UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS

INFLUÊNCIA DA UTILIZAÇÃO DE UMA METODOLOGIA DE DESENVOLVIMENTO SOBRE A QUALIDADE DO SOFTWARE: UM ENFOQUE QUANTITATIVO

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

CARLOS EDUARDO NEGRÃO BIZZOTTO

FLORIANÓPOLIS

SANTA CATARINA - BRASIL

DEZEMBRO DE 1992

"INFLUÊNCIA DA UTILIZAÇÃO DE UMA

METODOLOGIA DE DESENVOLVIMENTO

SOBRE A QUALIDADE DO SOFTWARE :

UM ENFOQUE QUANTITATIVO"

CARLOS EDUARDO NEGRÃO BIZZOTTO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA DE PRODUÇÃO, OPÇÃO GERÊNCIA DA PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO.

NERI DOS SANTOS, Ph. D

COORDENADOR DO PROGRAMA DE PÓS-GRADUAÇÃO

BANCA EXAMINADORA :

LUIZ CARLOS DUCLÓS, Ph. D

PRESIDENTE

BRUNO HARTMUT KOPITTKE, Ph. D.

NERI DOS SANTOS, Ph. D.

CONCEPCIÓN ULISES V. VELASQUES, M. Sc.

EDSON PACHECO PALADINI, M. Sc.

A quem soube dar um belo exemplo de vida, apesar de muitas vezes imcompreendido :

GERALDO

À grande responsável pela formação de minha personalidade:

TEREZA

Àqueles que souberam ser mais amigos do que irmãos:

JÚNIOR E FERNANDO

Àqueles que me acolheram com o maior carinho:

ÁLVARO E MAURITA
ALFREDO E MARIA JÚLIA

Àqueles que me ensinaram a ser mais paciente e compreensivo:

MARIA EDUARDA E CAIO

"Não importa o **quanto** se vive, mas **como** se vive."

A quem me proporcionou ,e proporciona, os melhores **comos** que eu poderia conseguir :

MARA REGINA

AGRADECIMENTOS

Ao Dr. Luiz Carlos Duclós, pela eficáz orientação e por sempre ter acreditado neste trabalho.

Aos Professores Ner dos Santos, Edison Pacheco Paladini e Bruno hartmut Kopttke pelas sugestões e comentários que contribuíram para o enriquecimento do trabalho.

A Ulises Velasques pelo acompanhamento contínuo e criterioso do trabalho, bem como às excelentes críticas construtivas realizadas.

À NTS - Núcleo de ecnologia de Software pelo fornecimento de apoios materiais e financeiros que contribuíram decisivamente para a conclusão do presente trabalho. Em especial a seu Diretor-Presidente, Dr. Luiz Carlos Duclós, por ter permitido a utilização da empresa para a aplicação prática do modelo elaborado.

À CAPES, pelo apoio financeiro fornecido durante a fase de créditos do mestrado.

À Universidade Federal de Santa Catarina, de modo especial ao Programa de Pós-Graduação em Engenharia de Produção, que forneceu as condições necessárias à realização dos objetivos estabelecidos.

A todos que direta ou indiretamente contribuiram para o bom andamento do presente trabalho.

RESUMO

O presente trabalho propõe um modelo para a quantificação da influência da utilização de uma metodologia de desenvolvimento sobre a qualidade do software.

Como resultado do detalhamento do modelo proposto, são sugeridos procedimentos para a quantificação da qualidade da metodologia utilizada e dos softwares desenvolvidos.

Com o objetivo de fornecer a base necessária à elaboração do modelo, o trabalho faz uma descrição dos fatores que se interrelacionam em um ambiente de desenvolvimento de software, a saber: Metodologia, Recursos Humanos e Gerência.

A fim de se verificar a adequação do modelo desenvolvido, procedeu-se a aplicação prática do mesmo. Esta aplicação forneceu, além de informações valiosas sobre o ambiente de desenvolvimento de software, os primeiros dados necessários á quantificação da influência que uma metodologia exerce sobre a qualidade do software.

ABSTRACT

These research presents a model for quantifying the influency of a development methodology over the software quality.

As a result from the proposed model, it is suggested procedures to the quality quantification of the methodology and the developed softwares.

The present work is a description of the factors that interact in a software development environment such as : methodology, personnel and management, with the objective of furnishing the necessary base to the model elaboration.

In order to verify the developed model fitness, was realized a practical application of the model. This application furnished, in adition to the valuable informations about the software development environment, the data to quantifying the methodology influency over the software quality.

SUMÁRIO

1 - DEFINIÇÃO DO TRABALHO 1

1.1. - HISTÓRICO DO ASSUNTO 1

1.2. - ORIGEM DO TEMA 3

1.3. - OBJETIVO 4

1.4. - IMPORTÂNCIA DO TRABALHO 5

1.5. - METODOLOGIA DE TRABALHO 6

1.6. - LIMITAÇÕES 6

1.7. - ESTRUTURA 6

2 - SOFTWARE E ENGENHARIA DE SOFTWARE 8

2.1. - INTRODUÇÃO 8

2.2. - CARACTERÍSTICAS DO SOFTWARE 13

2.3. - CRISE DO SOFTWARE 18

2.4. - QUALIDADE DO SOFTWARE 19

2.5. - CONTROLE DE QUALIDADE DE SOFTWARE 26

3 - AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE 31

3.1. - INTRODUÇÃO 31

3.2. - METODOLOGIA 31

3.2.1. - DESENVOLVIMENTO INCREMENTAL 37

3.2.2. - PROTOTIPAÇÃO 37

3.2.3. - SOFTWARE REUSÁVEL 38

3.2.4. - ENGENHARIA DA INFORMAÇÃO 38

3.2.5. - CICLO DE VIDA ESTRUTURADO 40

3.2.5.1. - ESTUDO 41

3.2.5.2. - ANÁLISE 43

3.2.5.3. - PROJETO 46

3.2.5.4. - IMPLEMENTAÇÃO 49

3.2.5.5. - CONTROLE DE QUALIDADE 50

3.2.5.6. - DESCRIÇÃO DE PROCEDIMENTOS 54

3.2.5.7. - CONVERSÃO DE BANCOS DE DADOS . . . 55

3.2.5.8. - INSTALAÇÃO 55

3.3. - GERÊNCIA 56

3.4. - RECURSOS HUMANOS 61

4 - METODOLOGIA E QUALIDADE DE SOFTWARE 66

4.1. - INTRODUÇÃO 66

4.2. - MODELO PROPOSTO 69

4.2.1. - RELACIONAR OS FATORES DE QUALIDADE DE SOFTWARE 72

4.2.2. - QUANTIFICAR A QUALIDADE DO SOFTWARE DESENVOLVIDO 74

4.2.3. - QUANTIFICAR A QUALIDADE DA METODOLOGIA UTILIZADA 79

5 - APLICAÇÃO DO MODELO 89

5.1. - INTRODUÇÃO 89

5.2. - QUALIDADE DO SOFTWARE 90

5.2.1. - CLASSIFICAÇÃO DOS DADOS 93

5.2.2. - QUANTIFICAÇÃO DAS RESPOSTAS 93

5.2.3. - QUANTIFICAÇÃO DOS FATORES DE QUALIDADE 94

5.2.4. - QUANTIFICAR A QUALIDADE DO SOFTWARE 94

5.2.5. - RESULTADOS OBTIDOS 95

5.3. - QUALIDADE DA METODOLOGIA 102

5.3.1. - QUANTIFICAÇÃO DOS SUBFATORES DE QUALIDADE 104

5.3.2. - QUANTIFICAÇÃO DOS FATORES DE QUALIDADE 104

5.3.3. - QUANTIFICAÇÃO DOS OBJETIVOS DE QUALIDADE 105

5.3.4. - CÁLCULO DA QUALIDADE DA METODOLOGIA 105

5.3.5. - RESULTADOS OBTIDOS 105

5.4. - ANÁLISE DOS RESULTADOS 112

6 - CONCLUSÕES E RECOMENDAÇÕES 117

6.1. - CONCLUSÕES 117

6.2. - RECOMENDAÇÕES 119

7 - BIBLIOGRAFIA 121

8 - APÊNDICE 1 130

9 - APÊNDICE 2 136

CAPÍTULO 1

1. DEFINIÇÃO DO TRABALHO

1.1. HISTÓRICO DO ASSUNTO

O homem sempre procurou mecanismos que aliviassem os trabalhos desde os mais pesados e complexos até os mais rotineiros. Devido a esse espírito inovador o próprio homem agilizou a descoberta do computador. Sua principal característica era realizar tarefas rotineiras mais rápida e corretamente do que seria possível manualmente. Estas tarefas seriam realizadas através de programas desenvolvidos por apenas um programador, pois tais programas seriam utilizados inicialmente, apenas para resolverem problemas simples.

Com este objetivo, o custo total do sistema (software + hardware), era devido quase que exclusivamente ao custo de construção do hardware.

Em função do alto custo do hardware, foram desenvolvidas várias pesquisas científicas para diminuir a participação daquele custo no custo total do sistema (hardware + software). Os resultados destas pesquisas concretizou-se na drástica redução dos custos de manufatura do hardware.

Um outro aspecto a ser ressaltado, é de que como o software, inicialmente denominado programa, era bastante simples de ser desenvolvido, estimulou ainda mais as pesquisas do hardware.

Paralelamente a pesquisas na área de hardware, houve um aumento na complexidade dos softwares solicitados. Esta situação provocou uma alteração radical na composição do custo do sistema, invertendo a situação inicial, e assim, o desenvolvimento de software passou a ser mais caro que a manufatura do hardware.

Entretanto, os avanços conseguidos na manufatura de hardware não influenciaram diretamente , na melhoria do processo de desenvolvimento de software, pois os dois produtos são fundamentalmente diferentes. Em termos práticos é possível "tocar" ou "pegar" as partes constituintes do hardware, não ocorrendo o mesmo com o software. Isto porque o hardware possui características físicas enquanto o software define-se através de características lógicas. Além disso, ambos possuem evoluções também distintas.

Ao contrário do hardware, durante as décadas de 50 e 60, o processo de desenvolvimento do software não era formalizado, sendo encarado mais como uma atividade artesanal. Somente no final da década de 60 é que o software começou a ser tratado como um produto. Ao invés de se pressionar os programadores a produzir mais em menos tempo, procurou-se mudar a maneira pela qual os softwares eram desenvolvidos.

A primeira mudança foi na elaboração de novas linguagens de programação. Estas linguagens diminuíram o esforço necessário para o desenvolvimento de software. Iniciaram-se também, investigações sistemáticas para se desenvolver softwares mais confiáveis, com menor tempo de desenvolvimento e de baixo custo. A partir destes objetivos conjuntos obteve-se como resultante a Programação Estruturada.

Apesar de fornecer um aumento de produtividade, a Programação Estruturada e as novas linguagens afetaram apenas a parte codificada do desenvolvimento do software, representando somente 20% do tempo e do custo total de desenvolvimento do mesmo [Teixeira,79].

A inovação seguinte enfocou o desenvolvimento de metodologias, as quais indicavam os passos a serem seguidos para o desenvolvimento de software. Embora resultassem em pequenos ganhos de produtividade, 7% ao ano segundo Goldberg [Goldberg,86], tais metodologias foram importantes no sentido de proporcionarem um maior entendimento do ciclo de vida do software. Além disso, facilitaram a comunicação entre os membros das equipes de desenvolvimento.

Após o surgimento das metodologias, a preocupação passou a ser o controle de qualidade. Este controle foi possível graças à padronização das atividades do desenvolvimento de software.

O próximo passo parece indicar para a automatização do processo de desenvolvimento de software. Ao invés de metodologias manuais, pensou-se num conjunto de ferramentas integradas que conduzissem todo o desenvolvimento desde a análise dos requisitos até à implementação.

Apesar do grande avanço nas pesquisas, a utilização destas descobertas nas empresas de software permanece aquém das expectativas. Isto se deve ao fato de que a transferência tecnológica necessita muito tempo para ser efetivada.

Segundo Goldberg [Goldberg,86], esse período de tempo gira em torno de 15 a 20 anos. De acordo com este raciocínio, as metodologias somente seriam aceitas em 1998, considerando-se que elas tenham tido início em 1978. Isso não quer dizer que em 1998 todos utilizariam alguma metodologia, mas sim que todos concordariam na necessidade de utilização de uma metodologia de desenvolvimento.

Em função disto, resta muito a ser feito para a efetiva utilização dos conhecimentos disponíveis atualmente.

1.2. ORIGEM DO TEMA

O rápido aumento da demanda por novos sistemas se contrapõe à capacidade das empresas em produzí-los, originando uma demanda reprimida de quatro anos [Boehm,81].

Por este motivo, as empresas estão tentando melhorar sua capacidade de desenvolvimento, através da melhoria da qualidade dos sistemas produzidos. Isto se deve ao fato de que a melhoria da qualidade proporciona dentre tantos motivos, a racionalização dos recursos utilizados, diminuição do retrabalho e aumento da motivação do pessoal.

Dentro deste contexto, o Núcleo de Tecnologia de Software - NTS, firmou com a Universidade Federal de Santa Catarina, um convênio para a realização de projetos técnicos. O primeiro projeto consiste na elaboração de duas dissertações de mestrado, em área de aplicação de interesse da empresa.

Uma das áreas de interesse é analisar a relação existente entre metodologia de desenvolvimento e qualidade de software.

Além da importância do tema dentro da Engenharia de Software, existem outros fatores que motivaram a realização deste trabalho, a saber:

- Dentre tantos trabalhos existentes sobre Metodologias de Desenvolvimento de Software, são poucos os que enfocam sua qualidade;

- Nenhum dos trabalhos consultados preocupa-se com a quantificação da qualidade da metodologia, e/ou do software, sob o ponto de vista do usuário.

- Existem poucos dados sobre a influência do uso de uma metodologia de desenvolvimento na qualidade do software;

- Nenhum dos trabalhos consultados estabelece um processo sistematizado para o cálculo da qualidade da metodologia.

1.3. **OBJETIVO**

O objetivo principal desta dissertação, é propor um modelo para quantificar a influência que a utilização de uma metodologia de desenvolvimento exerce sobre a qualidade do software.

Além disso, será realizada a aplicação prática do modelo proposto.

1.4. IMPORTÂNCIA DO TRABALHO

A maioria dos autores da área de informática, afirma que a utilização de uma metodologia de desenvolvimento provoca uma melhoria da qualidade do software. Como já foi ressaltado, existem poucos dados práticos que comprovem a relação mencionada.

A importância do presente trabalho situa-se no fato de poder indicar um caminho a ser seguido para se quantificar a relação entre metodologia e qualidade de software. Além disso, serão apresentados resultados preliminares que permitem tecer algumas considerações sobre a relação mencionada.

O conhecimento da relação entre o uso da metodologia e a qualidade do software permite prever, com antecedência, os benefícios originados a partir da implantação e/ou modificação de uma dada metodologia. Além disso, a aplicação deste trabalho propiciará as seguintes vantagens:

- Melhorar o relacionamento analista/usuário. Isto porque, as divergências entre analistas e usuários ficarão evidentes, podendo assim ser implementadas ações para amenizar seus efeitos ou eliminá-las.

- Redução dos recursos necessários para a modificação de uma dada metodologia. Os resultados obtidos com a aplicação deste trabalho identificarão as principais deficiências da metodologia utilizada;

- Identificação do grau de satisfação do usuário com relação ao software desenvolvido e à documentação associada. Isto facilitará o direcionamento dos esforços da empresa no sentido de melhorar a qualidade do software;

- Indicação do grau de satisfação dos analistas em relação aos aspectos técnicos da metodologia, caracterizando possíveis pontos de mudança.

1.5. METODOLOGIA DE TRABALHO

Com o intuito de se atingir os objetivos mencionados anteriormente, foram realizadas uma pesquisa teórica e uma pesquisa de campo.

A pesquisa teórica consistiu de uma revisão da literatura especializada, tanto nacional quanto estrangeira. Esta pesquisa envolveu consulta a livros, revistas, jornais, dissertações de Mestrado e teses de Doutorado.

A pesquisa de campo envolveu a aplicação de um questionário junto ao cliente, para que fosse possível a quantificação dos fatores de qualidade, tanto do software quanto da documentação entregue. Além disso, foram quantificados os fatores de qualidade da metodologia, através de um questionário enviado aos analistas programadores e consultores.

1.6. LIMITAÇÕES

O presente trabalho procurou ser o mais abrangente possível. Entretanto, por questões práticas, os clientes pesquisados limitaram-se à empresa NTS - Unidade Florianópolis, reduzindo assim o número de entrevistados e limitando a generalização dos resultados obtidos.

1.7. ESTRUTURA

Para apresentar o trabalho descrito anteriormente, em linhas gerais, a dissertação será dividida em 3 partes, a saber:

PARTE 1: Enfocará "O QUE" será feito:

Esta primeira parte terá os seguintes capítulos:

CAPÍTULO 1 - Fornece o objetivo, o escopo, a estrutura e as limitações do trabalho.

CAPÍTULO 2 - Será feita a conceituação de alguns termos importantes para a compreensão do trabalho apresentado.

CAPÍTULO 3 - Descrição do ambiente de desenvolvimento de software, enfatizando os principais fatores envolvidos: metodologia, recursos humanos e gerência. Além disso, serão enfatizados os aspectos gerenciais e humanos que afetam a qualidade do software

PARTE 2: Enfocará o "**COMO**" será feito:

CAPÍTULO 4 - Apresentação de um modelo para quantificar a influência da utilização de uma metodologia sobre a qualidade do software.

CAPÍTULO 5 - Apresentação e análise dos resultados práticos obtidos através da aplicação do modelo proposto.

PARTE 3: Análises e Conclusões.

CAPÍTULO 6 - Conclusões e Recomendações

CAPÍTULO 2

2. SOFTWARE E ENGENHARIA DE SOFTWARE

2.1. INTRODUÇÃO

Com o aumento da concorrência, as empresas intensificaram a utilização do computador (hardware) para a realização das mais diversas atividades. Os computadores são utilizados com grande sucesso desde a emissão de folhas de pagamentos até controles de mísseis, fornecendo resultados mais confiáveis e num tempo muito menor em relação a mesma tarefa executada manualmente.

A diversificação de atuação dos computadores começou a preocupar a comunidade científica, uma vez que a sociedade está cada vez mais dependente dos resultados fornecidos pelos mesmos. Como a exatidão destes resultados depende diretamente do software utilizado, os pesquisadores iniciaram estudos sobre o processo de desenvolvimento do mesmo.

Estes motivos resultaram no surgimento da Engenharia de Software, durante a "NATO-GARMISH SOFTWARE CONFERENCE"(1968).

Segundo Teixeira [Texeira, 1979], o termo Engenharia de Software é inadequado pois, o software não se adapta às regras normais de engenharia, principalmente no que diz respeito a aproximações e a coeficientes de segurança.

Deve-se salientar, no entanto, que o termo " Engenharia" é utilizado para enfatizar a necessidade de se formalizar o processo de desenvolvimento de software, em oposição ao processo artesanal utilizado anteriormente. Neste sentido, a Engenharia de Software é definida como sendo "***uma abordagem sistemática para o desenvolvimento, operação, manutenção e substituição de software***" [American National Standards Institute/Institute of Electrical and Eletronics Engineers - ANSI/IEEE, 82].

Uma outra definição de Engenharia de Software é dada por Goldberg [GOLDBERG,1986], a saber:

"***O estabelecimento e uso de sólidos princípios de engenharia (Métodos) para obter, de maneira econômica, software que seja confiável e que funcione em máquinas reais***".

As duas definições demonstram uma preocupação com relação ao uso de métodos formais de desenvolvimento de software, a fim de se obter um produto de melhor qualidade.

Segundo Duclós [DUCLÓS,82], a Engenharia de Software possui os seguintes objetivos:

- Gerenciar com sucesso o processo de desenvolvimento de software, satisfazendo as necessidades humanas da equipe de desenvolvimento e controlando os custos de processo;

- Desenvolver com sucesso um software orientado para o usuário e com controle de qualidade do produto.

Para que se tenha uma maior compreensão do que seja Engenharia de Software, é necessário que se defina o que se entende por software. Em muitos casos, confunde-se software com programa.

Entende-se software como sendo "***um conjunto de programas, procedimentos, regras e qualquer documentação associada com a operação de um sistema de processamento de dados***" [ANSI/IEEE, 82].

BrooKs [BROOKS,75] define quatro casos possíveis, enfatizando a diferença entre software e programa. Os quatro casos são:

I - PROGRAMA: Conjunto de instruções que serão executadas pelo computador e é escrito para resolver um problema específico possuindo baixa complexidade.

II - SISTEMA: Programa que se relaciona com outros, por isso deve possuir recursos de memória e precisa ser testado com os outros programas em diferentes combinações.

III - PROGRAMA PRODUTO: Programa que deve ser executado, testado, mantido, modificado e redocumentado por outros que não o desenvolveram. Deve ser muito bem documentado e testado. Além disso, deve possuir a seção de entrada de dados bem geral.

IV - SOFTWARE PRODUTO: Combinação dos casos II e III, caracterizando-se por possuir um alto grau de generalização e de complexidade.

Comparando-se os quatro casos, citados anteriormente, com relação à complexidade e ao grau de generalização, teremos a situação mostrada pela figura 1.

A relação entre os custos de desenvolvimento de cada um dos casos apresentados, é visualizada na figura 2. Por esta figura, observa-se que o software produto (IV) custa nove vezes mais do que um programa (I).

Observa-se, no entanto, que Brooks só considera como software aquele produto que tenha sido testado (executado ou mantido) por outros que não o desenvolveram. Concordar com isso, significa aceitar que a definição de um produto está vinculada ao seu processo de desenvolvimento. Seria como definir "AUTOMÓVEL" como sendo um veículo para transporte terrestre de passageiros, movido a motor e que tenha sido produzido em uma linha de montagem. Evidentemente que esta definição não esta correta, pois um produto deve ser definido em função de suas partes constituintes, e não em função de seu processo de produção.

A definição de Brooks é importante por distinguir software de programa. A definição adotada neste trabalho, por ser mais adequada, será aquela estabelecida pela ANSI/IEEE.

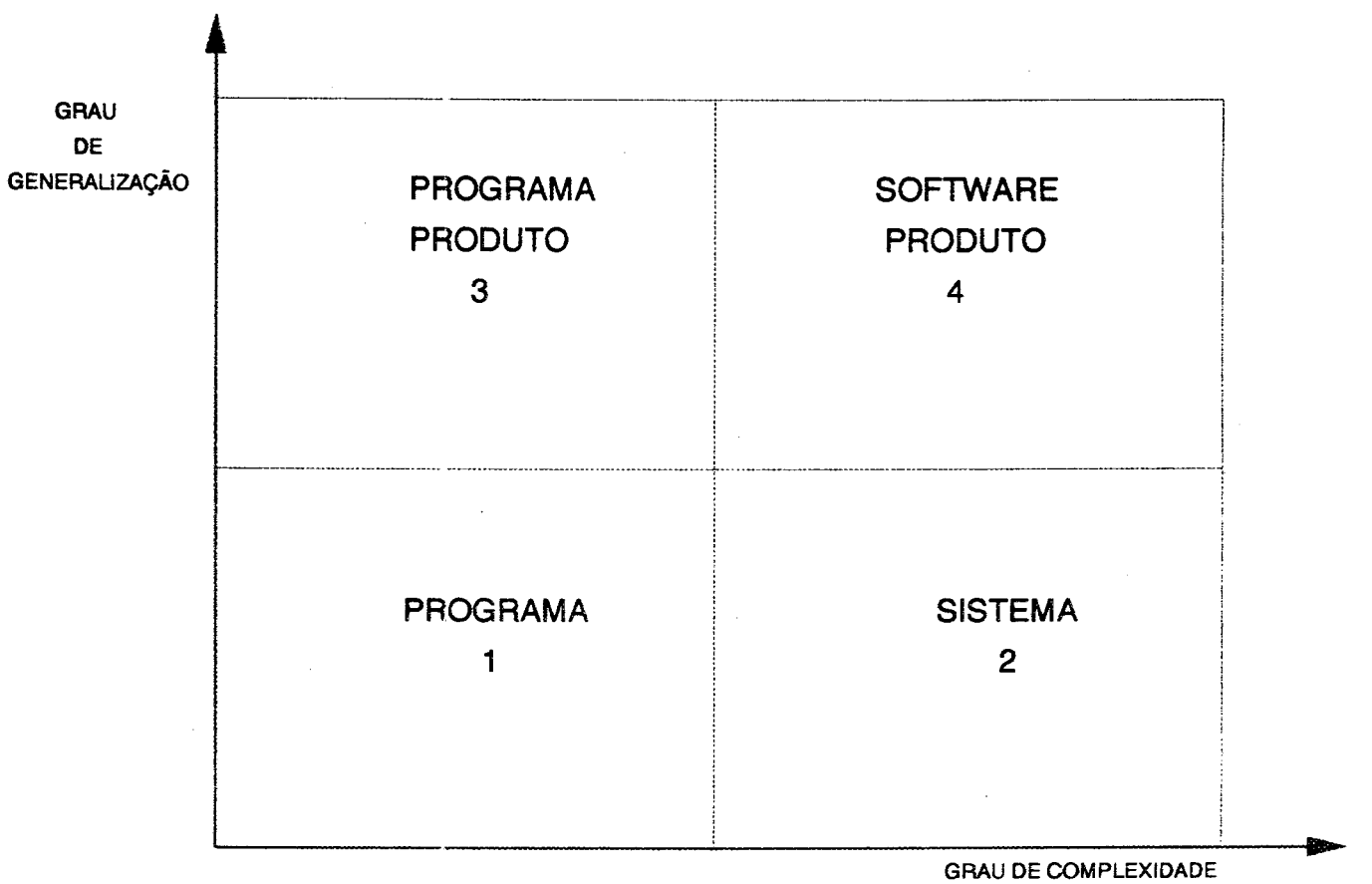

FIGURA 1 - DIFERENÇA ENTRE SOFTWARE E PROGRAMA

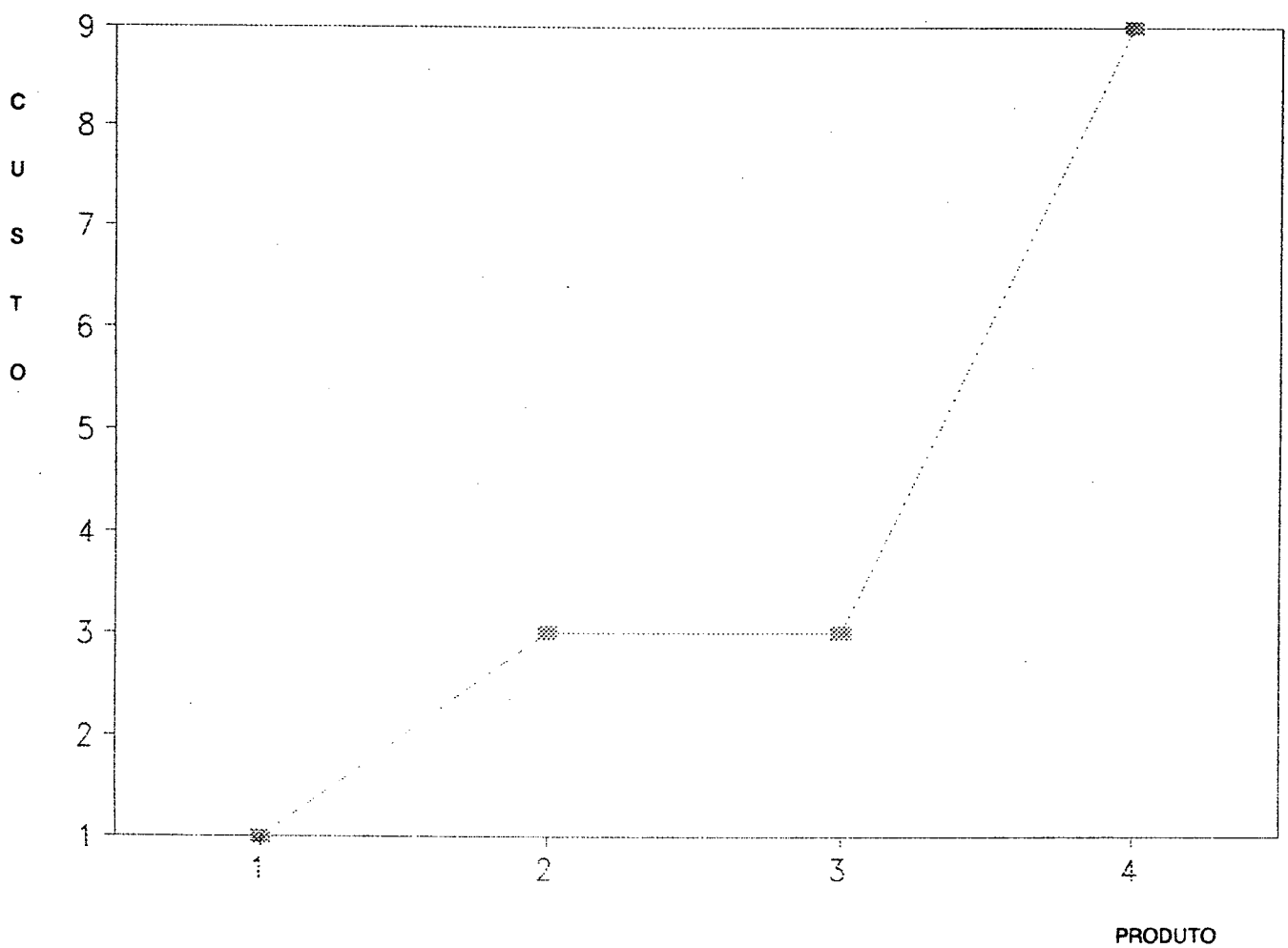

FIGURA 2 - RELAÇÃO ENTRE CUSTO DE PROGRAMA E CUSTO DE SOFTWARE

2.2. CARACTERÍSTICAS DO SOFTWARE

O software pode ser considerado um produto como outro qualquer, pois é desenvolvido para satisfazer as necessidades de um dado cliente. Entretanto, o software possui algumas características que o diferencia dos demais produtos.

