UFSM

Dissertação de Mestrado

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA SOB A ÓTICA DA QUALIDADE DE SOFTWARE

Marcos André Storck

PPGG

Santa Maria, RS, Brasil

2006

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA SOB A ÓTICA DA QUALIDADE DE SOFTWARE

POR

Marcos André Storck

Dissertação apresentada ao Programa de Pós-Graduação em Geomática, Área de concentração em Gerenciamento e Informática Rural, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Geomática.**

Santa Maria, RS - Brasil
2006

Storck, Marcos André

S884d

Desenvolvimento de sistemas de informação geográfica sob a ótica da qualidade de software / por Marcos André Storck ; orientador Enio Giotto . – Santa Maria, 2006.
65 f. : il.

Dissertação (mestrado) – Universidade Federal de Santa Maria, Centro de Ciências Rurais, Programa de Pós-Graduação em Geomática, RS, 2006.

1. Geomática 2. Sistemas de informação geográfica 3. SIG 4. Programas de computação 5. Informática 6. Software I. Giotto, Enio II. Título

CDU: 528.7/.9:004.4

Ficha catalográfica elaborada por
Luiz Marchiotti Fernandes – CRB 10/1160
Biblioteca Setorial do Centro de Ciências Rurais/UFSM

Contato: marcos.storck@gmail.com

Universidade Federal de Santa Maria
Centro de Ciências Rurais
Programa de Pós-Graduação em Geomática

A comissão examinadora abaixo assinada,
Aprova a Dissertação de Mestrado

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA SOB A ÓTICA DA QUALIDADE DE SOFTWARE

Elaborada por
Marcos André Storck

Como requisito para obtenção do grau de
Mestre em Geomática

Comissão Examinadora

_____
Prof. Dr. Ênio Giotto

_____
Prof. Dr. Pedro Roberto de Azambuja Madruga

_____
Prof. Dr. Eugênio de Oliveira Simonetto

Santa Maria, 22 de março de 2006.

AGRADECIMENTOS

São muitas as pessoas que merecem ser agradecidas neste trabalho, dentre elas, primeiramente gostaria de demonstrar a importância do apoio de minha família que foi crucial para não só esta dissertação existir, mas tudo em minha vida. Dentre muitas coisas agradeço:

Ao meu pai; agradeço o auxílio técnico e apoio moral em horas de dificuldade.

A minha mãe; agradeço por sempre estar perto quando precisei, me ajudando da sua maneira especial.

A minha irmã Carla; agradeço por não me atrapalhar quando devia e sua alegria que sempre contagia o ambiente.

A minha irmã Cátia, e ao novo membro de nossa família, Rafael seu marido, agradeço pela atenção e parceria constante.

Agradeço também ao programa de Pós-Graduação em Geomática, e principalmente meu Orientador e todos meus colegas de Mestrado, pois sem a ajuda e amizade destas pessoas eu nunca conseguiria chegar ao final do mestrado.

Agradeço meus amigos, queridos amigos, em que depositei minhas frustrações, procurei alivio e dei boas risadas nas horas tensas do decorrer deste caminho.

Finalizando, gostaria de fazer um agradecimento especial e uma homenagem póstuma a meu padrinho, Flávio Miguel Schneider, uma pessoa que deve servir de modelo tanto profissionalmente quanto pessoalmente. São muitas as palavras a serem ditas, mas creio que falando por todos que já o conheceram, terminar com um simples “sentimos sua falta” resumiria tudo.

Demonstre força aonde tens fraquezas e despreocupação aonde és forte, assim quem o subjugar será derrotado. Sun Tzu, 500 a.c.

RESUMO
Dissertação de Mestrado
Programa de Pós-Graduação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA SOB A ÓTICA DA QUALIDADE DE SOFTWARE
Autor: Marcos André Storck
Orientador: Ênio Giotto
Data e local da defesa: Santa Maria, 22 de março de 2006.

A procura pela qualidade em sistemas computadorizados nos dias atuais se tornou uma prática no mínimo necessária pela importância das atribuições que damos aos mesmos. Cada vez dependemos mais de computadores, para os mais devidos fins, e assim nos tornamos dependentes de sua exatidão. Devido a esta dependência que a procura pela excelência em termos de software se dá como um ato de extrema importância durante o desenvolvimento de sistemas. Sistemas de Informações Geográficas, ou SIG são programas que manipulam com dados que possuem a capacidade de serem referenciados geograficamente, e devido a complexibilidade destes dados e dos procedimentos utilizados para manipula-los, visualiza-los e processa-los, o desenvolvimento de SIG's se torna uma prática muito dificulta. Estudos feitos sobre Qualidade de Software, dentro da área da Engenharia de Software, resultaram em alguns métodos e técnicas que visam facilitam o alcance da qualidade durante o desenvolvimento de sistemas, e como muitas vezes SIG's são desenvolvidos por profissionais voltados as áreas rurais, estes podem não ter conhecimento de tais estudos sobre Qualidade de Softwares. Este trabalho procura suprir alguns conhecimentos sobre qualidade de software para profissionais não oriundos da informática, também propondo uma metodologia que auxiliará a organizar todo o desenvolvimento de um SIG, facilitando o alcance da qualidade de software.

Palavras-chave: Sistemas de Informação geográfica, Qualidade de Software, Engenharia de Software

ABSTRACT

Master's Dissertation
Master degree program in Geomatic
Federal University of Santa Maria

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO GEOGRÁFICA SOB A ÓTICA DA QUALIDADE DE SOFTWARE
(Developing Geographical information System Under the Software Quality Optic)
Author: Marcos André Storck
Adviser: Ênio Giotto
Defense: Santa Maria, march, 22 of 2006

The search for the quality of software in the current days became one practical, at least, necessary for the importance of the attributions that we give to them. Each time we depend more and more from software, for the most diverse ends, and thus we become dependents of its exactness. Had to this dependence that the search for the Excellency in software terms became as an act of extreme importance during the development of software. Geographic Information System, or SIG, are software that manipulate data that possess the capacity to have a geographically reference, and due the complexity of these data and the procedures used to manipulate, visualize and process them, the development of SIG becomes a very difficult practical. Studies made on Software Quality, in the area of Software Engineering, had resulted in some methods and techniques that they aim to facilitate the reach of quality during the software development, and, as many times, SIG are developed by professionals from the land areas, these can not have knowledge of such studies on software quality. This work looks for to supply some knowledge on software quality for professionals that are not from computer science area, also considering a methodology that will assist to organize the development process of a SIG, facilitating the reach of the software quality.

Key-words: Geographic Information Systems, Software Quality, Software Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente com o desenvolvimento das tecnologias, está sendo cada vez maior a informatização de sistemas, para os mais diversos fins, utilizando o poder de processamento dos computadores, para melhorar a qualidade das informações necessárias nas mais diversas áreas, bem como produzir as informações mais rapidamente e com um alto nível de precisão.

Com este avanço da tecnologia da informação, começou-se a utilizar o computador para processar, entre outros, dados geográficos, também chamados de dados espaciais. Estes dados possuem em sua atribuição uma referência no mundo real por meio de uma localização geográfica dentro de um sistema de coordenada. (LISBOA FILHO, 1995)

Programas de computador que utilizam estes dados são chamados de Sistemas de Informação Geográfica (SIG). Os SIG's, por possuírem esta característica de processar dados, que tem referência no mundo real, podem ter inúmeras aplicações.

Mesmo com tantas aplicações, nas mais diversas áreas, o desenvolvimento de um SIG geralmente é feito, ou tem como participante muitas vezes um profissional com o perfil de ter em sua atribuição lidar com dados geográficos. Estes profissionais podem ser Geógrafos, Engenheiros Florestais, Agrônomos, Engenheiros Cartógrafos, e outros profissionais provenientes das ciências da terra.

Esta realidade se dá pelo motivo destes profissionais serem oriundos das ciências exatas e da terra, e assim por trabalharem mais, e diretamente com dados geográficos tem uma maior compreensão dos dados que um SIG processa, assim podendo criar procedimentos que tratam de melhor forma possível os dados e retornam resultados mais compreensíveis pelo profissional e com maior nível de exatidão.

Um SIG não deixa de ser um software. Assim Brooks apud Fiorini et al (1998) descreve que o desenvolvimento de um software, independente da sua finalidade, é uma tarefa muito árdua, e que infelizmente não apresenta qualquer forma de ser

facilitada. O mesmo autor ainda afirma que a construção de um software infelizmente é uma tarefa impossível de ser simplificada.

O desenvolvimento de um sistema não pode então poder ser simplificada. No entanto estudos feitos sobre Qualidade de *Software*, dentro da disciplina da Engenharia de Software, resultaram em técnicas, métodos e meios para controlar, facilitar e estruturar da melhor maneira possível o desenvolvimento de um sistema computadorizado, para que este resulte num produto final de maior qualidade possível.

A Engenharia de Software é uma disciplina oriunda da área da informática, e, assim, estes estudos podem não ser de conhecimento do perfil dos desenvolvedores, mencionados anteriormente, por serem de áreas diferentes. Logo, a Qualidade de Software pode não estar sendo aplicada no desenvolvimento de Sistemas de Informação Geográfica. Também, é importante o conhecimento de como um produto de *software* torna-se de qualidade. Para que um programa de computador tenha qualidade é necessário que sua construção seja bem estruturada, pois é esta fase que prevê o futuro do produto de *software*, se vai ser ou não de qualidade. O desenvolvimento então é o ponto critico para que um software seja de qualidade.

Com a finalidade de se produzir sistemas de qualidade, foram desenvolvidas várias técnicas, normas, modelos e projetos que estruturam todo o desenvolvimento de um produto de *software* para que este no final se torne de melhor qualidade possível.

Um SIG, devido a complexibilidade dos dados e dos processos que envolvem todo o sistema, se torna um sistema computadorizado de difícil desenvolvimento necessitando de um processo estruturado e meticuloso para que no fim se obtenha um software funcional. Como um software que não executa as funções as quais foi desenvolvido se torna inútil, justifica-se a importância da qualidade em um software.

Neste trabalho foi criada uma metodologia para o desenvolvimento de SIG e feita uma revisão bibliográfica sobre Qualidade de *Software* que visa suprir conhecimentos que beneficiarão o desenvolvimento destes sistemas para que no fim, estes, possuam o maior número de itens que caracterizam um sistema de qualidade.

Foi desenvolvido um estudo sobre Qualidade de *Software* no desenvolvimento de um SIG, e uma proposta de uma estruturação para o desenvolvimento de um

SIG, com sugestões de técnicas que colaboram com um processo de desenvolvimento de software de qualidade.

Foi um estudo sobre a qualidade de software, suas características e métodos para se adquirir esta qualidade e seu uso no desenvolvimento de softwares.

Logo após será definido o que é um SIG para que o conceito deste tipo de sistemas seja bem entendido, e por fim será estabelecido uma sugestão estruturada para o desenvolvimento de SIG's sob a ótica da Qualidade de Software.

2 REVISÃO BIBLIOGRÁFICA

2.1 Qualidade de *software*

Um cliente, ao comprar um *software*, assim como qualquer outro produto, sempre tem a preferência dos melhores. Pode-se dizer que os melhores produtos são aqueles que possuem melhor qualidade, tanto no produto em si quanto no modo pelo qual é fabricado, e em se tratando de produtos de *software*, a qualidade é ainda mais importante devido às aplicações que são realizadas pelos mesmos.

A seguir será dissertada a história da qualidade de *software* para então introduzir o assunto propriamente dito.

2.1.1 A história da qualidade

A parte histórica da qualidade de *software* reflete na história da qualidade de modo geral. Antigamente, durante o século XX (vinte) a qualidade de algum produto era responsabilidade somente do artesão que o construísse.

Os primeiros indícios de preocupação com qualidade e controle do mesmo surgiram em 1916, e espalhou-se rapidamente dentre os países industrializados, e por meados dos anos 40 novas abordagens foram sendo criadas, surgindo um contínuo processo de aperfeiçoamento da gerência da qualidade, Pressman (2002).

A figura 1 demonstra um quadro resumo da evolução da qualidade. (Molinari, 2003).

Inicialmente a qualidade de software acompanhava o desenvolvimento de hardwares, pois a qualidade do software não era prioridade.

Assim, a garantia da qualidade em *softwares*, como nos produtos fabricados no século 20 , eram de total responsabilidade do programador nos primórdios da computação, entre o ano de 1950 e 1960.

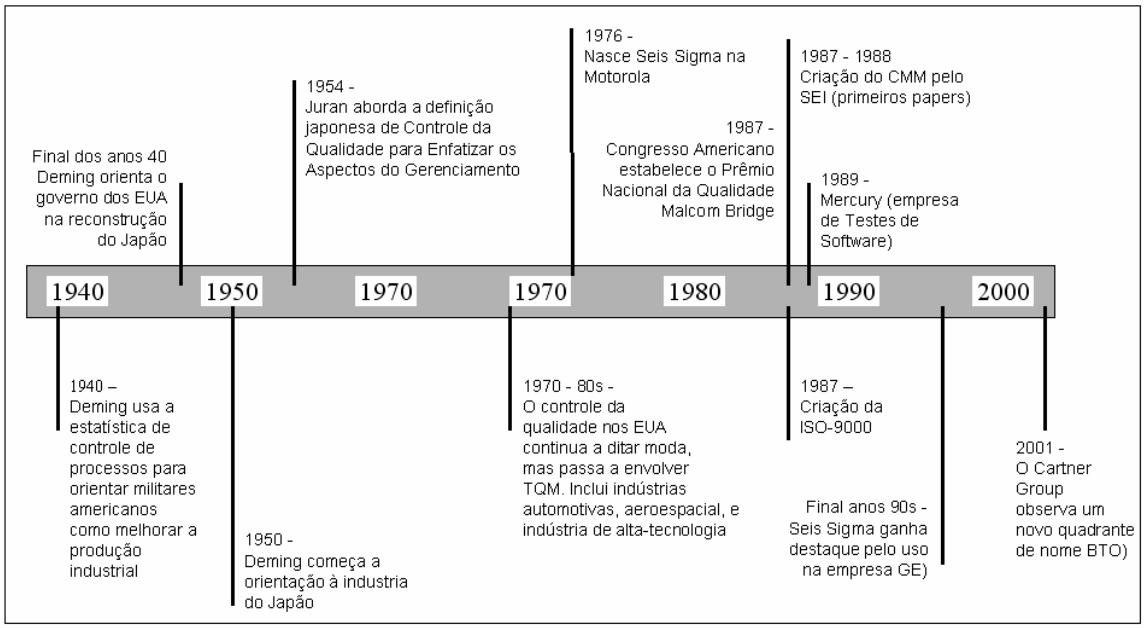

Figura 1: Marcos históricos da qualidade de *software.* (Molinari, 2003).

A padronização voltada à garantia de qualidade de *softwares* iniciou nos anos 70, onde foram criados os primeiros modelos de qualidade. No final da década de 80 criou-se a ISO – 9000, que era uma norma voltada a padronizar o processo de garantia de qualidade no mundo, mas não era muito usual para desenvolvimento de *softwares*. Posteriormente, por incentivo de militares foi criado um modelo de qualidade voltado para *software* chamado CMM (Capability Maturity Model), que será explanado de forma mais aprofundada mais adiante neste trabalho. (Fiorini et al. 1998)

Segundo Oliveira *apud* Lima (2001), o conceito de qualidade evoluiu com o tempo, encontrando uma inversão de perspectiva tradicional, onde não é o produto/serviço que gera qualidade, mas sim a qualidade que gera o serviço/produto. Mas, mudando um pouco o escopo da história e falando de qualidade de *software*, é importante deixar esclarecido o que é a qualidade de *software* e sua importância.

2.1.2 Conceitos de qualidade de *software*

O *software*, como produto, tem se tornado cada vez mais um componente absolutamente indispensável na sociedade. Logo, a qualidade nos produtos de *software* é essencial no desenvolvimento, e se tornou um item que não pode ser subjugado na consolidação das empresas desenvolvedoras, além de questões sociais envolvidas em aplicações em *software* em áreas críticas, como por exemplo, usinas nucleares e aeroportos. (Flores, 2001).

Qualidade de *software* em si é uma parte do estudo da Engenharia de *Software*, que é a disciplina ou ciência que se ocupa do estudo de todas as características do desenvolvimento de *software*, e segundo Inthurn (2001), engloba três elementos fundamentais que são: métodos, ferramentas e procedimentos.

Também, se pode definir a engenharia de software como uma metodologia de desenvolvimento e manutenção de sistemas modulares. (Rezende, 1999).

