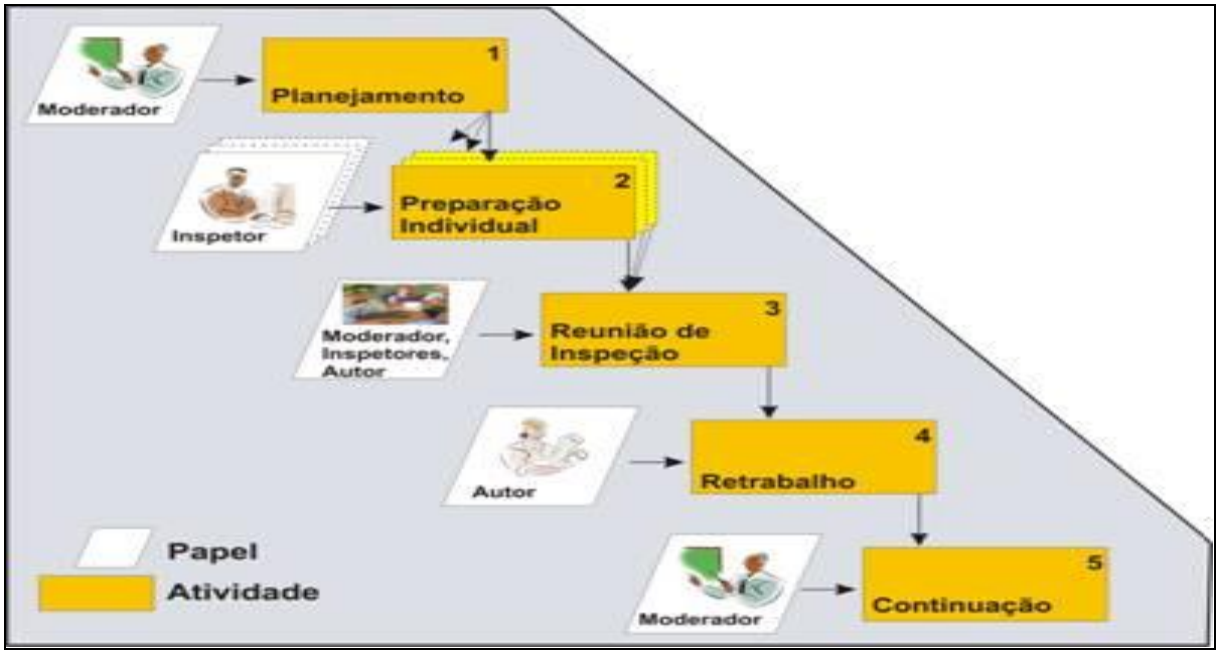

Figura 5 – Processo de Inspeção de *Software*
Fonte: KALINOWSKI, ([201-?]).

Na figura acima se pode verificar o relacionamento entre todas as atividades acima citadas no processo de inspeção. Por não ser obrigatória, a atividade de apresentação não é representada.

5 CUSTO DA QUALIDADE DE *SOFTWARE*

Bartie (2005) afirma que o custo da qualidade é o investimento realizado com o objetivo de atingir os padrões de qualidade determinados para um produto ou serviço.

Um *software* pode atingir o padrão de qualidade desejado pela utilização de processos bem definidos e estáveis ou pela correção de problemas antes da entrega ao usuário final. Pode também atingir esse padrão com a estabilização pós-implantação do *software*, considerando-se a possível aparição de ciclos de ocorrência de falhas, identificação e correção de defeitos conforme novas versões do *software* vão sendo liberadas para o usuário. A cada cenário acima relatado está associado um custo específico e, somados, estes custos constituem o custo da qualidade, que por sua vez faz parte do custo total da produção do *software*, onde também estão contemplados os custos de sua construção (GOMES, 2010b). A figura abaixo ilustra essa situação:

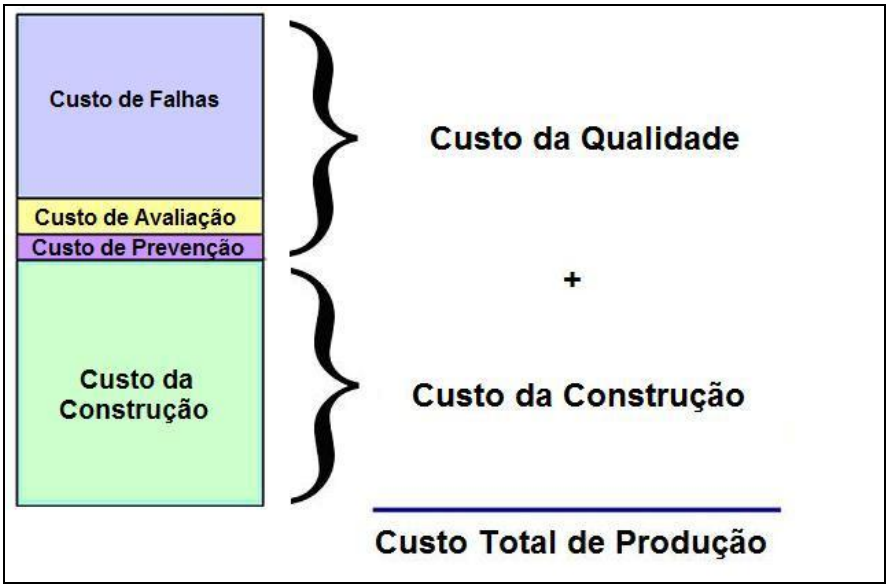

Figura 6 – Estrutura de custos de produção
Fonte: GOMES, 2010b.

No custo da qualidade de *software*, englobam-se tanto os custos da conformidade, que tem como objetivo prevenir e detectar erros do processo de desenvolvimento, visando a sua melhora e garantindo a sua qualidade, como os custos da não conformidade, cujo objetivo é a reparação de falhas encontradas no processo de desenvolvimento do *software*.

Englobam-se nos custos da conformidade os seguintes tipos de custo (GOMES, 2010b):

- Custo de prevenção: Custo associado a atividades com o objetivo de evitar defeitos de *software*. Ocorre antes da criação do produto e baseia-se em um investimento para a criação de métodos e procedimentos, treinamento de profissionais, aquisição de ferramentas, dentre outras medidas. Todas as atividades de garantia de qualidade estão associadas a este custo.

- Custo de avaliação: Custo relacionado à detecção de defeitos de *software*. Engloba todos os gastos com procedimentos de verificação e testes no produto de *software*, após a sua construção e antes de serem liberados para o usuário final.

Associado aos custos da não conformidade, podemos citar:

- Custo de falhas: Custo relacionado ao produto de *software* defeituoso. Englobam-se nesse tipo de custo os gastos com retrabalho para reparação do produto de *software* defeituoso, bem como a prejuízos causados por falhas no mesmo ou gastos para manutenção de uma equipe de *Help-Desk*. Podemos citar como custo de falhas. Pode ser causado antes da implantação do *software* por falhas internas (retrabalho, análise, reparação de falhas encontradas) ou por falhas externas (solução de queixas do cliente, devolução ou substituição do produto, manutenção da equipe de suporte, etc.) (GOMES, 2010b).

Conforme observado por Brito (2012), "é mais barato prevenir problemas do que melhorá-los. Se devemos reparar defeitos, custos com defeitos de falhas internas são menores do que os [custos com defeitos] de falhas externas".

Com base nas categorias de custos descritas acima podemos perceber que, com um investimento bem feito visando a prevenção e redução de defeitos, a redução nos custos de falhas torna-se cada vez maior, influenciando diretamente o custo da qualidade e por consequência o custo total da produção. As atividades de garantia de qualidade e de teste de *software* são responsáveis pela prevenção e redução de defeitos de *software* (GOMES, 2010b).

De acordo com Pressman (2010), os custos relativos para correção de erros em cada etapa do projeto de desenvolvimento de um produto de *software* são ilustrados na figura abaixo:

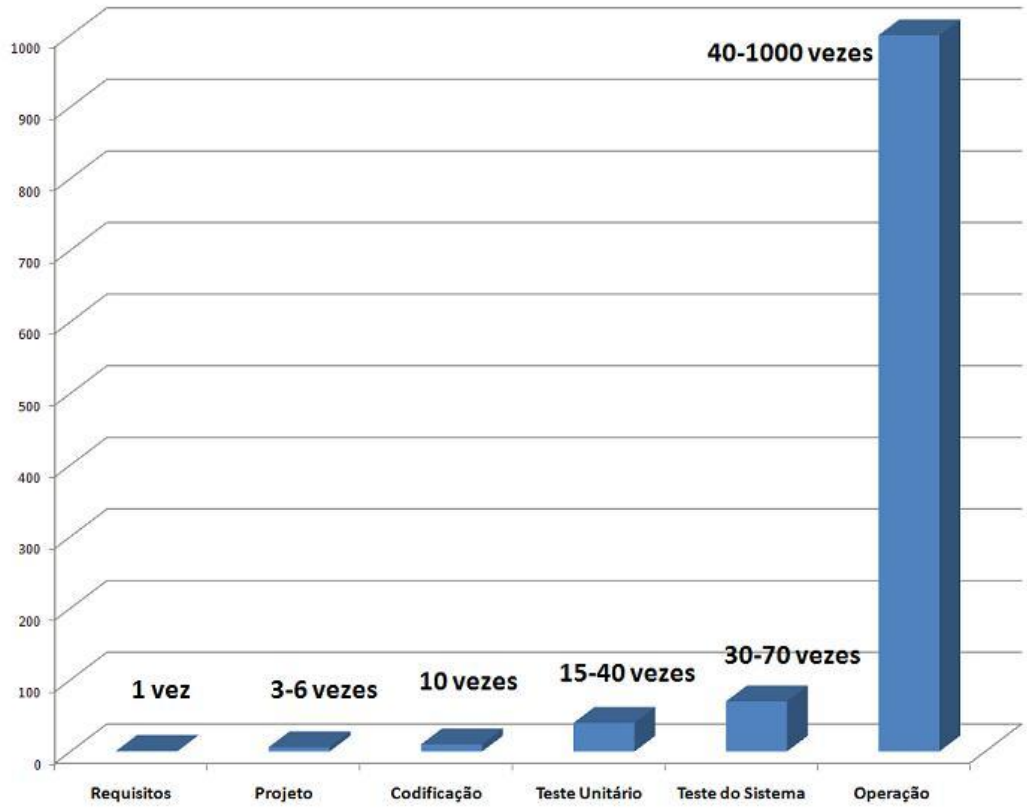

Figura 7 – Custos relativos para a correção de erros
Fonte: PRESSMAN, 2010.

De acordo com a figura acima, podemos perceber que, quanto mais tempo os erros demorarem a ser encontrados, verificados e corrigidos, maior se torna o custo da qualidade. Por isso existe a necessidade de se encontrar falhas e erros no projeto o mais cedo possível. Qualidade deve ser uma preocupação em todas as etapas do projeto, e a compreensão deste fato é a chave para a criação de um processo bem-sucedido para a disponibilização de aplicações de TI (SINFIC, 2006).

6 CONCLUSÃO

De acordo com os estudos realizados durante o desenvolvimento, a gestão da qualidade eficaz é obtida com a correta execução das etapas de planejamento, controle e garantia da qualidade. Porém, a qualidade possui um custo e exige mais tempo no desenvolvimento do projeto. Devido às necessidades urgenciais de grande parte dos clientes, é necessária atenção e esforços para que o projeto tenha o equilíbrio necessário entre escopo, custo e tempo, a fim de que sejam obtidos ganhos significativos em qualidade.

Muitos projetos de desenvolvimento de *software* buscam o equilíbrio acima citado, porém devido às exigências atuais do mercado e também à resistência em parte das organizações para a compreensão e prática da cultura de qualidade, ainda existem dificuldades para uma gestão eficaz da qualidade. Nos dias atuais, a qualidade é cada vez mais importante para o bom desempenho do produto de *software* e, consequentemente, para que o cliente obtenha satisfação com o produto final. Porém, existem situações em que o planejamento de um projeto de *software* não contempla o tempo e custo necessários para que os processos de gestão da qualidade sejam realizados corretamente devido a circunstâncias do projeto como, por exemplo, alta pressão interna (gerência) e externa (cliente) por resultados imediatos em projetos críticos.

Para a gestão eficaz da qualidade, é necessário que a alta gerência da organização desenvolvedora colabore e apoie a equipe do projeto visando o desenvolvimento de um produto com alto nível de qualidade, bem como é importante que a equipe do projeto tenha conhecimento sobre os conceitos fundamentais de qualidade de *software* e possua o comprometimento e colaboração necessários para o cumprimento de regras e processos exigidos pelos padrões de qualidade de *software* atualmente existentes. Porém, devido ao curto tempo disponibilizado pelos prazos exigidos, ainda existe resistência para a execução adequada de padrões de qualidade. Essa resistência se inicia na alta gerência e influencia diretamente tanto o gestor do projeto como a equipe executora do projeto.