Em primeiro lugar, devido ao fato de que o software possui características lógicas, o cliente não consegue identificar se o produto que está sendo desenvolvido é o que ele deseja. Por exemplo, no projeto de uma casa, o cliente consegue visualizar como ela ficará depois de construída, ou seja, dentre tantos aspectos como será a fachada ou a divisão interna. Isto não acontece no caso do software, pois o cliente tem dificuldade em relacionar os diagramas e fluxogramas do produto em desenvolvimento com o produto final.

Além desta característica, pode-se citar outras:

- TESTE: É impossível testar todos os caminhos possíveis de um software, pois o número é tão grande que seriam necessários vários anos para testar apenas trechos de programas [TEIXEIRA,79]. Medidas realizadas com programas supostamente corretos, forneceram os resultados apresentados na tabela 1. Estes resultados mostram que em programas já testados, mais de 50% dos comandos nunca haviam sido executados. Isto implica numa grande possibilidade de que o software venha a falhar durante sua operação.

- CORREÇÃO: Com a correção dos defeitos apresentados durante o desenvolvimento, seria natural esperar uma diminuição na taxa de falhas do software (figura 3).

  Na realidade o que ocorre é que com a correção dos erros, pode haver um aumento na taxa de falhas, como mostra a figura 4. Isto ocorre, pois a correção de um erro pode ter como consequência a execução de comandos, que ainda não haviam sido executados e que podem possuir erros.

COMAN-DOS / MEDIDAS	COMANDOS SIMPLES	COMANDOS DE TESTE E DESVIO	CHAMADA DE ROTINAS
NÚMERO TOTAL	24.491	13.585	2.019
EXECUTADO PELO MENOS UMA VEZ	44.50%	35.10%	37.2%
NUNCA EXECUTADOS	55.50%	64.90%	62.80

TABELA 1 -PERCENTAGEM DE EXECUÇÃO DE COMANDOS NO TESTE DE PROGRAMAS FORTRAN

– MANUTENÇÃO: Ao contrário de outros produtos (como carros, elétrodomésticos e aparelhos eletrônicos), o software apresenta muitos defeitos após ser entregue ao usuário. Além disso, as partes com defeitos não podem ser substituídas por novas, como ocorre com outros produtos. A necessidade de se desenvolver novamente as partes com defeitos, faz com que a manutenção do software seja tão caro quanto o seu desenvolvimento. Esta situação é demonstrada pela figura 5.

– PREVENÇÃO: Em muitos produtos é possível fazer uma revisão de certos itens para evitar futuros problemas. A revisão do automóvel e o check-up de saúde são exemplos dessa situação. Esta prevenção não é possível com relação ao software, pois ele não emite indícios, ou sinais, de que irá falhar no futuro.

– UTILIZAÇÃO: O software não se desgasta com o uso. Independente do número de vezes, em que for utilizado, o software fornecerá, nas mesmas condições, os mesmos resultados. No carro, por exemplo, depois de algum tempo de uso, existirão algumas

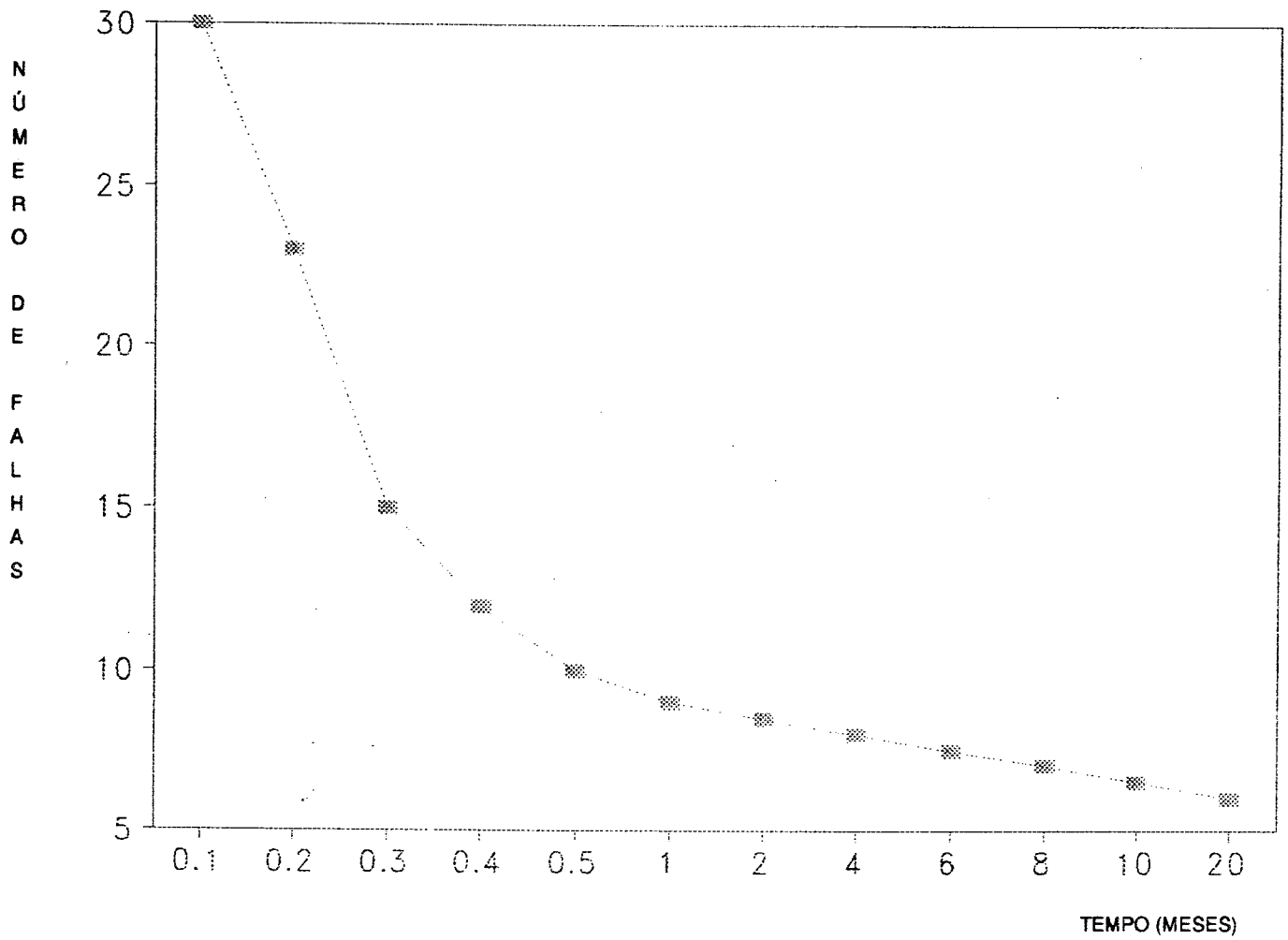

FIGURA 3 - CORREÇÃO DE FALHAS EM SOFTWARE - SITUAÇÃO IDEAL

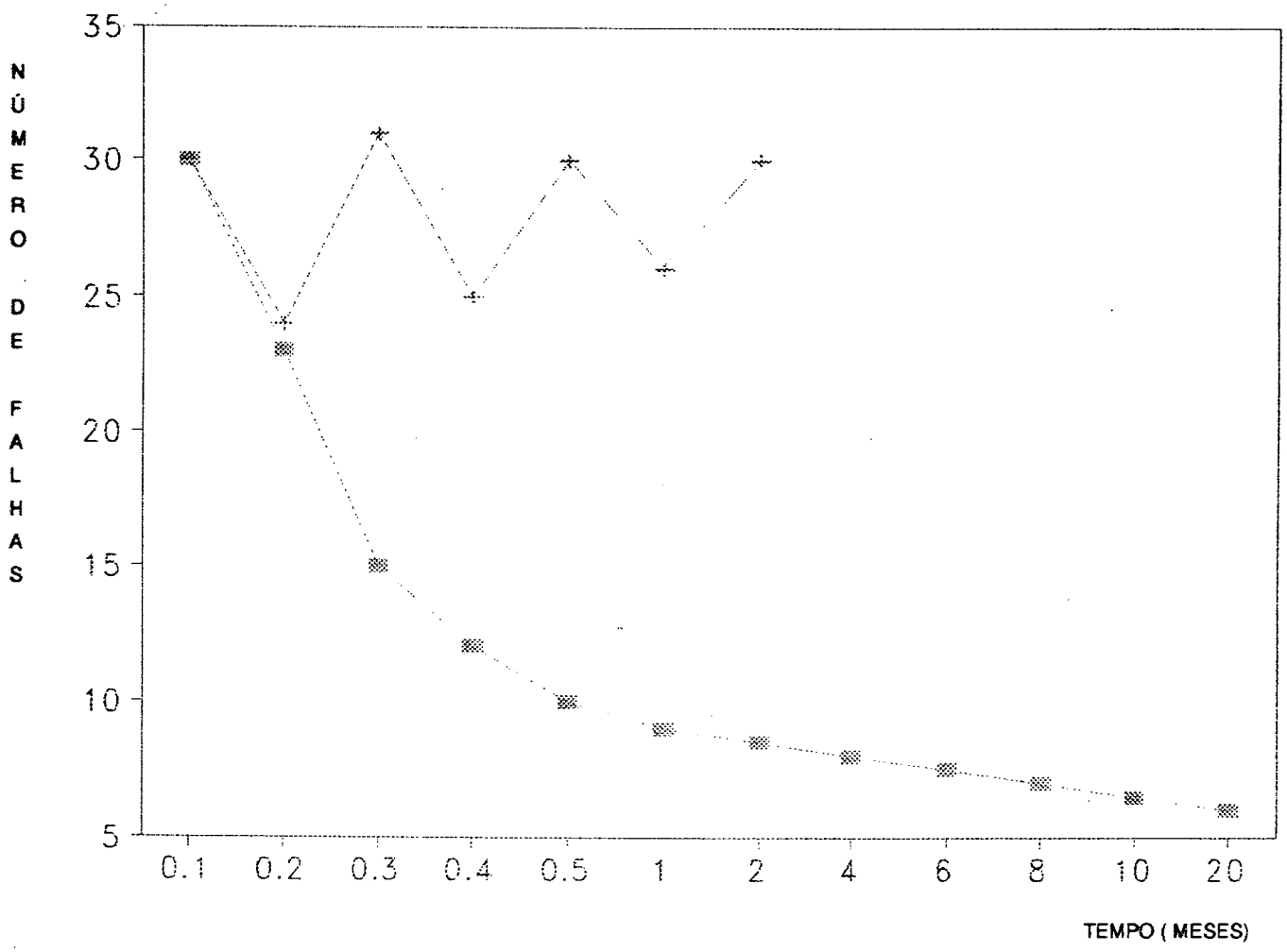

FIGURA 4 - CORREÇÃO DE FALHAS EM SOFTWARE - SITUAÇÃO IDEAL x SITUAÇÃO REAL

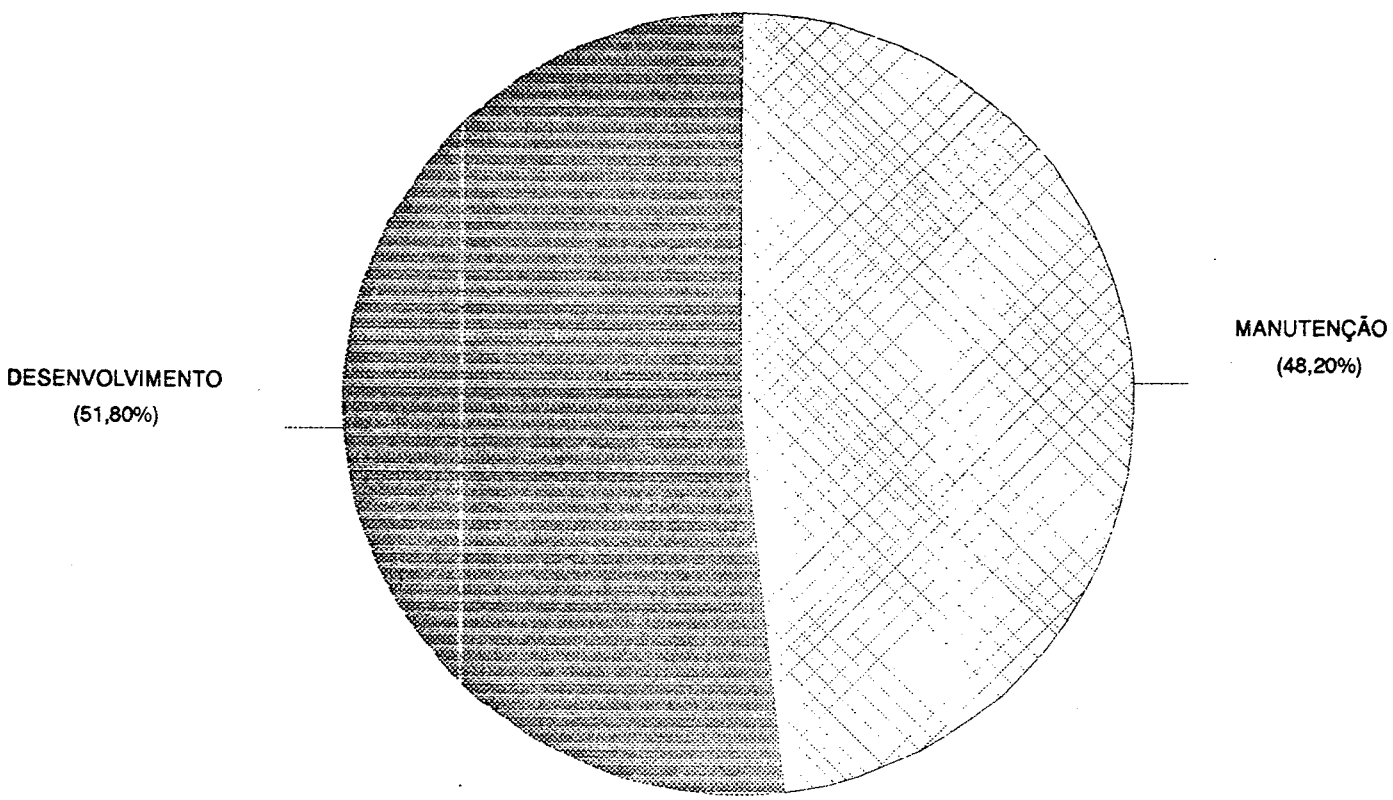

FIGURA 5 - PARTICIPAÇÃO DA MANUTENÇÃO NO CUSTO TOTAL DO SOFTWARE

peças desgastadas e que precisarão ser trocadas. Isso não acontece com o software.

2.3. CRISE DO SOFTWARE

Apesar dos grandes avanços conseguidos na área de software, existem muitos problemas com seu processo de desenvolvimento. Em primeiro lugar, o software demora muito tempo para ser desenvolvido, podendo levar alguns anos, e em muitos casos não atendendo as necessidades dos clientes.

Segundo DeMarco [DeMarco,89], 15% de todos os projetos de software são completamente incapazes de atingir as metas propostas.

Além disso, o custo de um software de grande porte é astronômico. Segundo Boehm, o "0S/360" da IBM custou Us$ 200.000.000,00 [Boehm,81]. Este alto custo é apenas uma das questões do que se convencionou chamar "Crise do Software". Este termo diz respeito a problemas relacionados com o processo de desenvolvimento de software.

Um outro aspecto da chamada "Crise do Software", é que, geralmente, o tempo requerido para o desenvolvimento do software é maior que o prazo estabelecido no início do projeto. Um dos motivos para que isto ocorra é o excesso de otimismo característico do pessoal que desenvolve software. Além disso, acredita-se que o tempo de desenvolvimento é inversamente proporcional ao número de pessoas envolvidas.

Baseado neste princípio, é comum adicionar analistas/progamadores a uma equipe de desenvolvimento, para que um projeto atrasado em relação ao seu cronograma, possa ser entregue no prazo pré-estipulado.

Segundo Brooks [Brooks,75], só é possível estabelecer uma relação proporcional entre tempo de desenvolvimento e número de trabalhadores, quando se elimina o processo de comunicação entre as pessoas responsáveis pelas diferentes sub-tarefas. Estas sub-tarefas são resultantes da divisão da tarefa global a ser realizada. A figura

6 mostra a relação entre número de trabalhadores e tempo de desenvolvimemto, para este caso.

No caso do software, cada parte de uma dada tarefa deve ser separadamente coordenada com as outras partes, aumentando em muito a necessidade de comunicação entre os membros da equipe. Com isso, o tempo ganho com a partição da tarefa é gasto em comunicação entre os membros da equipe. Isto conduz à situação da figura 7, onde o ganho inicial de tempo é devido à partição da tarefa.

Os problemas relacionados com a crise do software são agravados pelo fato de que a produtividade dos programadores/analistas não acompanha o aumento da demanda por software. A figura 8 mostra o crescimento da demanda por software em milhões de instruções geradas, através de quatro gerações do programa americano de vôo espacial tripulado. O aumento no número de instrucões indica um aumento no número das atividades automatizadas.

Além da baixa produtividade dos programadores/analistas, a qualidade dos softwares desenvolvidos, em muitos casos, é menor que a desejada.

Todos estes problemas citados, não poderão ser resolvidos de uma só vez. Serão necessários novos métodos e técnicas, para que seja possível melhorar a qualidade e a produtividade de software.

2.4. QUALIDADE DO SOFTWARE

Cada vez mais, as empresas estão investindo na qualidade como forma de diferenciação de seu produto no mercado. Isto faz com que o termo qualidade seja usado diariamente, tanto em propagandas quanto em conversas informais. Por este motivo, a qualidade é definida segundo os interesses e ponto de vista, de quem está definindo.

Segundo Garvin [in Velasquez,87 e Paladini,90], o termo qualidade é um conceito dinâmico e se modifica em função da atividade que estiver sendo desenvolvida (concepção, projeto, fabricação e comercialização do produto). Devido a estas

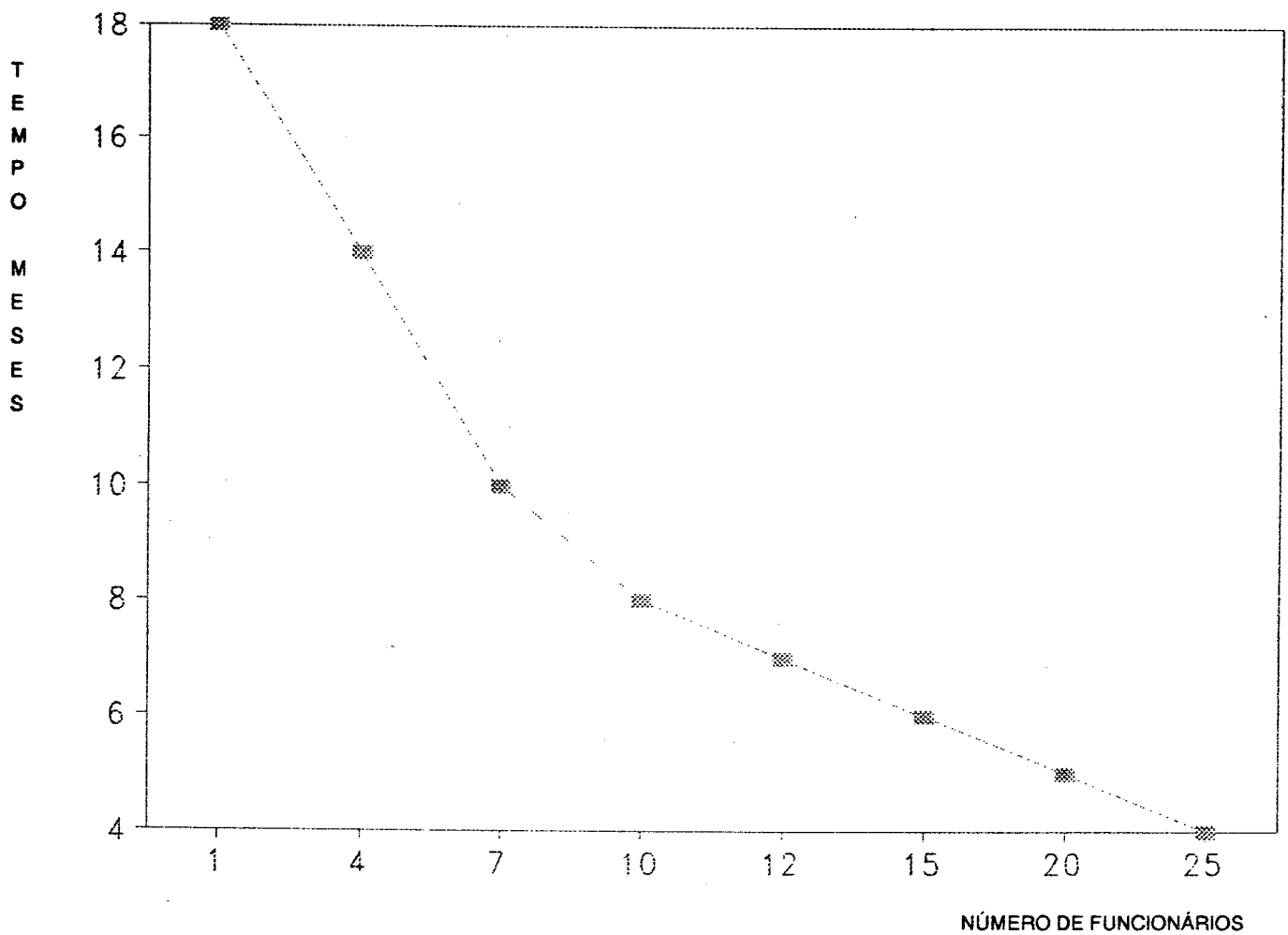

FIGURA 6 - RELAÇÃO ENTRE NÚMERO DE TRABALHADORES E TEMPO DE DESENVOLVIMENTO - SEM COMUNICAÇÃO ENTRE AS SUBTAREFAS

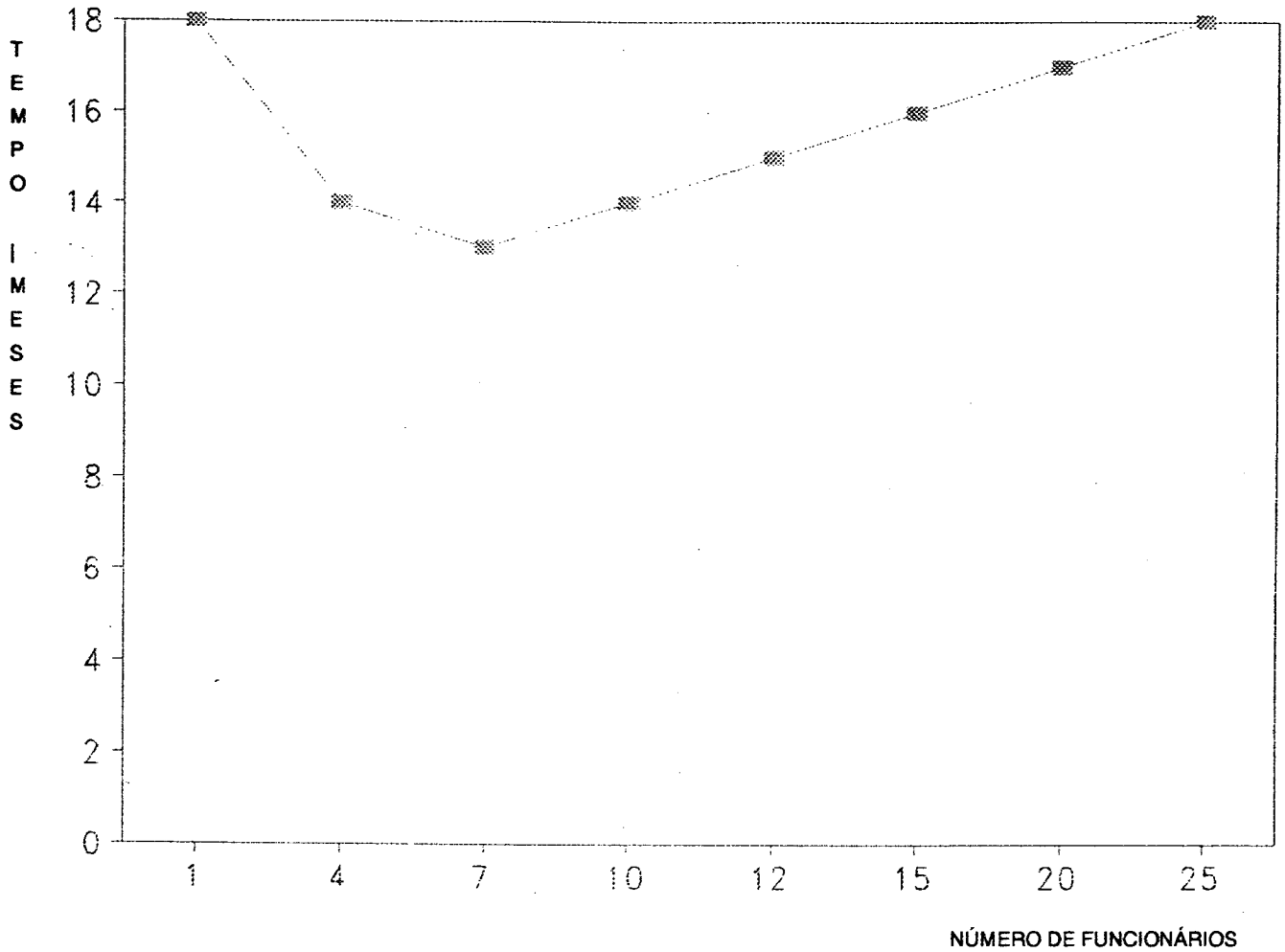

FIGURA 7 - RELAÇÃO ENTRE NÚMERO DE TRABALHADORES E TEMPO DE DESENVOLVIMENTO - COM COMUNICAÇÃO ENTRE AS SUBTAREFAS

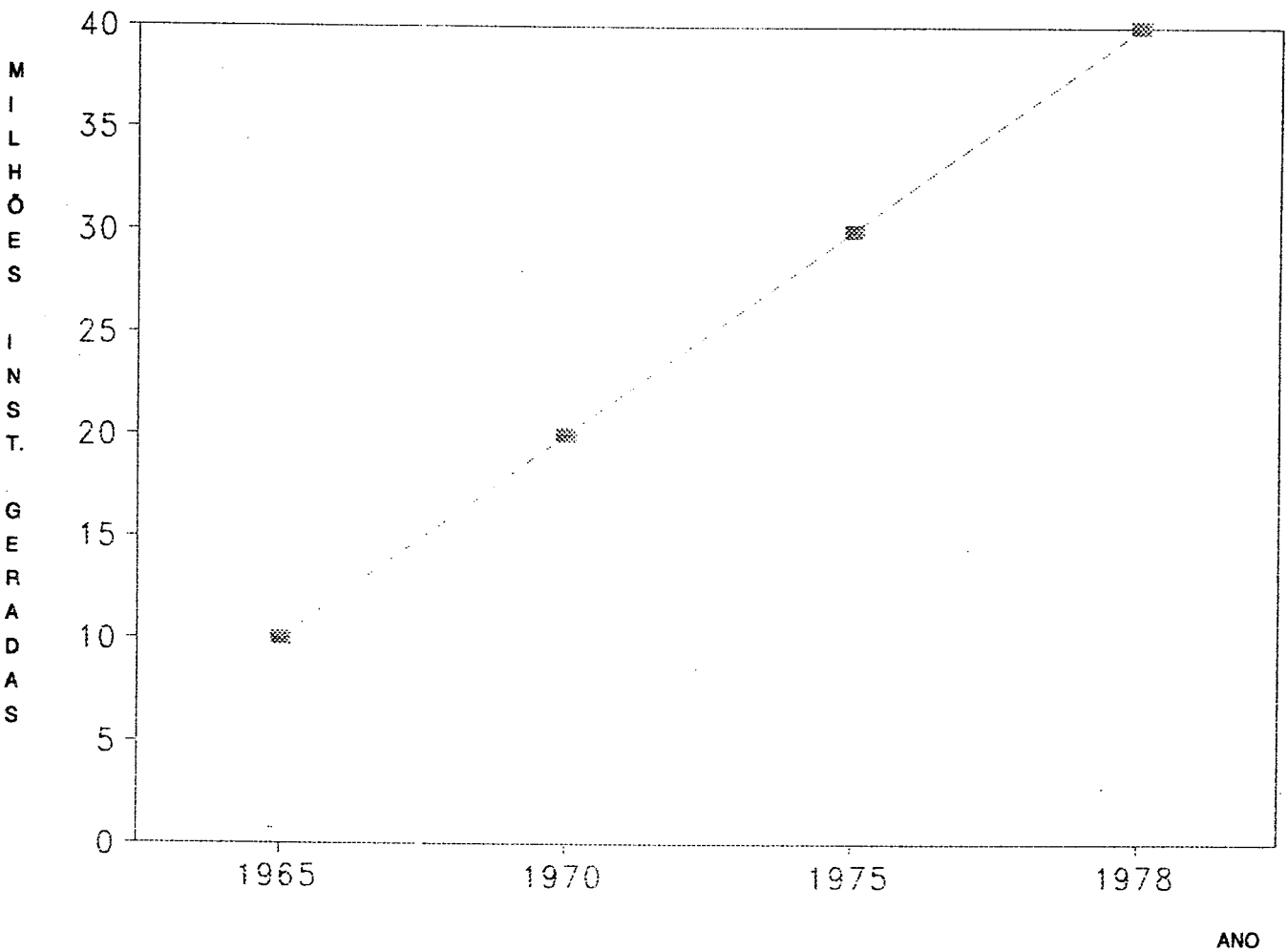

FIGURA 8 - CRESCIMENTO DA DEMANDA POR SOFTWARE

características, Garvin indentificou cinco abordagens gerais para definir qualidade, a saber:

1 - Abordagem Transcedental: Segundo esta abordagem, a qualidade não pode ser definida com precisão e a boa qualidade só é reconhecida com base na experiência. Um exemplo de definição baseada nesta abordagem é a de Pirsig (1974):

"***A qualidade não é pensamento nem matéria, mas uma terceira entidade indepedente das duas... ainda que a qualidade não possa ser definida, você sabe que ela existe***".