Outra definição para a Engenharia voltada a produção de *softwares* é que esta é a ciência que se preocupa em estudar e gerar soluções para o desenvolvimento de um *software* como um todo, aonde todos os processos do desenvolvimento são submetidos sob gestão de gerenciamento e melhoramento. (Sommerville, 2001).

Pressman (2002), define a qualidade de *software* como: “a totalidade das características de uma entidade, que lhe confere capacidade de satisfazer necessidades explícitas e implícitas”.

Consta nos princípios desta disciplina que um *software* de qualidade deve ser funcional em todos os aspectos; implícitos (requeridos) e explícitos (subjetivos). (Flores, 2001).

Para melhor entender o significado de qualidade voltada a *software*, deve-se entender sua definição, dentre elas temos:

- “A garantia da qualidade de *software* é um padrão planejado e sistemático de ações”. (Schulmeyer *apud* Pressman, 2001);
- “A qualidade não é apenas a ausência de defeitos. Na perspectiva do cliente, a qualidade significa tanto a presença de valor sim como a ausência dos defeitos”. (Arthur, 1994);
- “Qualidade é definida como a característica essencial de algo que calcula o seu grau de excelência”. (Webster’s Dictionary *apud* Lewis, 2000);

- “Garantia de qualidade: O estabelecimento de uma estrutura de procedimentos e de padrões organizacionais, que conduzam ao *software* de alta qualidade”. (Sommerville, 2001); e,
- “*Software* de qualidade é aquele que atende as necessidades dos usuários”. (Humphrey, 1995).

E ainda:

> Qualidade é a conformidade com os requisitos, ou, adaptabilidade ao uso, adequação ao cliente e/ou usuário; atendimento perfeito de forma confiável (sem defeitos), acessível (baixo custo), segura e no tempo certo às necessidades do cliente; é a ausência de desperdício, é a atitude (Rezende, 1999).

A qualidade deve ser vista sob a perspectiva do usuário, ou cliente, e por isso deve-se levar em consideração o que é importante para este. Um bom *software* atende as expectativas do cliente. (Zorzo, 2001). Assim, a problemática da excelência em *software* está em embutir qualidade em um software sem deixar de cumprir os objetivos que o software tem, pois um *software* deve cumprir os seus objetivos para se comportar da maneira esperada pelo cliente/usuário e assim ser funcional.

Existem três maneiras de um *software* falhar no cumprimento de seus objetivos. (Shiller, 1992) :

1) O *software* é implementado com base em uma má análise do sistema que não representa a realidade da política de negócios. Existem quatro critérios cujos analistas devem se preocupar no momento da especificação do sistema.

- Clareza - a especificação tem de ser clara, sem chances de má interpretação das informações nela contidas;
- Precisão - a especificação deve ser precisa quanto a sua validade das suas informações;
- Integridade - a especificação deve conter somente informações dentro do escopo relevante do sistema; e,
- Concisão: a especificação deve conter somente informações necessárias e relevantes ao escopo do sistema.

2) À parte de análise das especificações do *software* pode ser boa, mas se o mesmo é implementado de forma errada o sistema falhará nos seus objetivos. As principais falhas são:

- Má escolha de tecnologia - uso de módulos de serviços inadequados para o fim que o sistema tem; e,
- Otimização prematura - uso de otimizações durante o desenvolvimento que podem vir à destruição da estrutura do *software*.

3) A tecnologia ou recursos insuficientes para o desenvolvimento do *software*, como por exemplo: número de profissionais insuficientes (*Peopleware*), restrições no orçamento e tempo insuficiente.

Portanto, um software de qualidade deve eliminar estas três maneiras de falhas durante seu desenvolvimento, mantendo sua integridade e seus objetivos.

Existem algumas confusões populares quando se fala do assunto de qualidade de *software*, são elas. (Lewis, 2000):

- Qualidade requer comprometimento, toda a organização tem que trabalhar em conjunto;
- Aceitação de certos níveis de defeitos nos produtos de *software* causado por acreditar que é impossível o desenvolvimento de *software* com nível zero de erro;
- Qualidade está intimamente ligada a custos;
- O nível de especificações do sistema (a análise do cliente) deve ter um bom nível de detalhamento para que o produto seja de qualidade; e,
- Uso de procedimentos pré-definidos para alcance da excelência em *software*.

Arthur *apud* Rezende (1999) cita alguns conceitos que devem ser considerados para sistemas baseados em computadores que querem adquirir uma política de alta qualidade:

- Defeito zero - é importante para as aplicações aonde o acontecimento de erros, mesmos mínimos, podem acarretar em repercussões desastrosas;
- Requisitos funcionais adequados - a aplicação, ou *software* deve possuir um número de funções adequadas ao uso do cliente/usuário;
- Codificação estruturada e elegante - os programadores e engenheiros de *software* devem documentar e organizar a construção do produto de *software* para que futuras modificações sejam possíveis e simplificadas;
- Desempenho satisfatório - o tempo de resposta do *software* deve ser de acordo com as necessidades do cliente/usuário;

- Custo adequado - o investimento para um desenvolvimento de um *software* com qualidade deve espelhar nos resultados e retornos do mesmo;
- Desenvolvimento rápido e produtivo - uso de uma elaboração adequada e bem gerenciada do desenvolvimento do *software,* para que seja dado o término deste de modo rápido e eficiente; e,
- Facilidade para o cliente e/ou usuário - o produto de *software* desenvolvido deve ter fácil manipulação e suas funções devem ser bem acessíveis para comodidade do usuário/cliente.

Levando em consideração estes conceitos, acredita-se que o desenvolvimento de um produto de *software* tem mais possibilidade de adquirir várias vantagens como: qualidade maior no produto final, otimização no desenvolvimento, eliminação de atividades inúteis, maior desempenho no tempo de entrega do produto, entre outros. É impossível produzir um *software* de qualidade sozinho, pois isto é um grande processo que envolve toda uma organização/empresa, aonde todos, como um conjunto, devem trabalhar com o intuito de desenvolver produtos com excelência de qualidade. Pois neste processo deve-se observar itens como: estabelecimento de conceitos e objetivos, envolver e motivar todas as pessoas, de todos os escalões, definir produtos que serão externados, formalizar procedimentos, criar medidas de desempenho, performance e índices, fornecer condições e ambiente pró-ativos, medir e avaliar resultados. (Rezende, 1999).

Para coordenar uma grande operação de desenvolvimento de um *software* com qualidade foram criados modelos, normas e projetos pré-definidos que ajudam, utilizando uma série de regras pré-estabelecidas, em estabelecer, como alcançar a qualidade. Um programa para atingir qualidade e produtividade em informática é um grande empreendimento que deve absorver todos os setores e partes de uma organização; tais como: pessoas, atividades, métodos, entre outros.

Alguns itens muito importantes devem ser observados em um programa de qualidade e produtividade. (Rezende, 1999):

- Estabelecer conceitos e objetivos;
- Envolver e motivar todas as pessoas, de todos os escalões;
- Definir produtos que serão externados;
- Formalizar procedimentos;

- Criar medidas de desempenho, performance, índices;
- Fornecer condições e ambientes pró-ativos; e,
- Medir e avaliar resultados.

No momento que uma organização adota um programa de qualidade, este deve passar por um processo de melhoria contínua de seus processos, podendo ser exemplificado na Figura 2. (Humphrey, 1997).

Explicando este processo de melhoria contínua temos os seguintes passos:

- Definição das metas de qualidade - primeiramente deve-se estabelecer qual o nível de qualidade que se deseja atingir;
- Medição da qualidade do produto - verificar a qualidade atual dos produtos que estão sendo desenvolvidos;
- Compreender o processo - estabelecer as regras de como o processo do desenvolvimento irá funcionar;
- Ajustar os processos – com o passar do tempo, deve-se efetuar ajustes nos processos para aumentar sua efetividade;
- Utilizar os processos ajustados - após o ajustamento dos processos, deve-se iniciar imediatamente o uso dos mesmos;
- Medir os resultados - sob uma dada perspectiva se medem os resultados; e,
- Comparar os resultados com as metas - usando a medição dos resultados, se comparam estes dados com as metas do processo de melhoria. Caso a comparação der valor negativo; deve-se retornar ao ajustamento dos processos, e, assim, até que no momento de comparação o resultado seja positivo.

Existem muitas formas de conseguir atingir a qualidade, uma delas é a padronização de processos. No momento que um processo é padronizado os produtos decorrentes sempre serão iguais, e isto é um tipo de qualidade segundo Fiorini et al. (1998).

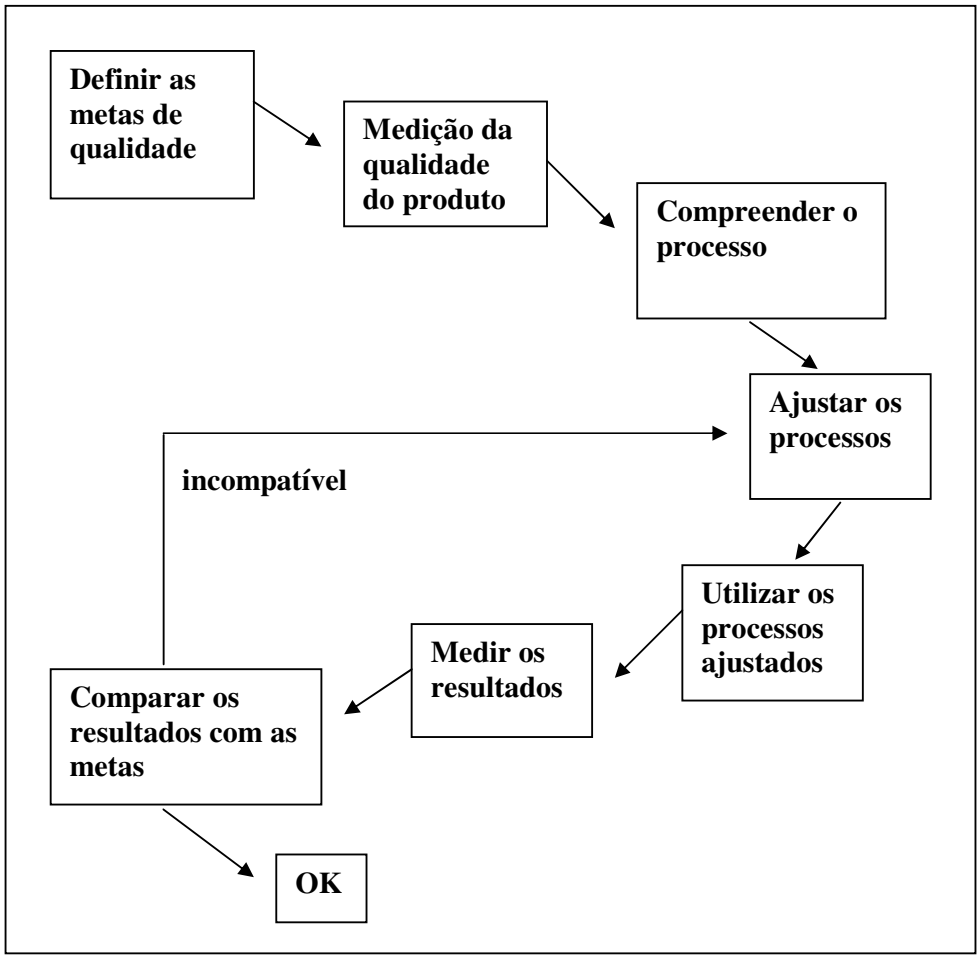

Figura 2: Fluxograma do esquema da melhoria contínua de processos. (Humphrey, 1997).

Com a padronização e a modelagem de processos, se pode atingir a qualidade, pois desta maneira se tem a vantagem de poder revisar os passos dos processos a procura de erros e a busca contínua da melhoria dos processos. Para estas revisões são comumente feitas as seguintes perguntas. (Pfleeger, 2001):

- Onde e como costumam ocorrer um tipo particular de erro?
- Como se pode achar os erros de forma mais rápida durante o processo de desenvolvimento?
- Existe alguma maneira de tornar os processos mais otimizados garantindo maior qualidade?

Existem dois tipos de padrões que podem ser estabelecidos em um sistema de qualidade de *software*. (Sommerville, 2001):

- Padrões de produtos - padronizações que se aplicam ao produto de *software*. Incluem padronização de documentos, padrões de codificação (Linguagem de Programação), entre outros.
- Padrões de processo - padronizações que se aplicam aos processos que devem ser seguidos no desenvolvimento de um produto de *software*. Incluem definições de especificação, processos de projeto, entre outros.

Ainda existem várias razões pelas quais a padronização, em termos de qualidade de *software*, é importante, (Sommerville, 2001), dentre elas:

- Eles, os padrões, fornecem um encapsulamento das melhores práticas e/ou pelo menos das mais bem sucedidas. Os padrões registram sabedoria que tem valor para a empresa;
- Fornecem um embasamento, no qual um processo de garantia de qualidade pode ser implementado; e,
- Padrões ajudam na continuidade dos processos, pois o trabalho padronizado de um funcionário pode ser assumido e continuado por outro.

Tanto se estuda em padronização para obter excelência em *software*, que na verdade o que isso significa é melhorar os processos que estão envolvidos no desenvolvimento de um *software*.

O interesse pela melhoria contínua dos processos de *software* é a busca pela garantia de qualidade do produto de *software*. A figura 3 exemplifica esta abordagem.

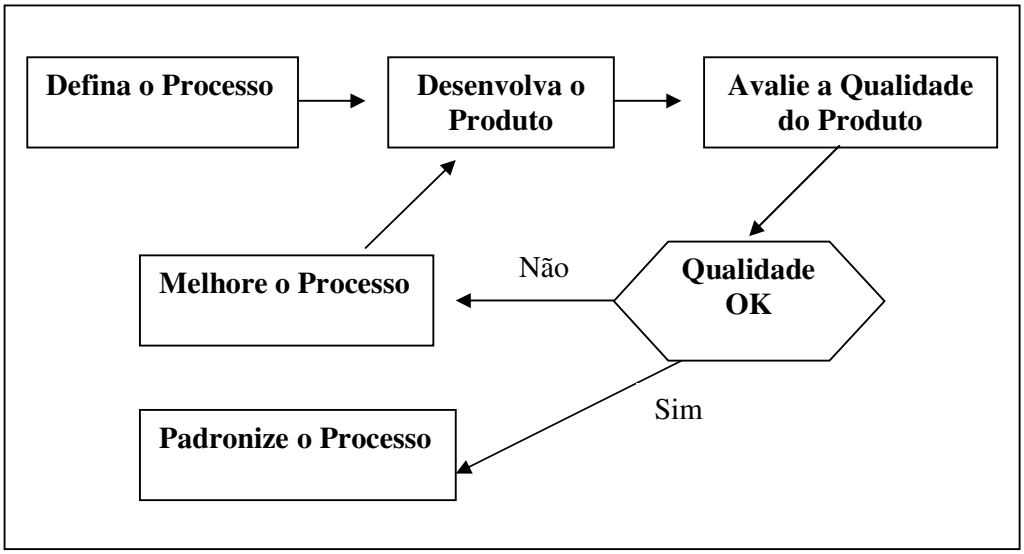

Figura 3: Diagrama de busca contínua pela melhoria dos processos de *software.* (Sommerville, 2001).

2.1.3 Caracterização de um software de qualidade

O modelo de qualidade definido na norma ISO/IEC 9126-1, (Weber et al. 2001), divide o processo de avaliação da qualidade em um software em duas partes. A primeira parte da avaliação é composta por características internas e externas, e a segunda parte é composto pelo modelo de qualidade em uso.

Ainda, pode-se classificar em seis, as características internas e externas de um *software*, ou seja:

- Funcionalidade - o produto deve satisfazer a todas as necessidades implícitas e explícitas de seus requerimentos específicos. Possui as sub características de adequação, segurança de acesso e conformidade;
- Confiabilidade - o produto de *software* deve desempenhar todas as funções para as quais foi desenvolvido num limite de tempo estabelecido. É importante a sua tolerância a erros e sua capacidade de recuperação;
- Usabilidade - o produto deve ser de manuseio fácil e agradável, tentando evitar telas e interfaces que possam vir a confundir o usuário, e também deve ser o máximo auto-explicativo;

- Eficiência - o produto de *software* deve conduzir suas funções às quais foi desenvolvido da maneira mais rápida e confiável possível;
- Manutenibilidade - o produto deve apresentar uma boa manutenibilidade, suportando sem esforços, demasiadas atualizações e modificações no programa tendo alta estabilidade e testabilidade; e,
- Portabilidade - o produto de *software* deve ter a capacidade de poder migrar de plataformas. Um produto com portabilidade é aquele produzido para ocupar o menor número possível de funções de *hardware* sendo aceito em maior número possível de ambientes.

