CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE JAHU

JÉSSICA STRINGHETTA

A IMPORTÂNCIA DO GERENCIAMENTO DA QUALIDADE EM PROJETOS DE SOFTWARES.

JAÚ – SP

2013

JÉSSICA STRINGHETTA

A IMPORTÂNCIA DO GERENCIAMENTO DA QUALIDADE EM PROJETOS DE SOFTWARES.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Jahu, como parte requerida dos requisitos para a obtenção do título de Tecnólogo da Gestão da Tecnologia da Informação.

Orientador: Prof. José Augusto Christianini Filho

Jaú

2013

“Seja um padrão de qualidade. As pessoas não estão acostumadas a um ambiente onde o melhor é o esperado.”

Steve Jobs

DEDICATÓRIA

Dedico este trabalho ao meu namorado Lucas Alexandrino pelo incentivo, confiança, motivação e paciência.

AGRADECIMENTO

Agradeço primeiramente a Deus pelo dom da vida, por ter me amado desde o primeiro momento em que passei a existir no ventre de minha mãe e, desde o início, sempre acreditou e nunca desistiu de minha vida.

Agradeço a minha família de sangue, pelo alicerce que me proporcionaram desde meus primeiros passos, por me ensinarem a batalhar pelos meus sonhos e a nunca desistir, independente da quantidade de obstáculos que poderiam surgir pelo caminho.

À minha segunda família, ao ministério de acolitato, sou eternamente grata por ser minha família de coração, por sempre me apoiarem e rezarem por mim, por compreenderem a minha ausência durante a batalha para a concretização deste trabalho de conclusão de curso e por me proporcionarem as melhores amizades.

Ao meu namorado, por me dar forças e motivação, por sempre me apoiar e caminhar ao meu lado. Agradeço, principalmente, por todo amor e carinho.

Agradeço aos meus queridos professores e mestres, em especial ao meu orientador José Augusto Christianini Filho, por todo apoio, orientação e aprendizado que proporcionaram para o meu crescimento acadêmico e profissional durante os três anos de curso.

Por último, mas não menos especiais, agradeço a todos meus colegas do curso de Gestão da Tecnologia da Informação, em especial as grandes amizades que nasceram da simplicidade. Obrigada por dividirem comigo esta conquista, por alegrarem as minhas noites, por cada compartilhamento de informação e, principalmente, por todo apoio e motivação.

RESUMO

Com o aumento da demanda pela busca de softwares que informatizem os processos internos de empresas, indústrias e organizações ocorreu, também, o aumento da concorrência entre empresas prestadoras de serviços na área da tecnologia da informação. O grande desafio dessas empresas no mercado é obter o reconhecimento pela excelência e qualidade dos serviços prestados e se diferenciar no mercado das demais empresas do ramo. Com o intuito de realizar um estudo sobre a aplicabilidade da qualidade em projetos de software, este trabalho abordará a qualidade aplicada em todo o software desde seus processos iniciais até os finais, visto que com tantas metodologias disponíveis para melhorar o processo qualitativo muitas organizações optam por permanecer na comodidade ao invés de seguir padrões que possibilitem uma melhoria dos produtos e serviços oferecidos aos seus clientes.

Palavras-chave: Gestão da Qualidade, Qualidade do Software, CMM, Ciclo de Vida, Teste de Software.

ABSTRACT

The increase of running to search software that computerizes internal processes of companies, industries and organizations, there was also an increase of competition among service providers in the area of information technology. The challenge of these companies in the market is getting recognition for their excellence and quality of services and be distinguished of other companies in the market industry. In order to conduct a study on the applicability of quality in software projects, this paper approaches the quality applied on software, from the beginning of the process to the end, as with so many methods available to improve the qualitative process many organizations choose to stay on convenience rather than follow patterns that allow an improvement of the products and services offered to their customers.

Keywords: Quality Management, Software Quality, CMM, Life Cycle, Software Testing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ISO – International Organization for Standardization.

IEC – International Electrotechnical Commission

SEI – Software Engineering Institute.

CMM – Capability Maturity Model.

ABNT – Associação Brasileira de Normas Técnicas.

NBR – Norma Brasileira.

ASQC – American Society for Quality Control.

SQA – Software Quality Assurance.

PBQP – Programa Brasileiro da Qualidade e Produtividade em Software.

MPS – Melhoria do Processo de Software.

SUMÁRIO

1 INTRODUÇÃO

A crescente aplicabilidade de softwares nos ambientes empresariais para o gerenciamento da tomada de decisão, junto com a busca por softwares de excelente qualidade e confiabilidade que ofereçam praticidade em seu manuseio, eficiência e segurança nas informações, gerou a necessidade das empresas de tecnologia da informação ter e executar em seus projetos a qualidade como fator essencial em todas as etapas de sua criação.

Em todos os processos envolvidos em um projeto de software, alguns pontos chaves são considerados para que a empresa obtenha o reconhecimento do produto oferecido ao mercado sendo, esses pontos chaves, altamente cobrados por sua qualidade. Por isso, é exigida a utilização de processos e metodologias eficientes que garantam a excelência da qualidade e funcionalidade do software. Porém, com o grande avanço da tecnologia, há diversas ferramentas que podem ser usadas para que sejam alcançados esses objetivos.

Contudo, a utilização das metodologias se diferencia na necessidade interna de cada empresa, pois seus processos e métodos internos possuem características especificas de seu modo de trabalho. A qualidade é analisada e aplicada desde os processos iniciais do software, sendo a análise de requisitos, em todo seu desenvolvimento até os processos finais de testes e implantação do software no cliente.

2 CONCEITOS E ABORDAGENS DO SOFTWARE

O software é uma ferramenta de grande importância na atualidade, que permite a informatização dos processos de diversas empresas e indústrias, melhorando sua segurança em relação às informações, à qualidade e ao tempo gasto nas diversas operações de seu cotidiano.

Segundo Sommerville (2003), “o software é o conjunto de vários artefatos e não apenas o código fonte”.

Já a ISO/IEC[1] conceitua o software como:

> Software é o conjunto completo ou apenas uma parte dos programas, procedimentos, regras e documentação associada de um sistema [computacional] de processamento de informação. (ISO/IEC 2382-1: 1993)

Inicialmente, o software é criado através de instruções escritas, denominadas linhas de códigos, que se tornam o código fonte do software. As instruções são baseadas nos desejos e nas necessidades do usuário e, após sua escrituração, os comandos são lidos por um computador que realizarão as operações conforme foram especificadas.

Através da agilidade nos processos, o software consegue transformar uma grande quantidade de dados em informações de maneira segura, apresentando valores válidos e informações verdadeiras que poderão ser utilizados como auxílio na tomada de decisão da organização de maneira rápida e prática.

2.1 A CRISE

Desde 1970, o termo da crise do software já era utilizado em meio às organizações, expressando as dificuldades encontradas no desenvolvimento e trabalho com os softwares diante da sua crescente demanda e da inexistência de processos qualitativos que permitissem seu correto funcionamento.

[1] International Organization for Standardization / International Electrotechnical Comission

Na Association for Computing Machinery Turing Award, em uma apresentação chamada “The Humble Programmer” em 1972, Edsger Wybe Dijkstra fez uma das primeiras referências ao termo da crise de software:

> A maior causa de crise no software é que as máquinas tornaram-se várias ordens de magnitude mais potentes! Em termos diretos, enquanto não havia máquinas, programar não era um problema; quando tivemos computadores fracos, isso se tornou um problema pequeno e agora que temos computadores gigantescos, programar tornou-se um problema gigantesco. (DIJKSTRA, 1972)

Atualmente, com o avanço da tecnologia é notável o aumento da gravidade da crise do software devido à disseminação da utilização de softwares em todas as atividades humanas. Com esse aumento na utilização dos softwares também ocorreu um grande aumento nas inconsistências encontradas pelos usuários ao manusearem os softwares, deixando-os insatisfeitos com o serviço adquirido.

Assim, surgiu uma das maiores preocupações das empresas prestadoras de serviço do ramo da informática, a oferta por produtos de alta qualidade que atendam os desejos e as necessidades de seus clientes e que diferenciem sua organização das demais empresas.