2 - Abordagem Baseada no Produto: a qualidade é considerada uma variável precisa e mensurável, sendo que as diferenças na qualidade refletem diferença na quantidade de alguns elementos ou atibutos, possuídos pelo produto. Este ponto de vista pode ser notado na definição de LEFFLER (1982):

"***Qualidade refere-se às quantidades de atributos inestimáveis contidos em cada unidade de atributo estimado***".

3 - Abordagem Baseada no Usuário: Esta abordagem considera o usuário como avaliador da qualidade do produto, ou seja, um produto só é considerado de boa qualidade se for bem aceito pelo usuário. Segundo JURAM (1974):

"Qualidade é a satisfação das aspirações do cliente", "Qualidade é adequação ao uso".

4 - Abordagem baseada na fabricação: Nesta abordagem a preocupação é produzir um produto de acordo com suas especificações básicas, determinadas durante o projeto. Um exemplo desta abordagem é a definição de P. B. CROSBY (1979):

"Qualidade é a conformidade do produto às suas especificações".

5 - Abordagem Baseada no Valor: Um produto é considerado de boa qualidade se oferece um bom desempenho a um preço aceitável.

> ***"Qualidade é o melhor possível sob certas condições do consumidor. Estas condições são referentes ao uso real e ao preço de venda do produto"*** (Feingen Baun/61).

Apesar da relevância dos cinco enfoques apresentados anteriormente, a qualidade será analisada, no presente trabalho, sob dois pontos de vista:

1 - Visão Externa: Esta visão relaciona-se com o grau de satisfação do usuário em relação ao produto. Isto significa que o produto deve ser bem aceito pelo usuário, tendo um preço compatível com o poder aquisitivo do público-alvo. Deve-se observar que esta visão engloba as abordagens baseada no usuário, no valor e a transcedental.

2 - Visão Interna: A filosofia desta visão enfatiza o produto que atenda a normas, especificações e padrões de desempenho, estabelecidos pela empresa. Esta visão engloba as abordagens baseada no produto e na fabricação.

A relação entre as abordagens descritas por Garvin e as duas visões citadas anteriormente, pode ser vista na Figura 9. Esta figura mostra também, que as visões interna e externa sã interdependentes. Isto ocorre, porque em função da percepção do usuário, pode-se alterar padrões e normas, ou mesmo criar novos padrões, de forma a adaptar o produto às aspirações do cliente. Por outro lado, a definição de padrões de qualidade pode aumentar a satisfação do cliente com relação ao produto ou serviço desenvolvido.

A qualidade poderia ser, desta forma, tanto a satisfação de exigências internas (padrões e normas do produtor), quanto de externas (necessidades e solicitações dos clientes).

Enfocar a qualidade da maneira como foi proposto neste trabalho, traz algumas vantagens sobre as abordagens individuais citadas por Garvin:

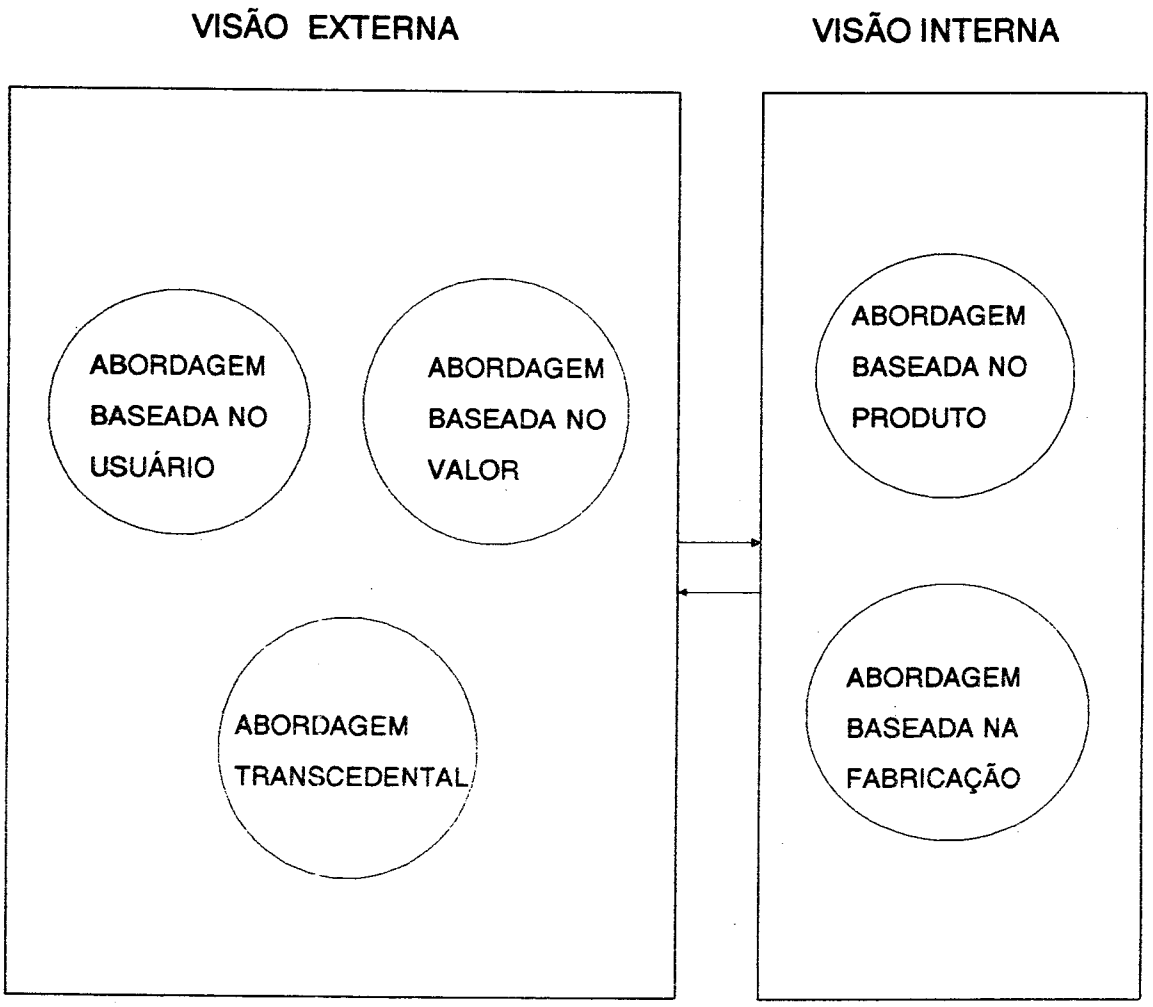

FIGURA 9 - RELAÇÃO ENTRE AS ABORDAGENS DE QUALIDADE

– As necessidades e aspirações dos clientes são sempre levadas em consideração.

– Os padrões e normas internas são repeitados como forma de satisfazer os clientes

– Melhor comprensão do que seja qualidade pelos componentes da empresa. Isto ocorre pois, esta abordagem descreve claramente os interesses envolvidos no desenvolvimento de um produto ou serviço.

No caso do software, todas as pessoas envolvidas com o desenvolvimento do produto devem atentar para as duas visões citadas. Em função disso, um software só será considerado de boa qualidade se for aprovado pelo produtor e pelo usuário. Este é um importante passo na implantação de um efetivo programa para a melhoria da qualidade do software.

2.5. CONTROLE DE QUALIDADE DE SOFTWARE

Faz parte do senso comum que toda empresa deve possuir um rigoroso sistema de controle de qualidade.

– Até que ponto esta afirmação é verdadeira?

– Porque implantar um programa de controle de qualidade?

Basicamente, existem três motivos que levam as empresas a implantarem um sistema de controle de qualidade, a saber:

– Para aumentar a competitividade dos produtos da empresa no mercado;

– Por exigências contratuais de outras empresas;

– Por razões técnicas: alto custo de assistência técnica, alta taxa de refugo e retrabalho, alto índice de reclamação de clientes, etc.

Devido às características já citadas anteriormente, as razões técnicas parecem ser o principal motivo para que uma empresa de software inicie um programa de controle da qualidade de seu produto.

Inicialmente, o controle de qualidade era realizado pela própria pessoa que desenvolvia o produto, não sendo realizado como uma ação formal a ser executada separadamente. Este procedimento consistia simplesmente em verificar se o produto atingia o padrão de qualidade do artesão e se correspondia ao que o cliente havia encomendado. Esta situação era análoga ao que ocorreu nas primeiras fases do desenvolvimento de software, onde a qualidade do produto dependia exclusivamente da habilidade e dos padrões pessoais de qualidade do programador.

A Segunda Guerra Mundial motivou pesquisas de processos de inspeção mais eficientes e de menor custo. Esta necessidade conduziu à adoção de conceitos estatísticos no controle do processo de produção, sendo definidos então os critérios de aceitação e rejeição. Este tipo de controle ficou conhecido como Controle Estatístico de Qualidade. A aplicação desse controle tornou-se possível devido à padronização do processo de produção.

Mesmo com a utilização de ferramentas estatísticas, o controle de qualidade continuava a ser apenas um processo de inspeção, pois não alterava a qualidade do produto. Isto porque os resultados da qualidade fornecidos pelas ferramentas, apenas retratavam o nível atual de qualidade. O controle de qualidade caracteriza-se portanto, pela utilização dos dados relativos à qualidade do produto para a sua própria melhoria e para a prevenção de defeitos futuros.

Com base no que foi exposto, o controle de qualidade pode ser definido como sendo ***"um sistema dinâmico e complexo, que abrange todos os setores da***

empresa - de forma direta ou indireta - com o objetivo de melhorar a qualidade do produto final e manter essa melhoria, operando em níveis economicamente aceitáveis". [PALADINI,90].

Apesar de resultar em inegáveis vantagens, vide tabela 2, existem muitos obstáculos à implantação de um programa de controle de qualidade.

Segundo Velasco [VELASCO, 88], o principal obstáculo é a noção equivocada de que a melhoria da qualidade exige altos investimentos. Isto ocorre porque é difícil obter os custos relacionados com a não-qualidade, ou seja, não se tem noção do quanto está sendo desperdiçado com mão-de-obra e tempo de máquina para a obtenção de um produto de má qualidade.

Para que seja possível a instalação de um programa efetivo de controle de qualidade é necessário que esteja bem claro o objetivo de qualidade a ser atingido.

Nesse sentido, o presente trabalho será de grande importância, uma vez que identificará as expectativas dos clientes. Estas expectativas deveriam fazer parte dos objetivos de qualidade da empresa.

Para a definição do objetivo de qualidade, é preciso estabelecer o nível mínimo de qualidade desejado. Nesse sentido, Staa [STAA,83] define software de qualidade como sendo aquele que produz resultados úteis e confiáveis na oportunidade certa; é ameno ao uso; é mensurável e auditável; é corrigível e modificável, opera em máquinas e ambiente reais; foi desenvolvido de forma econômica e no prazo estipulado; e opera com economia de recursos.

Em uma empresa que invista em qualidade, é necessário que se atente para os três fatores [ARTHUR ,85] que afetam tanto a qualidade quanto a produtividade de software.

Esses fatores são:

- Recursos Humanos;

VANTAGENS DO CONTROLE DE QUALIDADE
- MELHORIA NA QUALIDADE DO PRODUTO
- REDUÇÃO DOS CUSTOS DE FABRICAÇÃO
- REDUÇÃO DAS PERDAS POR RETRABALHO
- REDUÇÃO NOS PRAZOS DE ENTREGA
- MELHORIA NO MORAL DOS FUNCIONÁRIOS
- MAIOR PREVISIBILIDADE DO PROCESSO
- TAXAS ESTÁVEIS, E MAIORES, DE PRODUTIVIDADE

TABELA 2 - VANTAGENS DO CONTROLE DE QUALIDADE

- Gerência;

- Metodologia.

A interação dos fatores, citados anteriormente, dentro do ambiente de desenvolvimento de software, certamente afeta a qualidade do software desenvolvido.

O objetivo do presente trabalho é exatamente este: propor um modelo para quantificar a influência da metodologia sobre a qualidade do software.

No capítulo 3, os três fatores serão comentados isoladamente embora, na prática, seja difícil analisá-los dessa maneira. Tais descrições separadas serão feitas apenas como base de sustentação do modelo a ser proposto.

CAPÍTULO 3

3. AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

3.1. INTRODUÇÃO

Num ambiente de desenvolvimento de software, conforme comentado no capítulo anterior, existem vários fatores que afetam a qualidade do software desenvolvido.

Além da descrição de cada fator em separadamente, será comentada, quando aplicável, a influência de um dado fator sobre os demais e sobre a qualidade do software.

Como o objetivo principal do presente trabalho, é quantificar a influência da utilização de uma metodologia sobre a qualidade do software, o fator METODOLOGIA será descrito com maiores detalhes.

3.2. METODOLOGIA

A metodologia de desenvolvimento de software consiste num conjunto de métodos e técnicas, que tem como objetivo estabelecer os procedimentos necessários para a concepção, desenvolvimento, implantação e manutenção de software.

Uma das grandes vantagens em se utilizar uma metodologia é a formalização do processo de desenvolvimento. Esta formalização propicia a implantação de um eficiente controle de qualidade, uma vez que os marcos principais do projeto estarão definidos.

Outra vantagem é a melhoria da comunicação entre os analistas/programadores e usuários, o que será detalhado mais adiante,com reflexos positivos sobre a qualidade do software.

Desde a década de 70, foram sugeridas muitas metodologias, com o objetivo de se minimizar os problemas relacionados ao desenvolvimento de software.

Apesar das diferentes abordagens destas metodologias, todas elas entendem o ciclo de vida do software como sendo constituído de três grandes fases:

DEFINIÇÃO - Enfatiza "o que" deve ser realizado.

PROJETO - Especifica "como" o software deverá ser desenvolvido para satisfazer as definições estabelecidas.

MANUTENÇÃO - Focaliza as mudanças que serão necessárias para a correção de erros, adaptação e modificações visando a melhoria do software.

As três etapas descritas e as interações que ocorrem entre elas estão demonstradas na figura 10 [PRESSMAN,82].

O número de fases do ciclo de vida depende das subdivisões a que forem submetidas as etapas descritas acima.

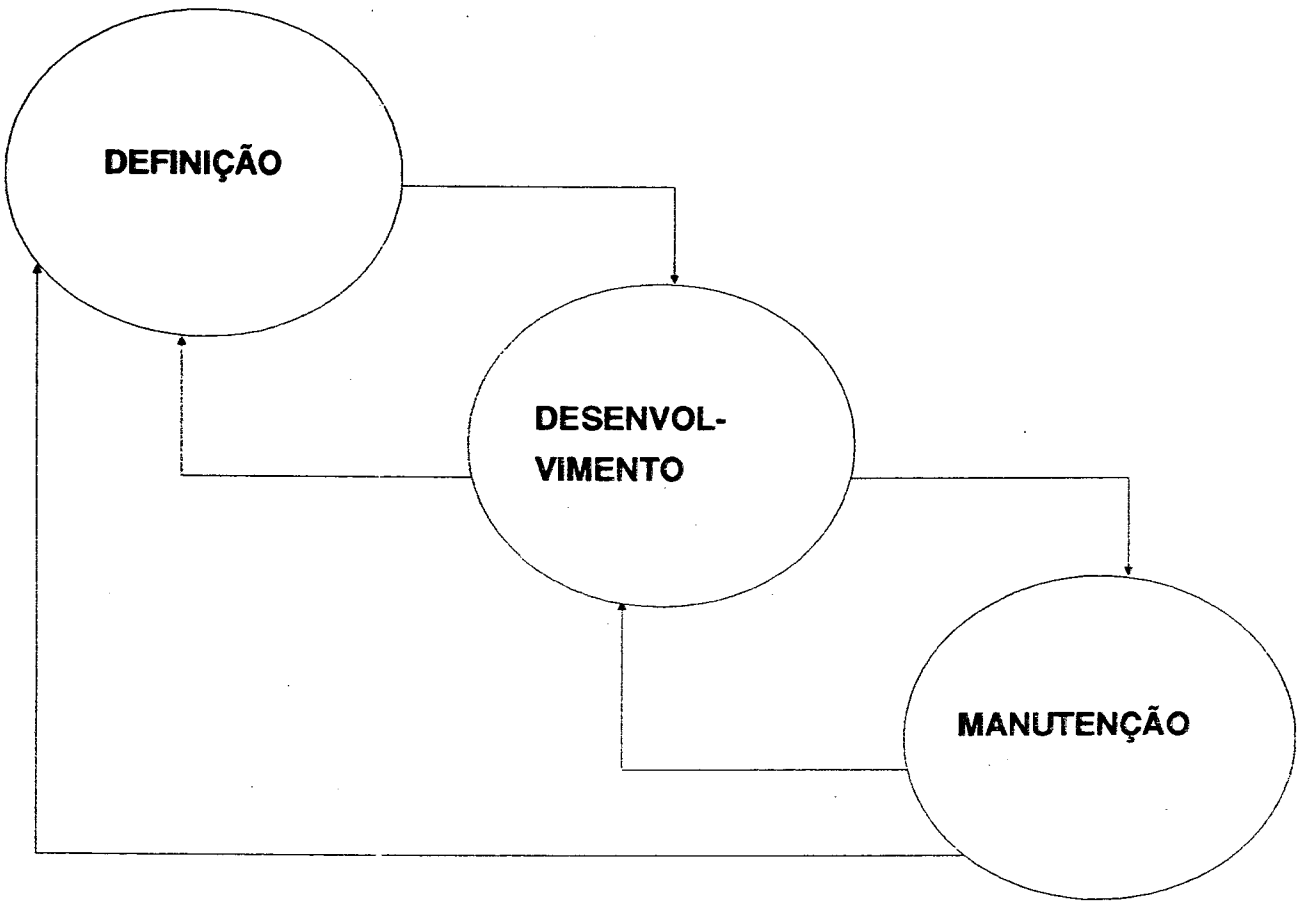

FIGURA 10 - VISÃO GERAL DO DESENVOLVIMENTO DE SOFTWARE

Independente do número de fases existentes, é essencial que se defina o ciclo de vida do software. A importância para tal fato centraliza-se nos seguintes postulados:

- Definição das atividades a serem executadas num projeto de software

- Introdução de coerência entre os diversos projetos desenvolvidos em uma mesma organização.

- Fornecimento de pontos que sirvam como marcos para o controle gerencial.

- Padronização do processo de desenvolvimento, facilitando a implantação de um Programa de Controle de Qualidade.

A primeira tentativa de se modelar o ciclo de vida do software foi feita por Royce [in Davis,88]. Desta pesquisa surgiu o modelo conhecido por "Waterfall", que apesar de suas limitações ainda é muito utilizado pelas empresas que desenvolvem software.

A figura 11 mostra as fases constituintes do modelo Waterfall.

O modelo Waterfall, portanto, foi a primeira proposta de utilização de uma metodologia para o desenvolvimento de software.

Na teoria, este modelo funciona muito bem, mas quando aplicado a projetos reais surgem alguns problemas:

- O modelo propõe um fluxo sequencial que raramente é seguido por projetos reais, pois ocorrem muitas interações entre as fases;

- O modelo assume que o cliente/usuário seja capaz de citar, explicitamente, no início do desenvolvimento, todos os requisitos do software. Esta suposição, na grande maioria das vezes, não é verdadeira;

- O controle de qualidade proposto pelo modelo está baseado exclusivamente na fase de testes. Isto faz com que um grande erro cometido no início do

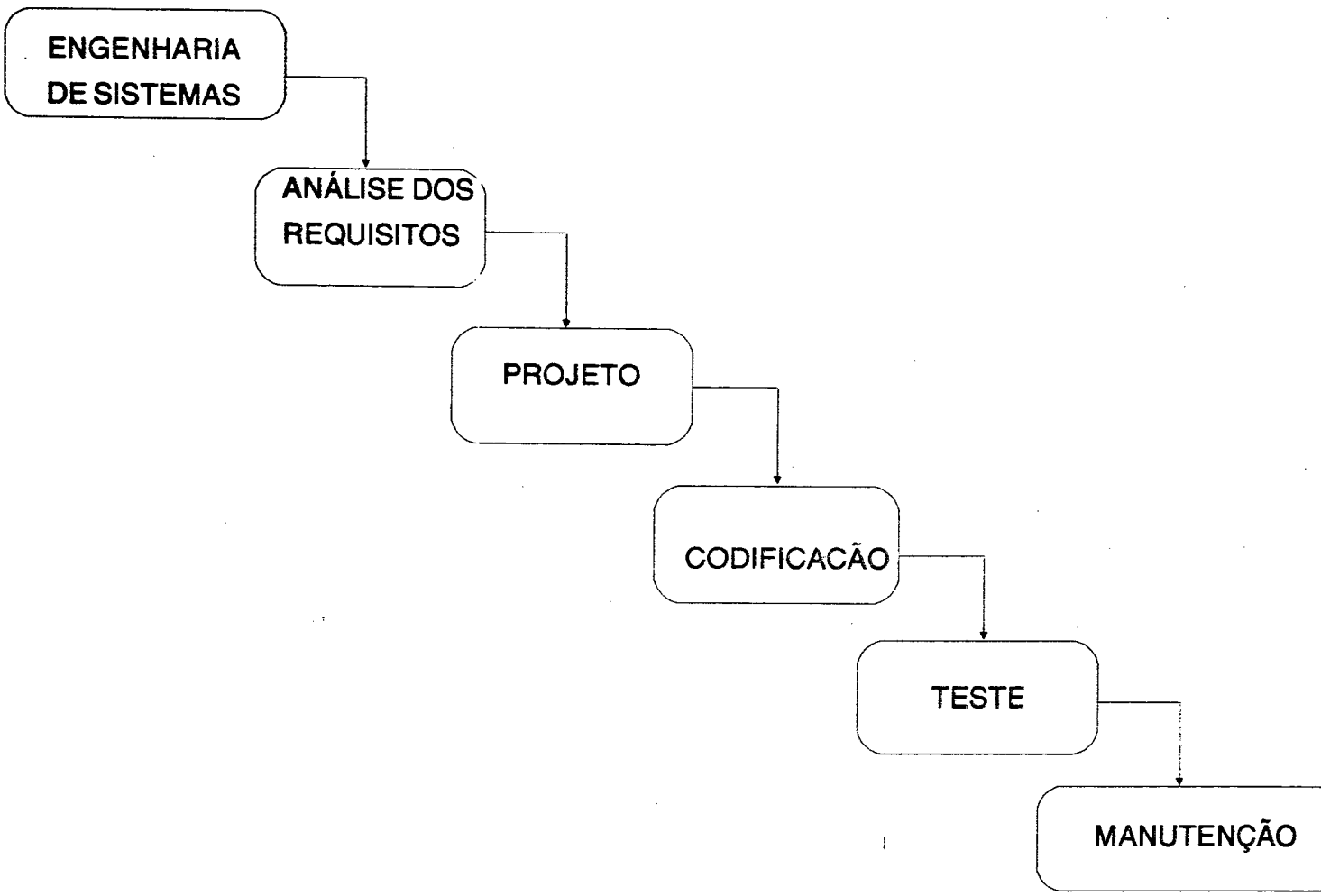

FIGURA 11 - MODELO WATERFALL

desenvolvimento tenha consequências desastrosas, pois só será detectado quando o software estiver pronto;

- A fase de análise realizada por empresas que utilizam esta abordagem clássica, tanto a nível de sistema quanto a nível de software, geralmente, é feita em português narrativo. Este fato dificulta a compreensão, modificação e atualização das especificações, ocasionando efeitos negativos sobre a qualidade;

- Após o término da fase de projeto e da fase de codificação, realizam-se os testes dos componentes na seguinte sequência: módulo, programa, subsistema e sistema. Esta sequência é claramente Bottom-up, ou seja, preocupa-se primeiramente com os níveis inferiores e segue em direção aos níveis mais elevados. As desvantagens em se utilizar o método Bottom-up são:

- Antes do término do teste do sistema não haverá nada para se mostrar ao usuário, a não ser uma pilha de listagens de programas;

- Os erros mais simples são encontrados no início da fase de testes, enquanto os erros mais complexos são encontrados durante o teste do sistema. O que se deseja é exatamente o contrário, pois quanto mais cedo se detectar os erros mais complexos, menores serão os impactos sobre os custos e sobre a qualidade do software;

- Se um erro é encontrado durante o teste do sistema, é difícil de se localizar o módulo que contém este erro. Isto ocorre pois, nesta etapa existirão centenas de módulos integrados, o que torna extremamente difícil a localização do erro.

Para tentar diminuir as deficiências do modelo Waterfall, foram sugeridos outros modelos.

Apesar de possuírem maneiras diferentes de enfocar o processo de desenvolvimento, os vários modelos sugeridos têm o mesmo objetivo: diminuição dos custos e dos atrasos de desenvolvimento. Além disso, os modelos procuram aumentar a confiabilidade e diminuir a necessidade de manutenção do software. Portanto, segundo

seus autores, a utilização de cada um dos modelos resultaria em uma melhoria da qualidade do software.

Dentre os métodos propostos, destacam-se os descritos a seguir:

3.2.1. DESENVOLVIMENTO INCREMENTAL

Este modelo, proposto por HIRSCH [IN DAVIS,88], propõe o desenvolvimento e a implantação de um sistema parcial em relação ao sistema total a ser desenvolvido. Sobre este sistema parcial são adicionadas novas funções, de forma a melhorar seu desempenho e confiabilidade. Esta abordagem proporciona uma redução nos custos, pois diminui o tempo necessário ao desenvolvimento. A redução do tempo é obtida devido a sucessivas implementações parciais, as quais melhoram o entendimento do sistema.

A vantagem dessa abordagem, é que o sistema é colocado em funcionamento já nas primeiras fases do processo de desenvolvimento, ao contrário da abordagem clássica, onde se produz uma grande quantidade de documentaçãoo para que se tenha o sistema operando. Como o tempo de desenvolvimento é menor, diminui a probabilidade de que as necessidades do usuário mudem durante o processo de desenvolvimento.

A desvantagem desta abordagem é que se admite o conhecimento da maior parte dos requisitos do sistema, o que, na maioria das vezes, não é verdadeiro.

3.2.2. PROTOTIPAÇÃO

A prototipação é bastante comum em outras áreas de conhecimento. Um exemplo clássico são as maquetes utilizadas na construção civil, as quais constituem os protótipos das obras a serem executadas.

No desenvolvimento de software, a utilização de protótipos parte do princípio de que não se conhece os requisitos do sistema [PRESSMAN,82].

As necessidades do usuário são conhecidas através da implementação de vários protótipos. A cada nova versão, o protótipo vai anexando novos requisitos, exigidos pelo usuário, que não constavam da versão anterior.

A participação efetiva do usuário no desenvolvimento, aumenta a probabilidade do software atender todos os requisitos pré-definidos.

A utilização de protótipos diminui o tempo total necessário ao desenvolvimento do software. Portanto, a mudança dos requisitos do usuário é menos provável.

Deve-se ressaltar, no entanto, que a prototipação não é um modelo de ciclo de vida de software. Trata-se de uma técnica de desenvolvimento que pode ser utilizada pelos modelos de ciclo de vida existentes.

3.2.3. SOFTWARE REUSÁVEL

A filosofia básica deste modelo é utilizar projetos e códigos que já tenham sido usados com sucesso em sistemas anteriores [PRESSMAN,82]. Para isso é necessário criar componentes reusáveis e técnicas de armazenamento e busca destes componentes . A vantagem obtida é um menor prazo de desenvolvimento, pois menos módulos deverão ser projetados, além de um software mais confiável.

Da mesma forma que a prototipação, o software reusável é apenas uma técnica de desenvolvimento.

3.2.4. ENGENHARIA DA INFORMAÇÃO

A Engenharia da Informação, através da utilização de técnicas similares às da engenharia tradicional, procura tratar as informações necessárias para o funcionamento ótimo de um empreendimento, de forma exata e utilizando conceitos formais a fim de desenvolver e manutenir sistemas de processamento de dados [Neto,88].

Baseado nesta definição, pode-se representar a Engenharia da Informação como uma pirâmide de quatro faces distintas: dados, atividades, tecnologia e pessoas. Cada uma destas faces possuem cinco fases, a saber:

- Planejamento Estratégico: Identifica as informações necessárias e estabelece o modelo corporativo de dados e o modelo funcional da empresa, fornecendo parâmetros para fixação de prioridades ao desenvolvimento de sistemas.

- Análise das Áreas de Negócios: Definidas as prioridades, executa-se o detalhamento das áreas mais carentes de apoio de sistemas automatizados.

- Projeto: Estabelece-se o modelo de dados de cada atividade envolvida, e transformam-se os diagramas de fluxo de dados para os correspondentes diagramas de estrutura.