Estendendo um pouco mais esta classificação de Weber (2001), temos outras três características muito importantes vistas por Peters & Pedrycz (2001), são elas:

- Testabilidade - esforço necessário para garantir que o *software* desempenhe as funções a que se destina;
- Interoperabilidade - esforço necessário para acoplar sistemas de *software*; e,
- Reusabilidade - até que ponto os módulos de software podem ser usados em diferentes aplicações.

É correto então afirmar que, quanto mais destas características o produto armazenar, maior será a qualidade do produto de *software*, pois ele vai deter todos os fatores que fazem um *software* ser uma ferramenta útil e funcional.

A Figura 4 exemplifica algumas relações interessantes entre os atributos externos e internos de software. Por outro lado, do ponto de vista de *software* já em uso, se divide os atributos da qualidade em quatro características. Estas características de qualidade de uso de um *software* são:

- Efetividade - capacidade do produto de software possibilitar os usuários atingir as metas requeridas;
- Produtividade - capacidade do produto de software em possibilitar aos usuários utilizarem uma quantidade adequada de recursos do sistema para alcançar suas metas;
- Segurança - capacidade do produto de software em oferecer níveis razoáveis e aceitáveis de riscos; e,
- Satisfação - capacidade do produto de software satisfazer as necessidades dos usuários.

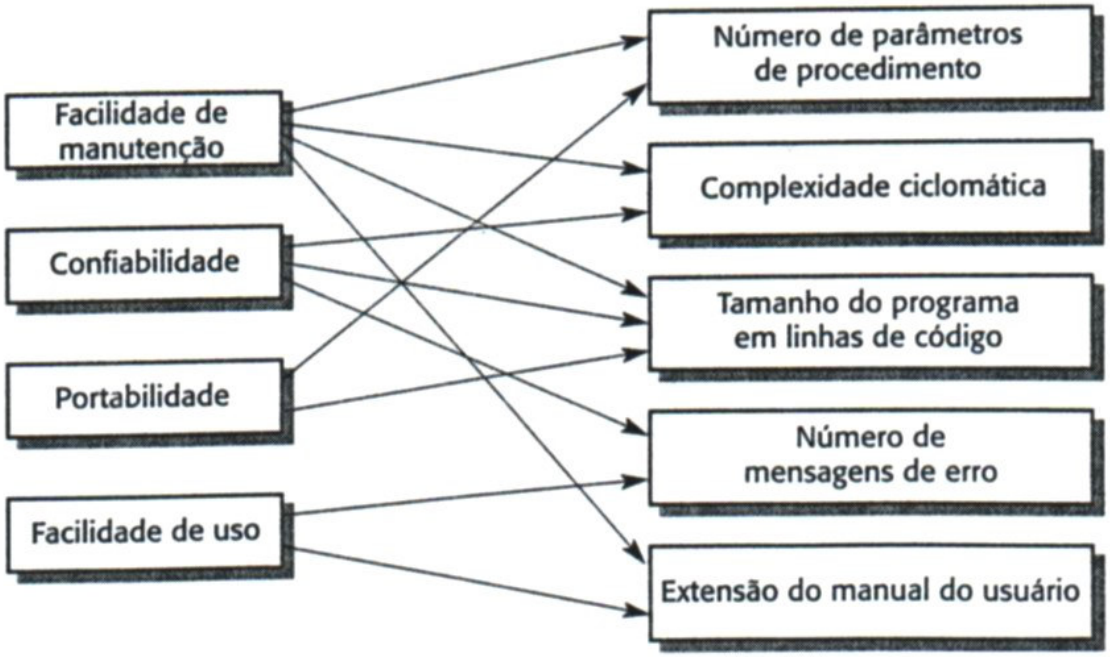

Figura 4: Relações entre os atributos internos e externos de *software*. (Sommerville, 2001).

No entanto, armazenar estas características no produto torna-se um grande problema, pois é muito difícil fazer com que durante o desenvolvimento do produto de *software*, sejam implementados meios para que este consiga finalizar todo o processo com qualidade.

2.1.4 Vantagens de se adotar a qualidade como política de trabalho

Dissertar quanto às vantagens que a adoção de uma política de qualidade tem atualmente é difícil devido ao tamanho da sua importância. A garantia de qualidade de *software* deve ser tratada estrategicamente como planejamento de risco. A qualidade reflete em custos e por isto deve ter uma abordagem de planejamento de risco na organização.

Lewis (2000) dita alguns exemplos de uma pobre qualidade em produtos de *software*:

- O produto entregue freqüentemente possui falhas;

- Conseqüências em níveis não aceitáveis por motivo de falha no sistema;
- O *software* entregue não supre as necessidades do cliente;
- Modificações no sistema são muito caras; e,
- Os custos de detecção e remoção de erros são excessivos e conseqüentemente caros.

Acredita-se que, no cenário atual, é impossível que uma organização que trabalha com desenvolvimento de *softwares* consiga sobreviver sem adotar uma política de qualidade. Obter uma política de qualidade é algo que é pesquisado há anos na área da engenharia de *software*, e até os dias de hoje é algo muito difícil, pois é um assunto muito complexo. Por isto, foram desenvolvidos modelos, normas e projetos com intuito de garantia a qualidade de *software*, estes modelos e normas serão tratados a seguir.

2.1.5 Modelos e normas de qualidade

A busca de qualidade deve ser o objetivo de qualquer empresa. Por isso muitas delas seguem à risca normas e modelos pré-estabelecidos para alcançar produtos de qualidade.

Estes modelos e normas são o que existe de mais moderno e eficaz em termos de qualidade de *software*, por isto vê-se a necessidade de conhecer estes modelos e normas e usá-los.

Na informática, existem vários modelos e normas que tendem a levar a produtos de qualidade, podendo-se dividir as normas mais reconhecidas pelo mercado em áreas de atuação:

- Quanto à qualidade da organização - onde se tem o CMM e BOOSTRAP – European System and *Software* Iniative – *ESSI* Programme;
- Quanto à qualidade do processo - onde se tem a ISO / IEC 9000, SPICE – ISO / IEC 15504 – *Software Process Improvement in Capability Determination* e ISO / IEC 12207 – Processos de Ciclo de Vida de *Software*; e,
- Quanto à qualidade do produto - onde se tem a ISO / IEC 9126 - Qualidade de Produtos de *Software*, ISO / IEC 9241 – Requisitos ergonômicos para o trabalho de escritório com terminais de computadores.

Tais normas, dentre outras não mencionadas, realizam seu papel apresentando moldes de procedimentos que podem levar a uma qualidade de *software*, porém não confundindo apenas o produto de *software* final, pois este é decorrente de uma estrutura bem moldada e organizada de todos os processos que decorrem do *software*. Esses moldes de procedimentos exemplificam o funcionamento e os objetivos (*software* de qualidade) das normas de desenvolvimento de *software*, tais como:

- *Software Process Improvement and Capability Determination* (SPICE): O projeto SPICE é um projeto que, sob uma estruturação fundamentada, visa à qualidade de processos de *software.*
- *Capability Maturity Model* (CMM): O CMM é um modelo usado para qualidade de organizações empresariais e sua capacitação.
- NBR ISO / IEC 12207 - é uma norma que globaliza processos de ciclo de vida de *software*. E sobre esta norma afirma-se que...

> "... Ela tem como objetivo auxiliar os envolvidos na produção de *software* a definir seus papéis, por meio de processos bem definidos, e assim proporcionar as organizações que a utilizam um melhor entendimento das atividades a serem executadas nas operações que envolvem, de alguma forma, o *software*..." Rocha (2001).

Além dessas normas, existem ainda outros modelos, normas e projetos, que ajudam a levar um *software* a ter qualidade. Contudo, as que foram citadas são as mais comumente utilizadas e as mais reconhecidas pelo mercado internacional.

A seguir serão dissertadas de forma mais detalhada algumas das normas mais importantes e reconhecidas, a SPICE e o CMM:

A) SPICE

A SPICE (*Software Process Improvement and Capability Determination*) é uma norma de iniciativa internacional que tem como objetivos criar um padrão internacional para a avaliação do processo de *software*.

Segundo consta em SPICE (2003), esta norma tem três objetivos principais, são eles:

- Desenvolver um esboço para a padronização do processo de avaliação de *software*.
- Conduzir experimentações na industria de *software* emergente.
- Promover uma troca mundial de tecnologia de processo de *software* entre as industrias.

Esta norma teve a postagem da sua primeira versão em Junho de 1995 e foi recentemente englobado pela *International Organization for Stantardization* (ISO) em que recebeu o nome de ISO 15504. Assim, a SPICE, ou ISO 15504 visa a orientação de empresas para uma melhoria contínua do processo de *software* com qualidade. Englobando todos os aspectos necessários para se alcançar a excelência em *software*, esta norma está sendo continuamente elaborada e melhorada num esforço conjunto de centros técnicos espalhados pelo mundo. No Brasil o órgão colaborador desta norma é a ABNT (Associação Brasileira de Normas Técnicas), (Rezende, 1999).

Esta norma funciona com base em um modelo de referência, o qual é usado como base para o processo de avaliação. Neste modelo de referência se encontram as emendas que visam a padronização dos processos envolvidos no desenvolvimento de um *software*.

O SPICE faz uma divisão das categorias de processos mais importantes, são cinco as categorias abordadas pelo modelo, e podem ser mais bem vistas na Tabela 1 adaptada de Rezende (1999).

Tabela 1: Divisão das categorias na norma SPICE. (Rezende, 1999).

Categoria	Descrição
CUS – Cliente Fornecedor	Processos que impactam diretamente os produtos e serviços de *software* no fornecedor para o cliente.
ENG – Engenharia	Processos que especificam, implementam, ou mantêm um sistema ou produto de *software* e sua documentação.
SUP – Suporte	Processos que podem ser empregados por qualquer um dos outros processos.
MAN – Gerência	Processos que contem praticas de natureza genérica que podem ser usadas por quem gerencia projetos ou processos dentro de um ciclo de vida de *software*.
ORG – Empresa	Processos que estabelecem os objetivos de negócios da empresa.

Além desta divisão das categorias de processo, o SPICE também define e divide em seis níveis de capacitação. Pode-se ver na Tabela 2 que a coluna de nível tem valores crescentes que representam o grau do controle de processos e conseqüentemente um ganho de qualidade no produto de *software* com o aumento do nível.

Em nível de capacitação se identifica o grau de competência que uma organização ou industria trata dos processos de *software*.

Em seguida será visto a norma CMM, a qual é uma das mais reconhecidas em âmbito internacional.

Tabela 2: Dimensões do processo de *software* para o modelo SPICE. (Zorzo, 2001).

Nível	Nome	Descrição
0	**Incompleto**	O caos impera. É o ponto de partida.
1	**Realizado**	Envolve práticas básicas, depende de conhecimentos e esforços individuais.
2	**Gerenciado**	O desempenho do processo é planejado e gerenciado.
3	**Estabelecido**	O processo é planejado e gerenciado usando um processo padrão.
4	**Previsível**	O processo definido é quantitativamente entendido e controlado.
5	**Otimizado**	O processo definido e o processo padrão suportam refinamentos e melhorias.

B) CMM

O CMM foi desenvolvido pelo SEI (*Software Enginnering Institute*), ligado à Universidade de *Carnegie Mellon*, e seus custos de desenvolvimento foram supridos pelo Departamento de Defesa Norte-Americano. No início, seu objetivo era desenvolver um padrão de qualidade de *software*, para o uso da instituição que o financiou. Mais tarde passou a ser utilizado para efetuar a qualidade em *software* em qualquer aplicação.

Segundo Paulk apud Fiorini et al. (1998, p9), tem-se que:

> O CMM é uma estrutura (*framework*) que descreve os principais elementos de um processo de *software* efetivo. O CMM descreve os estágios através dos quais organizações de *software* evoluem quando elas definem, implementam, medem, controlam e melhoram seus processos de *software*. O CMM fornece uma diretriz para a seleção de estratégias de melhoria de processos, permitindo a determinação da capacitação dos processos correntes e a conseqüente identificação das questões mais criticas para a melhoria de processo e qualidade de *software*. Desta forma provê e descreve

> um caminho de melhoria evolutiva a partir de um processo *ad-hoc* para um processo maduro e altamente disciplinado. Este caminho de melhoria é definido por cinco níveis de maturidade: Inicial, Repetitivo, Definido, Gerenciado e Em Otimização.

Observa-se que o CMM tem o seu enfoque voltado para a parte organizacional da empresas que fabrica um *software*. Desse modo, ele divide em cinco, as fases de uma empresa quanto a sua organização e estruturação. Assim, segundo o CMM, as fases de evolução de uma empresa de *software* têm as seguintes definições. (Pressman, 2002):

- **Nível 1 Inicial -** Empresa desorganizada, processo caótico em que poucos processos são definidos, todo o desenvolvimento depende de esforços extraordinários de algumas poucas pessoas, sem nenhum tipo de gerenciamento. A Figura 5 mostra como é o processo neste nível.
- **Nível 2 Repetitivo -** Conforme se pode observar na Figura 6, a empresa começa a se desenvolver, usando a repetição como base de qualidade de produtos, o gerenciamento de projeto já é estabelecido desde o princípio e é usado como base de programações em bibliotecas próprias para reaproveitamento.
- **Nível 3 Definido -** Documentação dos processos e atividades do desenvolvimento do *software*, usando um plano de qualidade de produto para manter a qualidade dos mesmos. A Figura 7 caracteriza este processo.
- **Nível 4 Gerenciamento -** Gerenciamento quantitativo do processo de desenvolvimento do *software*, que recebe um gerenciamento específico de qualidade, como pode-se observar na Figura 8.
- **Nível 5 Otimização -** Busca contínua para prevenção de problemas e busca de novas estratégicas de gerenciamento e de tecnologias. A Figura 9 exemplifica o funcionamento dos processos deste nível.

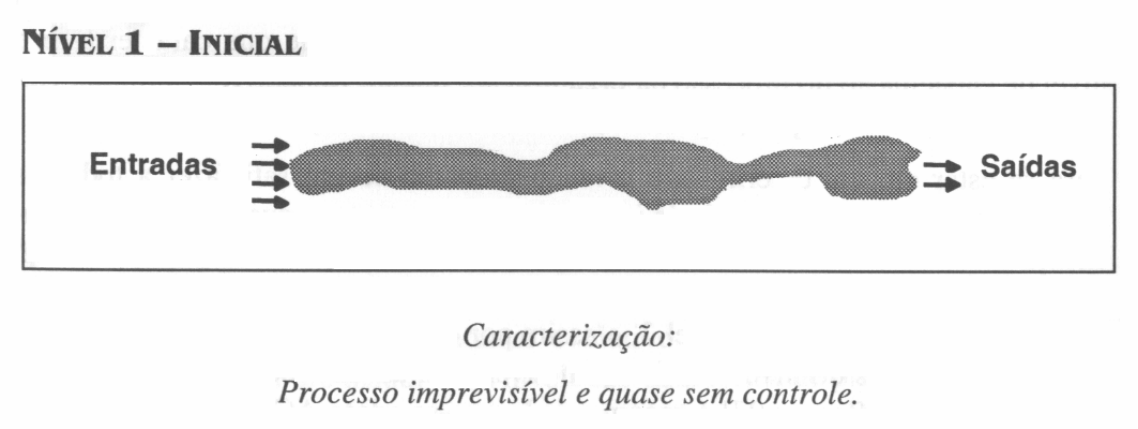

Figura 5: Nível 1 do CMM. (Fiorini et al. 1998).

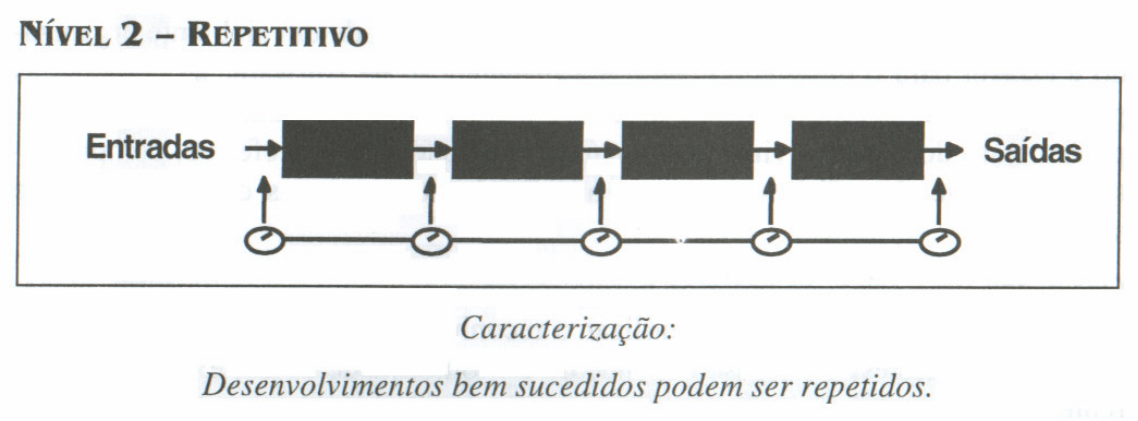

Figura 6: Nível 2 do CMM. (Fiorini et al. 1998).