2.2 CICLO DE VIDA DE UM SOFTWARE

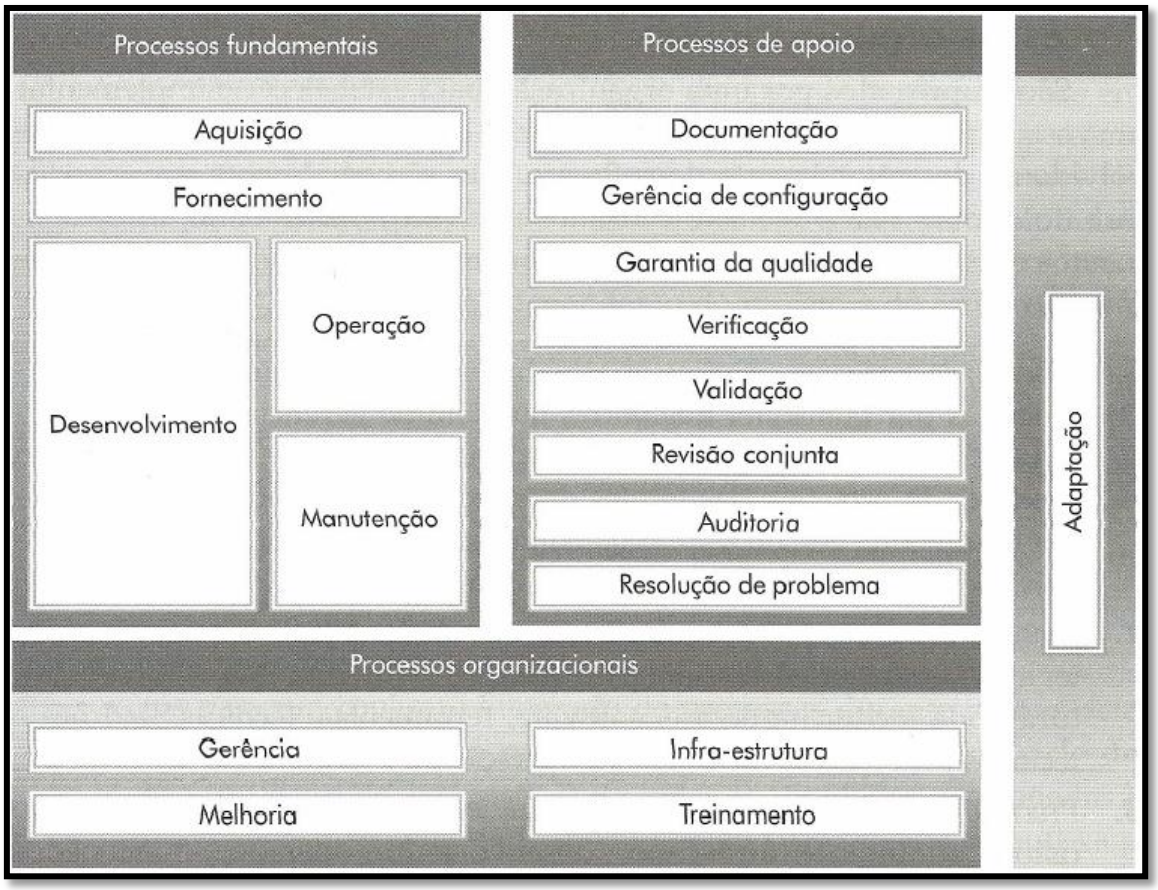
O ciclo de vida de um software é definido pela NBR ISO/IEC 12207 como:

> Ciclo de vida é a estrutura que contém processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema desde a definição de seus requisitos até o término do seu uso. (NBR *ISO/IEC* 12207, 1998)

Possuindo o objetivo de assessorar a definição dos papéis das pessoas envolvidas na produção do software, o ciclo de vida proporciona às organizações uma melhoria no entendimento das atividades e operações de um software que deverão ser executadas, através de processos precisos.

O ciclo de vida do software é baseado em quatro processos, sendo os processos fundamentais, de apoio, organizacionais e adaptação, conforme mostra a Figura 1.

Figura 1 - Processos de ciclo de vida do software.



Fonte: ROCHA; MALDONADO; WEBER, 2001, p. 13.

2.2.1 Processos fundamentais

Os processos fundamentais são responsáveis pelo atendimento das operações iniciais do ciclo de vida do software. Essas operações atendem a contratação da empresa prestadora de serviço pelo cliente e a realização do desenvolvimento, da operação e da manutenção do software. (ROCHA, A. R. C; MALDONADO, J. C.; WEBER; K. C., 2001)

Dividido em cinco tópicos, os processos fundamentais se baseiam nos processos de aquisição, fornecimento, desenvolvimento, operação e manutenção, conforme mostra a Tabela 1.

Tabela 1 – Processos fundamentais

Processos	Características
Aquisição	Definição dos requisitos necessários para que o software possa suprir suas necessidades. Nesta etapa o cliente é quem realiza o gerenciamento do funcionamento e operabilidade do software e, após a finalização dos requisitos, transmite as informações para auxiliar no processo e acompanhar as atividades até que seja feita a análise da entrega e a aceitação através dos padrões definidos inicialmente.
Fornecimento	Abrange atividades e processos referentes ao planejamento de um software, desde a elaboração de uma proposta inicial até a conclusão do mesmo. Nesta etapa são definidos os procedimentos e os recursos necessários para obter o resultado final do projeto.
Desenvolvimento	Realiza o planejamento, execução e concretização do projeto de software, agregando as atividades de ciclo de software: análise de requisitos, projeto, codificação, testes e manutenção.
Operação	Executa as atividades e tarefas condizentes ao uso operacional e realiza a liberação do software para o uso final. Posteriormente, oferece assistência para soluções e esclarecimentos de dúvidas.
Manutenção	Mantém o software atualizado e útil para o cliente, sempre atendendo as necessidades e preservando a sua integridade.

Fonte: Próprio do autor.

2.2.2 Processos de apoio

Os processos de apoio são responsáveis por auxiliar outros processos, sendo considerado como uma parte integrada, tendo o objetivo de contribuir

para que a qualidade do software seja alcançada, mostrados na Tabela 2. (ROCHA, A. R. C; MALDONADO, J. C.; WEBER; K. C., 2001)

Tabela 2 – Processos de apoio

Processos	Características
Documentação	Realiza a documentação das atividades durante o ciclo de vida do software, contribuindo para que haja uma melhor visão do software e criando uma fonte de consulta em relação ao seu uso e a manutenção.
Gerência de Configuração	Gerencia a grande quantidade de informações, sendo possível realizar alterações prevendo melhorias e evitando inconsistências.
Garantia da Qualidade	Realiza a validação do software e analisa se está de acordo com os requisitos solicitados pelo cliente e com as definições do planejamento.
Verificação	Avalia se todos os requisitos solicitados foram atendidos.
Validação	Avalia se o software atende ao objetivo requerido e ao uso operacional.
Revisão	Define métodos e atividades para avaliar o gerenciamento do projeto, tanto estratégico quanto técnico.
Auditoria	Avalia e/ou propõe revisões/alterações em todos os níveis do ciclo de vida do software.
Resolução de Problemas	Define atividades para avaliar e resolver problemas durante o ciclo de vida do software.

Fonte: Próprio do autor.

2.2.3 Processos organizacionais

Os processos organizacionais são responsáveis por estabelecer e implementar uma estrutura subjacente que será constituída através dos processos de ciclo de vida e pessoal, em busca da melhoria contínua da

estrutura e de seus processos, como mostra a Tabela 3. (ROCHA, A. R. C; MALDONADO, J. C.; WEBER; K. C., 2001)

Tabela 3 – Processos organizacionais

Processos	Características
Gerência	Gerencia as atividades tais como: planejar, organizar e executar tarefas para alcançar os objetivos especificados.
Infraestrutura	Define a infraestrutura para os processos envolvidos tanto no ciclo de vida de software quanto no uso operacional do mesmo. Engloba-se hardware, softwares e ferramentas.
Melhoria	Define métodos de avaliação e medição dos processos executados no software e assegurar a qualidade do software final.
Treinamento	Define as metodologias e processos que serão utilizados para o treinamento operacional dos usuários do software.

Fonte: Próprio do autor.

2.3 MODELOS DO PROCESSO DE SOFTWARE

Segundo Sommerville (2011), “um modelo de processo de software é uma representação simplificada de um processo de software”.

Através da utilização dos modelos de processo de software é possível receber uma maior orientação no ciclo de vida do software e obter uma maior visibilidade dos requisitos solicitados pelo cliente.