- Construção: Esgotadas as filtragens e refinamentos necessários na fase de projeto, passa-se a executar a construção do sistema em questão. Esta construção é feita através da utilização de todas as ferramentas disponíveis, desde geradores de código até linguagens de quarta geração, compatíveis com os equipamentos de processamento de dados da empresa.

- Manutenção: Desde que as quatro fases anteriores tenham sido executadas criteriosamente, a manutenção restringir-se-á a ocorrências eventuais.

A Engenharia da Informação enfoca todos os fatores relacionados com o desenvolvimento de software. Esta característica faz com que os softwares desenvolvidos sejam mais consistentes e se adaptem com maior facilidade ao ambiente do usuário.

Torna-se importante ressaltar que neste modelo, não existe uma preocupação explícita com o controle de qualidade. Os autores parecem partir do princípio de que um software desenvolvido utilizando os conceitos da Engenharia da Informação, será de boa qualidade.

Mesmo que a Engenharia da Informação forneça modernas técnicas de desenvolvimento, o acompanhamento e a medida da qualidade dos produtos intermediários,

com a devida documentação, é fundamental para o desenvolvimento de softwares de boa qualidade.

3.2.5. CICLO DE VIDA ESTRUTURADO

Apesar das vantagens apresentadas pelos métodos descritos anteriormente, o presente trabalho abordará o processo de desenvolvimento de software sob o enfoque da filosofia estruturada, baseado nos trabalhos de [DeMarco,89], e [Yourdon,89].

Embora esteja baseado na filosofia estruturada, o modelo a ser apresentado no presente trabalho poderá ser aplicado para qualquer modelo de ciclo de vida de software. Isto ocorre pois, a quantificação da influência da metodologia sobre a qualidade do software independe da filosofia utilizada no desenvolvimento.

A utilização do ciclo de vida estruturado deveu-se às seguintes vantagens proporcionadas pelo modelo:

- Devido às ferramentas utilizadas, é possível visualizar o software que será desenvolvido antes que o mesmo esteja concluído;

- Contem critérios objetivos para determinar o que são atividades "ideais" de análise, projeto, codificação e outros;

- Todas as atividades ficam mais visíveis ao gerente de projetos sendo mais fácil verificar se as atividades estão sendo desenvolvidas conforme planejado.

- Apresenta ferramentas adequadas para descrever as necessidades dos usuários, enfatizar o controle de qualidade e o desenvolvimento top-down.

Existem outras vantagens associadas à utilização da filosofia estruturada, e estas serão abordadas a seguir,durante a descrição do ciclo de vida.

Mesmo havendo muitas vantagens associadas à utilização do ciclo de vida estruturado, bem como às técnicas estruturadas, sua aplicação nas empresas é restrito. Isto deve-se ao fato de que esta filosofia significa uma grande mudança no

processo de desenvolvimento de software resultando em resistências de grande vulto. A fim de se diminuir estas resistências, deve-se investir no treinamento dos Recursos Humanos envolvidos, de forma que eles absorvam a filosofia estruturada.

Esta resistência ocorre pois, os analistas e programadores não estão acostumados a conviver com a participação do usuário no desenvolvimento e além disso, não possuem uma preocupação com a qualidade. Como pode ser observado no ciclo de vida apresentado na figura 12, as duas características fazem parte da filosofia estruturada, . Este ciclo de vida está baseado no trabalho de Yourdon [Yourdon,89].

As fases apresentadas na figura 12, serão descritas a seguir.

3.2.5.1. ESTUDO

Esta primeira fase, também conhecida como estudo de viabilidade, avalia os problemas e deficiências no ambiente do usuário. Dependendo da estratégia da empresa, o estudo pode ser incluído na fase de análise.

O objetivo desta atividade é identificar os aspectos do ambiente do usuário que são inadequados. Tais dados são obtidos através de entrevistas com os membros da comunidade do usuário.

Através de entrevistas com os usuários do sistema, procura-se obter as seguintes informações:

- Os requisitos do sistema, na forma de uma lista de funções e características operacionais, desejadas pelo usuário.

- A política do usuário, que descreve as características operacionais do ambiente de processamento.

Com a execução das atividades desta fase, obtem-se um primeiro esboço do ambiente do usuário. Este esboço é a base do projeto, tanto a nível do usuário quanto a nível da empresa responsável pelo desenvolvimento.

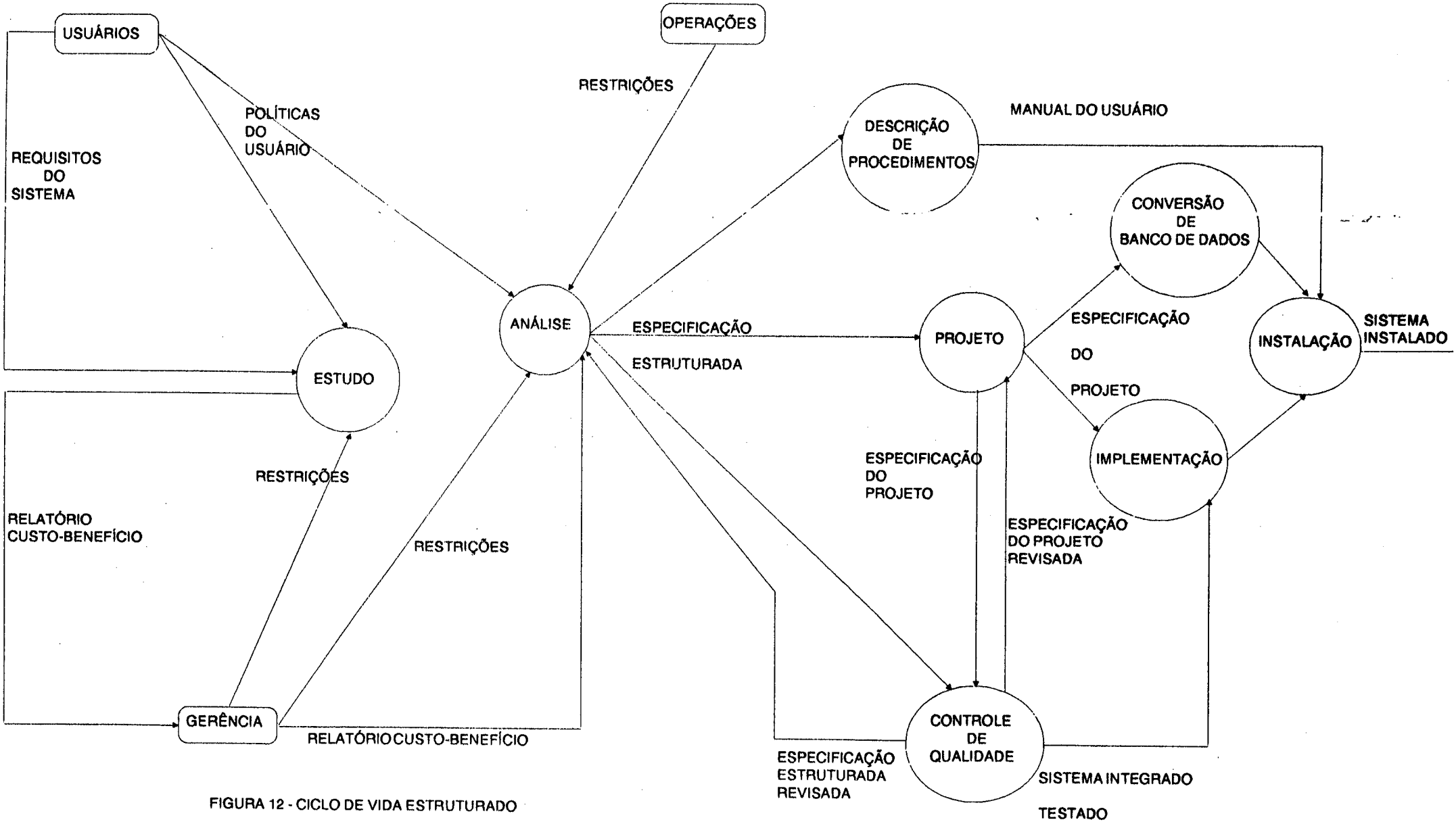

FIGURA 12 - CICLO DE VIDA ESTRUTURADO

O aprofundamento das informações fornecidas pela fase de estudo, será feito na próxima fase: a Fase de Análise.

3.2.5.2. ANÁLISE

A Fase de Análise, muitas vezes chamada de Especificação do Sistema, define as características do software, ou seja, preocupa-se com "o que" o produto será capaz de realizar após ser concluído. Nesta fase são realizadas as mesmas atividades da fase de estudo, só que de maneira mais detalhada.

Dentre os objetivos desta fase, pode-se citar:

– Identificar as interfaces do sistema

– Determinar "o que" o software irá fazer

– Definir a expectativa de qualidade

– Estabelecer as restrições do software

A análise é a base de todo o desenvolvimento do software. Por este motivo, qualquer erro cometido nesta fase irá refletir diretamente sobre a qualidade do produto final. Em função da importância do fato, esta fase será descrita com maiores detalhes do que as demais fases do processo de desenvolvimento.

Existem muitas dificuldades relacionadas com a epecificação de sistemas, que podem prejudicar totalmente as demais fases do desenvolvimento. Dentre os maiores problemas citam-se [DeMarco,89]:

1 - PROBLEMAS DE COMUNICAÇÃO: A falta de uma linguagem comum entre analistas e usuários, provoca máinterpretação e incompreensões. Aliado a isto, os analistas sentem dificuldades em descrever na forma textual convencional, os procedimentos a serem implantados.

2 - NATUREZA VARIÁVEL DOS REQUISITOS:

Durante o desenvolvimento do software é comum haver mudanças nos objetivos do projeto.Estas mudanças devem resultar na modificação das especificações produzidas. Como as especificações, descritas da forma textual convencional, são muito difíceis de serem modificadas, os gerentes de projeto tendem a congelar as especificações. Isto significa, ignorar as mudanças e perseguir um objetivo estável. Esta atitude acarreta grandes prejuízos ao projeto pois, poderá conduzir a um produto que não atenda às necessidades do usuário

3 - FALTA DE FERRAMENTAS ADEQUADAS: As especificações, geralmente, possuem ambiguidades, falta de clareza e contradições, indicando ausência de ferramentas e métodos que facilitem a Fase de Análise.

4 - PROBLEMAS COM A ESPECIFICAÇÃO DO PROJETO: Na abordagem convencional, a especificação é documentada na forma de texto. Isto provoca, além da dificuldade de manutenção já citada, alguns inconvenientes. Esta situação ocorre pois o

- Excessivamente redundante
- Excessivamente prolixo
- Mais voltado para características físicas
- Difícil de ser escrito
- Difícil de ser lido

texto torna-se:

5 - DISTRIBUIÇÃO DE TRABALHO: As especificações de melhor qualidade são, geralmente, aquelas produzidas por equipes pequenas, preferencialmente por uma pessoa. Isto ressalta a necessidade de se particionar melhor o problema. O objetivo é conseguir uma distribuição mais efetiva do trabalho a ser

realizado. Isto pode ser conseguido através de um planejamento mais efetivo das atividades a serem executadas. Os aspectos relacionados com o planejamento serão abordados posteriormente.

6 - POLÍTICA: A introdução de novos procedimentos de trabalho causam, na maioria das vezes, problemas políticos. Estes problemas estão relacionados à perda de status, mudança na rotina de trabalho e outros. Tais problemas não serão solucionados pela simples utilização de novas técnicas de análise. Entretanto, a definição clara e objetiva das responsabilidades e rotinas de trabalho pode diminuir as dificuldades políticas.

Todas as dificuldades apresentadas anteriormente, podem ser amenizadas através da utilização da análise estruturada. A aplicação da filosofia estruturada produz especificações com as seguintes características [Yourdon,89]:

- Particionada, ou seja, dividindo-se em partes menores que são compreensíveis por si mesmas. Isto significa que não é necessário ler todo o documento para se analisar um dado aspecto do sistema.

- Gráfica, utilizando-se de diagramas, gráficos e figuras, ao invés de descrição em forma de texto.

- Top-down, significando que o problema é tratado, primeiramente, nos níveis mais elevados, ou seja, a partir de um ponto de vista mais geral. Após decomposições sucessivas, atinge os níveis mais detalhados.

- Independente do restante do projeto. Portanto, mesmo que o projeto não prossiga, o produto obtido (especificação estruturada) será de grande utilidade para o usuário.

Estas características contribuem para que se desenvolva um software de melhor qualidade. A análise estruturada pode não resolver todos os problemas da definição de requisitos do sistema mas, será de grande ajuda na melhoria da interface analista/usuário.

3.2.5.3. PROJETO

As atividades do processo de desenvolvimento se preocuparam até agora, basicamente, com "o que" o software irá fazer. A fase de projeto tenta estabelecer o "como" será implementada a especificação produzida.

Como se observa, a fase de projeto está baseada totalmente nas especificações produzidas durante a fase de análise. Como já foi salientado, a qualidade da fase de projeto dependerá em alto grau da qualidade das especificações.

Tais fatos podem ser observados na definição de projeto estruturado de Yourdon [Yourdon,89]:

"O projeto estruturado é a determinação de quais módulos, interconectados de que forma, melhor solucionarão algum problema bem definido".

Esta definição considera que o projeto estruturado tenha sido precedido de uma análise estruturada.

As ferramentas geralmente utilizadas pelo projeto estruturado são os Diagramas de Fluxos de Dados, Diagramas de Estrutura, Especificações de Módulos e Dicionário de Dados. O Diagrama de Estrutura é uma ferramenta gráfica que representa a hierarquia dos módulos.

A utilização dessas ferramentas é condição necessária, mas não sufuciente, para que se alcance os objetivos estabelecidos na fase de projeto.

Existem dois objetivos principais, que devem nortear a fase de projeto, bem como todo o processo de desenvolvimento de software:

1 - Facilidade de Manutenção

A manutenção inclui, além da atividade de correção, a modificação do software. Quando os requisitos do usuário mudam, segundo Boehm, a manutenção é

responsável por 60% dos custos totais do software. Por este motivo é importante que a facilidade de manutenção seja uma preocupação constante, durante o desenvolvimento do software, mesmo que isto signifique um relativo aumento nos custos das outras fases.

Segundo Constantine, a facilidade de Manutenção está relacionada com:

- Módulos pequenos: esta característica diminui a probabilidade de que a mudança de um módulo afete um outro módulo qualquer.

- Independência modular: quanto menos um módulo depender de outro maior será sua independência e, portanto, maiores facilidades em mantê-lo.

- Característica de "caixa-preta": o módulo é desenvolvido de maneira que possa ser visto em termos de suas entradas e saídas, sem se preocupar com os detalhes internos.

- Isolamento de detalhes: deve-se procurar separar as partes do sistema que refletem sua filosofia básica daquelas que descrevem os detalhes.

2 - Facilidade de Depuração e Teste

A depuração e o teste são responsáveis por 75% dos custos de desenvolvimento (sem considerar a fase de manutenção). O projeto deve então, ser conduzido de forma a produzir softwares mais fáceis de serem testados. As mesmas caracteríticas que afetam a facilidade de manutenção, também influenciam a facilidade de depuração e teste.

Além dos objetivos citados, DeMarco cita outros que, apesar de serem secundários, aumentariam a qualidade do sistema produzido. Entre esses estão:

- A documentação do projeto deve ser um subproduto natural do processo de desenvolvimento.

- Deve-se diminuir a necessidade de comunicação entre os implementadores durante as fases subsequentes.

- O projeto deve apresentar característica top-down, ou seja, iniciar nos módulos de mais alto nível para depois partir para os níveis mais detalhados.

- Utilizar uma metodologia que facilite o particionamento.

Mesmo que a fase de projeto procure alcançar os objetivos citados anteriormente, podem surgir algumas dificuldades que prejudiquem sua qualidade

Uma dificuldade óbvia é quando o projeto se desvia das especificações produzidas, desenvolvendo um produto final que não atende as necessidades do usuário. Para se evitar esta situação, a melhor solução é a realização de uma revisão formal ao final do projeto, com o objetivo de garantir a integridade do software produto.

Apesar de sua grande utilidade, o diagrama de estrutura pode ser desenvolvido inadequadamente. A consequência será a produção de documentos que não correspondem ao código a ser depurado. Além disso o diagrama de estrutura pode não combinar com o DFD do qual foi gerado. Estes problemas ocorrem geralmente, pela não utilização de métodos sistemáticos de geração de diagramas de estrutura. Dois métodos que podem ser utilizados são: a Análise de Transformação e a Análise de Transação; ambos descritos em detalhes por DeMarco [DeMarco,89].

Um outro erro comum da fase de projeto é tratar as características do software na sequência em que o computador irá abordá-las, ou seja, na forma entrada-processo-saída. Esta abordagem considera mais importante as atividades que ocorrerão primeiro, durante a utilização do software. A filosofia do projeto estruturado aconselha um tratamento hierárquico das características, ou seja, as características de maior importância são consideradas em primeiro lugar (independente da sua ordem de ocorrência).

Além dos problemas citados, existem outros fatores que influenciam a qualidade do projeto. Dentre estes fatores, podem ser citados:

- ACOPLAMENTO: O acoplamento dá uma indicação da interdependência dos módulos, o que afetará em alto grau a facilidade de manutenção do software. Quanto maior for o acoplamento,maior será a probabilidade de que mudanças em um módulo afetem o funcionamento de outro. Por este motivo, é necessário minimizar o acoplamento, favorecendo asssim a facilidade de manutenção e em consequência, a qualidade do software.

- COESÃO: Um módulo altamente coeso possui seus elementos internos fortemente associados. Isto significa que estes elementos contribuirão para a conclusão de uma única tarefa ou tarefas muito relacionadas. Segundo DeMarco, "a coesão é uma medida da força de associação dos elementos dentro de um módulo". Existe uma correlação entre acoplamento em excesso e fraca coesão, ou seja, deve-se aumentar a coesão a fim de se desenvolver um software de qualidade.

- EMPACOTAMENTO: O empacotamento é a atividade que visa estruturar o projeto de maneira a adaptá-lo ao ambiente físico(máquina, sistema operacional, linguagem de programação, limitações de memória e disco, restrições de tempo,etc).

Uma abordagem detalhada dos fatores de qualidade do software, bem como das medidas de qualidade, é feita por Rocha [Rocha,92]

3.2.5.4. IMPLEMENTAÇÃO

Drante a fase de implementação o projeto será transformado (codificado) em linguagem de máquina. Além disso, grande parte do teste é executada nesta fase.

Caso tenham sido tomados todos os cuidados durante as fases de análise e projeto, a fase de implementação deverá transcorrer sem maoires problemas. Apesar dissso, alguns problemas de ordem gerencial podem ocorrer. Entre eles pode-se citar:

- Pode não haver hardware suficiente: as atividades de codificação e testes são realizadas no computador podendo haver, por esse motivo, um grande número de pessoas disputando poucos equipamentos

- O tempo designado para testes pode ser insuficiente: os módulos devem ser integrados e testados aos poucos. A falta de tempo pode fazer com que o analista integre e teste muitos módulos de uma só vez, prejudicando, em caso de erro, a identificação do módulo problema.

- Desvio da especificação do projeto: Isto ocorre geralmente quando pessoas(equipes) diferentes fazem o projeto e a especificação. A consequência é que o produto poderá executar tarefas desnecessárias ou omitir atividades importantes.

Nessa fase, ocorre a integração, codificação e teste dos módulos preparados na fase de projeto.

3.2.5.5. CONTROLE DE QUALIDADE

Tradicionalmente, o Controle de Qualidade, num ambiente de desenvolvimento de software, é realizado apenas na fase de teste do sistema. Esta prática faz com que se controle apenas a parte codificada do software.

Em outras palavras, os produtos das fases de Análise e Projeto, não sofrem nenhum tipo de controle. Isto faz com que 50% do tempo gasto no desenvolvimento de software, ocorra nos testes [Boehm, 81]. Além disso, como já foi salientado, um erro cometido na fase de análise que seja descoberto apenas na fase de teste, tem um custo muito grande para a empresa responsável pelo desenvolvimento.

Nos últimos anos, esta filosofia de teste tem sofrido mudanças. Começa a haver uma preocupação com o controle de qualidade dos produtos gerados nas primeiras fases do ciclo de vida do sistema.

A figura 12, mostrada no item 3.2.5, reflete esta mudança, apontando o controle de qualidade como uma preocupação ao longo de todo o desenvolvimento de software.

Uma técnica bastante utlizada atualmente, para se controlar a qualidade do software, é a revisão.

Segundo Yourdon [Yourdon, 90], cerca de 75% das organizações de processamento de dados dos Estados Unidos, usam alguma forma de revisão em um ou mais estágios do desenvolvimento de seus sistemas.

A revisão é uma forma de se analisar os principais pontos de um projeto em desenvolvimento. Conforme define Yourdon, ***"em uma revisão, um grupo de colegas - pessoas aproximadamente no mesmo nível na organização - encontram-se para rever e discutir um produto."***

Para assegurar a necessária independência, é recomendável que pessoas que não participaram do desenvolvimento sejam incluídas nas reuniões de revisão. Como exemplo pode-se citar: analistas externos à equipe de projeto, analistas externos à empresa, gerente responsável pela área técnica e, como sugere DeMarco, o usuário.

Nas revisões, é comum a utilização de listas de problemas que ocorreram em outros projetos, como forma de analisar o projeto em andamento. Estas listas são chamadas de Checklists (listas de checagem). As vantagens de se utilizar as checklists são [Hetzel, 87]:

- Estruturam a revisão
- Permitem que se aprenda com experiências acumuladas
- Garantem uma abordagem sistemática e abrangente
- Facilitam a medida e a quantificação dos resultados

Existem muitas vantagens em se realizar revisões dos principais produtos do desenvolvimento de software. Entre elas pode-se citar:

- Avaliam com segurança o andamento de um projeto

- Facilitam a identificação de talentos individuais

- Detectam padrões de erros repetitivos

- Fornecem um feedback rápido do processo de desenvol vimento e impedem a manifestação de erros mais graves

- Treinam e educam os participantes

A filosofia de se realizar revisões ao longo do processo de desenvolvimento, sugere um ciclo de vida de teste que se superponha ao ciclo de vida do software, como mostra a figura 13. Essa superposição tem por objetivo, disciplinar a maneira de se responder às seguintes perguntas: "o que testar?", "quando terminar?" e "quem será o responsável?".

Para maiores detalhes sobre a operacionalização dos testes de software, sugere-se a consulta ao trabalho de Hetzel [Hetzel,87]

A fim de que se possa atingir a excelência na produção de software, torna-se necessário a adaptação de procedimentos utilizados nas empresas industriais.

Neste sentido, uma abordagem que poderia ser utilizada no desenvolvimento de software é a filosofia da Qualidade Total. Segundo esta filosofia, descrita em Lobos [Lobos,91], a qualidade deve ser:

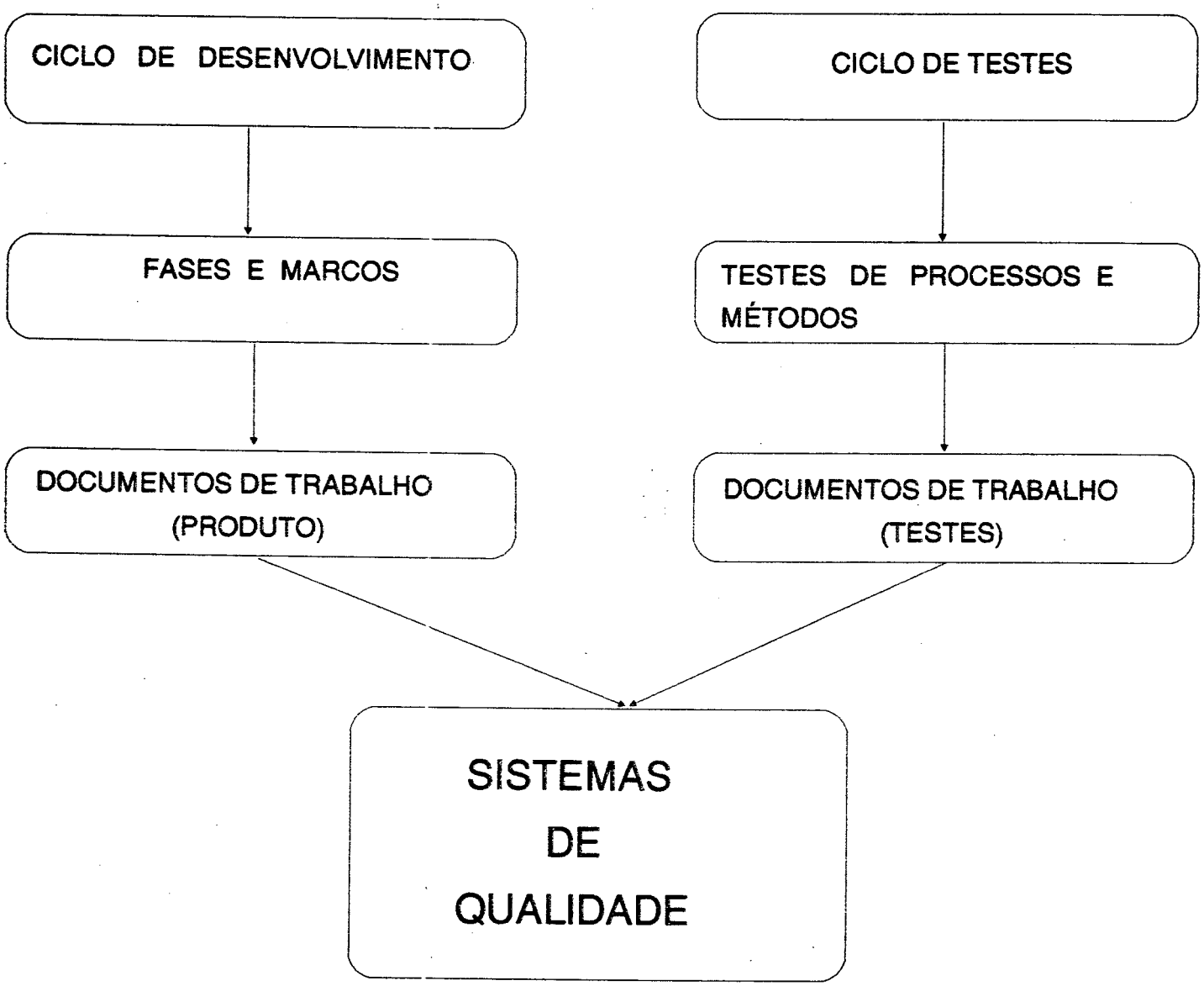

FIGURA 13 - SUPERPOSIÇÃO DO CICLO DE TESTE AO CICLO DEDESENVOL VIMENTO

– Plena, abrangendo todo o processo que origina o produto.

– Preventiva, evitando antecipadamente desvios do produto com relação aos requisitos do cliente.

– Proprietária, ou seja, assumida responsavelmente por todas as pessoas - e cada uma delas - que fazem parte da organização.

– Permanente, incorporando-se ao comportamento gerencial e individual no cotidiano.

– Progressiva, requerendo o contínuo aperfeiçoamento das diversas formas de execução na organização.

3.2.5.6. DESCRIÇÃO DE PROCEDIMENTOS

Durante a fase de Descrição de Procedimentos, desenvolve-se o Manual do Usuário.

Torna-se importante ressaltar que o desenvolvimento do manual do usuário depende, basicamente, das informações fornecidas pelo produto da fase de análise. Isto ocorre, pois é durante a fase de análise que são descritas as entradas e saídas do sistema, bem como a política do usuário.

A descrição de procedimentos pode, então, ter início logo após o témino da fase de análise, ocorrendo paralelamente ao restante do desenvolvimento.

Torna-se necessário salientar que esta fase também depende das informações de natureza física (hierarquia de telas, formato de entrada e saída, etc) fornecidas pelo projeto. Mas não é essencial que o projeto esteja concluído para se iniciar o desenvolvimento do Manual do Usuário. Estas duas fases podem ocorrer paralelamente, visto que as informações principais são fornecidas pela fase de análise.

3.2.5.7. CONVERSÃO DE BANCOS DE DADOS

O objetivo desta fase é transformar o banco de dados atual, com o intuito de adaptá-lo às características do novo sistema no que diz respeito a formato, organização de arquivo e outros aspectos. Esta fase só será realizada caso exista algum sistema em funcionamento(manual ou automatizado) pois, caso contrário não haverá banco de dados a ser convertido.

Pelo fato de demandar muito tempo, a conversão é realizada, geralmente, durante a operação do sistema. Por este motivo, o usuário é a pessoa mais indicada para executar essa atividade. Segundo Yourdon [Yourdon,89], este procedimento possui duas desvantagens:

- Poderá haver erros de estilo pois, ao longo do processo de conversão, pessoas diferentes estarão envolvidas. Isto ocorre devido ao longo tempo necessário à conversão do banco de dados.

- A produtividade será baixa, até que todo o banco de dados esteja convertido.

3.2.5.8. INSTALAÇÃO

A instalação trata da implementação do software no ambiente do usuário. Nesta etapa o cliente recebe o "Manual do Usuário" e o treinamento adequado para que possa operar o software.

Embora pareça trivial, a fase de implantação requer alguns cuidados [Yourdon,89]:

- Determinação de quando o sistema será implementado, para evitar que a rotina diária do usuário seja prejudicada.

- Planejamento antecipado da data de desativação do antigo sistema. Geralmente, o antigo e o novo sistema operam em paralelo, a fim de se identificar

problemas que ainda permanecem no sistema a ser implantado. A definição deste período de operação em paralelo é, portanto, muito importante.

- Implementação em partes, dependendo do tamanho do sistema, a fim de se diminuir a complexidade da implementação.

- Definição clara da maneira como os usuários serão treinados.