Projetos de desenvolvimento de *software* geralmente são caros e, por dificuldades apresentadas desde o seu planejamento até a sua execução, muitas vezes duram mais do que o inicialmente planejado. A qualidade do produto final de *software* vem se tornando cada vez mais importante à medida que uma das principais causas de insatisfação de cliente atualmente é a baixa qualidade do produto entregue. Apesar de as organizações aprenderem cada vez

mais com erros cometidos anteriormente, situações de fracasso ainda ocorrem, por motivos diversos . Porém, ainda é necessário que as organizações, desde a alta gerência até a equipe do projeto, aceitem e pratiquem a cultura de qualidade total.

A má gestão da qualidade ou a sua ausência podem elevar significativamente os riscos do projeto, e no mercado atual grande parte dos clientes já exige padrões de qualidade definidos como pré-requisito para contratação das organizações prestadoras de serviços. A falta de qualidade acarreta em custo e tempo extras para a correção de possíveis falhas, erros e defeitos no produto de *software*, diminuindo a satisfação do cliente e impactando até na reputação da organização contratada para a realização do projeto.

Com a aceitação e propagação da cultura de qualidade, clientes poderão atingir níveis cada vez maiores de satisfação e fornecedores terão diminuição no índice de retrabalho e consequente diminuição do risco de fracasso do projeto, acarretando em redução nos custos por retrabalho devido à implementação de processos de prevenção de erros e defeitos. Consequentemente, a melhoria na reputação do fornecedor frente ao mercado de TI será inevitável.

Como sequência do estudo, o autor recomenda a pesquisa detalhada dos modelos de qualidade ISO 9001, ISO/IEC 12207-1, SW-CMM e SPICE e sua aplicação em cada situação, de acordo com o projeto a ser desenvolvido.

REFERÊNCIAS

AGUIAR, T. C. Engenharia de *Software*: Introdução. **Universidade Federal Fluminense**, [200-?]. Disponível em: <www.eng.uerj.br/~fariasol/disciplinas/Engenharia_de_*Software*/apostilaengsoftw1_uff_teresa_cristina_de_aguiar.doc**>.** Acesso em: 05 mar. 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000-3**: Normas de gestão da qualidade e garantia da qualidade. Rio de Janeiro, nov 1993.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000**: Sistemas de gestão da qualidade – fundamentos e vocabulário. 2. ed. Rio de Janeiro, dez. 2005.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9001**: Sistemas de gestão da qualidade – requisitos. 2. ed. Rio de Janeiro, dez. 2008.

BALPARDA, André. **Qualimetria e gestão de qualidade em TI**. Artigo apresentado no encontro anual do Computer Measurement Group – Brasil, 2008, Brasília. Disponível em: <http://regions.cmg.org/regions/cmgbrazil/Downloads_files/AndreBalparda%28Eccox%29_Qualimetria_Paper.pdf>. Acesso em: 19 fev. 2013.

BARTIE, A. O custo da qualidade de *software*. **iMasters**, set. 2005. Seção *Software*. Disponível em: < http://imasters.com.br/artigo/3592/*software*/o-custo-da-qualidade-de-*software*/>. Acesso em: 07 mar. 2013.

BASILI, V. R. *et al.* The empirical investigation of perspective-based reading. **Empirical *Software* Engineering**, v. 1, n. 2, p. 133-164, nov. 1996.

BEIZER, Boris. ***Software* testing techniques**. New York: International Thompson Computer Press, 1990.

BONIFÁCIO, Alex; BORDIN, Fabio Zanelato; SOUZA, Thiago Nunes de; TESTAI, Vanessa. **Qualidade de *software* – aspectos essenciais**. 2008. 90 f. Trabalho de Graduação Interdisciplinar (Graduação em Sistemas de Informação) – Universidade Presbiteriana Mackenzie, São Paulo, 2008.

BRITO, L. O custo da qualidade de *software*. **Luis Brito: Desenvolvimento, Qualidade e Teste de *Software***, abr. 2012. Disponível em: <http://www.luisbrito.com.br/o-custo-da-qualidade-de-*software*/>. Acesso em: 07 mar. 2013.