Veja, abaixo, alguns modelos relacionados ao processo de software:

2.3.1 Cascata

O modelo de cascata, também conhecido como ciclo de vida do software, é um exemplo de processo destinado a planos onde, antes de

começar a trabalhar no desenvolvimento das atividades, elas deverão ser planejadas.

Abordando os estágios deste modelo é possível segmentá-lo em cinco principais estágios, sendo eles (SOMMERVILLE, 2011):

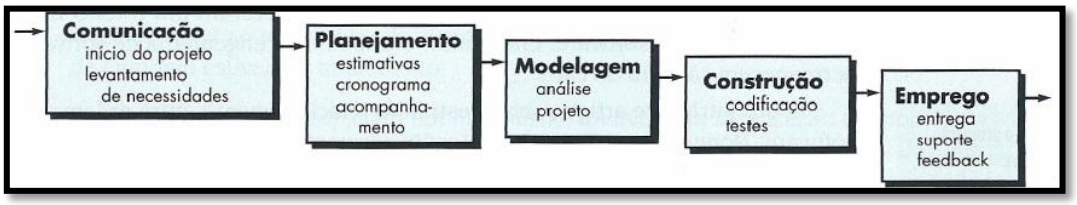
- Análise e definição de requisitos: As características sobre o sistema são estabelecidas de acordo com os desejos e necessidades do usuário, através de sua opinião e participação de um levantamento de dados. Depois de concluído o levantamento dos dados, através dessas informações será desenvolvido uma especificação para o sistema, que abordará todos os seus detalhes e funcionalidades.
- Projeto de sistema e software: Inicialmente é determinada uma arquitetura geral do sistema, envolvendo tanto os requisitos de hardware como os de softwares. Através da arquitetura elaborada começa o desenvolvimento do software, com base nas definições de requisitos especificadas anteriormente.
- Implementação e teste unitário: O software é desenvolvido segmentado por várias áreas, denominadas por conjunto de programas ou unidades de programa. Neste estágio começam a serem realizados os testes unitários em cada área segmentada para verificar se estão atendendo suas especificações.
- Integração e teste de sistema: Neste estágio é realizada a integração de todas as áreas segmentadas do sistema, tornando suas unidades individuais um único sistema. Após a integração, o sistema completo que foi gerado é testado para verificar se atende os requisitos especificados e para analisar o seu correto funcionamento e relacionamento entre todas as áreas do sistema. Depois que os testes forem concluídos, se o sistema estiver aprovado e não apresentar inconsistências, o mesmo será entregue para o cliente.
- Operação e manutenção: Na última fase dos estágios o sistema é colocado em uso na organização sendo essa fase considerada, em alguns casos, como a fase mais longa entre todos os estágios.

Ao entrar em operação, também estará começando o processo de manutenção, onde serão descobertas inconsistências das quais não haviam

sido notadas e começará o processo de correção dessas falhas, além de melhorias através de sugestões e críticas apresentadas pelo usuário.

Veja, na Figura 2, o modelo Cascata:

Figura 2 – Modelo Cascata

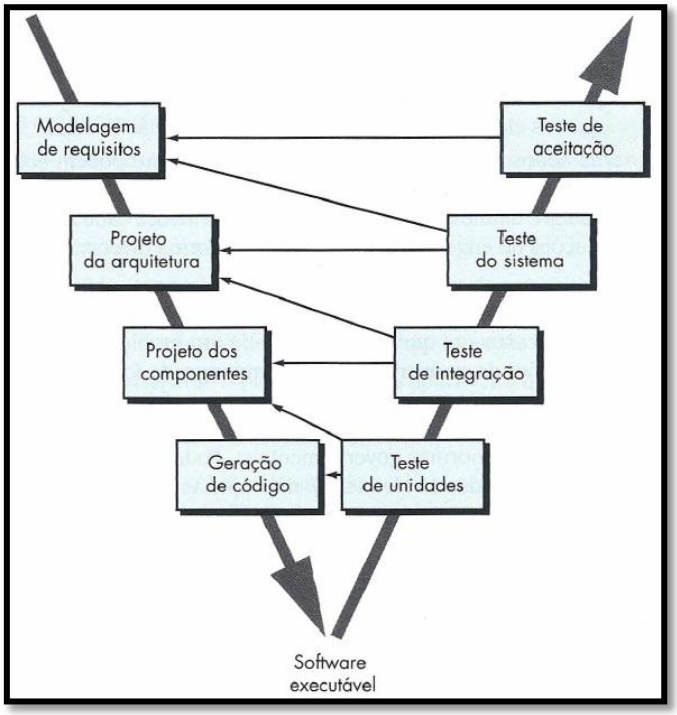


Fonte: PRESSMAN, 2011, p.60.

2.3.2 Modelo em V

O modelo V, considerado uma variante do modelo cascata, é responsável por demonstrar o relacionamento entre atividades de garantia da qualidade e atividades relacionadas à construção inicial do software, como mostra a Figura 3. (PRESSMAN, 2011)

Figura 3 – Modelo em V



Fonte: PRESSMAN, 2011, p.60.

Na representação do modelo V, conforme for descendo o lado esquerdo do V, os requisitos básicos do problema serão refinados detalhadamente.

Após a geração de código, ao descer o lado direito do V, será realizada uma diversidade de testes para a validação dos processos anteriormente executados.

Ao chegar na junção dos dois lados do V, após todos os processos serem aprovados, teremos o software executável.

2.3.3 Incremental

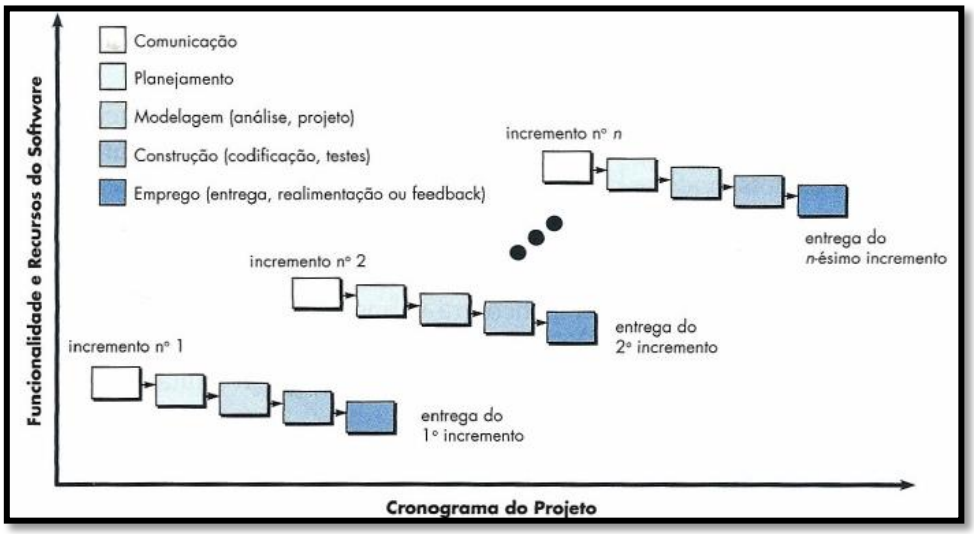
O modelo incremental é destinado no desenvolvimento de uma implementação inicial, visando expor esta implementação para o usuário com o intuito de receber elogios ou críticas sobre o projeto.

Este modelo é conhecido por receber um rápido feedback do usuário em relação às suas atividades, sejam elas de especificação, desenvolvimento ou validação dos processos.

Através das melhorias e correções são formadas várias versões neste modelo onde, a junção destas versões, entregará ao término do projeto sua versão final.