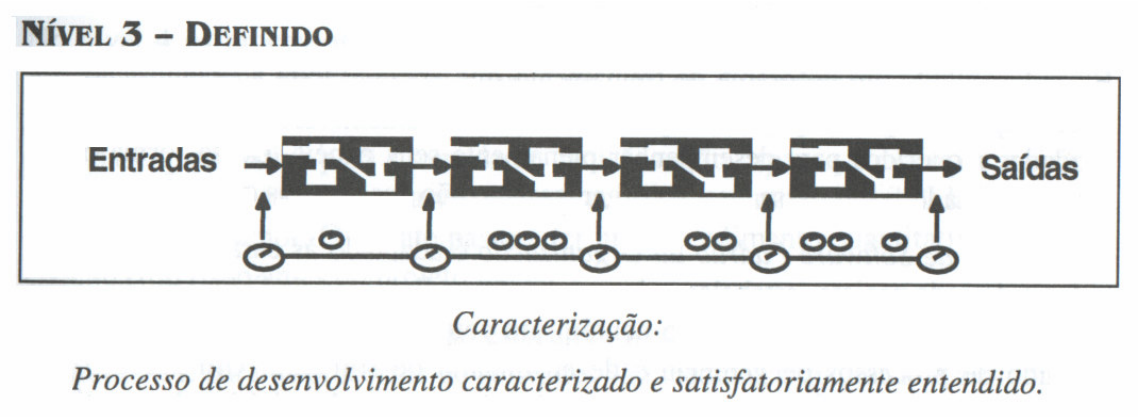

Figura 7: Nível 3 do CMM. (Fiorini et al. 1998).

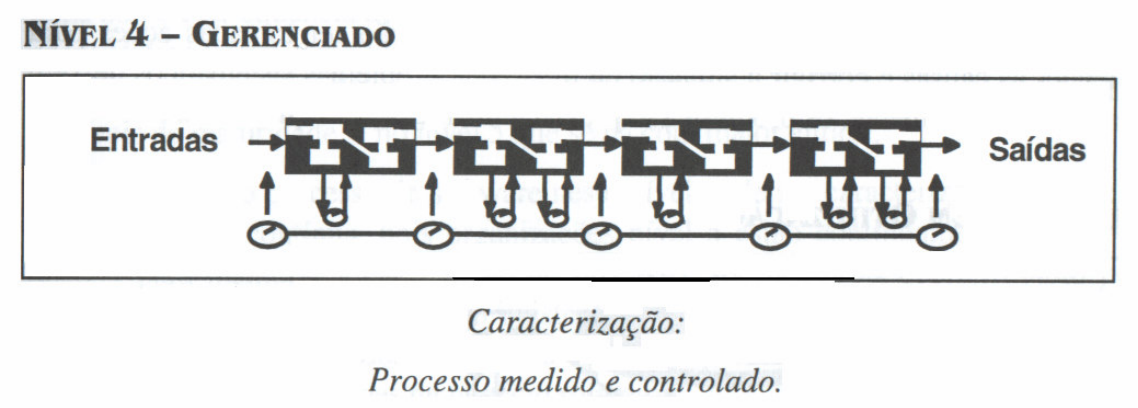

Figura 8: Nível 4 do CMM. (Fiorini et al. 1998).

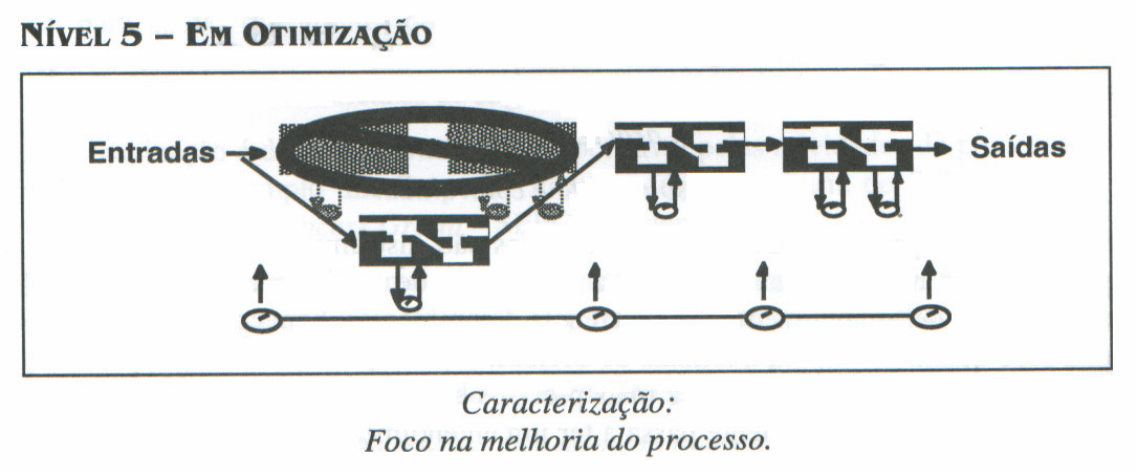

Figura 9: Nível 5 do CMM. (Fiorini et al. 1998).

Logicamente, quão mais alto é o nível de estrutura da empresa, usando-se a divisão do CMM, mais qualidade haverá nos processos e, decorrentemente, de *software* (Peters & Pedrycz, 2001). Porém, a tarefa de atingir um nível ou subir de nível é considerada muito difícil (Fiorini et al. 1998).

Em estudos sobre CMM, Fiorini et al. (1998), descrevem que, considerando-se todos os níveis que estruturam uma empresa quanto a sua organização, o mais difícil é conseguir sair do nível inicial, o de desorganização, para o segundo nível, o de operações repetitivas. Uma completa modificação na organização e na estrutura da empresa é necessária para que tal mudança seja possível, exercendo fortes impactos sobre os membros da empresa, o que infelizmente é algo que nem todas as empresas conseguem suportar.

No, entretanto, a dificuldade em subir de nível não se encontra apenas na transição do primeiro para o segundo nível. Todas as modificações que uma empresa pode sofrer causam um choque muito grande, pois sempre há o princípio da mudança e, como é próprio do o ser humano, o que é novo e diferente é sempre uma tarefa árdua. (Fiorini et al. 1998).

Para atingir um nível do CMM é necessário cumprir uma série de requisitos ou atividades, que são chamadas de ACP's (Áreas-Chaves de Processo), segundo Sommerville (2001). Cada ACP é subdividida em características comuns:

- Compromisso;
- Habilitação;
- Atividade;
- Medição e Análise; e,
- Verificação.

Na figura 10 podemos ver de forma clara as ACP's dos cinco níveis de maturidade e suas respectivas atividades que devem ser cumpridas em cada nível.

Uma organização, para conseguir passar de um nível para o seguinte deve cumprir todas as ACP's do determinado nível e ainda todas as dos níveis anteriores Pressman (2002). As ACP's e seus respectivos níveis são os demonstrados na Figura 10.

2.1.6 Situação atual dos modelos de qualidade

Atualmente os modelos e normas de qualidade de *software* são o que à de mais moderno na questão de excelência em *software*.

Pode-se ver claramente que várias empresas não só do ramo do desenvolvimento de *softwares*, tem visto a importância da adoção de sistemas de qualidade total, (Weber et al. 2001), como se pode verificar na Figura 12.

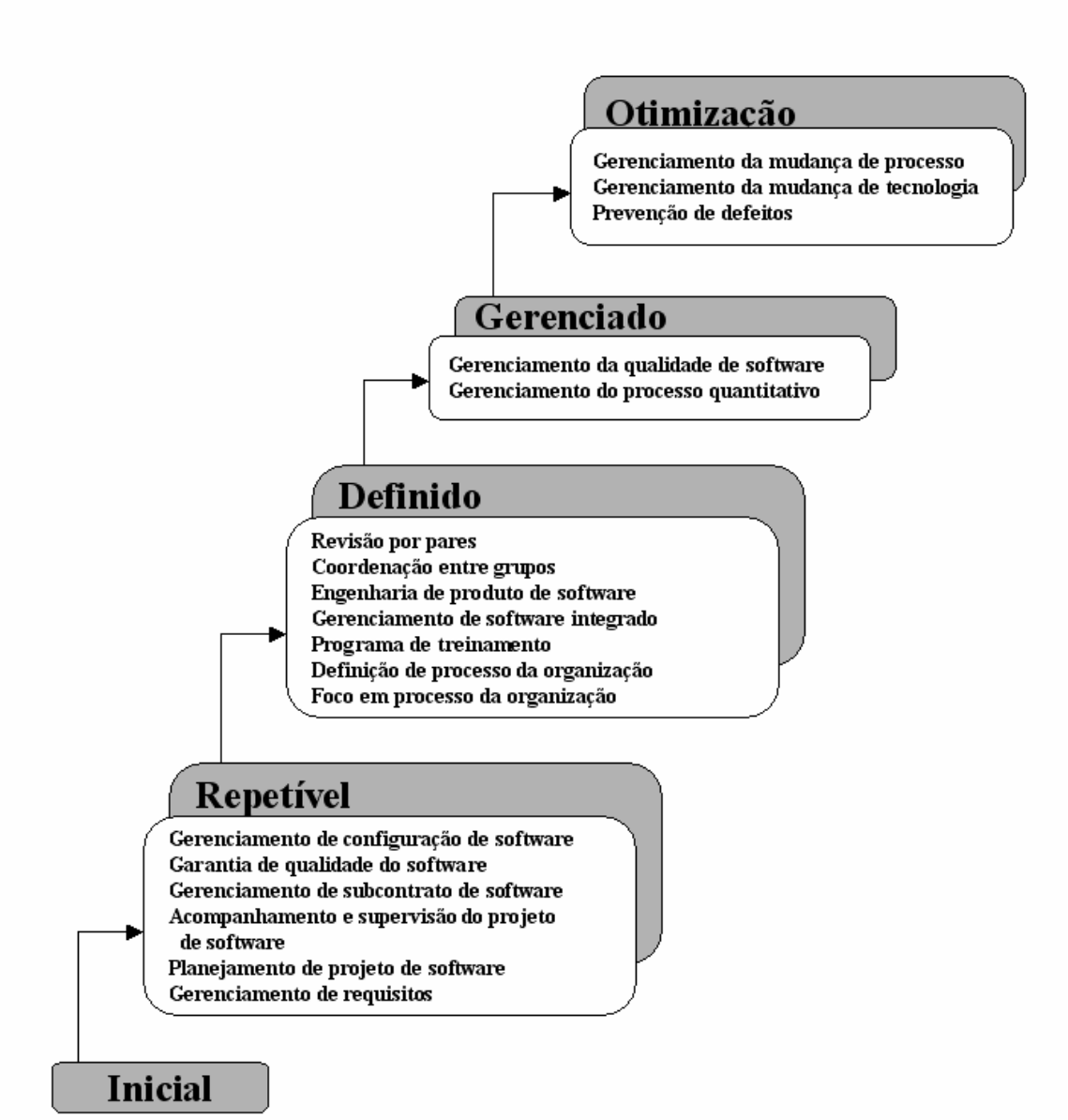

Figura 10: Níveis de maturidade do CMM e suas ACP's. (Sommerville, 2001).

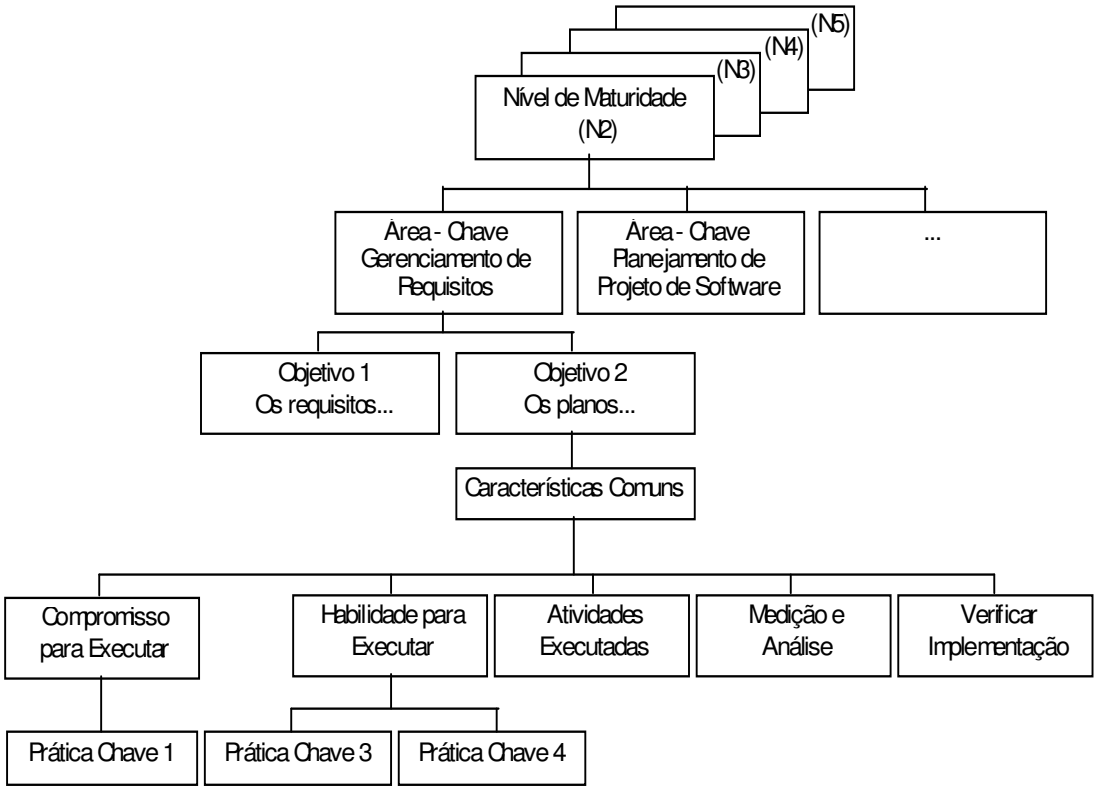

Figura 11: Figura Ilustrativa das ACP's e suas características comuns do CMM. (Fiorini et al. 1998).

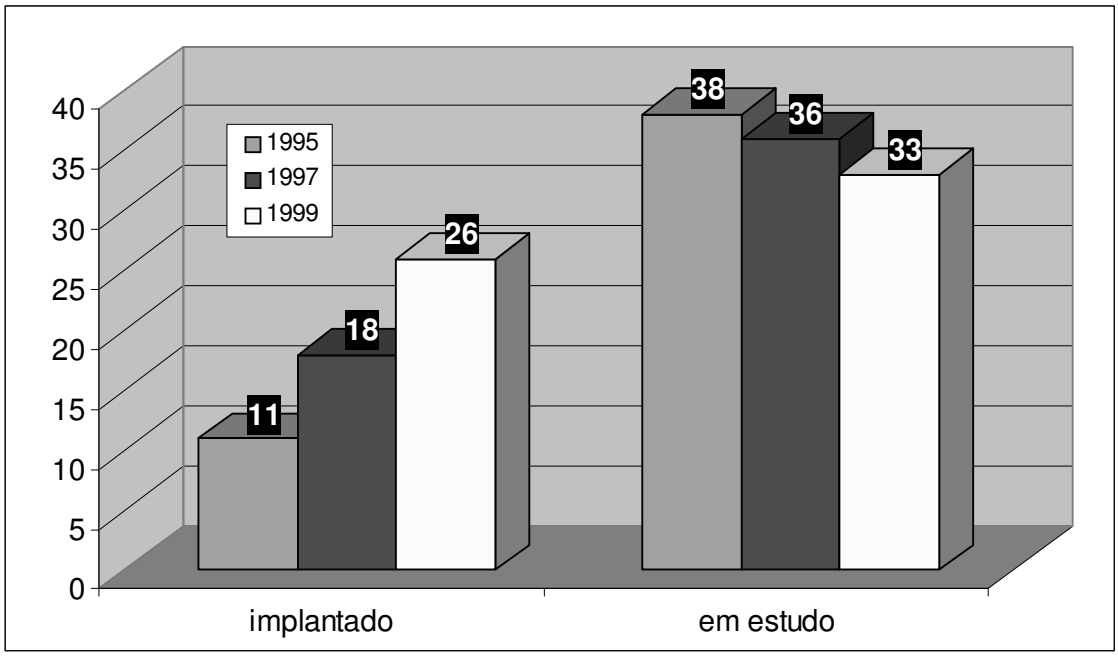

Figura 12: Uso de sistemas de qualidade. (Weber et al. 2001).