BUENO, C. F. S.; CAMPELO, G. B. Qualidade de *Software*. **Universidade Federal de Pernambuco**, [200-?]. Disponível em: <http://sistemas.riopomba.ifsudestemg.edu.br/dcc/materiais/1022789570_Qualidade%20de%20*Software*.pdf>. Acesso em: 01 mar. 2013.

CAMPOS, F. M. Qualidade, qualidade de *software* e garantia da qualidade de *software* são as mesmas coisas? **TestExpert**, mar. 2008. Disponível em: <http://www.testexpert.com.br/?q=node/669>. Acesso em: 19 fev. 2013.

CARNEGIE MELLON *SOFTWARE* ENGINEERING INSTITUTE. **Capability Maturity Model Integration for Systems Engineering (CMMI-2) – staged representation**. v. 1. Massachusetts: SEI, 2002.

CARRÉRA, J. Os sete princípios do teste de *software*. **Bytes don't Bite**. Seção Testes de SW. Disponível em: <http://bytesdontbite.com/2012/06/04/os-7-principios-do-teste-de-*software*/>. Acesso em: 01 mar. 2013.

CHERMONT, Gisele Salgado de. **A Qualidade na gestão de projetos de sistemas de informação.** 2001. 162 f. Tese (Mestrado em Engenharia de Produção) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001.

COHEN, M.; LUNDBLAD, M. Otimização da qualidade do *software*: equilíbrio entre transformações dos negócios e risco. **IBM**, mar. 2009. Disponível em: <http://www.trainning.com.br/download/Qualidade_de_*Software*.PDF>. Acesso em: 03 abr. 2013.

DANTAS, C. Gerência de Configuração de *Software*. **Revista Engenharia de *Software***, n. 2, [201-?]. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=9145>. Acesso em: 04 mar. 2013.

DEMARCO, Tom. **Controle do projeto de *software***: gerenciamento, avaliação e estimativa. Rio de Janeiro: Editora Campus, 1991.

DIAS, A. F. O que é gerência de configuração? **Pronus Engenharia de *Software***, set. 2011. Seção Artigos & Tutoriais. Disponível em: <http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/gerencia_configuracao.php?pagNum=2>. Acesso em: 04 mar. 2013.

FAGAN, M. E. Design and code inspections to reduce errors in program development. **IBM Systems Journal**, v. 15, n. 7, p. 182-211, 1976.

FAGAN, M. E. Advances in *software* inspections. **IEEE Transactions on *Software* Engineering**, v. 12, n. 7, p. 744-751, 1986.

FALBO, R. A. Revisões de *software*. **Universidade Federal do Espírito Santo**, 2008. Disponível em: <http://www.inf.ufes.br/~falbo/files/Aula%202%20-%20Revisao%20de%20*Software*%20-%20Parte%201.ppt>. Acesso em: 05 mar. 2013.

FARIAS FILHO, José Rodrigues de. **Gestão estratégica pela qualidade total percebida:** do conceito à forma e da forma à prática. 1996. 380 f. Tese (Doutorado em Engenharia de Produção) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1996.

FIGUEIREDO, S.; ROCHA, A. R.; SANTOS, G. Gerência de Configuração em Ambientes de Desenvolvimento de *Software* Orientados a Organização. **Universidade Federal do Rio de Janeiro**, 2004. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2004/007.pdf>. Acesso em: 04 mar. 2013.

FREITAS, A. NBR ISO 9000:2005 – 3.2.8 Gestão da qualidade – Sistema de gestão da qualidade – fundamentos e vocabulário. **Academia Platônica**, mai. 2012. Disponível em: <

http://academiaplatonica.com.br/2012/gestao/nbr-iso-90002005-3-2-8-gestao-da-qualidade-sistema-de-gestao-da-qualidade-fundamentos-e-vocabulario/>. Acesso em: 06 fev. 2013.

GOMES, E. Controle da qualidade de *software*. **Base de teste de *software***, jan. 2010a. Disponível em: < http://basedetestede*software*.blogspot.com.br/2010/01/controle-da-qualidade-de-*software*.html>. Acesso em: 01 mar. 2013.

GOMES, E. O custo da qualidade de *software*. **Base de teste de *software***, mai. 2010b. Disponível em: <http://basedetestede*software*.blogspot.com.br/2010/05/o-custo-da-qualidade-de-*software*.html>. Acesso em: 07 mar. 2013.