Veja, na Figura 4, o modelo Incremental:

Figura 4 – Modelo Incremental



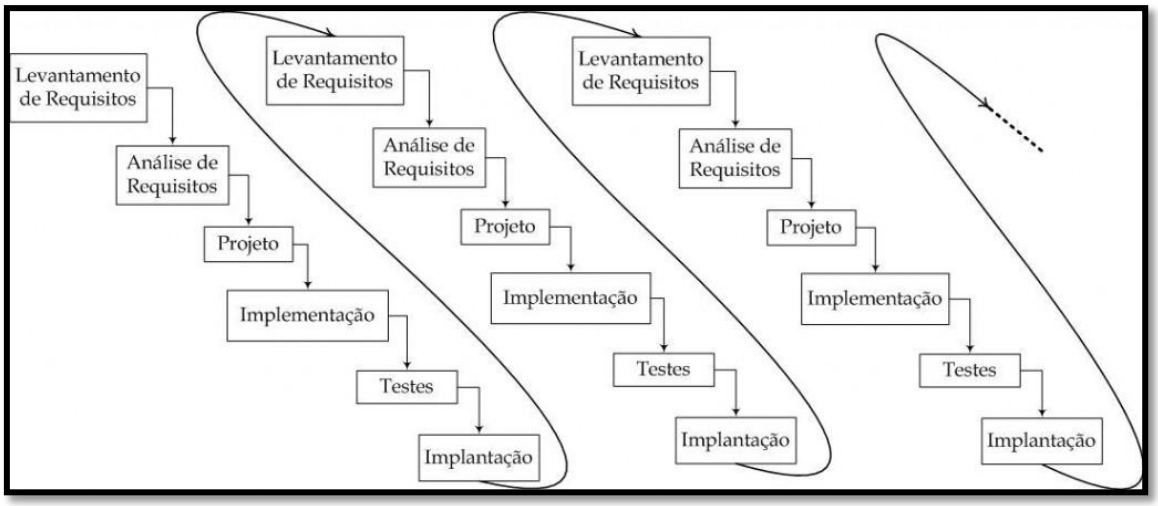
Fonte: PRESSMAN, 2011, p.61.

2.3.4 Iterativo

Baseado no modelo incremental, o modelo iterativo é um exemplo de processo, também considerado como uma estratégia, de planejamento de retrabalho. Através desse modelo de processo são pré-definidos o tempo de revisão e a melhoria de algumas partes do sistema.

A diferença entre o modelo iterativo do modelo incremental é que a saída de uma iteração é examinada para modificação, para revisão de objetivos e iterações posteriores, como mostra a Figura 5.

Figura 5 – Modelo Iterativo



Fonte: Página do portal Caverna do Software[2].

2.3.5 Espiral

Proposto por Boehm (1988), o modelo espiral é considerado como um framework destinado aos riscos de um processo de software.

O modelo é representado de uma forma espiral, não apresentando uma sequência de atividades para serem seguidas ao longo de seu desenvolvimento.

Segundo Sommerville, “cada volta na espiral representa uma fase do processo de software”. Estas voltas são divididas em quatro setores:

[2] Disponível em: <http://voat.com.br/rdal/?tag=iterativo> Acesso em Nov. 2013.

- Planejamento: Neste setor são estabelecidos os objetivos específicos desta fase, elaborando um plano detalhado do gerenciamento do projeto, identificando suas restrições e riscos do projeto.

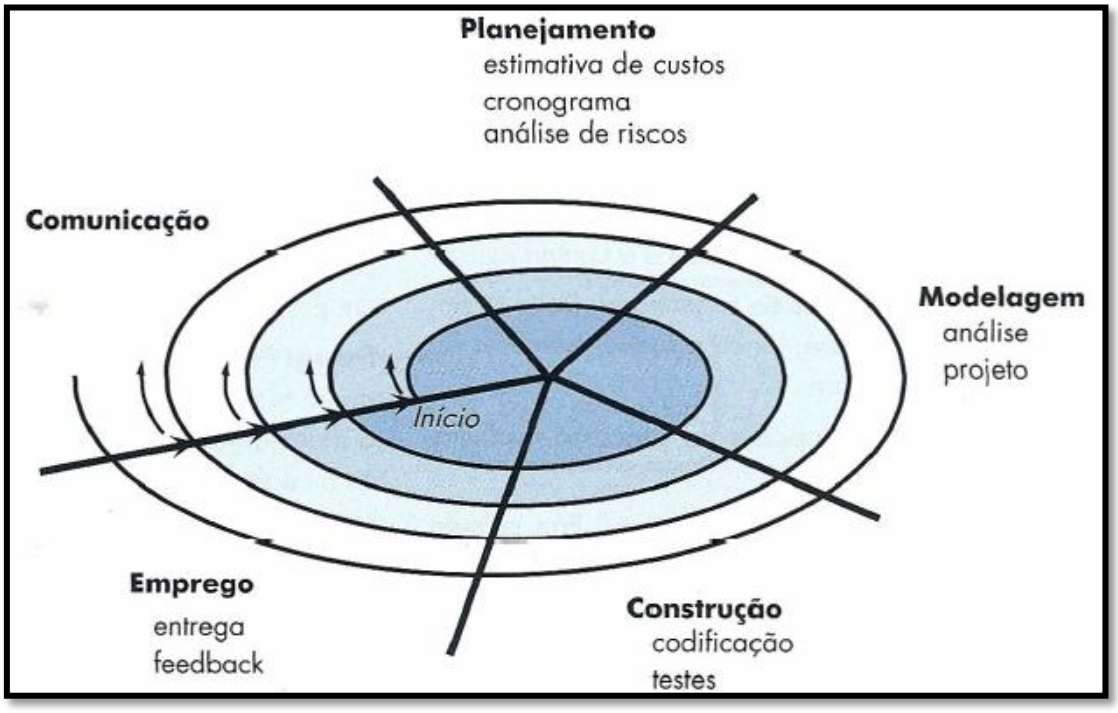
- Modelagem: Através de cada risco encontrado no setor anterior será realizada uma análise minuciosa, avaliando quais são as melhores medidas a serem tomadas para a redução do risco.

- Construção: Depois de realizar a avaliação dos riscos, um modelo de desenvolvimento será estabelecido para o sistema, realizando a sua validação.

- Emprego: Neste último setor, o projeto será revisado sendo, em seguida, decidido sobre sua conclusão ou continuidade através de mais uma volta no modelo espiral.

Veja, na Figura 6, o modelo Espiral:

Figura 6 – Modelo Espiral



Fonte: PRESSMAN, 2011, p.65.

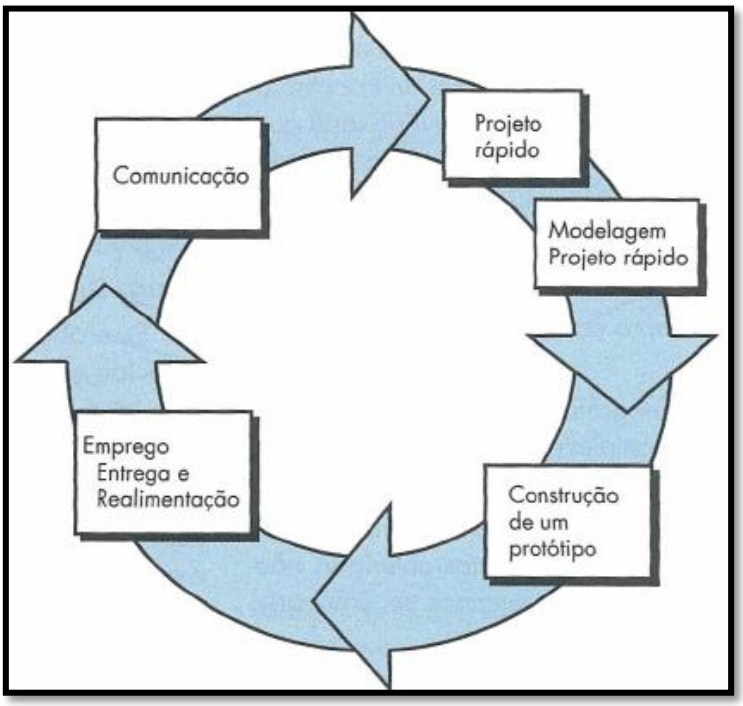
2.3.6 Prototipagem Evolutiva

O modelo de prototipagem evolutiva, considerado uma variante do modelo espiral, é utilizado para a construção de diversos protótipos, sendo responsáveis por cobrirem progressivamente os requisitos, até o software desejado ser alcançado.

Através de seu paradigma de prototipação, com o modelo de prototipagem evolutiva é possível ter uma melhoria na compreensão de seus requisitos, quando estiverem obscuros, facilitando as ações que deverão ser tomadas para a construção do software.

Veja, na Figura 7, o modelo de Prototipagem Evolutiva:

Figura 7 – Modelo do paradigma da prototipação



Fonte: PRESSMAN, 2011, p.63.

Cada modelo possui suas características, sendo necessário para cada modelo seguir sua metodologia. Apesar dos modelos serem parcialmente diferentes, a adoção de uma metodologia é extremamente importante para a obtenção da qualidade do software.