3.3. GERÊNCIA

A gerência do projeto é um fator intimamente ligado à qualidade do software, uma vez que é através da administração dos recursos disponíveis que chegar-se-á ou não aos objetivos pré-definidos.

Independente da existência de uma metodologia de desenvolvimento eficiente, a má atuação do gerente pode prejudicar o projeto como um todo. Além disso, a metodologia é utilizada por pessoas, que são coordenadas por um gerente. Devido a esta interdependência é que, quando se discute sobre fatores que afetam a qualidade do software desenvolvido, deve-se analisar os aspectos gerenciais envolvidos.

Em uma empresa que desenvolve software, na maioria das vezes, o analista é promovido a gerente devido ao bom desempenho de sua função. Como os dois cargos possuem perfis bastante distintos, o gerente sente muitas dificuldades para trabalhar com os vários interesses contrários envolvidos num projeto de desenvolvimento de software. Estes interesses são representados:

- Pelo cliente que adquiriu o sistema e deseja que o desenvolvimento seja feito o mais rápido possível e a um baixo custo.

- Pelos usuários do novo sistema, esperando que ele possua muitas funções (recursos), seja rápido, fácil de ser operado e que atenda a uma grande variedade de situações.

- Pela alta administração, a qual estabelece metas ambiciosas e não admite atrasos ou surpresas.

- Pelos membros da equipe de desenvolvimento, os quais desejam uma rápida ascensão profissional, têm preferência por projeto e geralmente, retardam a documentação.

A interrelação dos interesses descritos anteriormente, podem ser visualisados na figura 14.

O gerente, geralmente, não consegue adequar um estilo gerencial que se adapte à realidade mostrada na figura 14. Por este motivo, um ou mais interesses ficarão insatisfeitos, trazendo prejuízos para a empresa responsável pelo projeto.

A dificuldade em se encontrar um estilo gerencial adequado, reside no fato de que as teorias gerenciais exixtentes não são eficazes, quando aplicadas num ambiente de desenvolvimento de software.

A Teoria X, baseada nas idéias da administração científica de Frederick Taylor [TAYLOR, 71], defende a divisão do trabalho numa sequência definida de tarefas. Esta sequência de atividades, tem o objetivo de tornar as pessoas tão previsíveis quanto máquinas. A teoria está baseada em estudos precisos de tempos e movimentos e na pressão sobre os empregados, de modo a ter o trabalho realizado.

A realidade tem demonstrado que a Teoria X é totalmente deficiente, pois faz o empregado trabalhar desmotivado. Isto significa que as pessoas fazem o mínimo necessário para não serem demitidas. No desenvolvimento de software, a Teoria X torna-se ainda mais deficiente, uma vez que a maior parte do trabalho é intelectual, dificultando a medida de tempo e movimento. Além disso, os analistas e

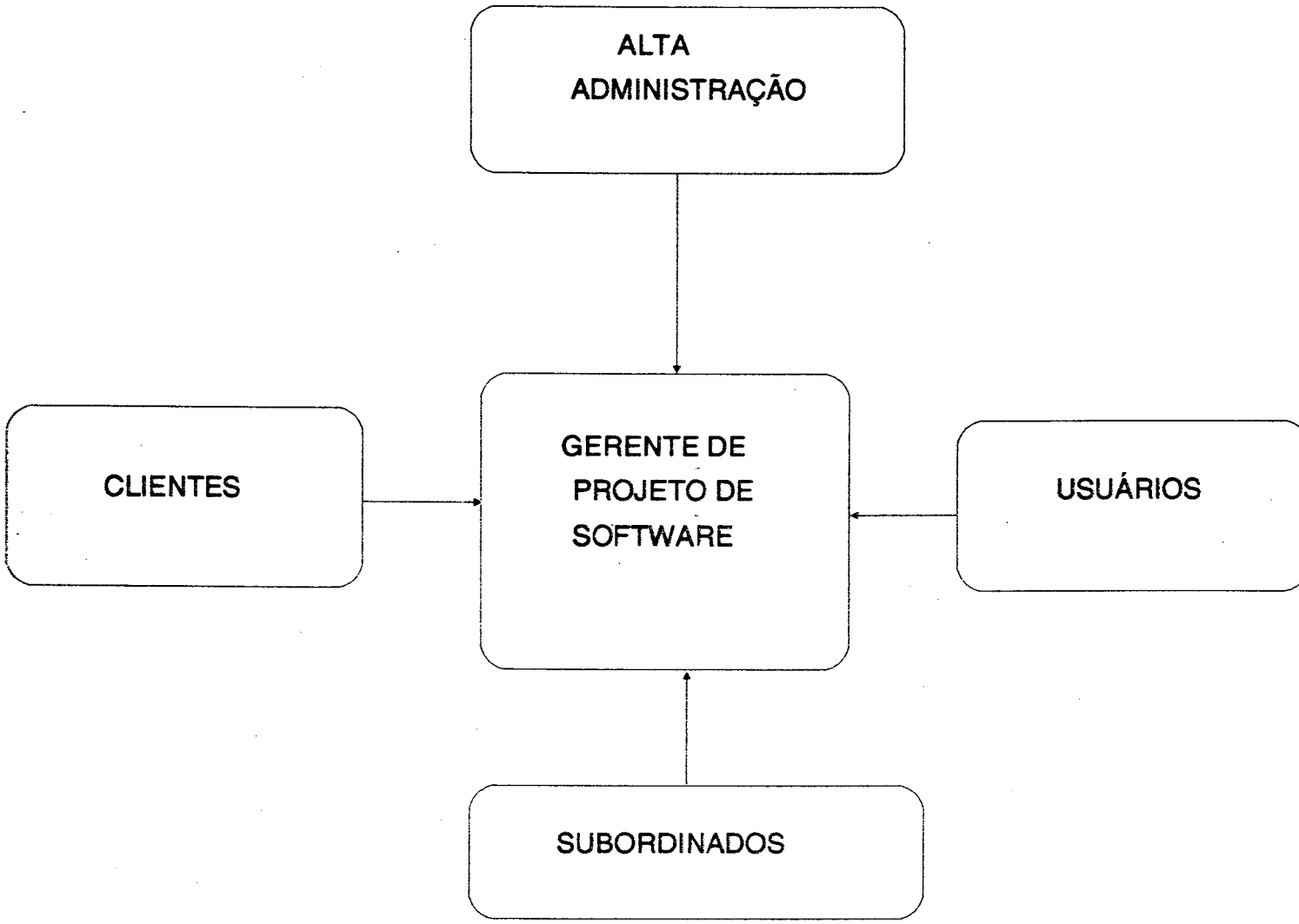

FIGURA 14 - INTERESSES ENVOLVIDOS EM UM AMBIENTE DEDESENVOLVIMENTO DE SOFTWARE

programadores possuem nível universitário, o que dificulta a utilização de pressão como instrumento de gerenciamento.

A Teoria Y, introduzida por Douglas Mcgregor (in [CHIANENATTO,80]), está baseada na filosofia de que a gerência deve estimular a criatividade e a iniciativa individual. Isto conduzirá, segundo os defensores desta teoria, a organizações mais adaptáveis e que melhor satisfazem as necessidades individuais. Apesar disso, a existência de muitas iniciativas individuais, competindo por recursos e instalações, criará um sério problema de coordenação.

A Teoria Z, introduzida por Ouchi [OUCHI, 82], acredita, que a maioria dos problemas podem ser resolvidos por consenso, entre as partes envolvidas. Isto significa que cada um dos envolvidos deve ceder um pouco para que se alcance uma situação aceitável para todos.

Quando existem múltiplos interesses envolvidos, com organizações de diferentes culturas e objetivos, fica muito difícil a resolução das situações-problema através de consenso. Portanto, a Teoria Z não se aplica perfeitamente ao ambiente de desenvolvimento de software.

A Teoria W, proposta por Boehm [Boehm,89], é uma tentativa de se ter uma teoria gerencial adaptada ao ambiente de software. Esta teoria defende que os gerentes de projeto serão bem sucedidos, se e somente se, transformarem "vencedores" todos os outros participantes do processo de desenvolvimento de software(superiores, subordinados, usuários, etc.).

Segundo a filosofia da Teoria W, o gerente deve assumir a posição de negociador entre as várias partes envolvidas. Além disso, o gerente também deve ser o fixador de metas e um monitorador do progresso em direção às metas pré-estabelecidas.

Nas situações do dia-a-dia, segundo a Teoria W, o gerente deve estar sempre atento a conflitos"ganha-perde" ou "perde-perde", onde pelo menos uma das partes sai prejudicada. A função do gerente, é transformar esses conflitos em situações "ganha-ganha".

O estilo gerencial a ser adotado depende muito da personalidade de cada gerente. Apesar disso, a Teoria W adapta-se melhor a um projeto de software do que as outras teorias gerenciais descritas. Isto ocorre pois, os passos sugeridos pela Teoria W , conseguem identificar os principais problemas do desenvolvimento de software. Além disso, cria-se um ambiente favorável à solução dos problemas existentes.

Segundo uma pesquisa desenvolvida por Thayer [Thayer,87], uma grande deficiência dos projetos de software é a ausência de um planejamento eficiente. Em função disso tanto a utilização dos recursos e da mão-de-obra, quanto os gastos a serem realizados e o tempo a ser consumido em cada tarefa, não são definidos com antecedência, causando consequências negativas para a qualidade do produto.

A utilização de uma metodologia de desenvolvimento de boa qualidade, fornece bases suficientes para o planejamento das atividades a serem realizadas durante o projeto. Apesar disso, raramente os gerentes utilizam a sequência de atividades contida na metodologia para planejar o andamento do projeto.

Tal situação é causada, em parte, pelo fato do gerente ser oriundo do staff técnico onde, geralmente, não são desenvolvidas atividades de planejamento. Além disso, o gerente não é treinado nas atividades de motivação de pessoal, planejamento, organização e controle.

Howes [Howes,84] sugere uma sequência de passos os quais possibilitam o pleno planejamento e controle de um projeto de software.

Além dos problemas citados anteriormente, o gerente, na maioria das vezes, não tem claramente definidas as suas responsabilidades. Por este motivo, ele não sabe qual é o limite de sua autoridade:

– Pode demitir?

– Pode substituir pessoas?

– Pode sugerir soluções alternativas?

– Pode recomendar o encerramento do projeto?

Definidas as responsabilidades, o gerente deve procurar conhecer os desejos e aspirações de seus subordinados, a fim de que possa motivá-los positivamente.

A administração dos recursos humanos é uma das tarefas mais difíceis de ser realizada pois, varia de equipe para equipe.

Portanto, pelo que foi exposto, o gerente de um projeto de software deve ter um perfil "administrativo", para liderar, administrar e remover barreiras; e um perfil "técnico", para encorajar inovações tecnológicas, para revisar especificações e catalisar atividades de projeto.

Esse profissional deverá realizar um planejamento eficiente, possuir um estilo gerencial adequado além de coordenar e motivar os membros de sua equipe.

3.4. RECURSOS HUMANOS

Segundo Chiavenato [Chiavenato,80], a maior parte dos problemas de uma empresa, está relacionada com seus Recursos Humanos. As principais causas destes problemas estão relacionadas com a falta de condições de trabalho e incapacidade ou desmotivação por parte dos integrantes das equipes de trabalho.

Esta situação torna-se mais dramática no caso das empresas produtoras de software, uma vez que a "matéria-prima" para o desenvolvimento de um projeto é o conhecimento humano. Se, por algum motivo, as pessoas estão insatisfeitas ou desmotivadas, as consequências sobre a qualidade do produto são desastrosas.

Em um artigo publicado na revista "Data Management", Awad descreve que a rotatividade de pessoal nas empresas que desenvolvem software, é em torno de 40% ao ano. Torna-se importante ressaltar, que este dado não diz respeito a uma empresa em particular mas sim, a um ramo de atividade.

Este dado mostra que as empresas não estão conseguindo identificar as necessidades de seus colaboradores. A causa básica desta realidade é a "importação" de conceitos e técnicas de gerenciamento de pessoal das indústrias de manufatura, desconsiderando diferenças de perfil entre os profisionais da área de software e os da indústria.

Algumas características importantes de analistas e programadores são:

- Preferência por trabalhar sozinho. Neste sentido, condições amigáveis de trabalho e reconhecimento social não são encarados como prioridades.

- Em geral, o desenvolvimento de carreira desejado não inclui promoção a cargos de gerência. O desejo de autoridade e a responsabilidade de dirigir outras pessoas, frequentemente não existe.

- A realização do trabalho é o principal motivador. "Realizar o trabalho" e "Fazer um bom trabalho" são as maiores fontes de motivação e, geralmente, mais prioritários que salário.

- Valorização da liberdade e da habilidade para organizar seus próprios trabalhos. Isto conduz a ressentimentos com relação a ordens impostas externamente.

- Apreciam receber tarefas desafiadoras e trabalhar nela sozinhos. Por este motivo, não se adaptam aos métodos de administração participativa.

O desconhecimento das características apresentadas anteriormente, conduz a decisões totalmente equivocadas. Dentre os principais erros, destacam-se:

- Promoção de bons analistas a gerentes.Como, em geral, os analistas não querem assumir cargos de gerência, essa prática faz com que

se transforme um competente analista em um péssimo gerente.

- Aplicação de técnicas japonesas de administração.As filosofias japonesa dão grande ênfase no trabalho em grupo. Os analistas e programadores, como foi salientado anteriormente, não se adaptam muito bem a trabalhos em grupo, o que torna as referidas técnicas ineficazes para a área de desenvolvimento de software.

- Implantação de círculos de controle de qualidade.Na indústria de manufatura, os círculos de controle de qualidade(CCQs) produzem excelentes resultados com relação à qualidade do produto produzido. Na área de software, devido às características já citadas, os resultados são muito piores. Em função dessas características, a implantação dos CCQs torna-se muito difícil. Quando a implantação é conseguida, o desinteresse para com as reuniões periódicas, necessárias para o andamento do programa, torna a continuação dos CCQs inviável.

Em função do que foi descrito, torna-se necessário a definição de um conjunto de políticas de pessoal que se adpte à área de desenvolvimento de software.

Estas políticas parecem tender para um aumento da responsabilidade individual, ênfase na avaliação de desempenho e baixa interação social.

Segundo Rahman [Rahman, 87], algumas estratégias que poderiam ser seguidas são:

- Dar a um analista ou programador, total responsabilidade por um dado pacote de trabalho.

- Utilizar a administração por objetivos, de forma a dar total liberdade a analistas e programadores para organizarem seus próprios trabalhos a fim de atingir os objetivos pré-fixados.

- Realizar promoções sem impor responsabilidades gerenciais, a menos que seja exatamente isso o que o analista deseja.

- Não dividir as equipes de trabalho ao final do projeto. Segundo DeMarco, a continuidade das equipes aumenta a sinergia entre os seus membros. Isto ocorre pois, as pessoas passam a se conhecer melhor.

- Fornecer às equipes, locais espaçosos e silenciosos para trabalhar. Segundo uma pesquisa de DeMarco e Lister [DeMarco,90], os melhores projetos, geralmente, são realizados por equipes que possuem um bom espaço físico e um baixo nível de ruído.

- Utilizar uma metodologia de desenvolvimento, de forma a diminuir a dificuldade de comunicação com o usuário. Além disso, a metodologia define o caminho a ser seguido, facilitando a comunicação interna.

A atuação da gerência, torna-se decisiva para a efetiva implantação das estratégias sugeridas. Com isso, segundo DeMarco, Rahmam e Awad, as vantagens a serem obtidas são:

- Maior produtividade.

- Melhor Qualidade.

- Maior motivação.

- Baixa rotatividade.

Evidentemente que essas vantagens só serão obtidas com a atenção igualitária sobre os três fatores estudados:

- Metodologia

- Gerência

- Recursos Humanos

CAPÍTULO 4

4. METODOLOGIA E QUALIDADE DE SOFTWARE

4.1. INTRODUÇÃO

Foi salientado no início deste trabalho que o desenvolvimento de software, no início da década de 60, era feito artesanalmente. Não havia uma padronização no processo de desenvolvimento. O surgimento das metodologias contribuiu para organizar o desenvolvimento de software.

Como houve uma melhoria na qualidade do software, pode-se pensar que o uso de uma metodologia de desenvolvimento provocasse um aumento na qualidade do produto final.

Mas, não é possível afirmar que existe uma relação causa-efeito entre o uso de uma metodologia e a qualidade do software produzido. O máximo que se pode dizer é que existe uma relação entre eles. Isto ocorre pois, paralelamente à implementação de metodologias, surgiram novas técnicas, métodos e linguagens de programação, que modificaram bastante a maneira de se produzir software.

Neste sentido, o trabalho procura identificar e quantificar a relação existente entre metodologia e qualidade de software.

O modelo a ser proposto, além de estabelecer a relação desejada, irá amenizar muitos dos problemas citados nos capítulos anteriores. A importância do modelo situa no fato de que não deva ser encarado como um fim em si mesmo, mas sim como sendo parte de um amplo programa para a melhoria da qualidade do software.

Para que o objetivo do trabalho seja atingido, torna-se necessário a definição do tipo de pesquisa a ser realizada, de forma a identificar a relação de interesse.

Segundo Aacker [Aacker,86], existem 3 tipos de estudos de pesquisa que possibilitam a análise das variáveis envolvidas em um dado problema:

1 - Pesquisa Exploratória

A pesquisa exploratória é aplicada a situações onde se possui poucas informações com relação ao que será estudado. O objetivo deste tipo de pesquisa é explorar o assunto, fornecendo certas informações sobre as variáveis desejadas. Dentre as situações onde a pesquisa exploratória pode ser usada encontra-se:

- Qual o novo produto que deve ser desenvolvido?

- Qual apelo publicitário que será efetivo na divulgação de um dado produto?

- Como um dado serviço pode ser melhorado?

2 - Pesquisa Descritiva

O objetivo da pesquisa descritiva é fornecer um quadro preciso da situação-problema. Este tipo de pesquisa funciona como uma fotografia da situação atual, ou seja, apenas descreve o que está acontecendo.

O resultado de uma pesquisa descritiva não permite analisar uma relação de causa-efeito. Por exemplo, um estudo de pesquisa exploratória indica que as vendas de uma dada empresa são maiores onde é maior o número de vendedores, em relação ao número de habitantes.

Este resultado NÃO nos permite concluir que aumentando o número de vendedores nos locais onde as vendas são baixas, aumentará a quantidade vendida. Muitos outros fatores podem estar envolvidos e por isso, só um estudo causal poderá fornecer a relação de causa-efeito desejada.

A pesquisa descritiva pode ser aplicada, entre outras, nas seguintes situações:

- Avaliação dos atributos de produtos competidores, sob o ponto de vista do cliente.

- Características demográficas e sócio-econômicas dos leitores de uma dada revista.

- A proporção de eleitores que irão votar em um dado candidato,na próxima eleição.

3 - Pesquisa Causal

Uma pesquisa causal deve ser usada quando se deseja provar que uma dada variável é a causa de outra variável. Este tipo de pesquisa consegue isolar as variáveis consideradas. Por esse motivo, é possível garantir que não existem outros fatores causais, que poderiam ter afetado a relação em estudo.

Algumas situações onde uma pesquisa causal pode ser usada, são listadas abaixo:

– Verificar se uma dada propaganda conseguirá aumentar as vendas de um dado produto.

– Verificar se a emissão de dióxido de carbono exerce influência sobre a temperatura da terra.

– Verificar se a utilização de uma metodologia de desenvolvimento influencia a qualidade do software produzido.

Os três tipos de pesquisas, descritos anteriormente, podem desempenhar papéis complementares em certos estudos. Neste caso, a pesquisa exploratória indicaria as variáveis envolvidas, fornecendo as informações necessárias ao conhecimento do problema. Após o estudo exploratório, poder-se-ia ser realizada uma pesquisa descritiva. Além de identificar relações entre algumas variáveis, esta pesquisa pode filtrar muitas das causas possíveis. Para se encontrar a causa(ou as causas) mais prováveis, deveria ser realizada uma pesquisa causal. A figura 15, ilustra o processo descrito.

Como ficou claro pela discussão anterior, o estudo necessário para se verificar a influência da metodologia sobre a qualidade do software, fará parte de uma pesquisa causal.

4.2. MODELO PROPOSTO

Existem algumas alternativas de procedimento para que se realize a pesquisa desejada, ou seja, para que se alcance o objetivo estabelecido.

A abordagem a ser adotada depende da realidade de cada empresa. O modelo proposto está baseado numa empresa que já utiliza uma metodologia de desenvolvimento e que possui sistemas instalados em diferentes usuários.

As etapas sugeridas pelo modelo são:

1 - Relacionar os Fatores de Qualidade de Software

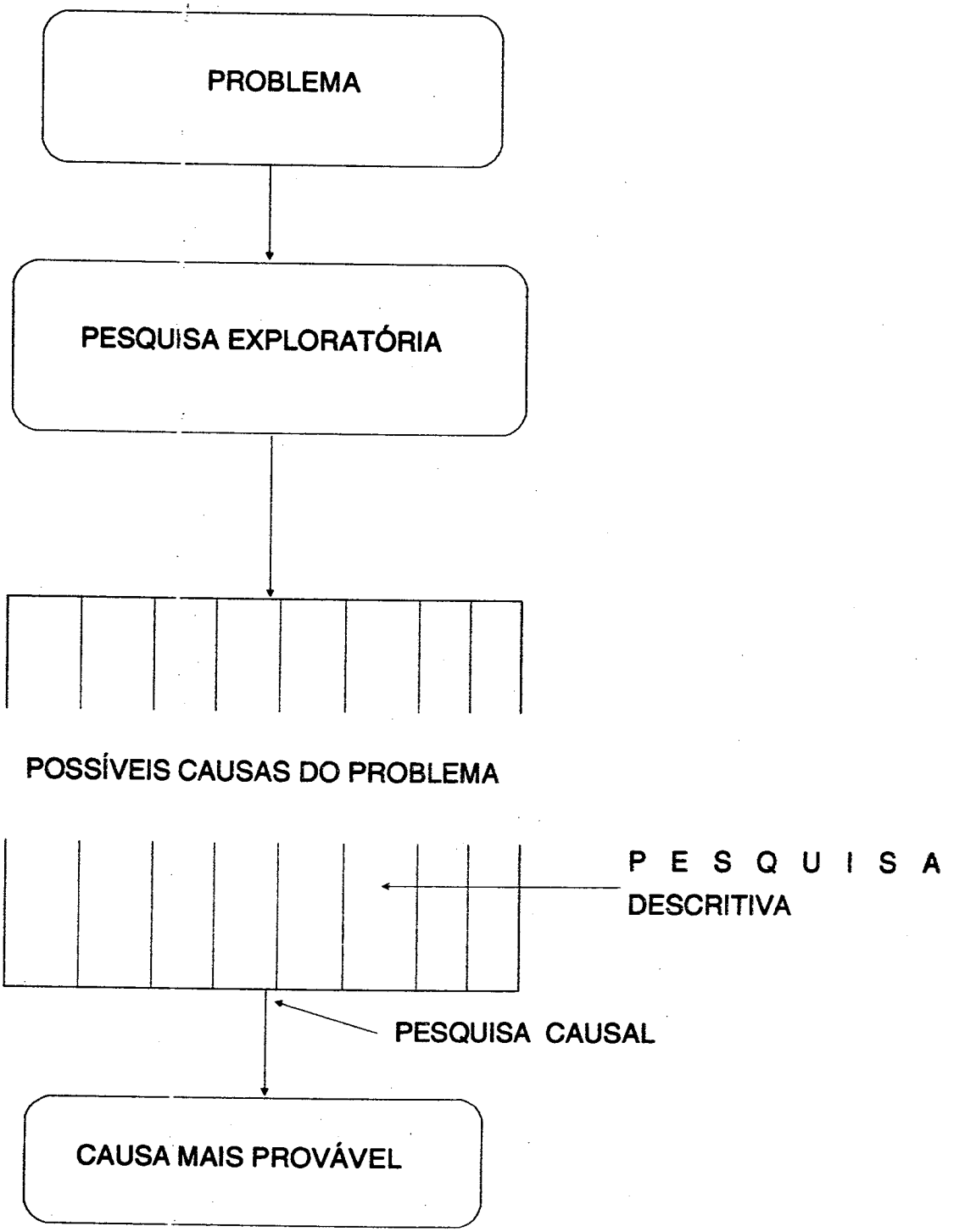

FIGURA 15 - PROCESSO DE IDENTIFICAÇÃO DA VARIÁVEL CAUSAL

2 - Quantificar a Qualidade do Software Desenvolvido

3 - Quantificar a Qualidade da Metodologia Utilizada

4 - Relacionar a qualidade da metodologia utilizada com a dos softwaresdesenvolvidos.

A idéia básica do modelo é analisar a alteração que sofre a qualidade dos softwares desenvolvidos, quando se realiza uma mudança na metodologia utilizada(substituição ou atualização).

Após algumas medidas, poder-se-á quantificar a relação existente entre qualidade de metodologia e qualidade de software. Por extensão, será possível verificar se o uso de uma metodologia de desenvolvimento melhora a qualidade do software produzido.

Os benefícios adicionais, a serem conseguidos, com a aplicação dos passos sugeridos pelo modelo, são:

- Obtenção de um diagnóstico da satisfação do cliente, com relação ao software desenvolvido. Este diagnóstico deve ser utilizado pela empresa como um mapa, o qual fornece o caminho para reverter as situações desfavoráveis.

- Identificação das deficiências do desenvolvimento, funcionando como um ótimo feed-back para futuros projetos.

- Identificação dos aspectos da metodologia que podem ser melhorados. Isso facilita bastante a alteração de uma dada metodologia, uma vez que os pontos deficientes já foram identificados.

- Melhoria na comunicação com o usuário, uma vez que se identifica as deficiências da documentação entregue ao usuário e\ou de analistas e programadores.

Portanto, com base no que foi comentado anteriormente, o próprio modelo torna-se um influenciador da qualidade do software desenvolvido.

A seguir, serão descritas, em detalhes, todas as etapas constantes do modelo. O contexto e as etapas do modelo são apresentados na figura 16.

4.2.1. RELACIONAR OS FATORES DE QUALIDADE DE SOFTWARE

O objetivo deste primeiro passo é, identificar os fatores que deveriam ser analisados na quantificação da qualidade de software. Segundo o que foi salientado no capítulo 3, a qualidade de um produto deve sempre ser analisada sob os enfoques interno e externo. Isso significa analisar a qualidade sob o ponto de vista de padrões e medidas internas e sob o ponto de vista do usuário.

Alternativamente, pode-se analisar a qualidade do software apenas sob o enfoque externo. Isso se deve ao fato de que o modelo proposto, como foi salientado anteriormente, deve fazer parte de um programa para a melhoria da qualidade do software. O primeiro passo de um programa dessa natureza é a identificação da qualidade do produto, sob o ponto de vista do usuário ou cliente.

Este primeiro passo fornece, portanto, a base suficiente para a definição das diretrizes do programa de qualidade. Além disso, obtem-se as informações necessárias para que se alcance os objetivos propostos no presente trabalho.

Existem muitos fatores de qualidade que podem ser identificados pelo usuário. Entre os mais importantes, pode-se citar:

- Corretismo: Esse fator identifica o grau no qual o software realiza, corretamente, as especificações do usuário.
- Integridade: Diz respeito a quanto o software está protegido de falhas de segurança.
- Confiabilidade: Está relacionado com o grau de confiança que se pode depositar nos resultados fornecidos pelo software.
- Capacidade de Utilização: Relaciona-se com o esforço requerido para aprender, operar e utilizar o sistema.

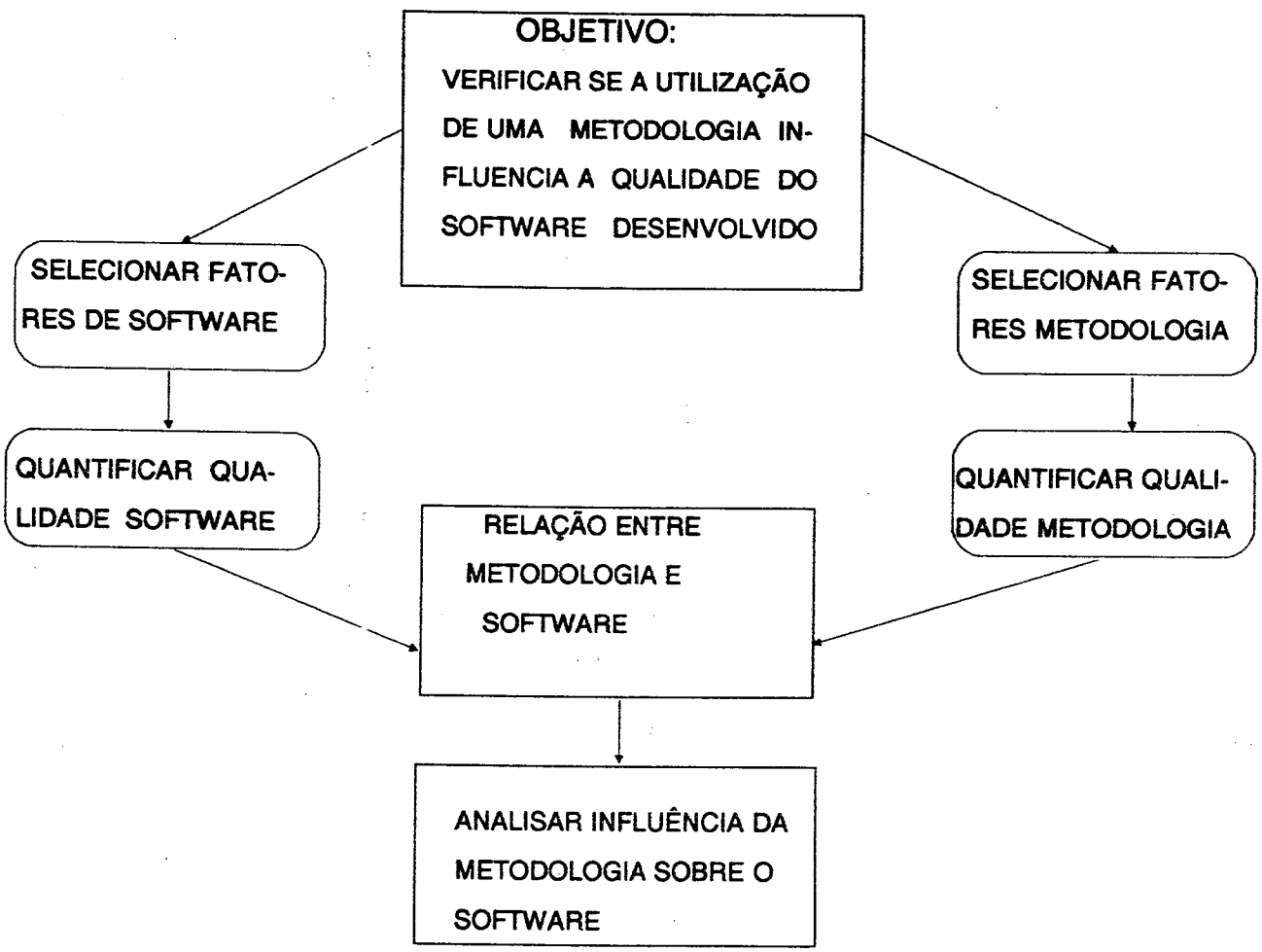

FIGURA 16 - CONTEXTO DO MODELO PROPOSTO

– Manutibilidade: Está relacionado com a frequência necessário à manutenção do software.