GP3. O triângulo das restrições de gerenciamento de projetos. **GP3**, [200-?]. Seção Artigos. Disponível em: <http://www.gp3.com.br/index.php?/o-triangulo-das-restricoes-de-gerenciamento-de-projetos.html>. Acesso em: 03 abr. 2013.

KALINOWSKI, M. Introdução à inspeção de *software*. **Revista Engenharia de *Software***, [201-?]. Disponível em: < http://www.devmedia.com.br/artigo-engenharia-de-*software*-introducao-a-inspecao-de-*software*/8037>. Acesso em: 06 mar. 2013.

LAURINDO, F. J. B.; MORAES, R. O. **Teste de *software* e qualidade de *software***: uma visão geral. Artigo apresentado em XIX ENEGEP - Encontro Nacional de Engenharia de Produção, 1999, Rio de Janeiro. Disponível em: < http://www.abepro.org.br/biblioteca/ENEGEP1999_A0198.PDF>. Acesso em: 01 mar. 2013.

MACORATTI, J. C. Testes em desenvolvimento de *software*. **Macoratti.net**, [200-?]. Disponível em: <http://www.macoratti.net/tst_sw1.htm>. Acesso em: 01 mar. 2013.

MATOS, F. F. Planejamento e gestão da qualidade. **Webartigos**, out. 2009. Seção Administração e Negócios. Disponível em: <http://www.webartigos.com/artigos/planejamento-e-gestao-da-qualidade/26759/>. Acesso em: 21 mar. 2013.

MELO, S. M. Inspeção de *software*. **Universidade de São Paulo**, [200-?]. Disponível em: <http://moodle.stoa.usp.br/mod/resource/view.php?id=12776>. Acesso em: 06 mar. 2013.

MORELL, L. J. A model for code-based testing schemes. **Fifth Annual Pacific Northwest *Software* Quality Conf.,** [S.l.], p. 309-326, 1987.

MYERS, Glenford J. **The art of *software* testing**. New York: John Wiley & Sons, 1979.

OLIVEIRA, J. Qualidade no desenvolvimento de *software* – planejamento da qualidade. **Sistemas & Cia**, ago. 2009. Disponível em: <http://sistemasecia.freehostia.com/component/jccmultilanguagecontent/article/49-qualidade-no-desenvolvimento-de-*software*.html?start=2>. Acesso em: 21 mar. 2013.

OLSON, T. **How to improve your *software* inspection process**. 1999**.** Trabalho apresentado na 11th *Software* Engineering Process Group Conference, Atlanta, 1999.

PORTO, Daniel de Paula. **CRISTA: Um apoio computacional para atividades de inspeção e compreensão de código**. 2009. 246 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos, 2009.

PRESSMAN, Roger S. **Engenharia de *Software***. 3 ed. São Paulo: Makron Books do Brasil, 1995.

PRESSMAN, Roger S. **Engenharia de *Software***. 6 ed. Porto Alegre: Editora MCGrawHill, 2010.

PROJECT MANAGEMENT INSTITUTE, INC. **Um guia do conhecimento em gerenciamento de projetos (Guia PMBOK)**. 4 ed. Pensilvânia: PMI Publications, 2008. p. 160-180.

SINFIC. O valor da qualidade de *software*. **Sistemas de Informação Industriais e Consultoria (SINFIC)**, 2006. Seção Artigos. Disponível em: < http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=24417>. Acesso em: 07 mar. 2013.

STAA, A. V. Controle da qualidade de *software*. **Pontifícia Universidade Católica do Rio de Janeiro**, ago. 2012. Disponível em: <http://www.inf.puc-rio.br/~inf2134/docs/INF2134_Modulo08_TecnicasControleQualidade.pdf>. Acesso em: 01 mar. 2013.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC. **Guide to the *Software* Engineering Body of Knowledge (SWEBOK)**. v. 2004. California: IEEE Computer Society, 2004.

TRAVASSOS, Guilherme H. **Revisão e Inspeção de *Software*.** Curso ministrado no IV Experimental *Software* Engineering Latin America Workshop (ESELAW 2007), 2007, São Paulo. Disponível em: <http://lens.cos.ufrj.br:8080/eselaw2007/ESELAW07-TRAVASSOS%28SC2%29.pdf>. Acesso em: 05 mar. 2013.