2.4 A IMPORTÂNCIA DA UTILIZAÇÃO DE UMA METODOLOGIA

Com o aumento da busca pela informatização, seguir uma metodologia na criação de um software se tornou algo fundamental para as empresas prestadoras de serviço do ramo da tecnologia da informação.

Devido a grande quantidade de metodologias existentes é necessário fazer uma análise minuciosa para encontrar a metodologia adequada para o projeto que será criado.

Cometer um erro na escolha da metodologia pode ser prejudicial à todo o projeto, porém, pior do que cometer o erro é não utilizar uma metodologia.

A grande importância que as metodologias apresentam é o controle do projeto como um todo, desde a análise do projeto, seu desenvolvimento, possíveis melhorias, correções ou prevenções de futuras inconsistências, previsão de tempo gasto no desenvolvimento, cumprimento do escopo do cliente e resultados que serão alcançados com o término do projeto.

3 GESTÃO DA QUALIDADE

A qualidade corresponde ao grau de eficiência e excelência obtida como resultados.

A qualidade é definida pela International Organization for Standardization (ISO[3]) como um “conjunto das propriedades e características de um produto, processo ou serviço, que lhe fornecem a capacidade de satisfazer as necessidades explícitas ou implícitas”.

Já a Sociedade Americana para o Controle da Qualidade (ASQC – American Society for Quality Control) define que a qualidade implica em um nível de atendimento, sendo “definida como a totalidade de características e atributos de um produto ou serviço que possuem a habilidade de satisfazer uma certa necessidade”.

Em relação às abordagens da qualidade, Garvin relata a existência de cinco abordagens principais em sua definição, sendo transcendental, baseada no produto, no usuário, na produção e no valor, conforme mostra a Tabela 4. (Garvin, 2002).

Tabela 4 – Abordagens da qualidade segundo Garvin

Abordagens da qualidade segundo Garvin	
Transcendental	A qualidade é vista não apenas com o funcionamento, mas a busca pelo padrão mais alto de ser alcançado.
Baseada no produto	“Diferenças de qualidade correspondem a diferenças de quantidade de algum ingrediente ou atributo desejado” (Abbott, 1955).
Baseada no usuário	Capacidade de satisfazer os desejos dos usuários.
Baseada na produção	Cordialidade com a especificação do projeto.
Baseada no valor	Analisada através do grau de excelência e oferecido por um preço justo e aceitável.

Fonte: Próprio do autor.

[3] ISO (International Organization for Standardization) Organização não governamental, fundada em 1946, que possui sua sede na cidade de Genebra (Suíça), estando presente em cerca de 157 países.

3.1 QUALIDADE DO SOFTWARE

Segundo Peters (2001), “qualidade de software é avaliada em termos de atributos de alto nível chamados fatores, que são medidos em relação a atributos de baixo nível chamados de critérios”.

Pressman define a qualidade de software como:

> Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais. (PRESSMAN, 1995)

Segundo Bartié (2002), a qualidade do software pode ser definida como “um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos”.

Para alcançar a qualidade de software é necessária a aplicabilidade de fatores qualidade.

3.2 FATORES DE QUALIDADE

Os fatores da qualidade podem ser divididos em seis características, sendo elas: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Para cada uma das características existem, também, suas subcaracterísticas.

3.2.1 Funcionalidade

A característica da funcionalidade representa um conjunto de funções, providas das operações das quais o software é capaz de realizar, que possuem o objetivo de satisfazer as necessidades do usuário, sejam elas implícitas ou explícitas.

As subcaracterísticas desta característica são (CÔRTES;CHIOSSI, 2001):

- Acurácia: Analisa os resultados que estão sendo gerados pelo software, verificando sua exatidão e se estão dentro dos resultados que são esperados.
- Adequação: Verifica se o software está realizando apropriadamente as operações que foram especificadas.
- Conformidade: Analisa se estão sendo seguidos os padrões e regras que foram estabelecidos para o software, verificando se ele está de acordo com as normas e leis.
- Interoperabilidade: Analisa a capacidade que o software possui em interagir com outros sistemas que foram especificados para essa interação.
- Segurança de Acesso: É responsável por prevenir acessos não autorizados ao sistema, base de dados, entre outros.

3.2.2 Confiabilidade

A confiabilidade mensura a capacidade que o software possui em manter o seu nível de desempenho através das condições que foram estabelecidas para um determinado período, evitando o surgimento de falhas.

Suas subcaracterísticas são (CÔRTES;CHIOSSI, 2001):

- Maturidade: Indica o nível de maturidade do software, ou seja, informa a frequência de apresentação de falhas e inconsistências no software causados por defeitos do sistema.
- Recuperabilidade: Analisa a capacidade que o software possui em recuperar os dados e fazer seu reestabelecimento em caso de ocorrências de falhas.
- Tolerância de Falhas: Analisa a capacidade que o software possui em reagir às falhas e manter um determinado nível de desempenho mesmo com a presença das inconsistências.

3.2.3 Usabilidade

A característica da usabilidade mensura o esforço necessário à utilização do software pelo usuário, seja-o implícito ou explícito. Entre a mensuração do esforço encontra-se a avaliação da utilização, analisando-a se possui facilidade de compreensão e de manuseio.

As subcaracterísticas referentes à usabilidade são (CÔRTES;CHIOSSI, 2001):

- Apreensibilidade: Analisa a facilidade de uso do software, ou seja, o grau necessário que um usuário deve aplicar para aprender a utilizar o sistema.
- Inteligibilidade: Realiza a medição da facilidade que o usuário dispõe para interagir com o software e reconhecer as lógicas de seus processos.
- Operacionalidade: Analisa se o software é fácil de ser operado e se oferece um simples controle de suas operações.

3.2.4 Eficiência

A eficiência é a característica responsável por analisar, através de condições preestabelecidas, o relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados.

Suas subcaracterísticas são (CÔRTES;CHIOSSI, 2001):

- Comportamento com relação ao tempo: Realiza a medição do tempo de processamento e de resposta das execuções de suas operações.
- Comportamento com relação ao uso de recursos: Faz a medição da quantidade de recursos que serão utilizados na execução das operações do software, além de estabelecer o tempo de duração de suas utilizações.

3.2.5 Manutenibilidade

A característica da manutenibilidade é responsável por avaliar o esforço necessário para a realização das alterações no software que foram especificadas. Entre essa análise, o grau de facilidade à modificação do software é verificado.

As subcaracterísticas da manutenibilidade são (CÔRTES;CHIOSSI, 2001):

- Analisabilidade: Realiza a medição do esforço necessário aplicado na identificação de uma falha e sua causa, além de identificar os dados que deverão ser alterados para realizar a correção das falhas.
- Estabilidade: Analisa os riscos existentes do surgimento de efeitos inesperados no software, providos das alterações realizadas no sistema.
- Modificabilidade: Analisa a facilidade de realizar alterações e adaptações no software para eliminar as falhas encontradas no sistema ou adequá-lo para determinadas operações ou mudanças em seu ambiente.
- Testabilidade: Verifica se o software possui simplicidade de validação após suas falhas serem corrigidas, realizando a medição dos esforços aplicados em seus testes.

3.2.6 Portabilidade

Através da característica da portabilidade é mensurada a capacidade que o software possui em ser transferido do ambiente no qual se encontra para outro ambiente operacional, ou seja, é analisado se o software possui a facilidade de ser utilizado em diversos ambientes.

Suas subcaracterísticas são (CÔRTES;CHIOSSI, 2001):

- Adaptabilidade: Analisa a facilidade que o software possui em se adaptar em outros ambientes operacionais, diferente de seu ambiente comum.
- Capacidade de coexistência: Verifica se o software está em conformidade com os padrões de portabilidade.

- Facilidade de instalação: Realiza a medição do esforço que deve ser aplicado na instalação do software, ou seja, verifica se a instalação do mesmo é simples.
- Facilidade de substituição: Faz a medição do esforço aplicado na utilização do software em substituição de outro sistema que tenha sido previamente especificado.

Para dar garantia aos fatores de qualidade é necessária a execução de testes para validar suas funcionalidades.

3.3 TESTE DO SOFTWARE

O teste do software é o processo responsável pela execução do software, analisando se o mesmo atingiu as especificações que foram estabelecidas no início do projeto.