– - Tolerância aos Erros: É necessidade de verificar se os erros ocorridos afetam a operação do sistema, ou se é possivel continuar com a operação, mesmo com a ocorrência do erro.

Portanto, este primeiro passo faz uma descrição dos pontos fundamentais da qualidade do software. Naturalmente, dependendo de cada caso, pode-se acrescentar ou eliminar fatores de qualidade, de forma a adequar o modelo à situação em estudo.

4.2.2. QUANTIFICAR A QUALIDADE DO SOFTWARE DESENVOLVIDO

Este passo é composto por três atividades sequenciais, as quais serão descritas abaixo.

1 - Elaborar um Questionário para Quantificar a Qualidade do Software

Após a seleção dos fatores de qualidade desejados, deve-se elaborar um questionário para ser aplicado junto ao usuário. Neste questionário, todos os fatores devem ser analisados em pelo menos uma pergunta.

Torna-se importante ressaltar que, ao elaborar as perguntas deve-se pensar nos tipos de respostas que se espera.

Segundo Aacker [Aacker,86], existem dois tipos de respostas:

– Respostas Abertas:Este tipo de resposta ocorre quando o respondente tem total liberdade para falar, não sendo necessário que se encaixe em faixas ou classes. A vantagem deste tipo de resposta, é que o usuário não fica influenciado pelas classes, ou faixas existentes. Por outro lado, este tipo de resposta gera dificuldades para anotação, o que pode prejudicar o resultado da pesquisa. Além disso, a análise dos resultados pode ser grandemente dificultada, uma vez que o respondente fala livremente sobre o assunto em questão.

– Respostas Fechadas:Neste caso, são fornecidas ao respondente algumas respostas possíveis. Estas respostas podem ter a forma de uma lista com todas as rspostas plausíveis ou uma escala de valores, variando de "totalmente favorável" a "completamente contrário"

As vantagens desse tipo de respostas são:

– As questões são fáceis de serem respondidas

– Requer menor esforço do entrevistador

– Facilita a análise dos resultados

– Poucas chances de respostas erradas

A limitação desse tipo de resposta está na necessidade de uma pesquisa exploratória, ou seja, é necessário um grande conhecimento do assunto. Além disso, a rigidez das respostas pode limitar a espontaneidade do respondente.

Aacker [Aacker,86] salienta que antes de se escolher o tipo de resposta, deve-se pensar em como elas serão analisadas. Para os objetivos deste trabalho, as respostas fechadas parecem ser mais interessantes.

2 - Realizar a Pesquisa Junto ao Usuário

O questionário elaborado anteriormente, deve ser aplicado junto ao usuário, de forma a quantificar a qualidade do software desenvolvido. Existem três meios básicos para a aplicação de um dado questionário, a saber: pelo correio, por telefone ou pessoalmente. A escolha de um dado método, deve ser feita em função do objetivo da empresa e da natureza da pesquisa. As vantagens de cada método são descritas a seguir:

- ENTREVISTA PESSOAL

VANTAGENS:

- O entrevistador pode, se for o caso, utilizar de artifícios para aumentar o interesse do usuário para com a entrevista.

- Pode-se formular perguntas complexas, pois o entrevistador pode desfazer as masinterpretações.

- Coleta uma grande quantidade de informação, além da impressão pessoal do pesquisador.

- Bastante flexível quanto a ordem de se formular as perguntas.

DESVANTAGENS:

- Consome muito tempo e recursos.

- A maior parte do tempo é improdutiva

- O controle administrativo é difícil.

- **ENTREVISTA POR TELEFONE**

VANTAGENS:

- Maior número de entrevistados por unidade de tempo

- Baixos custos

- Pode ser repetida várias vezes no mesmo dia

- Melhor controle administrativo

DESVANTAGENS:

– Não pode tratar de temas complexos

– Não se utiliza de imagens

– O tempo necessário a cada entrevista não pode ser grande.

– **ENTREVISTA PELO CORREIO**

VANTAGENS:

– Melhores resultados

– Baixos custos

DESVANTAGENS:

– Esclarecimentos não podem ser feitos

– O tempo de resposta geralmente é grande

– A ordem de formulação das perguntas, geralmente, não são seguidas pelos clientes

– Questionários longos tendem a não ser respondido

– A quantidade de resposta é pequena

Com a aplicação do questionário, torna-se possível a análise dos dados obtidos.

3 - Analisar os Resultados Obtidos

Na verdade, o tipo de análise a ser realizada, deve ser definido antes da definição do tipo de resposta esperada. Uma análise possível para o questionário mencionado anteriormente, pode ser:

– Associar uma nota a cada resposta:

No caso de respostas fechadas seria simples, cada classe corresponderia a uma nota. Exemplo:

3 -- BOM

2 -- REGULAR

1 -- RUIM

Para respostas abertas, deve-se procurar outros tipos de relacionamentos, para se conseguir a quantificação das respostas em termos de notas.

– Determinar a importância de cada fator

Deve-se determinar, sob o ponto de vista do usuário, o grau de importância relativa de cada fator de qualidade.

– Calcular a média dos fatores

Calcula-se a média ponderada dos fatores de qualidade, usando como fatores de ponderação o grau de importância de cada fator.

4 - Calcular a qualidade do software

A média ponderada calculada é uma medida da qualidade do software, sob o ponto de vista do usuário.

Após a conclusão deste passo, ter-se-á uma descrição completa dos pontos fortes e fracos dos softwares desenvolvidos. Como estes softwares foram desenvolvidos através da utilização de uma metodologia, será possível estabelecer uma primeira relação entre metodologia e qualidade de software.

4.2.3. QUANTIFICAR A QUALIDADE DA METODOLOGIA UTILIZADA

Até aqui, foi quantificada apenas a qualidade do software. Mas, para que se estabeleça uma relação entre a utilização de uma metodologia e qualidade de software, faz-se necessário o cálculo da qualidade da metodologia utilizada. A justificativa para este cálculo baseia-se em dois motivos:

- Verificar em que medida a metodologia atinge seu objetivo, ou seja, em que grau consegue fornecer apoio para a construção, verificação, validação e modificação de um dado software.

- Quantificar o grau de influência da utilização de uma metodologia sobre a qualidade do software. Isto só será possível, como foi salientado, após a medição de algumas versões da metodologia e da qualidade dos softwares produzidos.

A avaliação da qualidade da metodologia, será feita através da adaptação do modelo proposto por Rocha [Rocha,87] para a avaliação da qualidade do software. Esta adaptação não deve trazer prejuízos para a avaliação da qualidade da metodologia. Como poderá ser observado a seguir, o modelo proposto por Rocha, proporciona a base para o cálculo da qualidade de um produto, seja ele software ou metodologia.

O referido modelo está baseado nos seguintes conceitos:

1 - Objetivos de Qualidade: São propriedades gerais que um produto deve possuir.

2 - Fatores de Qualidade: São os atributos ou características, que o produto deve possuir.

3 - Critérios: São os atributos primitivos e portanto, passíveis de serem quantificados.

4 - Processos de Avaliação : Determinam o processo e os instrumentos a serem usados, para se medir o grau de presença de um dado atributo.

5 - Medidas: Consiste no resultado da avaliação do produto segundo os critérios.

6 - Medidas Agregadas: É o resultado da agregação das medidas, segundo os critérios. Estas medidas agregadas quantificam os fatores

A figura 17, ilustra o modelo descrito anteriormente.

Através da pesquisa de campo realizada e da consulta à literatura especializada, foi possível criar objetivos, fatores e subfatores de qualidade, que afetam a qualidade global da metodologia. A figura 18, apresenta os objetivos de qualidade criados, bem como as respectivas subdivisões em fatores e subfatores.

Cada ítem constante da figura 18 apresentada, será descrito em detalhes a seguir.

Os **OBJETIVOS** de qualidade de uma metodologia de desenvolvimento de software são:

- APLICABILIDADE: A metodologia deve se adaptar às características do projeto, no qual for utilizada. Além disso, seus usuários devem compreender totalmente o seu conteúdo.

- ESTRUTURA DIDÁTICA: A metodologia deve ser apresentada numa forma de fácil utilização, possuindo uma linguagem que facilite a compreensão

Cada um dos objetivos de qualidade apresentados é composto de **FATORES DE QUALIDADE**.

A **APLICABILIDADE** é composta por:

- Compreensibilidade:

Está relacionado com a facilidade com que o usuário compreende os conceitos contidos na metodologia.

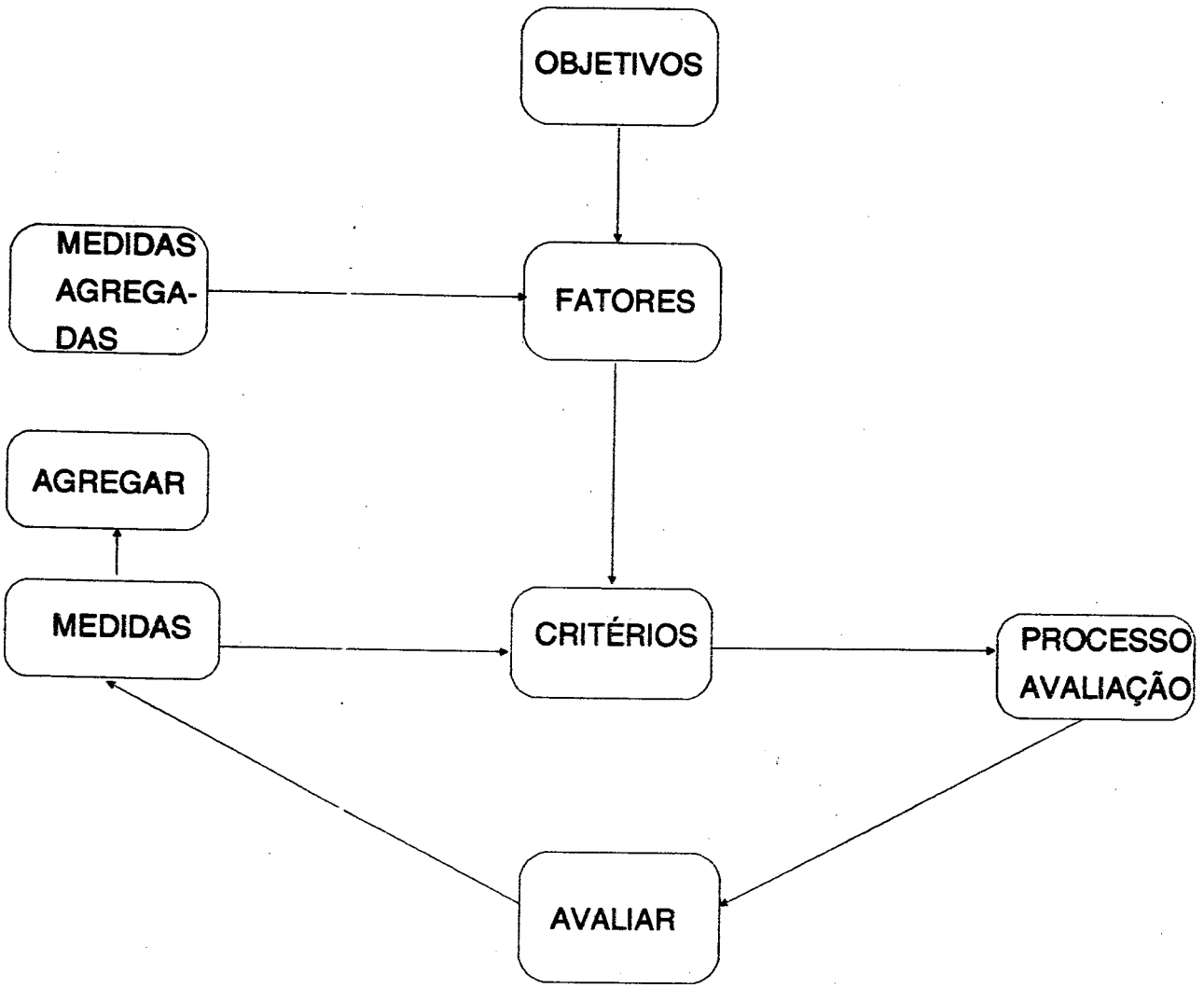

FIGURA 17 - MOD. DE AVALIAÇÃO DA QUALIDADE DA METODOLOGIA

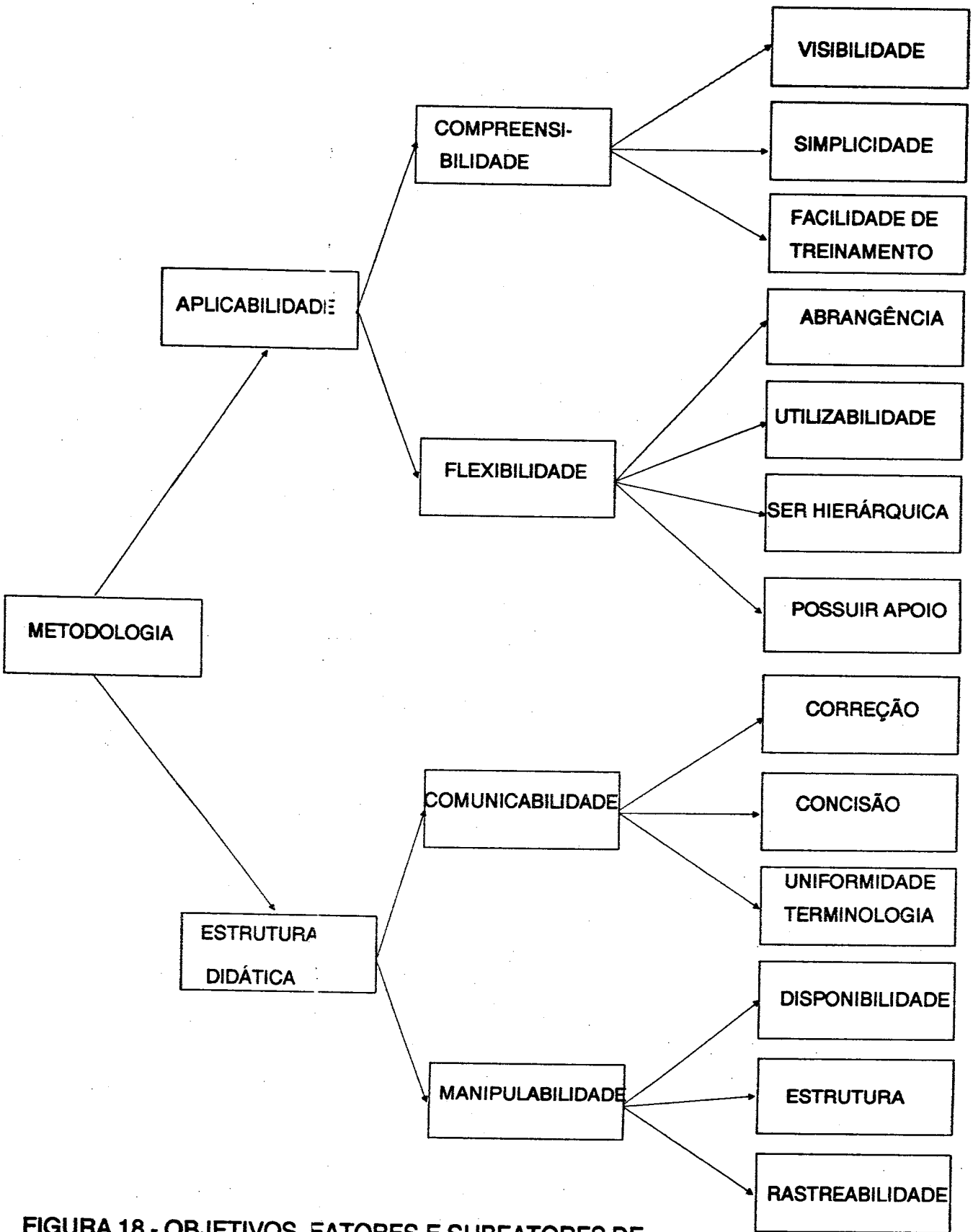

FIGURA 18 - OBJETIVOS, FATORES E SUBFATORES DE QUALIDADE DA METODOLOGIA

– Flexibilidade:

Diz respeito à facilidade com que a metodologia se adapta às características dos diferentes projetos.

A **ESTRUTURA DIDÁTICA** se divide em:

– COMUNICABILIDADE: As informações contidas na metodologia devem ser escritas em uma linguagem facilmente entendida pelos usuários.

– MANIPULABILIDADE: A forma de apresentação da metodologia deve facilitar sua aplicação num projeto real.

Os **FATORES** apresentados, por sua vez, são divididos em **SUBFATORES DE QUALIDADE**.

O fator **COMPREENSIBILIDADE** se divide em:

– VISIBILIDADE: As pessoas que trabalham com a metodologia devem conseguir entender como ela funciona, como controlá-la e qual sua abrangência.

– SIMPLICIDADE: Avalia a facilidade de entendimento da documentação entregue ao usuário.

– FACILIDADE DE TREINAMENTO: avalia os recursos disponíveis e o esforço necessário para o treinamento do pessoal que utilizará a metodologia.

A **FLEXIBILIDADE** é composta pelos seguintes subfatores:

– ABRANGÊNCIA: Avalia se a metodologia engloba todo o ciclo de vida do software.

– UTILIZABILIDADE: Está relacionada com a facilidade da metodologia ser usada em projetos com características diferentes.

– SER HIERÁRQUICA: Avalia se as ferramentas utilizadas pela metodologia permitem decomposições sucessivas e a correta vinculação destas decomposições com as visões abrangentes existentes.

– POSSUIR APOIO: Diz respeito ao apoio dado pela metodologia à:

– Criação de especificações.

– Verificação e validação das especificações e dos produtos

– Modificação de especificações e produtos.

– Gerência do processo de desenvolvimento.

Os subfatores que compõem o fator **COMUNICABILIDADE** são:

– CORREÇÃO: Está relacionado com a notação, sintaxe e semântica da linguagem utilizada pela metodologia, além do formato da documentação.

– CONCISÃO: Avalia o volume de texto a ser manipulado, em comparação com o volume de informações fornecido.

– UNIFORMIDADE DE TERMINOLOGIA: Avalia se os termos são utilizados com o mesmo significado ao longo de toda metodologia.

A **MANIPULABILIDADE** é composta por:

– DISPONIBILIDADE: Avalia a facilidade de acesso à versão mais nova da metodologia.

– ESTRUTURA: Está relacionado com a facilidade de localização de um dado aspecto no texto da metodologia.

Torna-se importante observar que a construção deste modelo difere um pouco daquele proposto por Rocha [Rocha,83]. A quantificação dos fatores e subfatores de qualidade será feita através de perguntas de quantificação, ao invés de ser feita a partir da quantificação dos critérios.

As perguntas de quantificação têm o objetivo de avaliar o grau de presença de um determinado aspecto de um dado subfator. Cada pergunta será classificada em notas que variam de 1 a 5, de acordo com a percepção do usuário da metodologia. Nesse sentido, uma nota 5 para uma dada questão significa que aquele aspecto, do subfator considerado, está muito bem desenvolvido na metodologia. Abaixo, é descrito o significado de cada nota:

1 - PÉSSIMO

2 - RUIM

3 - REGULAR

4 - BOM

5 - ÓTIMO

A nota de um dado subfator pode ser calculada pela média aritmética simples das notas dadas às questões, relacionadas a esse subfator. Evidentemente, cada empresa deve desenvolver um processo de avaliação que melhor se adapte às suas características. A abordagem decrita é apenas uma alternativa, dentre as muitas possíveis.

O cálculo da medida da qualidade da metodologia pode ser feito seguindo-se os passos a seguir.

1 - Após o cálculo das notas dos subfatores, definir a importância relativa de um dado subfator em relação aos outros. Isso pode ser feito através dos seguintes pesos:

1 - NÃO MUITO IMPORTANTE

2 - IMPORTANTE

3 - MUITO IMPORTANTE

2 - A nota de cada fator será obtida pela média ponderada das notas dos subfatores correspondentes. O fator de ponderação será a importância relativa de cada subfator(3,2 ou 1).

3 - A nota de cada objetivo de qualidade será obtida pela média ponderada das notas dos fatores correspondentes. O fator de ponderação será a importância relativa de cada fator, de forma análoga ao que foi feito para os subfatores.

4 - Com a nota de cada objetivo de qualidade é possível calcular a medida da qualidade da metodologia. Esta será obtida através da média aritmética simples da nota de cada objetivo. A justificativa de se utilizar a média simples é que os objetivos de qualidade possuem a mesma importância relativa.

5 - Cada empresa deve definir o nível de qualidade aceitável da metodologia utilizada.

Torna-se importante observar que a quantificação dos subfatores fornece um mapeamento da qualidade da metodologia, o qual será de grande auxílio quando se desejar implantar uma nova versão da metodologia. As modificações poderiam começar pelos subfatores que apresentassem as menores notas.

Seguindo, novamente, a filosofia proposta no capítulo 2, a qualidade da metodologia deve ser calculada sob o ponto de vista interno e sob o ponto de vista externo.

Anteriormente, foi descrita com detalhes a avaliação interna da qualidade da metodologia. Para se avaliar a qualidade sob o ponto de vista externo, deve-se proceder analogamente ao que foi feito para o software. Esse procedimento inclui:

1 - Relacionar os Fatores de Qualidade da Metodologia

O único aspecto da metodologia com o qual o usuário tem contato é a documentação que lhe é entregue. Portanto, para se quantificar a qualidade da metodologia, sob o ponto de vista do usuário, deve-se avaliar a qualidade da documentação entregue. Os fatores de qualidade a serem observados são:

- Compreensibilidade: Relaciona-se com a facilidade de se compreender o documento como um todo.

- Simplicidade da Terminologia: Verifica se os termos utilizados nos documentos, são facilmente entendidos pelos usuários

- Organização da Informação: Está relacionado com a sequência na qual as informações são fornecidas. Verifica se existe uma sequência lógica das informações.

- Rastreabilidade: Facilidade de se encontrar um dado aspecto de interesse do usuário.

2 - Quantificar a Qualidade da Metodologia Utilizada

Analogamente ao que foi feito para o software, deve-se formular perguntas para cada um dos fatores descritos. Além disso, o tipo de resposta esperado deve ser definido anteriormente.

Torna-se interessante que esse questionário seja similar ao que foi elaborado para o software, uma vez que eles podem ser aplicados em conjunto.

Esta abordagem deve ser adotada para se obter, de uma só vez, o diagnóstico completo da qualidade do produto desenvolvido pela empresa. A documentação entregue, a parte visível da metodologia ao usuário, foi tratada como produto, pois é parte integrante do software.

No capítulo 5, o modelo apresentado anteriormente, será aplicado a uma situação real.

CAPÍTULO 5

5. APLICAÇÃO DO MODELO

5.1. INTRODUÇÃO

No presente capítulo, serão apresentados os resultados obtidos com a aplicação do modelo proposto no capítulo 4. Esses resultados práticos são essenciais para que se possa avaliar se o modelo atinge, ou não, os objetivos estabelecidos.

O referido modelo, foi aplicado na **NTS - Núcleo de Tecnologia de Software LTDA.**

A **NTS**, fundada em 1984, é uma empresa que atua no setor de informática, prestando serviços predominantemente nas áreas de consultoria, treinamento e desenvolvimento de sistemas de informações. Faz parte de seu espectro de atividades, o desenvolvimento de produtos de software/hardware integrados, com objetivo de atender às necessidades-fim das organizações.

As nove unidades da **NTS** estão instaladas em aproximadamente 9.000 m2, abrigando 06 Doutores, 10 Mestres, 14 Especialistas, 85 Graduados e 22 Técnicos.

No conjunto das atividades propostas a seus clientes, a **NTS** tem como meta de trabalho proporcionar um nível elevado de qualidade nos serviços prestados, a um preço aceitável.

Em função disto, é que surgiu o interesse em se quantificar a influência da utilização de uma metodologia sobre a qualidade do software desenvolvido. O modelo a ser utilizado com tal objetivo foi apresentado no capítulo 4.

O referido modelo sugere a aplicação dos seguintes questionários:

- aos clientes da empresa, a fim de se quantificar a qualidade do software desenvolvido.

- aos analistas da empresa, com o objetivo de se quantificar a qualidade da metodologia utilizada.

A aplicação destes questionários, forneceu informações detalhadas sobre os problemas enfrentados no desenvolvimento de software. Estas informações são muito importantes, uma vez que existem poucas pesquisas de campo sobre a realidade brasileira.

A realidade encontrada, parece confirmar a situação descrita por livros e artigos americanos. Isto ficará claro, a partir da apresentação dos dados obtidos.

5.2. QUALIDADE DO SOFTWARE

Seguindo o que foi proposto pelo modelo, elaborou-se um questionário para se quantificar a qualidade do software, sob o ponto de vista do usuário, baseado nos fatores de qualidade descritos no capítulo 4.

A característica mais marcante da aplicação do questionário, foi a receptividade dos clientes. O fato da empresa fornecedora preocupar-se com a satisfação de seus clientes, é extremamente positivo para a imagem da mesma. De maneira geral, o cliente não está acostumado a ter sua satisfação avaliada, com relação a um dado produto ou serviço.

Toda esta situação pode ser retratada na declaração de um cliente:

> ***"Aqui no Brasil, é comum pagarmos preço de Primeiro Mundo por produtos de Terceiro. Já estava na hora de alguém nos dar um atendimento de Primeiro Mundo."***

Isto demonstra uma grande **"demanda reprimida"** por bom atendimento, o qual, segundo Júlio Lobos [LOBOS,91], contitui um dos componentes da qualidade.

Por questões de custo e tempo, limitou-se a aplicação do questionário aos clientes da Grande Florianópolis. Em função disto, foi possível obter os dados desejados através de entrevista pessoal com cada cliente.

A escolha da entrevista pessoal como método de aplicação do questionário deveu-se, também, ao fato de ser possível obter valiosas informações adicionais sobre a satisfação dos clientes.

Além de perguntas sobre os fatores de qualidade do software propriamente dito, foram incluídas perguntas sobre a qualidade da documentação entregue durante o desenvolvimento, a qualidade do manual do usuário, o treinamento oferecido, e a assistência técnica existente.

Este procedimento foi adotado para se obter um diagnóstico completo de todos os aspectos relacionados ao desenvolvimento, implantação e manutenção de um sistema de processamento de dados. Além disso, desta maneira tem-se uma primeira avaliação da qualidade da metodologia utilizada, quantificando-se a qualidade da documentação entregue ao cliente.

Cada fator de qualidade, relacionado diretamente ao software ou não, foi avaliado por pelo menos uma pergunta, denominadas perguntas de quantificação.

Com o objetivo de facilitar a análise dos dados, optou-se por respostas do tipo fechada. Para isto, foram criadas quatro respostas possíveis:

A --- BOM

B --- REGULAR

C --- RUIM

D --- NÃO SE APLICA

As poucas opções de resposta não prejudicou a avaliação da satisfação do cliente, pois o método de aplicação do questionário foi a entrevista pessoal. Este método possibilitou a formulação de perguntas adicionais a fim de se identificar, perfeitamente, a realidade de um dado cliente. Por este motivo, é muito importante a habilidade do entrevistador em conduzir o entrevistado, de forma a obter respostas que representem o ponto de vista do cliente, com relação a um dado fator de qualidade.

Baseado no modelo proposto, e nas considerações feitas anteriormente, elaborou-se o questionário para a quantificação da qualidade dos softwares desenvolvidos, o qual é mostrado no Apêndice 1.

A relação entre os fatores de qualidade e as perguntas que quantificam estes fatores, pode ser visualizada na tabela 3, apresentada a seguir.