A qualidade dos processos de *software* nas empresas Brasileiras tem sido acompanhadas sob as normas: modelo CMM, projeto SPICE (ISO 15504), certificação ISO 9000 (Norma da ISO para processos empresariais), e, a norma internacional ISO / IEC 12207 (Norma ISO: *Information Technology – Software Live Cycle Processes*).

É crescente o uso de sistemas de qualidade total no ramo industrial brasileiro, e como o enfoque deste trabalho é sobre qualidade de *software* pode-se ver nas Figuras 13 e 14 o crescimento do conhecimento e uso das normas CMM e SPICE no âmbito nacional. Estes são dados retirados em pesquisas feitas no Brasil, e se pode perceber que a partir destes dados a implantação de normas de qualidade vem crescendo, mesmo que devagar. Ainda em nível nacional, o Brasil é um dos paises que se mostram com significativas aparições do uso do modelo CMM.

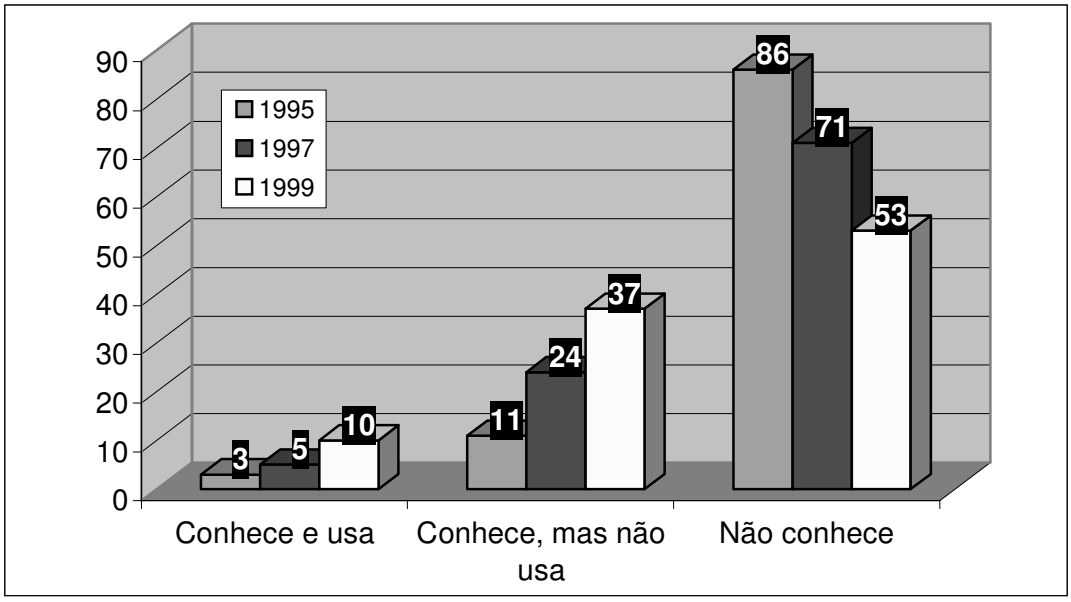

Figura 13: Gráfico do conhecimento do CMM no Brasil por Weber (2001).

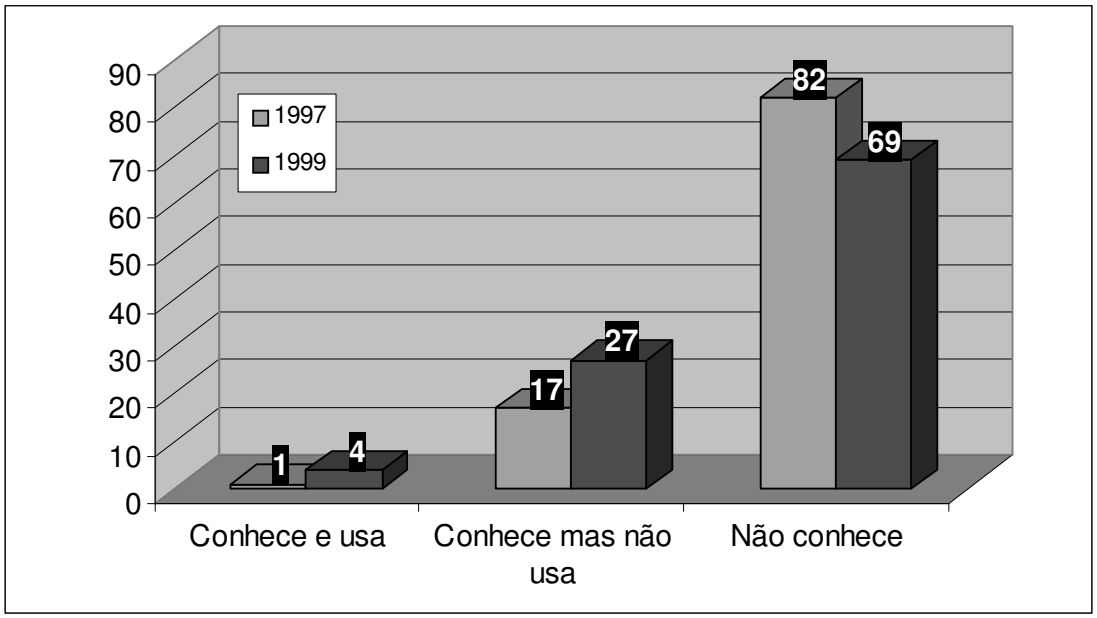

Figura 14: Gráfico do conhecimento do SPICE no Brasil por Weber (2001).

Dados retirados do SEI (*Software Engineering Institute*), aonde o CMM foi desenvolvido, mostram que no Brasil oficialmente existem 17 aparições do CMM em organizações de desenvolvimento de *software*, aonde existem organizações que se encontram até no nível três de maturidade. (CAPABILITY MATURITY MODEL. CMM, 2003)

Reconhecidos pelo CMM só existem sete países (Canadá, China, Índia, Japão, Coréia, Singapura e Estados Unidos) que possuem empresas com o ultimo nível (nível 5) de maturidade.

Na figura 15, nota-se como o uso do CMM está crescendo a nível mundial nos últimos anos.

Analisando todos estes dados pode-se ver que a situação atual do uso de modelos de qualidade tem uma crescente aceitação em nível nacional e mundial.

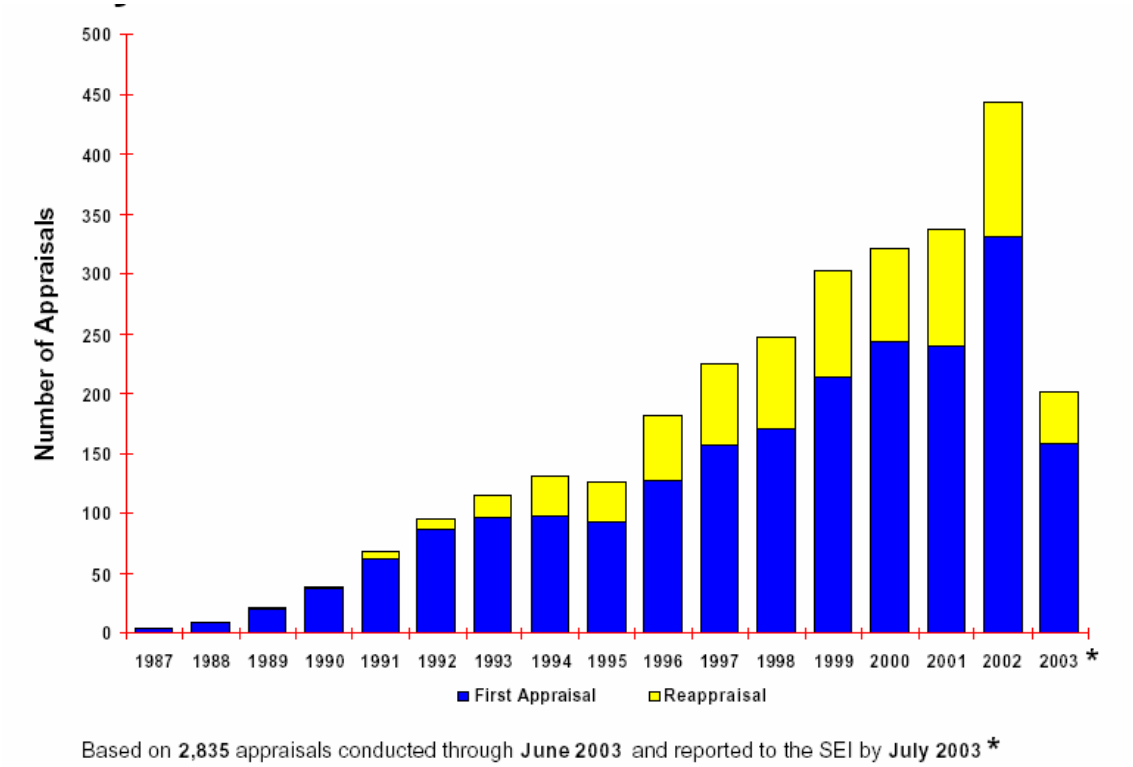

Figura 15: Crescimento mundial das aparições do uso do modelo CMM. (CAPABILITY MATURITY MODEL. CMM, 2003).

2.2 Sistemas de informação geográfica

Softwares de geoprocessamento que também recebem o nome de Sistemas de Informações Geográficas (SIG) ou Sistemas de informação Georrefereciadas são sistemas que realizam um tratamento computacional de dados geográficos.

Sistemas de Informação Geográfica podem ter várias definições, dentre elas temos o seguinte:

> O termo Sistemas de Informação Geográfica (SIG) é aplicado para sistemas que realizam o tratamento computacional de dados geográficos e recuperam informações não apenas com base em suas características alfanuméricas, mas também através de sua localização espacial; oferecem ao administrador (urbanista, planejador, engenheiro) uma visão inédita de seu ambiente de trabalho, em que todas as informações disponíveis sobre um determinado assunto estão ao seu alcance, inter-relacionadas com base no que lhes é fundamentalmente comum – a localização geográfica. Para que isto seja possível, a geometria e os atributos dos dados num SIG devem estar

> *georreferenciados*, isto é, localizados na superfície terrestre e representados numa projeção cartográfica (CÂMARA et al. 2004a).

2.2.1 Descrição

A definição de um Sistema de Informação Geográfica, também conhecido pela sigla SIG, é de uma ferramenta que captura, gerencia e processa dados que possuem uma referencia no mundo real por meio de uma localização geográfica. (LISBOA FILHO, 1995)

Ainda temos a definição de Lisboa Filho (2001) "O termo Sistema de Informação Geográfica (SIG) caracteriza os sistemas de informação cuja principal característica é possibilitar a realização de análises espaciais envolvendo dados referenciados geograficamente".

Além de dados referenciados geograficamente, um SIG ainda pode trabalhar com mapas cartográficos, nisso temos:

> Qual a característica fundamental de um Sistema de Geoprocessamento? A faculdade de armazenar, recuperar e analisar *mapas* num ambiente computacional. Um mapa é uma representação gráfica dos fenômenos espaciais. Num ambiente computacional, a noção de mapa deve ser estendida para incluir diferentes tipos de dados como *imagens de satélite e modelos numéricos de terreno* (CÂMARA, 1995).

Moura (2003) explica que só por um sistema baseado em computadores, lidar com informações geográficas, não quer dizer que lidam com dados desta natureza. Sistemas de Informação Geográficas, juntamente com dados geográficos processam dados comuns, mas o fato de poderem trabalhar com dados geográficos é o que os classificam como desta natureza.

Estes dados são utilizados há muito tempo por profissionais de diversas áreas, tais como geólogos, ecologistas e outros cientistas. Mas, a importância da utilização destes dados não está só na necessidade de se preservar estes profissionais e suas ciências, diz BOURROUGH (1998), pois estes sistemas resultam informações quem podem ser utilizados para os mais diversos fins e áreas, como planejamento urbano, construção de vias publicas, rastreamento de epidemias e outras.

Em um exemplo desta realidade temos:

> Os dados que produzem as informações num SIG são os mais variados e dependem do propósito a que se destinam. Por exemplo, um SIG destinado à educação interessa-se não somente pela idade das pessoas, infra-estrutura, sistemas de transporte, segurança, qualificação dos professores, renda das pessoas, dentre outros, mas também pela localização (posição) de alguns, ou todos os dados envolvidos (MONICO, 2000, p.227).

Em cima deste caso entendemos que se pode unir um dado que contém uma posição ou localização no globo terrestre com outros tipos de dados.

Um SIG é considerado uma ferramenta ou *software* (CHRISMAN, 1997). Por isso não pode existir fora de um ambiente digital e por ser um sistema baseado em computadores, ele trata de dados e informações, tem procedimentos e funções como qualquer outro software, com a característica de trabalhar com dados espaciais que possam ser representados geograficamente.

Definir o que é um Sistema de Informação Geográfica pode ser muito difícil, pois segundo Silva (2003), existem várias razões que criam esta dificuldade, ou seja:

- Por ser uma tecnologia muito recente e, nos últimos 30 anos, aconteceu um crescimento muito rápido da teoria da organização, um crescimento tanto teórico, quanto tecnológico e organizacional;
- A orientação comercial da utilização dos SIGs gerou figuras de linguajem que engrandecem ou diminuem muito a verdade dos fatos;
- O valor de compra de computadores cada vez menor, possibilitando a aquisição de maquinas cada vez melhores e que assim suportam cada vez mais SIGs e figuras retórica e neologismos;
- O número de usuários deste tipo de sistema é cada vez maior, e isto causou um aparecimento de informações conflitantes sobre o que realmente significa um SIG;
- O amplo uso de SIGs em diversas áreas criou nestas áreas conceitos diferentes da definição de um SIG; e,
- Discordância acadêmica da definição de SIG, envolvendo o enfoque principal destes *softwares*.

De maneira mais ampla este tipo de sistema pode ser aplicado nas mais diversas áreas, como agricultura, geografia, urbanismo e até medicina. Segundo Assad & Sano (1998) existem três maiores usos de um SIG, e são eles:

- Como meio para produção de mapas;
- Como suporte para analise espacial de fenômenos; e,
- Como um Banco de Dados Geográfico, com funções de armazenamento e recuperação das informações espacializadas contidas no mesmo.

Os mesmos autores ainda explicam que estas definições mostram a grande amplitude de usos deste tipo de sistema, em cima disto se pode indicar duas principais características de um SIG: a primeira é de em uma única base de dados, armazenar as informações obtidas de dados cartográficos, dados de censo de população, de cadastro urbano e rural, e ainda imagens de satélites, redes e modelos numéricos de terreno. A segunda característica é a de oferecer meios para combinar estas informações, processa-las, consultá-las, recupera-las e finalmente visualiza-las. O uso dos Sistemas de Informação Geográfica se dá pelo fato dos SIG's procurar simular a realidade do espaço geográfico (Câmara & Freitas, 1995). E os mesmos ainda ditam duas grandes e principais, características deste tipo de sistema:

- Integrar informações espaciais provenientes de dados cartográficos, dados de censo e cadastro urbano e rural, imagens de satélite, redes e modelos numéricos de terreno; e,
- Oferecer mecanismos para analise geográfica, através de facilidades para consultar, recuperar, manipular, visualizar e plotar o conteúdo de base de dados geocodificadas.

Comumente se atribui o uso de SIG para fins geográficos, mas a utilização deste tipo de sistema não se restringe só a este uso. Sistemas de Informações Geográfica são utilizados nos mais diversos fins e áreas segundo Bourrough & McDonnel (1998). Para maior compreensão da amplitude de uso, dados e aplicações de SIG's a Tabela 3 cita os maiores usos de SIG nas mais diversas áreas.

Tabela 3: Current Applications for SIG, (BOURROUGH & MCDONNEL, 1998).