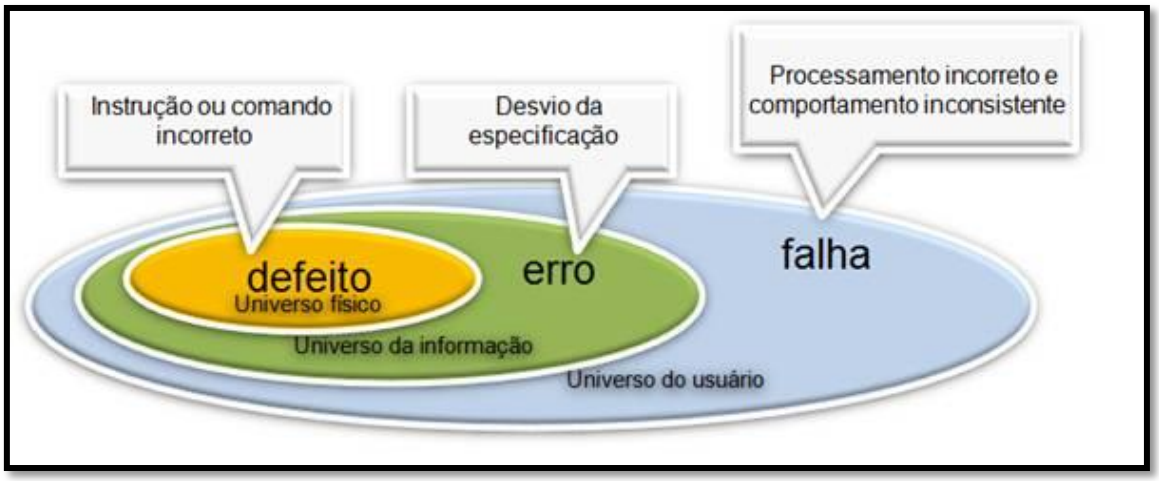
Como objetivo principal, o teste de software deve identificar as possíveis falhas que o software possa apresentar durante sua operacionalidade, visando sua correção e alcance dos requisitos solicitados.

3.3.1 Conceitos básicos encontrados no Teste do Software

No teste de software, três conceitos fundamentais de identificação são defeitos, erros e falhas, conforme mostra a Figura 8.

- Defeito: Processo incoerente resultante de uma instrução ou comando incorreto, após ser executada uma determinada ação.
- Erros: Concretização de um defeito onde, após a ação executada, resulta em algo imprevisto ou incoerente.
- Falhas: Atuação operacional não esperada do software devido a erros.

Figura 8 – Defeito x Erro x Falha



Fonte: Página do DEVMEDIA.[4]

3.3.2 Níveis de Teste

Segundo Lewis (2005), teste de software "é uma estratégia popular para o gerenciamento de risco". Dividido em 4 níveis, o teste do software é segmentado como mostra a Tabela 5:

Tabela 5 – Níveis de teste do software

Níveis de teste do software	
Unidade	Realiza a exploração das unidades do projeto, em busca de falhas ocasionadas por defeitos de lógicas ou erros que não foram percebidos pelo desenvolvedor.
Integração	Busca por falhas na interface e nas relações entre os módulos do sistema que possuem integração.
Sistema	Busca por falhas em todo o sistema, representando o usuário final que utilizará o software.
Aceitação	Realização de simulações das operações que serão utilizadas pelos usuários finais em suas rotinas, analisando se o software está de acordo com o que foi especificado.

Fonte: Próprio do autor.

[4] Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035#ixzz2kUAJ2HAg> Acesso em Nov. 2013.

Existe, também, o teste de regressão. Esse teste não se refere a um determinado nível, sendo considerado como uma estratégia para testes. Seu objetivo é ser aplicado em cada nova versão do software que for gerada, revendo todos os níveis de teste que já foram analisados. Assim, com o teste de regressão, é possível ter a certeza de que nenhum processo novo ou alteração realizada possa afetar os processos já existentes.

3.3.3 Técnicas de Teste de Software

As técnicas para testar um software são inúmeras, mas todas possuem o mesmo objetivo de localizar erros e falhas no software.

São duas as técnicas mais conhecidas e utilizadas sendo a estrutural, também denominada por teste da caixa branca, e a funcional, também conhecida por teste da caixa preta.

Técnica Estrutural

A técnica estrutural é responsável por avaliar o comportamento interno dos componentes do software. Essa técnica opera diretamente sobre o código fonte do componente, avaliando aspectos como testes de condição, fluxo de dados, ciclos e de caminhos lógicos. (PRESSMAN, 2011)

Possui o nome de técnica estrutural devido todas as estruturas de condições serem testadas, sendo destinada essa técnica para os níveis de teste de unidade e integração.

Teste Funcional

Diferente da técnica estrutural, o teste funcional não avalia o comportamento interno dos componentes do software, sendo conhecido como teste da caixa preta, onde não é possível enxergar o seu interior.

Através de resultados já esperados, essa modalidade executa os dados que são fornecidos como entrada e obtém seus resultados. Em seguida, é feita uma análise entre os resultados obtidos e os esperados e, se os dois estiverem de acordo, essa modalidade de teste é considerada bem sucedida.

4 CAPABILITY MATURITY MODEL

O CMM (Capability Maturity Model) é um conjunto de processos que foi desenvolvido pelo SEI (Software Engineering Institute) com o objetivo de ajudar as organizações na melhoria de seus processos de software, desde um processo caótico até um processo maduro.

Construído sobre o conceito de processos e baseado na maturidade dos processos de software, o CMM é dividido em cinco níveis de maturidade que propõem demonstrar uma escala crescente sobre a visão dos processos e dos resultados do projeto de software. (CÔRTES;CHIOSSI, 2001)

4.1 NÍVEIS DE MATURIDADE

Os cinco níveis de maturidade do CMM são: inicial, repetível, definido, gerenciado e em otimização.

4.1.1 Inicial

Considerado o primeiro nível de maturidade do CMM, o nível inicial é conhecido por seus procedimentos não pertencerem ao projeto, mas sim às pessoas.

Devido ao fato anterior, a organização acaba assumindo compromissos que não é capaz de cumprir, muitas vezes relacionados às questões de prazo e custo, desiste de manter um padrão de procedimentos e apenas realiza o que é considerado fundamental.

4.1.2 Repetível

O cumprimento da realização dos procedimentos para o gerenciamento do software é obedecido no nível repetível, assim como o acompanhamento de prazos, estimativa e custos.

Conhecido por seus procedimentos pertencerem à equipe, o nível repetível possui processos que são extremamente gerenciais. Uma das

principais características deste nível é que o planejamento de novos projetos é realizado com base em projetos semelhantes que já foram criados, agindo sempre formalmente.

4.1.3 Definido

Dando continuidade ao nível anterior, o nível definido pertence à organização e é responsável por definir os processos que serão utilizados e padronizá-los em toda a empresa. Assim, os processos de engenharia de software caminham juntamente com os processos gerenciais.

4.1.4 Gerenciado

No nível gerenciado, a gestão passa a pertencer às bases quantitativas dos processos do software, sendo realizada uma coleta sobre as medidas de qualidade e produção para uma análise contínua do desempenho do software.

4.1.5 Em otimização

Baseado na melhoria contínua dos processos do software, ações preventivas são realizadas, assim como alterações relacionadas ao custo e benefício.

Todas as melhorias realizadas nesse processo são consideradas parte dos procedimentos de rotina.

4.2 BENEFÍCIOS

Através da implementação do modelo CMM, diversos objetivos benéficos podem ser alcançados e levados para dentro das organizações. Esses objetivos podem ser encontrados desde a etapa de execução e controle do software até as etapas referentes à qualidade e aos relacionamentos de uma organização.

Nas etapas de execução e controle é possível analisar a melhoria da distribuição das atividades e alocação dos recursos, assim como a melhoria na organização, economia de tempo e custos, facilidade em alcançar as metas e aumento da produção. Já as etapas referentes à qualidade e aos relacionamentos é notável uma melhoria na qualidade do software, na comunicação e identificação das necessidades e desejos, assim como liderança da concorrência. (MEZZENA, 2005)

5. ANÁLISE ESTATÍSTICA

Através do Programa Brasileiro da Qualidade e Produtividade em Software, realizado em 1995, iniciou-se uma pesquisa sobre a qualidade do software no Brasil e, até hoje, vem sendo realizada.

A Secretaria de Política em Informática (SEPIN) do Ministério da Ciência e Tecnologia (MCT) é a responsável pelo desenvolvimento e divulgação dessa pesquisa.