FATOR DE QUALIDADE	PERGUNTA(S) DE QUANTIFICAÇÃO
PRAZO	1
DOCUMENTAÇÃO ENTREGUE	2, 3, 4
MANUAL DO USUÁRIO	5, 6, 7, 8
TREINAMENTO	9, 10, 11, 12
SATISFAÇÃO	13
CONFIABILIDADE	14
FACILIDADE DE OPERAÇÃO	15
MANUTIBILIDADE	16, 17
ASSISTÊNCIA TÉCNICA	18

TABELA 3 - RELAÇÃO FATORES x PERGUNTAS DE QUANTIFICAÇÃO

Através da análise dos dados obtidos foi possível, como era esperado, fazer um mapeamento da qualidade do software, identificando-se os pontos positivos e negativos dos softwares desenvolvidos. Para que esta análise fosse possível, realizou-se uma sequência de passos, a qual será descrita a seguir.

5.2.1. CLASSIFICAÇÃO DOS DADOS

Para efeitos de quantificação dos fatores de qualidade, foram desconsideradas as respostas "D - Não se aplica". Tal medida foi adotada pois, quando o cliente optou por essa resposta, deveu-se à um acordo entre contratante e contratada, não resultando em diminuição da qualidade do software desenvolvido. Por outro lado, estas respostas foram levadas em consideração, a fim de se identificar pontos-potenciais para melhoria do atendimento ao cliente.

5.2.2. QUANTIFICAÇÃO DAS RESPOSTAS

A fim de se quantificar os fatores de qualidade, passo essencial para o cálculo da qualidade do software, torna-se necessário quantificar as perguntas relacionadas a cada fator. Para que isto fosse possível, associou-se uma nota para cada tipo de resposta existente, ou seja:

RESPOSTA	NOTA
A	3
B	2
C	1

Calculou-se assim a nota média de cada pergunta. Esta nota média foi obtida através de média ponderada das notas de cada resposta, utilizando-se como fator de ponderação o número de respondentes.

5.2.3. QUANTIFICAÇÃO DOS FATORES DE QUALIDADE

Cada fator de qualidade é avaliado por uma ou mais perguntas, chamadas de perguntas de quantificação.

Dado um fator de qualidade, sua quantificação é feita através da média simples das notas-médias de suas respectivas perguntas de quantificação.

Utilizou-se a média simples pois, acreditou-se que as perguntas de quantificação de um mesmo fator possuíam o mesmo grau de importância. Isto ocorre porque cada pergunta de quantificação enfoca um dado aspecto do fator de qualidade, que não é nem mais nem menos importante que os outros.

Caso entenda-se que existam diferenças de importância entre as perguntas, de um dado fator de qualidade, pode-se dar pesos diferentes a cada uma delas.

Com isto, calcular-se-ia a média ponderada das notas médias das perguntas de quantificação, a fim de se quantificar um dado fator de qualidade.

5.2.4. QUANTIFICAR A QUALIDADE DO SOFTWARE

Para se calcular a qualidade do software, deve-se calcular a média entre as notas dos fatores de qualidade. Mas, os fatores de qualidade, diferentemente das perguntas de quantificação, possuem graus de importância diferentes. Portanto, faz-se necessário o uso de média ponderada, utilizando como fator de ponderação a importância de cada fator.

O fator de ponderação (peso de cada fator), foi obtido através de uma entrevista com os clientes. Através desta entrevista, observou-se que os fatores relacionados diretamente ao software, foram considerados mais importantes. A tabela 4, mostra o peso de cada fator, de acordo com o ponto de vista dos clientes.

Os pesos dos fatores de qualidade, tendem a ser diferentes para empresas distintas. Por este motivo, é recomendável realizar uma consulta aos clientes, para se saber o peso de cada fator.

FATOR DE QUALIDADE	IMPORTÂNCIA RELATIVA (PESO)
PRAZO	2
DOCUMENTAÇÃO ENTREGUE	1
MANUAL DO USUÁRIO	1
TREINAMENTO	1
SATISFAÇÃO	3
CONFIABILIDADE	3
FACILIDADE DE OPERAÇÃO	3
MANUTIBILIDADE	3
ASSISTÊNCIA TÉCNICA	2

TABELA 4 - RELAÇÃO FATORES x IMPORTÂNCIA RELATIVA (PESO)

Com isto, será possível determinar a distribuição de qualidade dos softwares desenvolvidos, através da utilização da versão 2.0, da metodologia de desenvolvimento e manutenção da NTS.

5.2.5. RESULTADOS OBTIDOS

Em primeiro lugar, observa-se que a análise da distribuição das médias de qualidade, fornece uma descrição adequada da satisfação do cliente com relação ao software desenvolvido. Deduziu-se tal fato ao comparar-se o valor obtido pelo software

com o total de pontos possíveis, o qual permite também verificar se existem problemas de insatisfação dos clientes.

Apesar disto, deve-se pesquisar técnicas que otimizem a confiabilidade dos valores obtidos. Uma alternativa viável neste sentido é a utilização da teoria de Conjuntos Difusos, conforme sugerido por Paladini [PALADINI, 92].

Torna-se importante salientar que, geralmente, altos valores para a qualidade originaram-se nos dados obtidos de usuários/clientes que entendiam de informática. Normalmente, estas pessoas participaram do desenvolvimento do projeto e sentem-se como "pais" do software. Qualquer crítica ao software, funciona como crítica a elas próprias. Portanto, elas tendem a minimizar os problemas, evidenciando as qualidades do sistema.

Por outro lado, os valores mais baixos para qualidade surgiram de clientes/usuários leigos em informática. Neste caso, a comunicação entre analistas e usuários foi o fator limitante. Os clientes não sabiam o que estava sendo desenvolvido mas, os analistas tinham certeza que estavam desenvolvendo um software de alta qualidade.

Este conflito está caracterizado nas declarações coletadas durante a pesquisa de campo. A tabela 5, mostrada a seguir, apresenta estas declarações.

Todas essas contradições afetam frontalmente, a qualidade do software, pois é com base na comunicação analista/usuário que o mesmo é desenvolvido.

Este é um dos aspectos da importância do presente trabalho. Pode-se verificar, com maior precisão, em que medida a mudança de um dado fator de qualidade da metodologia e que afeta a comunicação entre usuário e analista, irá influenciar a qualidade do software desenvolvido.

A distribuição das médias de qualidade, obtidas junto aos clientes, pode ser vista na figura 19. As médias de qualidade foram transformadas, para maior clareza de observação, numa escala de notas variando entre 0 e 10. Neste sentido, uma média

ANALISTAS	USUÁRIO
"O CLIENTE NÃO SABE O QUE QUER" "A REALIDADE DELE (CLIENTE) É BASTANTE DIFERENTE" "O CLIENTE VIVE MUDANDO DE OPINIÃO"	"NÃO ERA ISSO QUE EU ESPERAVA" "A MINHA REALIDADE NÃO DIFERE EM NADA DAS OUTRAS EMPRESAS" "DURANTE O DESENVOLVIMENTO, PERCEBO POSSIBILIDADES, NAS QUAIS NÃO HAVIA PENSADO ANTES"

TABELA 5 - DIVERGÊNCIAS ENTRE ANALISTAS E USUÁRIOS

igual a 6,00, indica que o software considerado atingiu 60% do total dos pontos possíveis.

Observando-se a figura 19, nota-se que existem dois patamares de qualidade. Um patamar mais baixo, variando de 3,50 a 5,00 e outro patamar mais alto, variando de 8,50 a 10,00.

Além disso, existe um valor bastante baixo, qualidade = 2,50, que não se encaixa em nenhum dos patamares existentes. Este valor mais baixo, está relacionado com um software que o usuário adquiriu de outra empresa e solicitou à NTS, adaptação do mesmo para sua realidade. Esta peculiaridade pode ter sido uma das causas da baixa qualidade deste software em relação aos demais.

Existe uma faixa de valores muito grande entre os patamares mencionados anteriormente. Esta faixa é denominada de "Faixa de Transição". Devido ao pequeno número de entrevistados, não se obteve nenhuma informação acerca da mesma. Outra

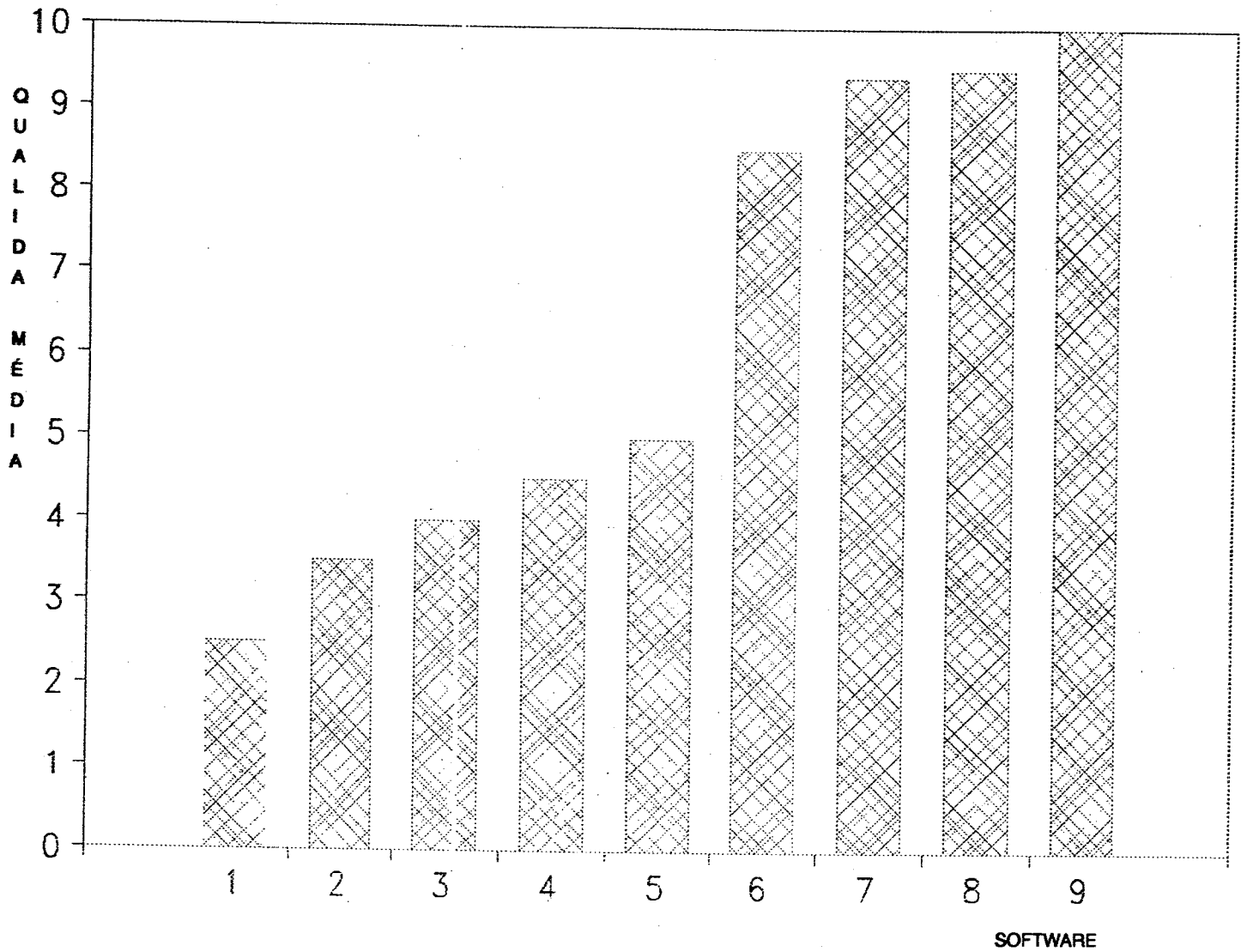

FIGURA 19 - QUALIDADE MÉDIA DOS SOFTWARES PESQUISADOS

justificativa para este fato é que não existe, para a grande maioria dos usuários, "meia-qualidade". Em outras palavras, o sistema é de boa qualidade ou é ruim, não existindo meio termo.

Com base nas informações adicionais, obtidas durante a pesquisa de campo, verificou-se que:

- Os softwares cujas médias pertencem ao patamar mais baixo, estão relacionados com clientes insatisfeitos.

- Os softwares cujas médias pertencem ao patamar mais alto, estão relacionados com clientes satisfeitos.

- As médias pertencentes à "Faixa de Transição", correspondem a softwares de qualidade regular.

Em função destes resultados, pode-se estabelecer o Limite de Qualidade Aceitável como sendo igual a 8,00. Isto significa dizer que, um software cuja média de qualidade seja inferior a 8,00, está abaixo do padrão mínimo de qualidade. Evidentemente, esta nota também denominada Nota de Corte, varia de empresa para empresa, dependendo da política de qualidade adotada.

No presente trabalho, decidiu-se por uma Nota de Corte igual a 8,00, pelo fato desta nota representar a nota mínima dada pelos clientes bastante satisfeitos com os softwares adquiridos.

Além da média de qualidade de cada software, foi possível verificar o desempenho individual de cada fator de qualidade. A figura 20 mostra este desempenho.

Com isto, detalha-se todo o mapa de qualidade do software, sob o ponto de vista do usuário. A partir deste mapa, é possível identificar os fatores que necessitam ser melhorados. Este procedimento já está sendo realizado atualmente na NTS.

Analisando-se o desempenho obtido pelos fatores de qualidade do software, pode-se obter valiosas informações, as quais justificam as médias de qualidade obtidas pelos softwares.

1 - PRAZO
2 -DOCUMENTAÇÃO ENTREGUE
3 - MANUAL DO USUÁRIO
4 - TREINAMENTO
5 - SATISFAÇÃO
6 - CONFIABILIDADE
7 - FACILIDADE DE OPERAÇÃO
8 - MANUTIBILIDADE
9 - ASSISTÊNCIA TÉCNICA

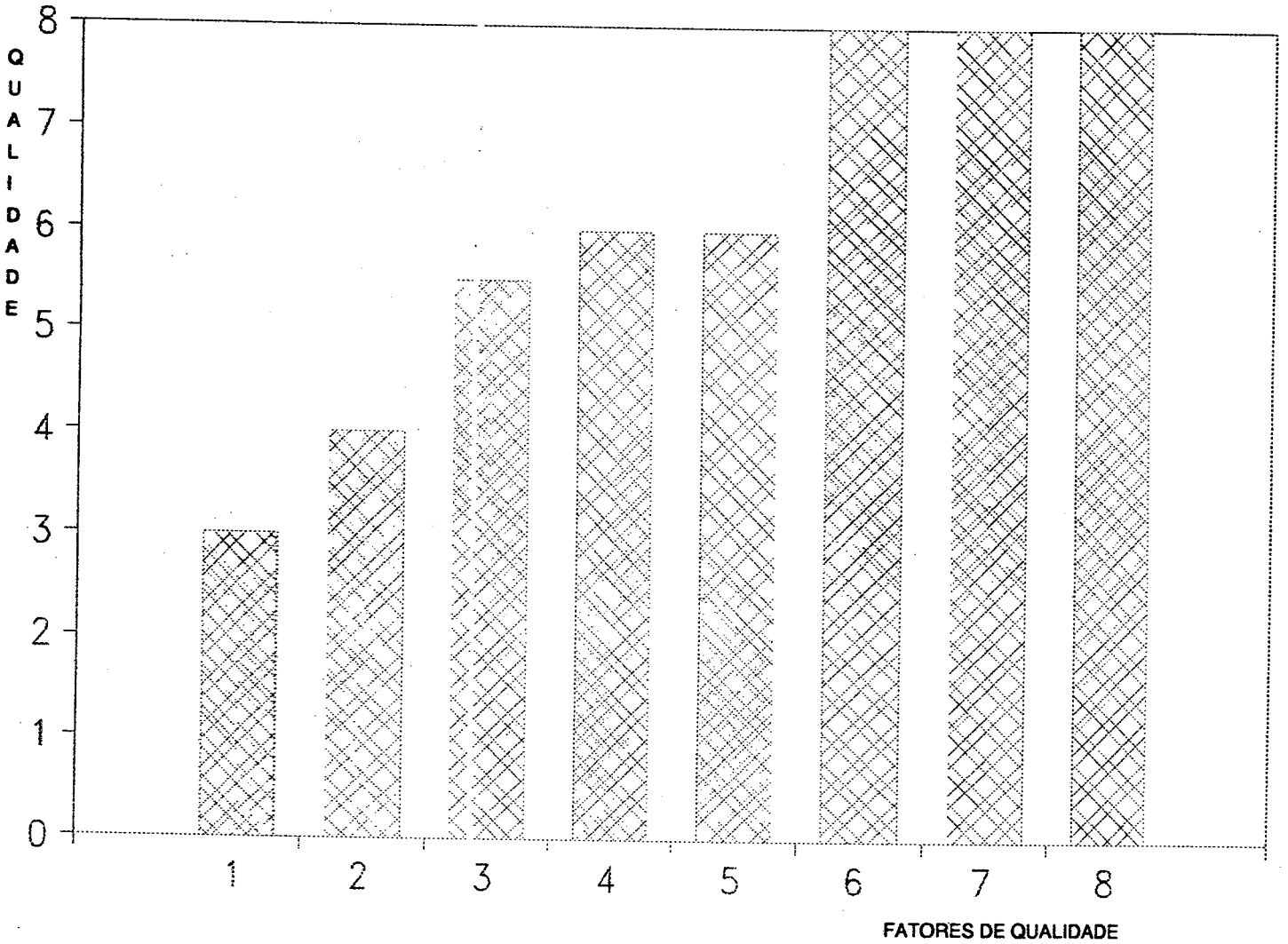

FIGURA 20 - DESEMPENHO DOS FATORES DE QUALIDADE DE SOFTWARE

As principais informações obtidas são:

- Dos fatores de qualidade de maior importância, 75% dos que possuem peso 3, apresentam índices abaixo do Limite de Qualidade Aceitável, definido anteriormente. Nesse sentido, a "Satisfação das Necessidades" e a "Confiabilidade" receberam notas muito baixas equivalentes à 6,00. Esta é uma situação muito grave pois, se o usuário não teve suas necessidades satisfeitas ou não pode confiar nos resultados fornecidos pelo software, qual é a utilidade dos mesmos?

- A Manutibilidade, outro fator de qualidade muito importante e com baixo desempenho, parece seguir a tendência mundial, ou seja, alta necessidade de manutenção.Cs valores baixos para este fator, foi uma das principais causas do fraco desempenho dos softwares desenvolvidos.

- O prazo de desenvolvimento, na grande maioria dos casos, não foi respeitado. Este fato, além de diminuir a satisfação dos clientes, atua como motivo de frustração para a equipe de desenvolvimento, pois, o não cumprimento do prazo funciona como uma derrota para a equipe.

- Uma surpresa obtida dos dados, é que o Manual do Usuário possui uma qualidade muito baixa, nota = 3,00. Isto está em total desacordo com a percepção dos analistas entrevistados, para os quais a qualidade do Manual do Usuário é muito boa. A principal causa desta divergência é que o analista utiliza uma linguagem excessivamente técnica, que lhe é bastante familiar, mas que dificulta a compreensão do usuário.

- Uma informação aparentemente contraditória em relação à qualidade do Manual do Usuário, é a boa nota obtida pela documentação entregue, segundo

o ponto de vista do usuário. A causa provável é que o usuário, geralmente, não lê esta documentação e, portanto, tende a atribuir a ela uma boa qualidade.

– Os pontos positivos dos sistemas analisados dirigiram-se para a facilidade de operação do software e a assistência técnica fornecida pela empresa.

Em função de todas as informações coletadas e descritas anteriormente, foi possível verificar que a utilização de uma metodologia de desenvolvimento influencia positivamente a qualidade do software produzido.

Com a continuidade da pesquisa relativamente à satisfação do cliente para novas versões da metodologia, poder-se-á quantificar a influência exercida.

Para que isto seja possível, deve-se calcular a qualidade da metodologia utilizada, como proposto pelo modelo. O resultado possibilitará relacionar qualidade da metodologia e qualidade do software.

5.3. QUALIDADE DA METODOLOGIA

Durante a aplicação do questionário para a quantificação da qualidade da metodologia, confirmou-se a grande dificuldade de comunicação entre analista e cliente.

O comentário mais ouvido é que ***"o maior culpado pelos problemas que ocorrem durante o desenvolvimento de software, é o cliente"***.

Este fato é agravado pela inoperância do gerente do projeto. A formação essencialmente técnica do gerente, conforme comentado no capítulo 3, faz com que ele tenha dificuldades em negociar com os clientes, influenciando negativamente a qualidade do software. Neste caso o gerente passa a atuar mais como um analista, esquecendo seu verdadeiro papel.

O gerente acaba, conforme identificado na prática, deixando de lado suas atividades principais: planejamento organização, controle, motivação de recursos

humanos e negociação. Tal situação, além de prejudicar a qualidade do software, provoca um aumento nos custos de desenvolvimento pois, não são realizados um planejamento ou um controle adequados dos recursos existentes.

Outro fato que foi comprovado na prática, é que o analista é extremamente confiante em seu trabalho. Com base nisto, a maioria acredita que o software desenvolvido é bom, o usuário é que não sabe utilizá-lo.

Além de identificar as características relacionadas ao processo de desenvolvimento, o questionário aplicado, como era esperado, possibilitou a quantificação dos subfatores de qualidade da metodologia.

Isto foi possível através da atribuição, por parte dos analistas, de uma nota a cada pergunta para cada subfator.

A relação entre objetivos, fatores e subfatores de qualidade, foi apresentada na figura 18.

O questionário utilizado para a quantificação dos subfatores de qualidade é mostrado no Apêndice 2.

Os analistas responderam às perguntas, com base na pontuação apresentada no capítulo 4, a saber:

5 --- ÓTIMO

4 --- BOM

3 --- REGULAR

2 --- RUIM

1 --- PÉSSIMO

Com os dados em mãos, calculou-se a qualidade da metodologia de desenvolvimento e manutenção de sistemas, versão 2.0, da **NTS - Núcleo de Tecnologia de Software**. Os passos executados, são apresentados a seguir.

5.3.1. QUANTIFICAÇÃO DOS SUBFATORES DE QUALIDADE

Cada subfator de qualidade, como já foi comentado, é quantificado através de uma ou mais perguntas de quantificação.

Dado um subfator de qualidade, sua quantificação é feita à partir da média simples das notas dadas às suas respectivas perguntas de quantificação.

Analogamente ao cálculo da qualidade do software, utilizou-se a média simples pois, acredita-se que as perguntas de quantificação de um mesmo subfator, possuem o mesmo grau de importância. Isto ocorre pois, cada pergunta enfoca um dado aspecto do subfator considerado, que não é nem mais nem menos importante que os outros.

5.3.2. QUANTIFICAÇÃO DOS FATORES DE QUALIDADE

Após a quantificação dos subfatores de qualidade, pode-se proceder à quantificação dos fatores de qualidade.Cada fator é composto por subfatores de qualidade. Através do cálculo da média das notas dos subfatores, obtém-se a qualidade dos fatores.

Para decidir sobre a utilização de média simples ou ponderada, foi realizada uma pesquisa junto aos analistas. Com esta pesquisa, identificou-se a importância relativa de cada subfator. Através dos dados obtidos, concluiu-se que todos os subfatores considerados possuem, segundo os analistas entrevistados, o mesmo grau de importância.

Assim, a qualidade dos fatores foi calculada através da média simples entre as notas dos subfatores de qualidade.

5.3.3. QUANTIFICAÇÃO DOS OBJETIVOS DE QUALIDADE

Para se quantificar os objetivos de qualidade, calculou-se a média simples dos fatores de qualidade.

A média simples foi utilizada pois, segundo pesquisa feita junto aos analistas, identificou-se que os fatores de um dado objetivo possuem o mesmo grau de importância.

5.3.4. CÁLCULO DA QUALIDADE DA METODOLOGIA

A metodologia, por definição, deve atingir, igualmente, seus dois objetivos, devendo possuir portanto, o mesmo grau de importância. Assim, a qualidade da metodologia é calculada através da média simples das notas obtidas pelos objetivos de qualidade

5.3.5. RESULTADOS OBTIDOS

Em primeiro lugar, vale ressaltar que o modelo utilizado para o cálculo da qualidade da metodologia (adaptação do modelo proposto por Rocha [ROCHA,87] para o cálculo da qualidade do software, conforme comentado no capítulo 4) adequou-se perfeitamente, possibilitando a obtenção de valiosas informações.

Durante a entrevista realizada, percebeu-se que os analistas entrevistados dão maior importância à fase de programação do desenvolvimento de software. Isto faz com que a análise, fundamental para o desenvolvimento de software de qualidade, seja relegada a segundo plano.

Tal resultado pode ser consequência da dificuldade de comunicação entre analista e usuário. Isto ocorre pois, a análise está baseada unicamente na comunicação analista/usuário, enquanto a programação é um trabalho solitário do analista (Esta característica faz parte da realidade observada, onde o analista também

programa. Nos E.U.A. é procedimento comum o analista executar apenas a parte de análise).

Esta situação provoca consequências negativas sobre a qualidade do software pois, os erros mais graves, conforme comentado no capítulo 3, ocorrem nas fases iniciais do desenvolvimento.

Outra característica importante identificada, é que o analista utiliza a metodologia apenas para gerar a documentação do processo de desenvolvimento. Isso significa que os passos sugeridos pela metodologia não são seguidos, prejudicando a qualidade do software. A causa disso é que, os passos realizados pelo analista, podem não ser os mais indicados podendo, inclusive, deixar de realizar tarefas importantes do desenvolvimento. Geralmente, estes passos não realizados envolvem tarefas relacionadas com o auto-controle de qualidade. Esta realidade cria uma dependência muito grande da competência técnica do analista.

O treinamento para a utilização plena da metodologia parece ser o passo mais acertado para eliminar este problema.

As críticas feitas pelos analistas, enfatizam a necessidade da metodologia ser mais abrangente, fornecer melhor apoio e que o treinamento para sua utilização seja mais eficaz.

Em função destas críticas, o desempenho de qualidade da metodologia deixou muito a desejar.A baixa qualidade parece justificar, parcialmente, a baixa qualidade dos softwares desenvolvidos.

A figura 21, mostra a qualidade média da metodologia de desenvolvimento, segundo o ponto de vista dos analistas entrevistados. Novamente foi utilizado, assim como será feito para os gráficos posteriores, uma escala de 0 a 10 para a qualidade.

O objetivo de qualidade que mais contribuiu para o baixo desempenho da metodologia, foi o objetivo Aplicabilidade. Segundo os analistas, a metodologia não é facilmente compreensível, nem se aplica, com a mesma facilidade e profundidade, a

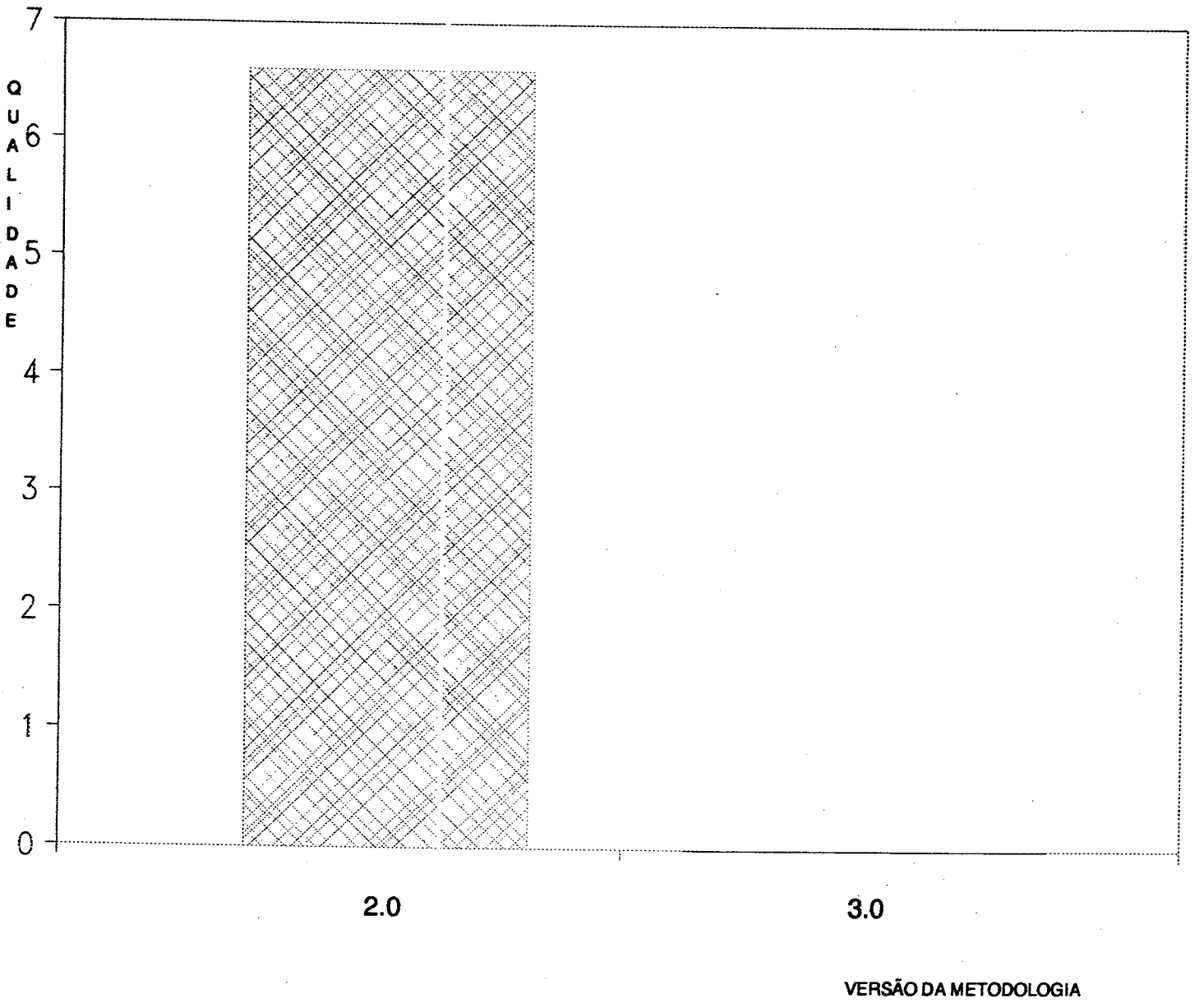

FIGURA 21 - DESEMPENHO DA METODOLOGIA - VERSÃO 2.0

projetos diferentes. Essa deficiência pode ter sido uma das responsáveis pela grande variação da qualidade dos softwares desenvolvidos.