Área	Uso/Exemplo
Agricultura	Agricultura de precisão.
Arqueologia	Uso de mapas temáticos para descrição da área.
Meio Ambiente	Modelar, monitorar e gerenciar a degradação do solo e o planejamento rural; qualidade da água, pragas, qualidade do ar, tempo e previsão do tempo.
Epidemiologia e Saúde	Local de epidemias e sua relação com fatores locais.
Florestamento	Gerenciar, planejar e otimizar a extração de árvores e reflorestamento
Serviços Emergenciais	Otimizar o serviço e as rotas utilizadas pelo corpo de bombeiros, policia e ambulâncias.
Navegação	Ajudar na navegação pelo ar, água e terra.
Marketing	Otimizar a logística da entrega em grupos e relacionar áreas de consumidores com interesses em comum.
Administração Pública	Gerenciar aspectos legais sobre cadastro de propriedades, seus valores e impostos.
Planejamento Urbano	Desenvolvimento de plantas de expansão urbana, seus custos, manutenção e gerenciamento.
Estudos Sociais	Estudo do desenvolvimento demográfico de uma região.
Turismo	Gerenciamento de locais turísticos e suas atrações.
Prefeituras	Localização, gerenciamento e planejamento do abastecimento de água, eletricidade, telefone, etc…

2.2.2 Estrutura de um SIG

Um SIG, segundo Druck et al. (2004), utilizando uma visão mais abrangente de seu uso, possui os seguintes componentes:

- interface com usuário;
- entrada e integração de dados;
- funções de processamento gráfico e de imagens;
- visualização e plotagem; e,
- armazenamento e recuperação de dados (organizados em um Banco de Dados Geográfico).

Estes componentes que caracterizam um SIG se relacionam de acordo com a Figura 16, demonstrada abaixo.

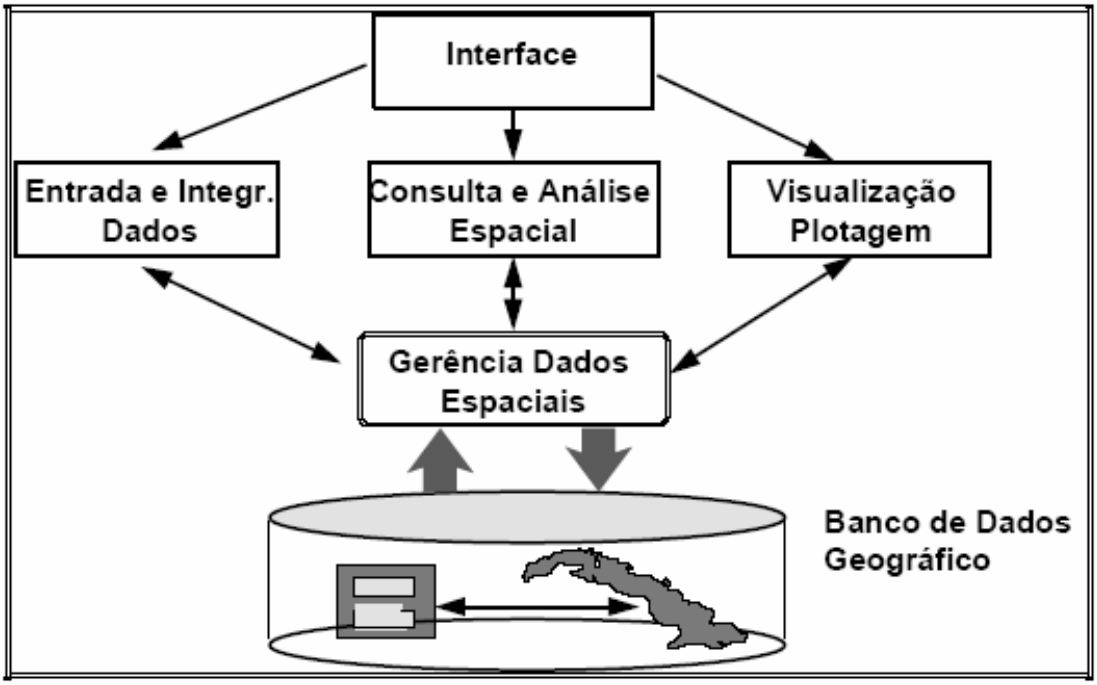

Figura 16: Arquitetura de Sistemas de Informação Geográfica. (CÂMARA et al. 2004b).

Todos estes componentes se relacionam hierarquicamente, e segundo Câmara et al. (2004a) temos a seguinte classificação dos níveis de um SIG :

- nível de usuário - nível mais próximo ao usuário do SIG, o qual é operado e controlado por meio de uma interface homem-máquina;
- nível intermediário - é o nível onde acontece o processamento dos dados espaciais; e,
- nível interno: é o nível onde se armazenam os dados espaciais e seus atributos em um banco de dados.

Os níveis de arquitetura de um SIG pode ser melhor visualizado na Figura 17.

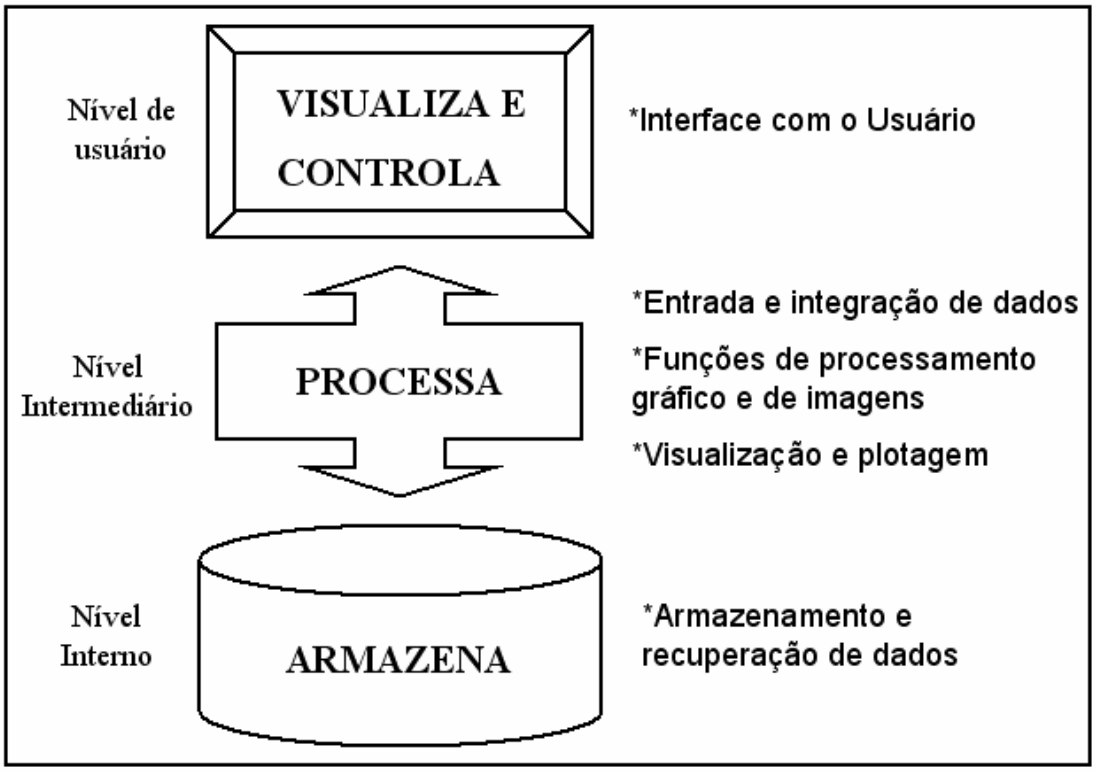

Figura 17: Níveis da arquitetura de um SIG.

Sobre a visualização de dados em um SIG, tem-se:

> SIG utilizam diversos recursos de apresentação dos dados, como em uma carta, mapa, planeta ou esquema. Neste tipo de apresentação (em papel) ou na tela do computador, técnicas de cartografia desenvolvidas há centenas de anos são empregadas, de modo a preservar a familiaridade do usuário com os tipos convencionais de visualização de fenômenos naturais ou desenvolvidos pelo homem. Assim, a grande maioria dos SIG comerciais oferece recursos de visualização que se aproximam daqueles utilizados tradicionalmente em cartografia convencional. (Davis Jr. 1999)

Assim exemplificando a possibilidade e capacidade de implementação de técnicas tradicionais, que podem ser classificadas analogicamente, para meios digitais, demonstrando o poder de utilização de sistemas informatizados para fins não só cartográficos, mas para várias outras finalidades.

Mas as técnicas e dados utilizados em SIG não podem se deter à mera implementação de técnicas e informações ditas “analógicas”, pois um SIG trabalha com um grande fluxo de dados de tamanha complexidade que uma base de dados

de um SIG não poderia simplesmente guardar informações cartográficas, pois assim se perderia muita informação que um SIG pode produzir (Davis Jr & Laender, 1999).

Existem diversos produtos de SIG a disposição, aonde cada um possui uma implementação única com um propósito especifico, como o de trabalhar com analise ambiental, planejamento urbano, agricultura de precisão, e assim por diante. Segundo Davis Jr (1999), estas diferenças nas implementações são causadas pelas percepções distintas da realidade dos desenvolvedores, causando disparidade na representação de dados espaciais utilizados em um SIG.

2.2.3 Dados espaciais

Como descrito anteriormente, um SIG é um *software* caracterizado pelo processamento de dados geográficos. Também chamados de dados georreferenciados ou mais comumente conhecidos como dados espaciais.

Estes dados se encontram em formato digital e possuem uma informação geográfica, ou seja, possuem referencia no mundo real em um sistema de coordenadas geográficas.

Um dado deve possuir atributos de localização para ser considerado um dado espacial, pois assim permite a localização desta informação, estes atributos são: longitude, latitude e altitude. Com estas informações pode-se referenciar um dado a qualquer posição na superfície da terra.

E existem basicamente duas formas de se representar dados espaciais em um SIG: Vetorial ou *Vector* e Matricial ou *Raster*, (COURSEWARE, 2000)

Um dado vetorial tem a seguinte definição:

> Os mapas são abstrações gráficas nas quais linhas, sombras e símbolos são usados para representar as localizações de objetos do mundo real. Tecnicamente falando, os mapas são compostos de pontos, linhas e polígonos. Internamente, um SIG representa os pontos, linhas e áreas como conjunto de pares de coordenadas (X, Y) ou (Longitude, Latitude). Os pontos são representados por apenas um par. Linhas e áreas são representadas por seqüências de pares de coordenadas, sendo que nas áreas o ultimo par coincide exatamente com o primeiro.(ROCHA, 2000, p.55)

Dados matriciais são compostos por matrizes de células, aonde a cada célula pode ser atribuído um valor. Este valor é utilizado para definir uma cor para sua visualização. Com a totalidade da matriz contendo valores agregados em suas células chega-se a uma imagem digital que permite o reconhecimento de objetos (LAMPARELLI et al. 2001). Cada célula de um dado matricial, que recebe o nome de *pixel*, representa uma medição de alguma grandeza física, que corresponde a algum objeto no mundo real.

A figura 18 demonstra as características visuais de uma área original vista pelo método matricial e pelo método vetorial.

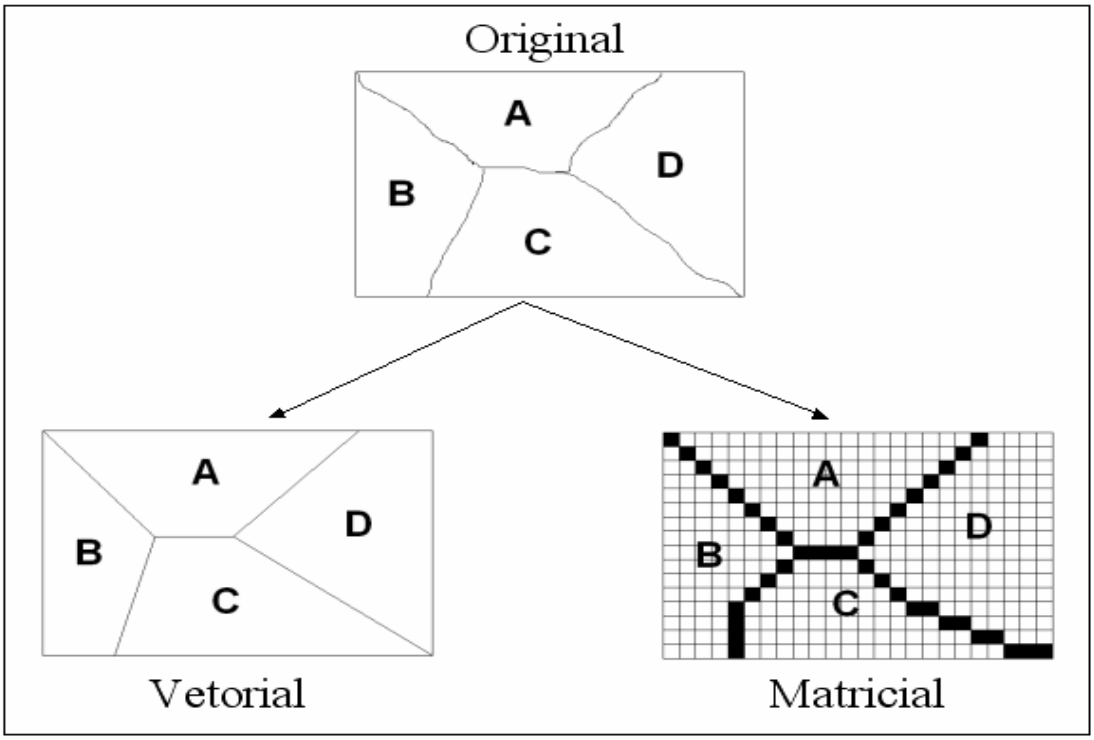

Figura 18: Diferenças na visualização de dados vetoriais e matriciais.

Pela figura 19, podemos ver que o tipo de dado vetorial pode guardar muito mais informações, e com mais detalhes se forem comparados com os dados do tipo matricial. Tendo em vista que existem situações onde dados matriciais são de melhor uso, como em sobreposição e processamento de imagens.

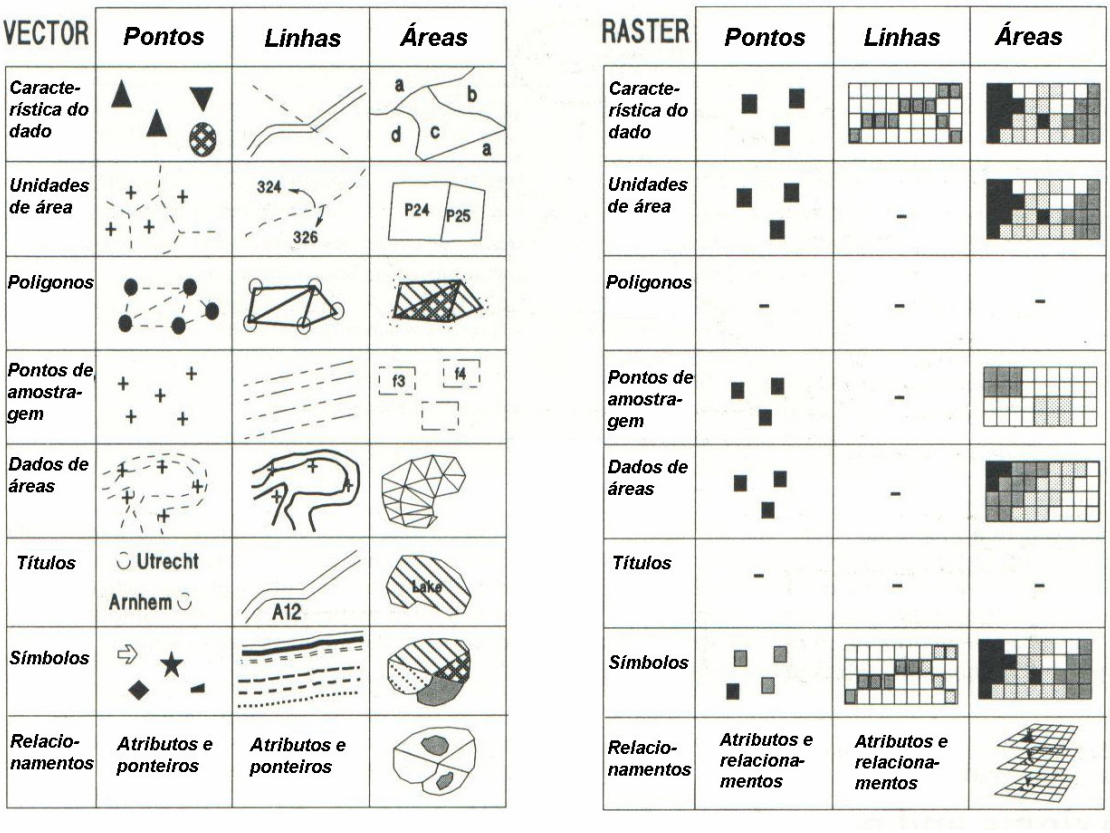

Figura 19: Figura ilustrativa da diferença entre dados espaciais do tipo Vetorial (à esquerda) e Raster (à direita). (BOURROUGH & MCDONNEL, 1998)

Cada um destes dois tipos de dados tem suas vantagens e desvantagens. Segundo Lamparelli et al. (2001), temos os dois quadros comparativos demonstrando algumas destas vantagens e desvantagens nos tipos de dados Vetorial, na Tabela 3, e Matricial, na Tabela 4.