5.1 DADOS DA PESQUISA

De acordo com a pesquisa de qualidade no setor de software brasileiro do ano de 2009, sendo a última pesquisa gratuita liberada, a base digital de dados cadastrais chegou a 2.587 organizações, de diversos portes e lugares, do ramo da tecnologia da informação que possuíam atividades de software e manifestaram interesse em participar da pesquisa.

Entre as organizações que responderam a pesquisa, foi possível analisar alguns dados estatísticos, sendo a grande maioria das organizações privadas e independentes, localizadas no Sudeste do país.

Com relação à gestão e a qualidade, foram apresentados os seguintes resultados de distribuição:

5.1.1 Certificação utilizando a Norma ISO 9000

Entre as organizações que responderam a pesquisa, 64% das organizações possuem a certificação vigente da ISO 9000, como mostra a Figura 9. Essa norma visa a implementação de um sistema da qualidade na organização como um todo, não sendo voltada apenas para o desenvolvimento do software.

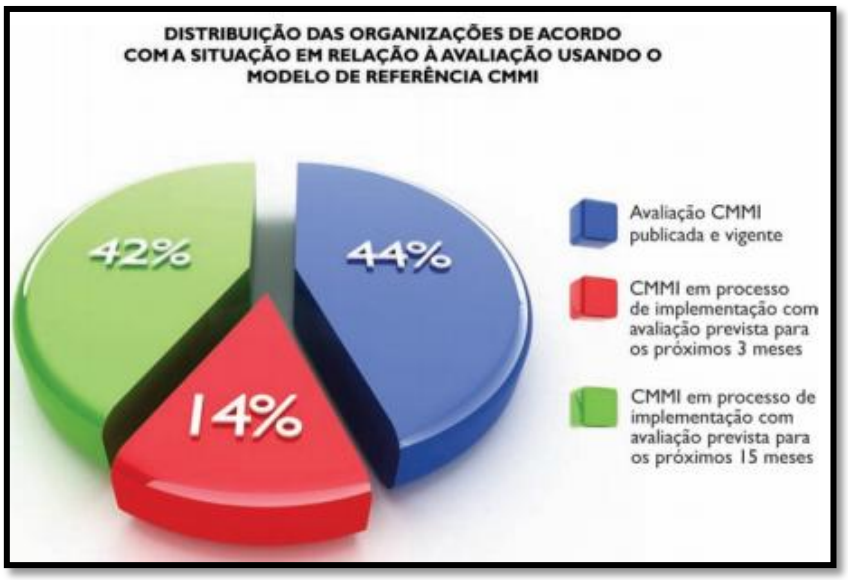
Figura 9 – Distribuição das organizações em relação à certificação usando a Norma ISO 9000.

Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.[5]

5.1.2 Utilização do modelo de referência do CMMI

Das respostas obtidas na pesquisa, 44% das organizações utilizam o modelo de CMMI de forma vigente, mostrado na Figura 10.

Figura 10 – Distribuição das organizações de acordo com a situação em relação à avaliação usando o modelo de referência CMMI.



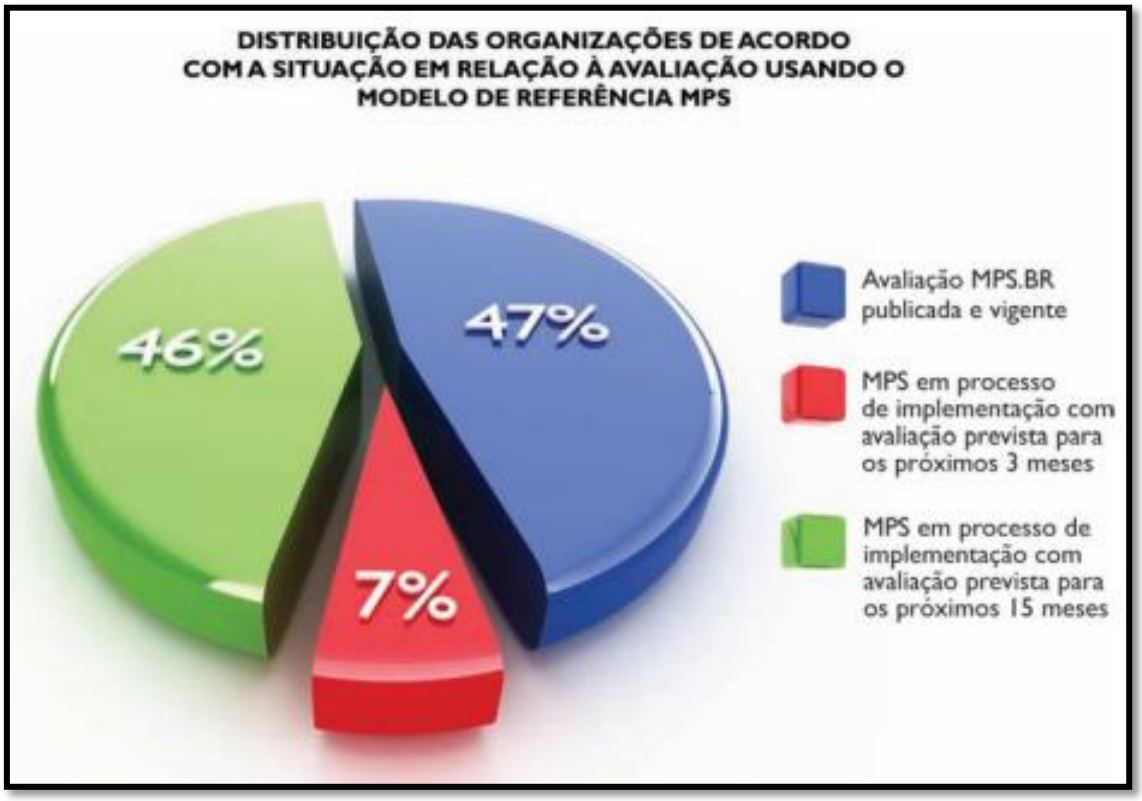
Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.

[5] Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf> Acesso em Nov. 2013.

5.1.3 Utilização do modelo de referência do MPS

Entre as respostas obtidas na pesquisa, 47% das organizações utilizam o modelo de CMMI de forma vigente, conforme apresenta a Figura 11.

Figura 11 – Distribuição das organizações de acordo com a situação em relação à avaliação usando o modelo de referência MPS.



Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.[6]

5.1.4 Medição do desempenho dos processos de software

Entre as organizações que responderam a pesquisa, é notável a igualdade entre as empresas que realizam a medição de desempenho de seus processos de software de forma sistemática e eventual, sendo 40% das organizações para cada caso, como mostra a Figura 12.

[6] Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf> Acesso em Nov. 2013.

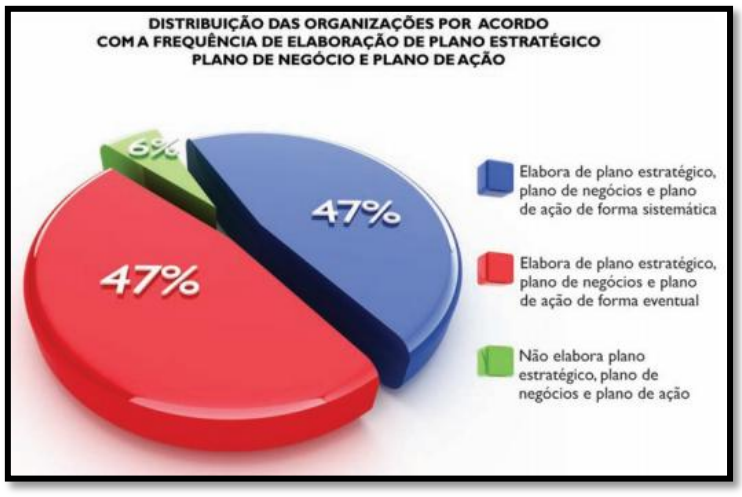
Figura 12 – Distribuição das organizações de acordo com a frequência de medição do desempenho do processo de software.

Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.[7]

5.1.5 Elaboração de planos estratégicos, de negócios e ações

Na análise da distribuição referente às organizações que elaboram planos estratégicos, de negócios e ações é notável os valores igualitários na elaboração desses planos de maneira sistemática e eventual, conforme mostra a Figura 13.

Figura 13 – Distribuição das organizações de acordo com a frequência de elaboração do Plano Estratégico, Plano de Negócios e Plano de Ação



Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.

[7] Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf> Acesso em Nov. 2013.