Tabela 4: Quadro comparativo de vantagens e desvantagens de dados vetoriais.

Vantagens - Vetorial	**Desvantagens - Vetorial**
- Possui uma estrutura de dados compacta, ocupando menor espaço de memória. - É recomendado para mapas que necessitem de melhor qualidade visual. - Recuperação, atualização e generalização de gráficos e atributos são realizadas de forma eficiente.	- A estrutura de dados é complexa - As operações de superposição são difíceis de serem implementadas. - A tecnologia é cara, especialmente para *softwares* e *hardwares* mais sofisticados.

Tabela 5: Quadro comparativo de vantagens e desvantagens de dados matriciais.

Vantagens - Matricial	**Desvantagens – Matricial**
- A tecnologia é barata e tem sido bastante desenvolvida. - Operações de superposição são facilmente implementadas e de maneira eficaz. - As operações de modelagem e simulação são facilitadas porque cada unidade espacial tem a mesma forma e tamanho. - Permite realizar operações matemáticas com precisão. - A estrutura de dados é simples.	- A estrutura de dados toma muito espaço na memória. - As transformações de projeção consomem muito tempo a não ser que algoritmos especiais sejam usados. - O produto final pode não ser esteticamente agradável por causa do efeito “escada”, em função do *pixel*. - O uso de grandes células, ou *pixels*, para reduzir o volume de dados significa que estruturas fenomenologicamente reconhecíveis podem ser perdidas, resultando em sérias perdas de informação.

2.3 Desenvolvimento de SIG com qualidade

Desenvolver um software de qualidade é considerado uma tarefa muito árdua, e infelizmente não existe nenhuma forma de fazer com que este processo seja facilitado, segundo Brooks apud Fiorini et al. (1998) é uma tarefa impossível de ser simplificada.

Antigamente o desenvolvimento de um software era uma atividade muito trabalhosa, pois era, e ainda é, uma atividade muito difícil devido a sua complexidade. Esta dificuldade se residia pelo meio caótico em que se desenvolvia um software, sem nenhuma estruturação, ou controle do desenvolvimento. O programador agia por si próprio e simplesmente fazia o que achava ser necessário de acordo com o que ele achava que era preciso. Então se pode dizer que antigamente o desenvolvimento de um software dependia basicamente de uma, ou pouquíssimas pessoas que carregavam a responsabilidade de executar todo o desenvolvimento. Estes fatores, segundo Pressman (2001), resultavam em softwares de péssima qualidade, com altos níveis de custo e manutenção que era continua pelo sistema ser instável e com níveis de confiabilidade duvidosos.

Esta situação demonstrava que existia algo de errado, pois computadores são máquinas extremamente lógicas que teoricamente não cometem erros, logo o problema estava no software. Problema causado pelo modo que se desenvolvia um software, e então se notou a necessidade de criar tecnologias voltadas ao desenvolvimento de softwares.

Tais tecnologias desenvolvidas pela Engenharia de Software criaram técnicas que fazem com que o processo de desenvolvimento de um software seja mais organizado, estruturado e seguro. Muitas destas técnicas estão compiladas em modelos e normas de qualidade que unem várias técnicas com a intenção de alcançar um software com o máximo de qualidade possível. Neste trabalho não será adotado nenhum destes modelos em questão, pois se optou por uma abordagem mais ampla sobre qualidade de software para criar então um referencial para o desenvolvimento de SIG's.

Para que um *software* tenha qualidade é necessário que seu desenvolvimento seja bem feito e bem estruturado, pois é nesta fase que se prevê o futuro do software, se ele será de boa ou má qualidade. E para no fim do desenvolvimento de um software se chegue a um produto de qualidade foram desenvolvidas várias técnicas dentro da disciplina da engenharia de software.

Tal disciplina diz que o processo de um bom desenvolvimento de um software se subdivide em ciclos (PRESSMAN, 2002). Estes ciclos são chamados de ciclos de vida de um software, e é especificadamente nestes ciclos que se devem tomar medidas para que no final se obtenha um software de qualidade.

O mesmo autor define o ciclo de vida clássico do desenvolvimento de um software, também chamado como modelo em cascata, em seis partes, são elas:

- Engenharia de sistemas - é o estabelecimento de requisitos do sistema;
- Análise de requisitos - é a coleta de requisitos e informações necessárias para a implementação de um sistema;
- Projeto - é a determinação de todos os passos que serão executados durante o desenvolvimento. É onde se estabelece a arquitetura, estrutura de dados, detalhes procedimentais e interface do sistema;
- Codificação - é a tradução do projeto para uma linguagem computacional para então ser compilado em um objeto executável no computador;
- Teste - após a codificação o teste é a validação dos procedimentos que o sistema executa, aonde se procura por erros e procura-se corrigi-los; e,
- Manutenção - como o desenvolvimento é feito por pessoas, mesmo após os testes, erros podem persistir em existir, para isto a manutenção é necessária para prever e corrigir problemas futuros, ou prover modificações mediante necessidade.

Na figura 20, se tem um melhor entendimento de como este ciclo de vida de um software funciona aonde a engenharia de sistemas é base para a analise de requisitos e o projeto do software. Este seria o ciclo de vida clássico de um software.

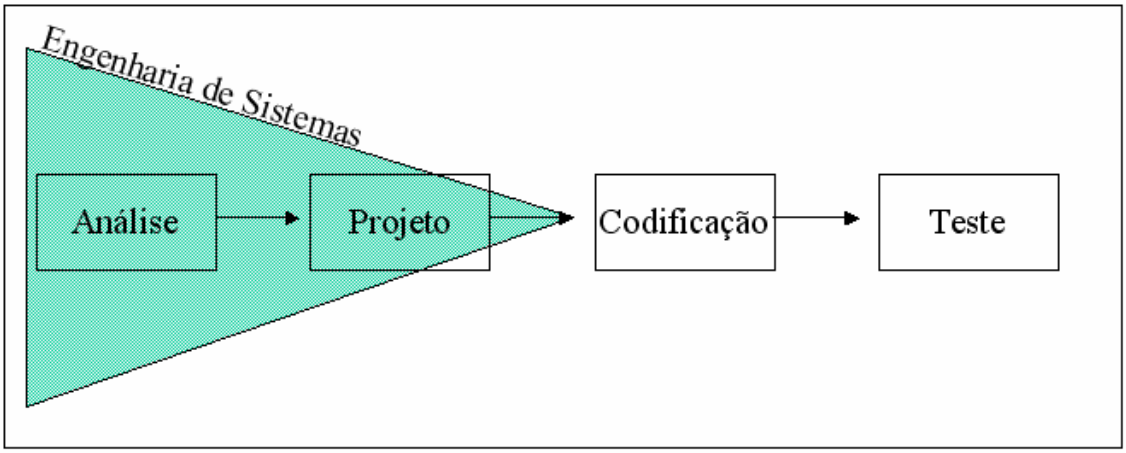

Figura 20: Esquema seqüencial do ciclo de vida clássico de um software.

Existem muitos outros criados pela Engenharia de Software, mas este foi escolhido para ser utilizado neste trabalho devido a sua simplicidade e flexibilidade para uso nos mais diversos tipos de software, em questão em Softwares de Geoprocessamento, no qual este trabalho se dedica. Ainda, deixando claro que o desenvolvimento de um software pode ser visto como arte, pois segundo Aurélio (1986), arte é a capacidade que tem o homem de por em prática uma idéia. Logo, o desenvolvimento de um software depende da capacidade de seu desenvolvedor de compilar em linguagem entendível pelo computador algo que este tem em mente.

Neste trabalho serão sugeridas algumas técnicas para um melhor desenvolvimento de um SIG, em cada um dos ciclos de vida de um software, não deixando de respeitar esta arte do desenvolvedor, garantindo a liberdade do desenvolvedor de executar todo o processo de forma mais conveniente ao mesmo.

A seguir será descrito e sugerido estas técnicas para cada ciclo de vida de um software que auxiliarão no desenvolvimento de um Sistema de Informação Geográfica.

2.3.1 Engenharia de sistemas

Nesta fase se determinam duas questões primordiais: o que deve ser feito e o que se espera ter como resultado, mas ainda não entra em questão como o trabalho será realizado, (Fiorini et al 1998).

A Engenharia de Sistemas é uma fase que dá sustentação para a fase de Análise e Projeto de Software, pois estas duas fases dependem das prerrogativas definidas nesta fase.

Em outras palavras, este primeiro ponto do desenvolvimento de um software é onde se define uma série de requisitos para o sistema. Requisitos que estabelecem o que o software terá de fazer de acordo com a sua aplicação, pois de acordo com a sua aplicação será estabelecida uma série de funções que são requeridas pela mesma.

Estes requisitos do sistema incluem:

- Os procedimentos e operações que o software deverá realizar;
- O tipo de Hardware ao qual o software será desenvolvido;
- Como o usuário irá interagir com o sistema;
- Como o programa visualizará os dados; e,
- O que o software não poderá fazer.

Existem várias funções e procedimentos que um SIG pode executar, e dependendo do uso que será dado ao SIG a ser desenvolvido, este terá funcionalidades específicas para o seu funcionamento.

Após o estabelecimento dos requisitos do sistema, estas informações deverão ser documentadas e distribuídas para todas as partes participantes do desenvolvimento do software, pois para o bom andamento do desenvolvimento, todas as pessoas envolvidas devem ter em mente o que será o produto final.

2.3.2 Análise de requisitos

Após o estabelecimento de requisitos na fase da Engenharia de Sistemas, a fase da Análise de Requisitos do Sistema é responsável pelo estudo destes requisitos e sua estruturação. Este estudo é voltado somente para o software, sem levar em consideração o hardware. Dependendo dos requisitos, algumas informações serão necessárias para que se possa desenvolver o projeto de software, que é a próxima fase do desenvolvimento global.

Pressman (2002) divide a análise dos requisitos de software em cinco áreas de esforço, são elas:

- Reconhecimento do Problema;
- Avaliação e síntese;
- Modelagem;
- Especificação; e,
- Revisão.

Nesta primeira fase de reconhecimento do problema se estabelece o que o referido requisito é. Exemplo: Visualização de um dado do tipo Vetorial em tela.

Logo em seguinte temos a fase de avaliação e síntese, onde o foco é estabelecer informações sobre o requisito. Exemplo: como é a estrutura de um dado do tipo Vetorial.

A modelagem é onde se estrutura a solução de acordo com o estudo do requisito. Exemplo - especificar que é necessário um componente gráfico para a visualização do dado vetorial.

Na especificação, como o nome já diz, se especifica detalhes da modelagem. Exemplo - as funções que o componente gráfico deve ter para poder executar a visualização dos dados vetoriais.

E, finalmente se revisa todas as fases anteriores a procura de erros, caso não haja, se passa para o próximo requisito do software.

Todas estas informações, sobre cada requisito de software, deve ser documentado de maneira mais conveniente e anexada aos documentos gerados na fase de Engenharia de Sistemas.

2.3.3 Projeto

O projeto de um software é a parte mais importante da construção de um software, pois é onde se definem todas as atividades que serão executadas durante o desenvolvimento do software. Nesta fase se define um plano de realização do trabalho, ou plano de desenvolvimento de software, estabelecendo tarefas a serem executadas pelas partes envolvidas, (Fiorini et al. 1998).

No projeto, todas as partes envolvidas, devem se reunir e estabelecer algumas características do SIG a ser desenvolvido, como:

- qual Banco de Dados será utilizado;
- qual linguagem de programação será utilizada;
- como deverão ser as telas de interface maquina/usuário;
- quem será responsável por cada parte do projeto;
- definir a estrutura dos dados;
- definir a arquitetura do software; e,
- e todos detalhes técnicos envolvidos.

Deve ser redigido um documento que contenha todos os detalhes técnicos do sistema e, após o consentimento de todos os envolvidos no desenvolvimento deve-se criar um cronograma de atividades.

Um cronograma de atividades é um processo que também deve ser documentado, e resguardado junto ao resto dos documentos já gerados, onde as partes envolvidas no projeto de software recebem tarefas a serem executadas.

Neste cronograma, além das atividades será definido um tempo para conclusão do mesmo. Tempo de conclusão deve ser negociado com quem o irá executar e assinado pelo mesmo para documentar sua concordância com as datas. Um exemplo de cronograma se encontra na tabela 6.

Quando um procedimento é cronogramado e ele tiver como pré-requisito outra tarefa, se deve ter o cuidado para estabelecer um tempo extra entre o término de um e o começo do outro para eventuais atrasos do cumprimento das tarefas. Exemplificando, na tabela 6, entre o término da tarefa 1 e o inicio da tarefa 3, foi estipulado um tempo para suprir atrasos que são comuns no desenvolvimento de softwares.

Como um SIG pode ter várias aplicações, para manter a qualidade do Software a ser desenvolvido, este, deve ser adaptado ao seu fim. Existe uma linguagem técnica adequada de cada área que um SIG pode ser utilizado, logo, este deve utilizar em suas telas de interface com o usuário esta linguagem, pois assim se adapta o software para o seu usuário final o tornando de melhor e mais fácil utilização resultando em uma qualidade melhor do software. Na figura 21, se exemplifica a adaptação de um SIG voltado a topografia, com seus termos utilizados.

Tabela 6: Exemplo de cronogramação.

Tarefa	Descrição da tarefa	Responsável	Data	Data de Entrega	Assinatura do responsável
001 Montagem do Banco de Dados	Definir as tabelas do banco de dados de e criar um banco de dados funcional usando MySQL	Marcos André Storck	01/05/2005	23/05/2005	
002 Criação do Procedimento 1	Desenvolver um procedimento de visualização em tela de dados Vetoriais de extensão ".VET" utilizando linguagem de programação Delphi com uso do Componente Gráfico CANVAS	Luiz Antônio de Almeida	06/05/2005	30/05/2005	
003 Criação do Procedimento 2	Desenvolver um procedimento na Linguagem de programação Delphi que leia os dados da tabela "GEORREFERENCIA" do Banco de Dados desenvolvido na "Tarefa 001" e recupere a georreferência gravada da imagem (numa determinada pasta) a qual o item da tabela está relacionado	Alexandre Thomas	27/05/2005	19/06/2005	
004 Criação do Procedimento 3	---------	---------			
-------	---------	---------			
-------	---------	---------			

Figura 21: Tela de interface com o usuário do Sistema de Processamento de Levantamentos Topográficos do sistema **CR – CAMPEIRO 5**.

Para se notar a importância de definir e executar um projeto de software, se demonstra na Figura 22, o custo de se fazer alguma modificação no software durante o seu desenvolvimento tendo em vista mudanças de requisitos, causados por uma má analise.

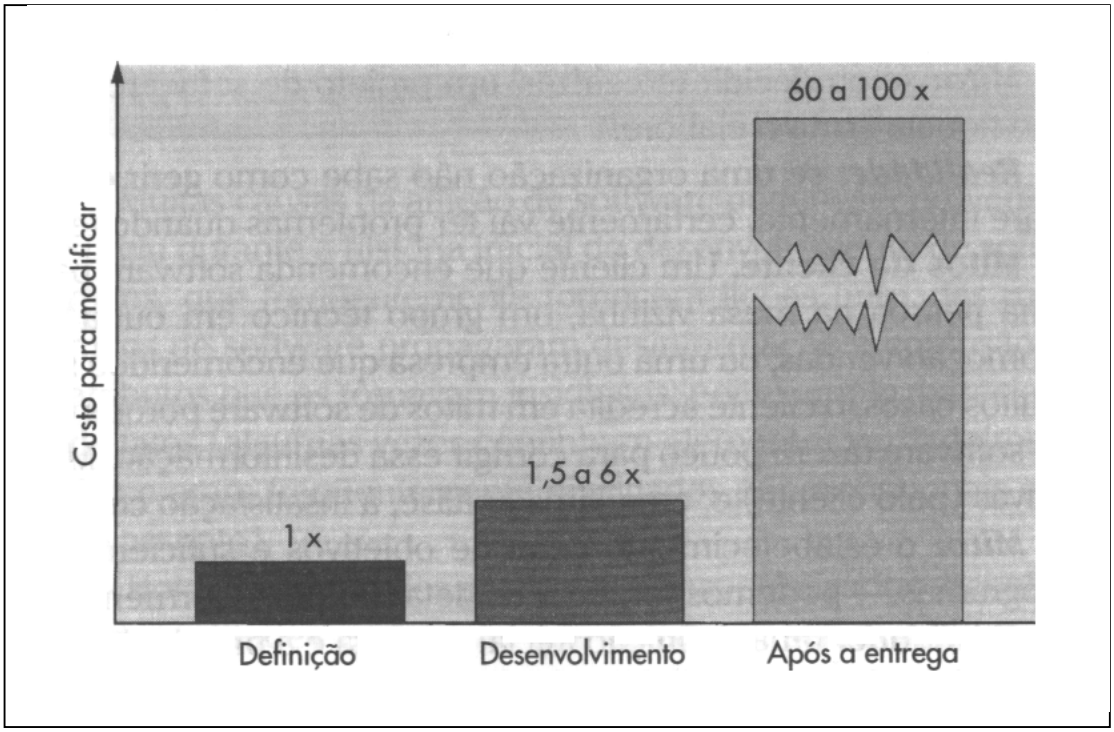

Figura 22: Gráfico de custos de modificação de software (PRESSMAN, 2001).

Como se pode notar, quanto mais tardia for a modificação em maiores custos de dinheiro, tempo e mão de obra acarretará, assim demonstrando a importância de uma boa analise para que então um projeto de software competente, e que supra as necessidades as quais o software será desenvolvido, seja estabelecido.

2.3.4 Codificação

Já definida a linguagem de programação que será utilizada, na definição do projeto do software, a codificação é o uso desta linguagem para criar os algoritmos que irão implementar os requisitos do sistema.

É a fase da literal construção do SIG, é onde o que foi chamado de arte aparece mais claramente, pois para codificar em linguagem entendível pelo computador uma idéia ou conceito dos requisitos do sistema, é necessário além de um domínio da linguagem de programação, uma capacidade criativa de transformar a idéia em um algoritmo.

Uma pratica muito usada no desenvolvimento de um programa, é dividir o programa em módulos ou procedimentos. Assim a mesma linha de código pode ser utilizada em várias partes do programa evitando a redundância de linha de código, tornando o programa mais ágil e de mais fácil manutenção.

Ainda para facilitar a manutenção e modificações durante a própria codificação é muito importante usar comentários de linha de código. Comentários de Linha de código são observações feitas entre as linhas de código do programa que não são compiladas pela linguagem de programação. Possuem a função de facilitar o entendimento da codificação. Exemplificando a diferença entre linguagens de programação e seus comentários, temos a Tabela 7.

Tabela 7: Quadro comparativo entre as linguagens de programação Delphi e Visual Basic.

Delphi	**Visual Basic**
Procedure abre_arq;	**sub abre_arq()**
//abre o procedimento de nome "abre_arq"	*'inicia o procedimento de nome "abre_arq"*
var	
//define as variáveis "F" e "S" do procedimento	**dim S as string**
F: TextFile;	*'define variavel "S"*
S: string;	
begin	**s= " teste de entrada "**
s:= ' teste de entrada ';	*' atribui valor a variável "s"*
//atribui valor a variavel "S"	
AssignFile(F, 'teste.txt');	**Open "teste.txt" For Output As #l**
//atribui nome lógico (f) ao nome fisico teste.txt)	*' abre o arquivo.*
Reset(F);	
//posiciona o ponteiro no inicio do arquivo	**Print #l, S**
writeln(F, S);	*' grava a seqüência de caracteres no*
//grava o conteudo da variavel "S"	*'arquivo.*
//ao arquivo de nome lógico "F"	
label1.caption := 'O texto '+ S + 'gravado';	**Label1.caption = "O texto"+S+"gravado"**
//mostra no componente "label1"	*'mostra no componente "label1" uma*
//uma mensagem que valor	*'mensagem que valor*
//da variavel "S" foi gravada	*'da variavel "S" foi gravada*
CloseFile(F);	**Close #1**
//fecha o arquivo	*'fecha o arquivo*
end;	
end;	**End sub**

O ato de fazer comentários em linha de código é uma prática muito importante, pois linguagens de programação são diferentes quando comparadas entre si, e o próprio código tem uma leitura e entendimento muito difícil. Isto pode ser visto na Tabela 6, que demonstra a diferença entre duas linguagens de programação, codificando procedimentos iguais.

2.3.5 Teste

Esta é a fase final do desenvolvimento de um SIG, antecede a entrega da versão final do software. A fase de teste, como o próprio nome já diz, é a fase onde se testa o software como um todo para verificar sua funcionalidade.

Deve-se atribuir um tempo generoso para esta fase do desenvolvimento de software e faze-lo de maneira meticulosa, pois apesar de não ser o teste de software que garante a qualidade de um software, é esta fase que a afirma.

A palavra chave do teste de software é “validar”, ou seja, confirmar a exatidão dos processos que o SIG deverá executar.

A validação de um software começa pela validação de cada um de seus processos e requisitos. Isto é, o ato de confirmar que o algoritmo aplicado em cima de uma linguagem de programação foi elaborado de maneira correta.

Esta confirmação é feita comparando o resultado que se tem com o procedimento desenvolvido com um resultado obtido de maneira manual e com outros softwares de confiança.

Como exemplo de validação podemos comparar um procedimento de transformação de um sistema de coordenada para outro criado no desenvolvimento do SIG e comparar o resultado com um calculo feito à mão e um resultado utilizando outro software que executa a mesma função. Assim, se afirma a exatidão de seus processos, garantindo a qualidade do SIG desenvolvido. Mas, se ao tentar validar um procedimento do SIG, este mostrar disparidade entre os resultados, isto significa que existe algo errado no algoritmo desenvolvido, logo, é preciso retornar a codificação e revisar todo o código a procura do causador do erro.

Nesta situação se exemplifica a importância de aplicar comentários na linha de código do programa, pois esta é uma situação em que os comentários feitos são de

extrema importância na ajuda para entender o que o código de programação significa, e então corrigir o erro verificado durante a validação nos testes de software.

Um software nunca está livre de erros, muito pelo contrário, pois algumas correções sempre são necessárias, então durante os testes de software o desenvolvimento do SIG se depara com um ciclo de correções até que o software chegue no ponto planejado no principio.

2.3.6 Manutenção

Em si a manutenção de um software não faz parte do desenvolvimento, pois é feito após o software já estar pronto, mas é peça fundamental para o crescimento e evolução de um SIG.

A manutenção envolve corrigir erros que sobreviveram aos testes de software, e também faz parte desta fase aplicar modificações futuras no sistema, como novos procedimentos e funções do sistema.

Esta fase é exclusivamente contínua na vida de um software, e por meio desta prática um SIG pode manter-se sempre atualizado e otimizado.

3 CONCLUSÃO

Depois do estudo referente à qualidade de *software*, chegou-se a conclusão que esta prática é absolutamente necessária para reduzir os casos de insucessos econômicos, profissionais e de credibilidade no mercado.

Esta necessidade na qualidade de *software* se vê pela importância e a dependência que a sociedade como um todo tem pelos *softwares*, e devido às aplicações que um Sistema de Informação Geográfica (SIG) pode ter, e sua importância nas mais diversas áreas, estes sistemas devem possuir características de um software de qualidade.

Após um breve estudo sobre SIG, se verificou uma grande complexibilidade, em nível de estruturação, que estes softwares possuem, fazendo com que o seu desenvolvimento seja uma tarefa muito difícil, e que sem algum controle neste processo de desenvolvimento seria considerada uma tarefa impossível, pois repetindo as palavras de Brooks apud Fiorini et al(1998), o desenvolvimento de um software é uma tarefa que não pode ser simplificada devido a complexibilidade que os softwares possuem.

Visto que os SIG's geralmente são desenvolvidos ou tem participação de profissionais oriundos das ciências exatas e da terra, e, porque estes profissionais terem maior conhecimento de lidar e processar os dados que um SIG processa, este trabalho reuniu técnicas desenvolvidas na área da informática sobre qualidade de software e os compilou em uma metodologia de desenvolvimento de SIG's com preocupação no controle de qualidade do software. A finalidade foi a de auxiliar estes profissionais mencionados no desenvolvimento de SIG's, como ferramenta de grande ajuda para o crescimento desta tecnologia.

Este trabalho ainda não sofreu uma validação para confirmar a metodologia desenvolvida, mas todos os resultados esperados foram concluídos de forma satisfatória. Esta tarefa, juntamente com o desenvolvimento de documentação pré-definida pode ser alvo de futuras pesquisas.

4 REFERÊNCIAS BIBLIOGRÁFICAS

ARTHUR, L. J. **Melhorando a qualidade do *software* – Um guia completo para o TQM**, Rio de Janeiro: Infobook, 1994. 304p.

ASSAD, E. D.; SANO, E. E. **Sistemas de informações Geográficas – Aplicações na Agricultura**: Brasília: EMBRAPA, 1998. 434p.

AURÉLIO, B. de O. F. **Novo dicionário da língua portuguesa**: português. Rio de Janeiro: Nova Fronteira, 1986.

BOURROUGH, P. A.; MCDONNEL, R. A. **Principles of Geographical Information Systems**. New York: Oxford, 1998. 333p.

CÂMARA, G et al. **Banco de Dados Geográficos**. Brasília: EMBRAPA, 2004. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/bdados/> Acesso em: 18 mai. 2005.

CÂMARA, G.; DAVIS, C.; MONTEIRO, A.V.M. **Introdução à Ciência da Geoinformação**. Brasília: EMBRAPA, 2004. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/introd/> Acesso em: 18 abr. 2005.

CÂMARA, G.; FREITAS, U. M. de. Perspectivas em Sistemas de Informação Geográfica. In: **Simpósio sobre CAD/CAM e Computação Gráfica, SOBRACON**, 1995, São Paulo. **Anais Eletrônicos...** São Paulo, 1995. Disponível em: <http://www.dpi.inpe.br/gilberto/fatorgis95.pdf>. Acesso em 13 abr. 2005.

CAPABILITY MATURITY MODEL. CMM. Carnegie Mellon University - Software Engineering Institute, 2003**.** Disponível em: <http://www.sei.cmu.edu/cmm/>. Acesso em: 17 nov. 2003.

CHRISMAN, N. **Exploring Geographic Information Systems**. New York, Johm Wiley & Sons, 1997. 298p.

COURSEWARE. In: **Courseware em Ciências Cartográficas** - Unesp - Campus de Presidente Prudente – Introdução ao Geoprocessamento. 2000. Disponível em: <http://www.multimidia.prudente.unesp.br/arlete/hp_arlete/courseware/intgeo.htm> Acesso em 23 jun. 2005.

DAVIS JR., A. D. Múltiplas representações em bancos de dados geográfico. In: V **GIS Brasil**, 1999, Salvador. **Anais eletrônicos**... Salvador, 1;999. Disponível em: <http://www.pbh.gov.br/prodabel/cde/publicacoes/1999/davis2_1999.pdf>. Acesso em 10 mai. 2005.

DAVIS JR., A. D.; LAENDER., A. H. F. Multiple Representations in GIS: materialization through map generalization, geometric and spatial analysis operations. In: INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 7., 1999 Kansas City. **Anais eletrônicos…** Kansas City, 1999. Disponível em: <http://www.pbh.gov.br/prodabel/cde/publicacoes/ 1999/davis1999.pdf>. Acesso em 10 mai. 2005.

DRUCK, S. et al. **Análise Espacial de Dados Geográficos**. Brasília: EMBRAPA, 2004. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/analise/> Acesso em: 18 abr. 2005.

FIORINI, S. T.; STAA, A. Von; BAPTISTA, R. M. **Engenharia de *software* com CMM**, Rio de Janeiro: BRASPORT, 1998. 346p.

FLORES, R. F. **Um Estudo Sobre Qualidade Em Pacotes de *Software*.** 2001. 29f. Monografia (Trabalho Final de Graduação em Sistemas de Informação) - Centro Universitário Franciscano, Santa Maria, 2001.

HUMPHREY, W. S. **A discipline for *software* engineering.** Boston: Addison-Wesley, 1995. 816p.

HUMPHREY, W. S. **Introduction to the personal *software* process.** Boston: Addison-Wesley, 1997. 278p.

INTHURN, C. **Qualidade & Teste de *Software*.** Florianópolis: Visual Books, 2001. 110p.

LABORATÓRIO DE GEOMÁTICA, Universidade Federal de Santa Maria. **CR – CAMPEIRO**. Santa Maria, 2004. Versão 5 AV2. 1 CD. Sistema Operacional Windows XP.

LAMPARELLI, R. A. C.; ROCHA, J. V.; BORGHI, E. **Geoprocessamento e Agricultura de Precisão –** Fundamentos e Aplicações**.** Guaíba: Agropecuária, 2001. 118p.

LEWIS, W. E. **Software Testing and Continuous Quality Improvement**, Auerbach, 2000. 656p.

LIMA, T. G. **Avaliação da Qualidade de *Software* Educacional**. 2001. 30f. Monografia (Trabalho Final de Graduação em Sistemas de Informação) - Centro Universitário Franciscano, Santa Maria, 2001.

LISBOA FILHO, J. **Introdução a SIG – Sistemas de Informações Geográfica**: Porto Alegre, Curso de Pós-Graduação em Ciência da Computação, URFGS, 1995. 69p.

LISBOA FILHO, J. Projeto de Banco de Dados para Sistemas de Informação Geográfica. **Sociedade Brasileira de Computação – Revista Eletrônica de Iniciação Cientifica,** n.2, 2001. Disponível em: <http://www.sbc.org.br/reic/ edicoes/2001e2/tutoriais/ProjetodeBDparaSistemasdeInformacaoGeografica.pdf >. Acesso em 13 abr. 2005.

MOLINARI, L. **BTO – Otimização da Tecnologia de Negócios – Qualidade de Software na Prática**, São Paulo: Érica, 2003. 216 p.

MONICO, J. F. G. **Posicionamento pelo NAVSTAR-GPS: descrição, fundamentos e aplicações.** São Paulo: UNESP, 2000. 287p.

MOURA, A. C. M. **Geoprocessamento na Gestão e Planejamento Urbano**. Belo Horizonte, Ed. Da Autora, 2003. 294p.

PETERS, J F; PEDRYCZ, W. **Engenharia de Software – Teoria e Prática**, São Paulo: Campus – 2001. 602 p.

PFLEEGER, S. L. **Software Engineering – Theory and Practice**, New Jersey: Prentice-Hall, 2001. 659 p.

PRESSMAN, R. S. **Engenharia de *Software***. 5. ed. Rio de Janeiro: Mc Graw Hill, 2002. 843 p.

REZENDE, D. A. **Engenharia de Software E Sistemas de Informação**. Rio De Janeiro: Brasport, 1999. 292 p.

ROCHA, A. R. C. da; MALDONADO, J. C.; WEBER, K. C. **Qualidade de *Software*, Teoria e Prática**. São Paulo: Pretice Hall, 2001. 303 p.

ROCHA, C. H. B. **Geoprocessamento: Tecnologia Transdisciplinar.** Juiz de Fora: Ed. do Autor, 2000. 220p.

SHILLER, L. **Excelência em *Software***. São Paulo: Makron Books, 1993. 295 p.

SILVA, A. de B. **Sistemas de Informações Geo-Referenciados – Conceitos e Fundamentos**. Campinas: UNICAMP, 2003. 236p.

SOMMERVILLE, I. **Software Engineering**. 6. ed. Boston: Addison-Wesley, 2001. 688 p.

SPICE. In: **The SPICE Website**. Software Quality Institute. 2003 Disponível em: <http://www.sqi.gu.edu.au/spice/> Acesso em: 17 nov. 2003.

WEBER, K. C.; ROCHA, A. R. C. da; NASCIMENTO, C. J. **Qualidade e Produtividade em Software**. 4. ed. São Paulo: Makron Books. 2001. 188 p.

ZORZO, J. J. **Melhorando o processo de desenvolvimento de software em pequenas organizações, através da interpretação de modelos de qualidade.** 2001. 168f. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Santa Maria, Santa Maria, 2001.