5.1.6 Inclusão de metas ou diretrizes de qualidade e produtividade

Com os dados da distribuição de inclusão de metas ou diretrizes, 53% das organizações as praticam de forma sistemática 44% de forma eventual, como mostra a Figura 14.

Figura 14 – Distribuição das organizações de acordo com a frequência de inclusão de metas ou diretrizes de qualidade e produtividade nos planos

Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009[8].

5.1.7 Nível de maturidade do MPS para contratação de fornecedores

De acordo com a análise do nível de maturidade do MPS exigido para a contratação de fornecedores é notável que a grande maioria das organizações exija o MPS de nível F – Gerenciado e, em segundo o lugar, o nível E – Parcialmente Definido, conforme mostra a Figura 15.

[8] Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf> Acesso em Nov. 2013.

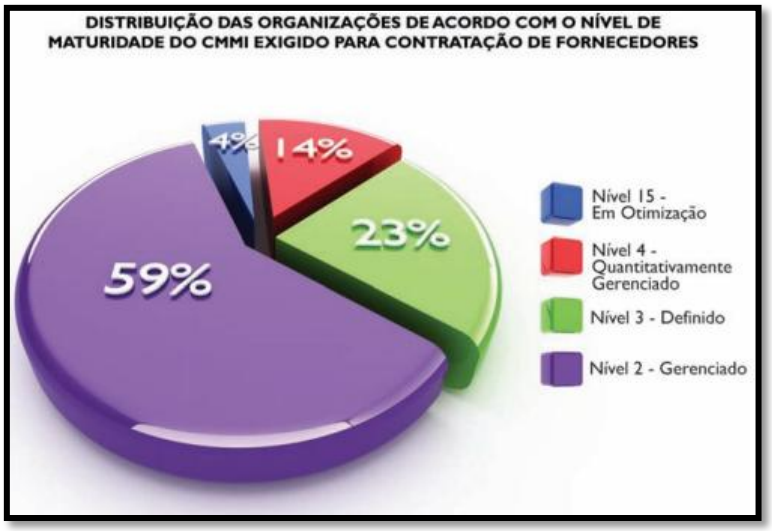
Figura 15 – Distribuição das organizações de acordo com o nível de maturidade do MPS exigido para contratação de fornecedores.

Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009[9].

5.1.8 Nível de maturidade do CMMI para contratação de fornecedores

De acordo com a análise do nível de maturidade do CMMI exigido para a contratação de fornecedores é notável que a grande maioria das organizações exija o CMMI de nível 2 – Gerenciado, mostrado na Figura 16.

Figura 16 – Distribuição das organizações de acordo com o nível de maturidade do CMMI exigido para contratação de fornecedores



Fonte: Pesquisa de Qualidade no Setor de Software Brasileiro 2009.

[9] Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf> Acesso em Nov. 2013.

6 CONCLUSÃO

A análise da pesquisa de qualidade no setor de software brasileiro foi de grande importância para a conclusão deste trabalho.

Atualmente, as organizações prestadoras de serviço do ramo da tecnologia da informação consideram importante a aplicabilidade da qualidade do software. Porém, como foi possível concluir através dos dados relevantes da pesquisa, muitas empresas ainda se ausentam na aplicabilidade da qualidade de software.

Existem diversas ferramentas para trabalhar com a qualidade de um software como, por exemplo, citados neste trabalho CMM, CMMI e MPS. Entretanto, não existe uma determinada ferramenta que seja a melhor para se trabalhar, mas existe a ferramenta que seja mais recomendada para o tipo de software que será criado.

Através deste trabalho, concluímos que a qualidade do software necessita ser melhorada nas organizações brasileiras, visto que com tantas orientações e possibilidades para serem seguidas, as organizações preferem ficar na comodidade ao invés de aplicar um padrão de qualidade que possa melhorar seus produtos e serviços.

REFERÊNCIAS BIBLIOGRÁFICAS

BARTIÉ, A. **Garantia da Qualidade de Software**. Rio de Janeiro: Editora Campus, 2002.

BOEHM, B. **A Spiral Model of Software Development and Enhancement**. Vol. 21. IEEE Computer: 1988.

BUENO, C. F. S. **Qualidade de Software.** Centro de Informática UFPE. Disponível em: <www.cin.ufpe.br/~qualisoft/documentos/diversos/quality.doc>. Acesso em: 4 de out. 2013.

CÔRTES, M. L.; CHIOSSI, T. C. S. **Modelos de Qualidade de Software.** Campinas: Editora da Unicamp, 2001.

DIJKSTRA, E. W. **The humble programmer.** Communications of the ACM. ACM: Commun. 1972.

FILHO, W. P. P. **Engenharia de Software: Fundamentos, Métodos e Padrões.** 3ª ed. Rio de Janeiro: LTC, 2012.

GARVIN, D. A. **Gerenciando a qualidade: A visão estratégica e competitiva.** Rio de Janeiro: Qualitymark, 2002.

GOMES, N. S. **Qualidade de Software – Uma necessidade.** ROAI. Disponível em: <http://roai.joinville.udesc.br:8080/xmlui/bitstream/handle/123456789/15/Qualidade%20de%20Software.pdf?sequence=1>. Acesso em 13 nov. 2013.

ISO/IEC 2382-1:1993. **Information Technology - Vocabulary** - **Part 1: Fundamental terms**. 1993.

JUNIOR, I. M.; CIERCO, A. A.; ROCHA, A. V.; MOTA, E. B.; LEUSIN, S. **Gestão da qualidade.** 10ª ed. Rio de Janeiro: FGV, 2010. (Gestão Empresarial)

LEWIS, W. **Software Testing and Continuous Quality Improvement**. Auerbach, 2005.

MARCHI, M. C. **Qualidade de Software: Uma Abordagem ao PMBOK.** 2008. 46 f. Trabalho de Conclusão de Curso (Tecnólogo em Informática) - Faculdade de Gestão Financeira, Faculdade de Tecnologia de Jahu, Jaú, 2008.

MEZZENA, B. **Benefícios e Dificuldades da Implantação do Modelo CMM: Estudo de Caso.** EAD USP. Disponível em: <http://www.ead.fea.usp.br/tcc/trabalhos/TCC_Bruno%20Mezzena.PDF>. Acesso em: 18 nov. 2013.

NASCIMENTO, J. C. B. **Gestão de Projetos e Qualidade de Software.** Fatec Taquaritinga. Disponível em: http://fotos.fatectq.edu.br/a/6052.pdf. Acesso em: 22 out. 2013.

NBR ISO/IEC 12207. **Tecnologia de informação:** Processos do ciclo de vida do software. Rio de Janeiro: Nova York, 1998.

NETO, A. C. D. **Engenharia de Software – Introdução a teste de software.** DevMedia. Disponível em: <www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035#ixzz2kUAJ2HAg>. Acesso em: 12 nov. 2013.

PAVÃO, I. C. **Gestão da Qualidade para Testes de Software conforme NBR ISO/IEC 12207.** Universidade Católica de Santos. Disponível em:

<http://biblioteca.unisantos.br/tede/tde_busca/arquivo.php?codArquivo=190>. Acesso em: 30 de set. 2013.

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

PRESSMAN, R. S. **Engenharia de Software:** Uma abordagem Profissional. 7ª ed. Porto Alegre: Bookman, 2011.

ROCHA, A. R. C.; MALDONADO, J. C. ; WEBER, K. C. **Qualidade de Software: Teoria e Prática.** São Paulo: Pearson, 2001.

SODRÉ, C. C. P. **Norma ISO/IEC 9126: Avaliação de Qualidade de Produtos de Software.** Instituto Federal Catarinense. Disponível em: <http://www.ifc-camboriu.edu.br/~catia/BSI11/tcc-cibele-sodre-2006.pdf>. Acesso em: 10 de out. 2013.

SOMMERVILLE, I. **Engenharia de software**. 6. ed., São Paulo: Addison Wesley, 2003, 592p.

SOMMERVILLE, I. **Engenharia de Software.** 9ª ed. São Paulo: Pearson, 2011.

ZULIANI, M. R. **A Qualidade de Software no Brasil: Uma Questão de Sobrevivência das Empresas Fabricantes de Softwares.** 2003. 83 f. Trabalho de Conclusão de Curso (Tecnólogo em Informática) – Faculdade de Gestão da Produção Industrial, Faculdade de Tecnologia de Jahu, Jaú, 2003.