A figura 22, apresenta o desempenho dos objetivos de qualidade da metodologia. Um dos objetivos, Aplicabilidade, relaciona-se à parte técnica da metodologia, enquanto o outro, Estrutura Didática, diz respeito à forma de apresentação.

Através da figura 22, observa-se que o objetivo Estrutura Didática teve, um desempenho abaixo do nível mínimo de qualidade estabelecido neste trabalho. Apesar disso, este objetivo, bem como seus respectivos fatores e subfatores, teve uma qualidade sensivelmente superior, em relação ao objetivo Aplicabilidade.

A causa principal disso parece ser que o analista sente facilidade em compreender a linguagem técnica e a estrutura utilizados na apresentação da metodologia.

Dentre os fatores de qualidade, o baixo desempenho também está relacionado à parte técnica da metodologia, embora os fatores relacionados à Estrutura Didática também estejam abaixo do desejado. Isto pode ser observado na figura 23.

A falta de flexibilidade da metodologia em se aplicar a projetos diferentes e a dificuldade de compreensão de seus conceitos, são as principais informações deduzidas da figura 23.

O mapeamento detalhado da qualidade da metodologia, pode ser visto na figura 24. Esta figura apresenta o desempenho dos fatores de qualidade da metodologia.

Segundo os analistas, a deficiência do treinamento para sua utilização é um dos princpais problemas apresentados pela metodologia. Esta situação, representada pelo baixo desempenho do subfator Facilidade de Treinamento, faz com que os conceitos contidos na metodologia sejam malcompreendidos. Este problema tem como consequência a aplicação ineficiente da metodologia, o que pode provocar o desenvolvimento de softwares de máqualidade.

O ponto mais crítico da metodologia, conforme mostrado na figura 24, é a falta de apoio para análise, codificação e gerenciamento do ciclo de vida do software.

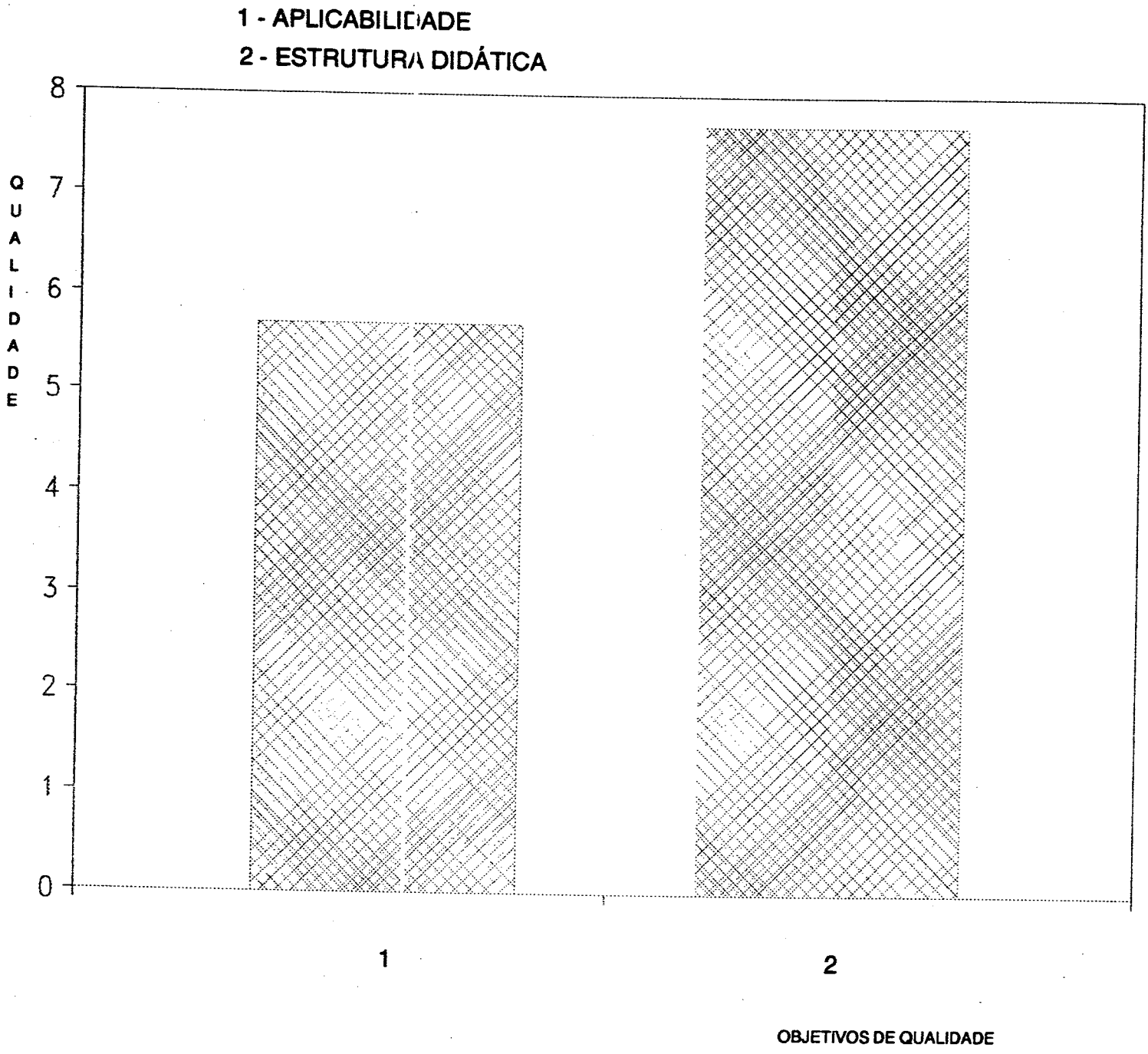

FIGURA 22 - DESEMPENHO DOS OBJETIVOS DE QUALIDADE

1 - COMPREENSIBILIDADE
2 - FLEXIBILIDADE
3 - COMUNICABILIDADE
4 - MANIPULABILIDADE

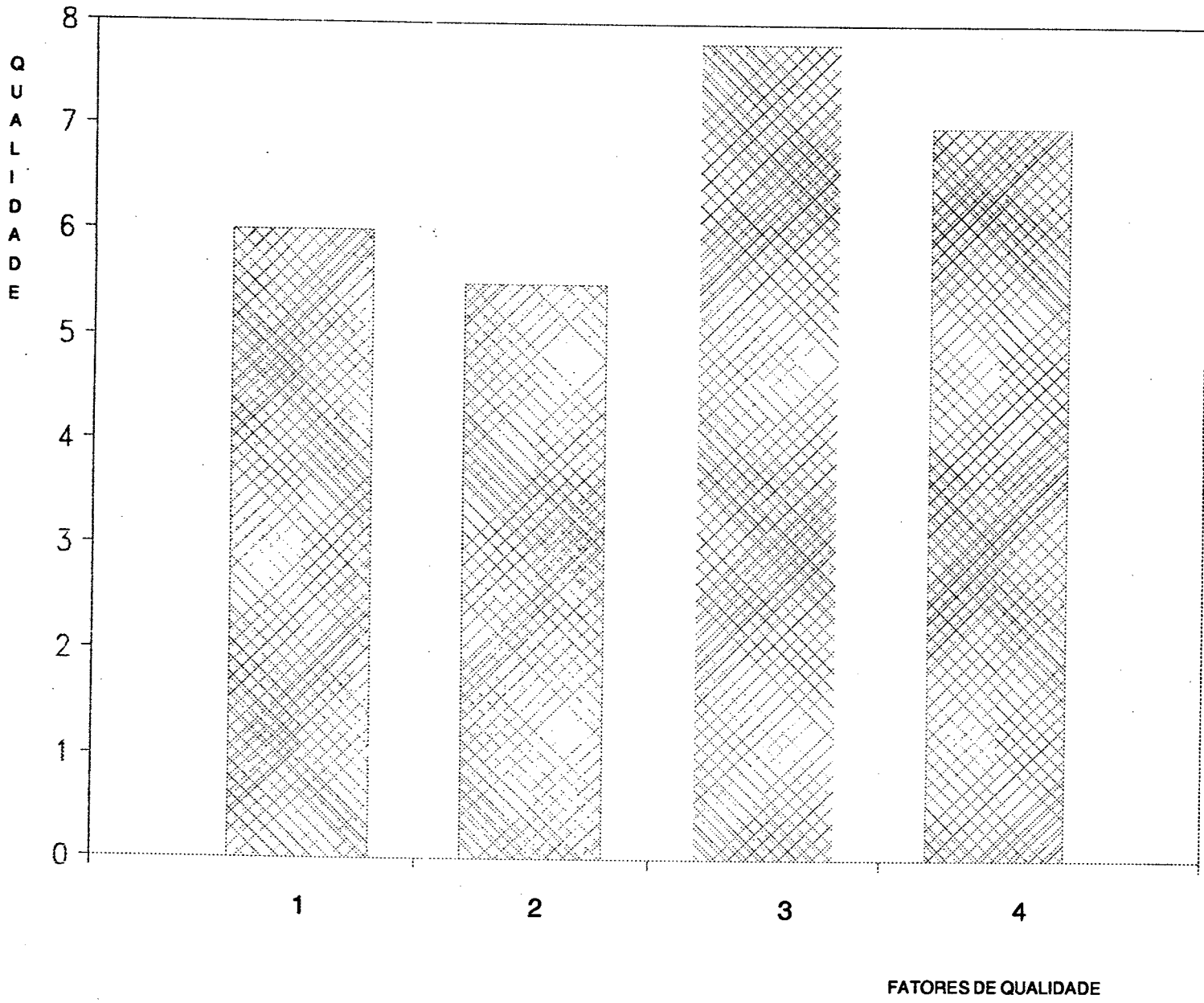

FIGURA 23 - DESEMPENHO DOS FATORES DE QUALIDADE

1 - VISIBILIDADE
2 - SIMPLICIDADE
3 - FACILIDADE DE TREINAMENTO
4 - ABRANGÊNCIA
5 - UTILIZABILIDADE
6 - SER HIERÁRQUICA
7 - POSSUIR APOIO
8 - CORREÇÃO
9 - CONCISÃO
10 - UNIFORMIDADE DE TERMINOLOGIA
11 - DISPONIBILIDADE
12 - ESTRUTURA
13 - RASTREABILIDADE

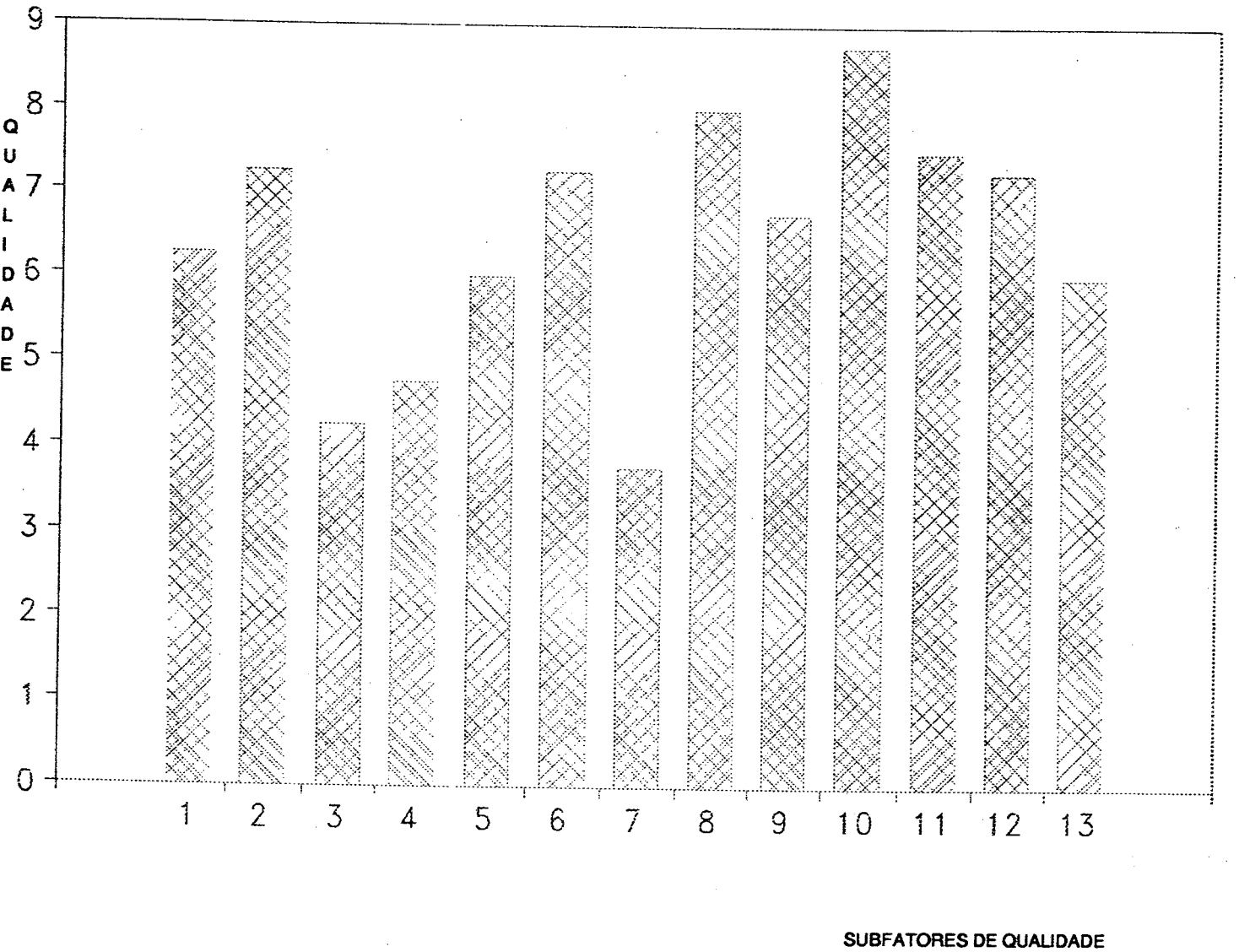

FIGURA 24 - DESEMPENHO DOS SUBFATORES DE QUALIDADE

Este fato associado a um treinamento deficiente, provoca danos muito grandes para a qualidade do software desenvolvido. Isto ocorre pois, sem um apoio efetivo e sem um treinamento adequado, fica muito difícil aplicar os conceitos contidos na metodologia.

Torna-se importante salientar que, se for utilizada a Nota de Corte igual a 8,00, o que parece razoável segundo informações obtidas durante a entrevista, a quase totalidade dos subfatores, fatores e objetivos de qualidade estarão abaixo do mínimo.

Por este motivo, existe muito a ser feito para melhorar a qualidade da metodologia de desenvolvimento, versão 2.0, e, consequentemente, a qualidade do software desenvolvido.

5.4. ANÁLISE DOS RESULTADOS

Através dos dados obtidos junto aos analistas e clientes da **NTS - Núcleo de Tecnologia de Software**, foi possível o cálculo da qualidade da metodologia e do software, respectivamente.

Em função destes dados e de informações adicionais obtidas durante a entrevista, observou-se que existe uma relação diretamente proporcional entre a qualidade da metodologia e a qualidade do software desenvolvido. Essa percepção baseia-se em experiências anteriores de analistas e usuários em desenvolvimento de softwares sem a utilização de uma metodologia.

A quantificação da influência da utilização de uma metodologia de desenvolvimento sobre a qualidade do software, será possível através da elaboração de um gráfico, onde se relacione qualidade da metodologia e qualidade do software.

A figura 25, apresenta o gráfico elaborado com os dados calculados nos ítens anteriores. Os valores apresentados no gráfico originam o primeiro de uma série de pontos a serem calculados através de pesquisas posteriores, de forma a quantificar a relação desejada.

Na figura 26, é feita uma simulação de como ficaria o gráfico, apresentado na figura 25, após várias repetições da pesquisa realizada.

Observa-se que, no limite, quando a metodologia atingir sua qualidade máxima, a qualidade do software continuaria a aumentar. Isto porque, existem outros fatores que além da metodologia, influenciam a qualidade do software, conforme mostrado no capítulo 3.

Segundo a filosofia japonesa, não existe qualidade máxima pois, qualquer produto pode ser continuamente melhorado, em função do surgimento de novas tecnologias, novas necessidades dos clientes, e outros.

Neste sentido, a qualidade máxima da metodologia, citada anteriormente, foi utilizada apenas como simulação de uma situação futura.

Em termos reais sempre haveria uma possibilidade de melhoria da qualidade da metodologia utilizada pois, o ambiente de desenvolvimento de software está em contínua transformação.

Neste sentido, a inclinação da curva da figura 26, não começa na origem do sistema cartesiano (ponto que representa a qualidade zero, tanto da metodologia quanto do software). Isto ocorre pois, mesmo sem a utilização de uma metodologia, o software desenvolvido possuiria uma determinada qualidade. Tal qualidade seria resultado dentre tantos fatores existentes da experiência das pessoas envolvidas, dos equipamentos utilizados e dos recursos financeiros disponíveis.

Portanto, baseado nas considerações feitas anteriormente, o modelo proposto atinge o objetivo pré-estabelecido. Além disso, sua aplicação resulta, conforme identificado na prática, na organização de uma estrutura de qualidade na empresa.

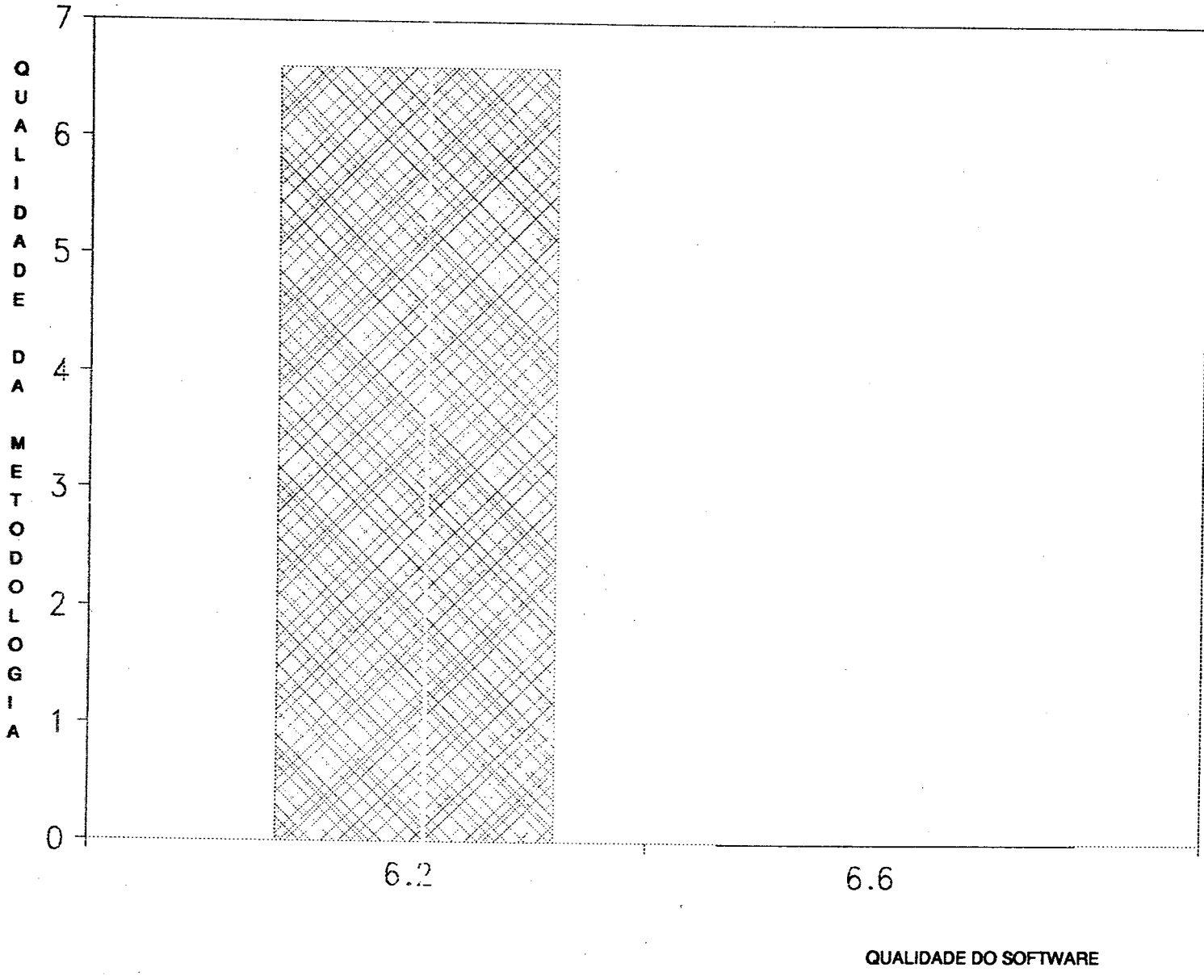

FIGURA 25 - RELAÇÃO METODOLOGIA SOFTWARE - RESULT. OBTIDOS

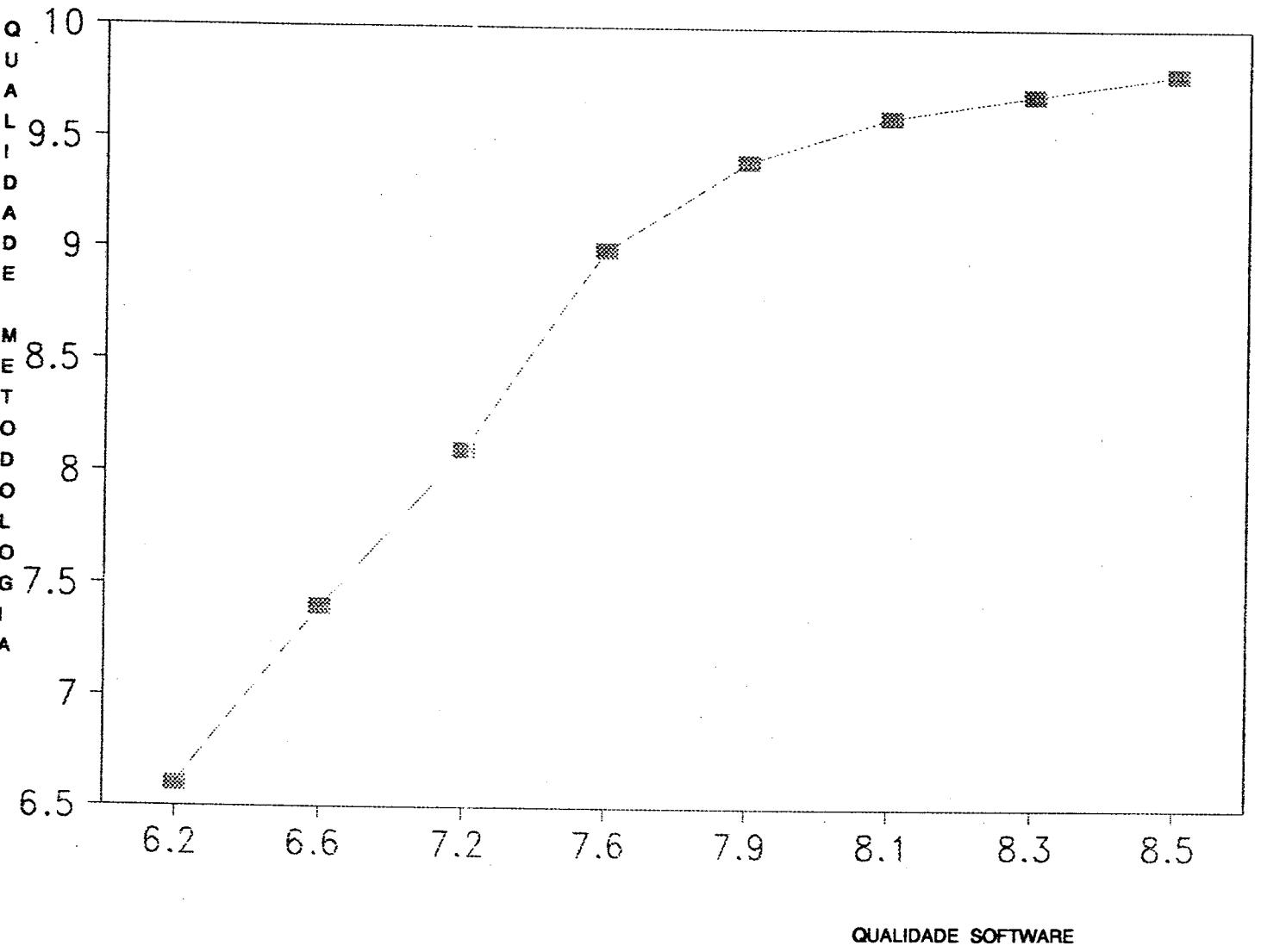

FIGURA 26 - RELAÇÃO METODOLOGIA SOFTWARE - SIMULAÇÃO

Independente da estrutura montada, o sistema* de qualidade atua como elo de ligação entre a empresa e o cliente, conforme mostrado na figura 27.

* Entidade constituída por elementos interdependentes, que se interagem segundo determinado princípio para atingir um objetivo específico.

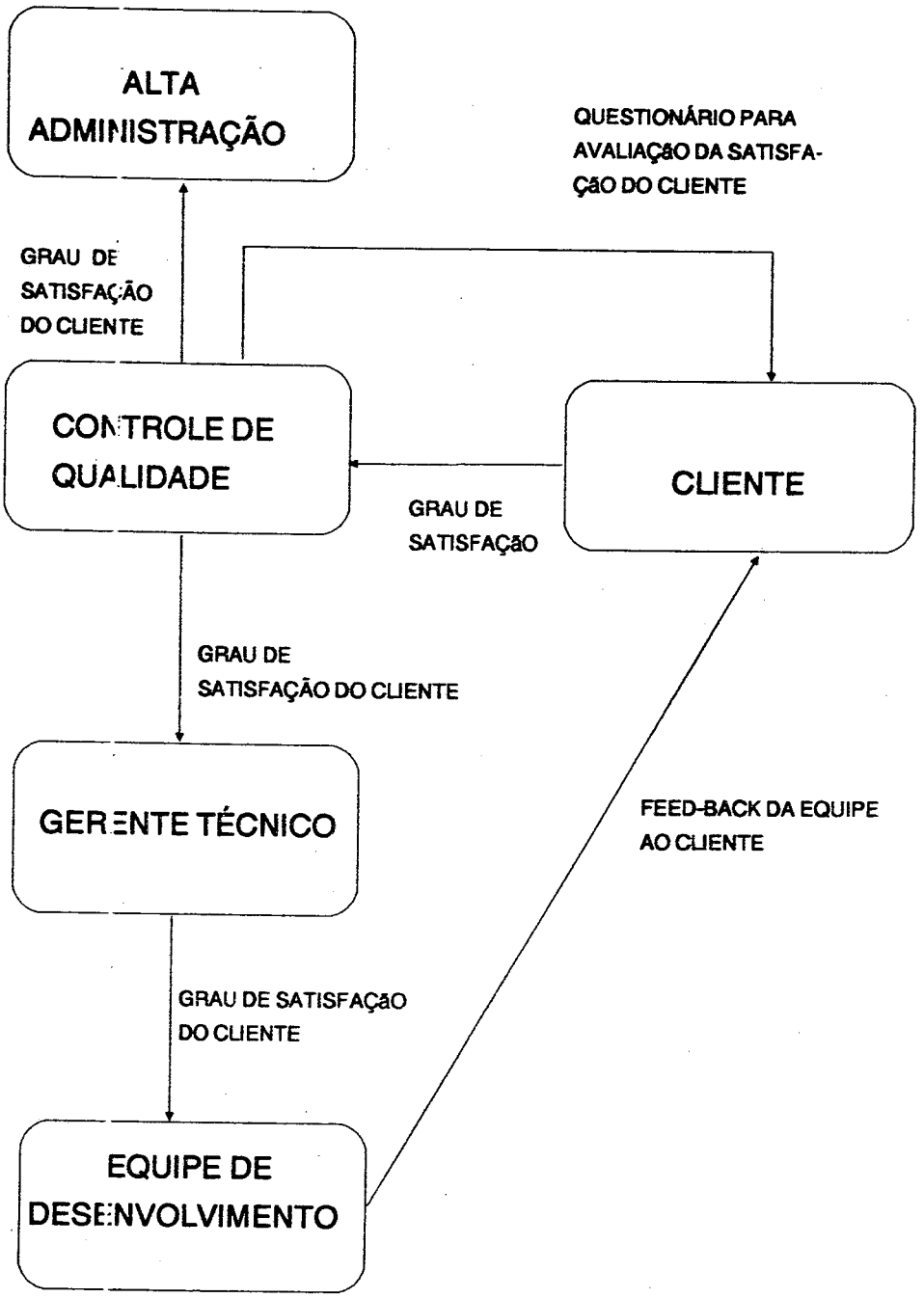

FIGURA 27 - ATUAÇÃO DO CONTROLE DE QUALIDADE

CAPÍTULO 6

6. CONCLUSÕES E RECOMENDAÇÕES

6.1. CONCLUSÕES

O presente trabalho abordou a elaboração de um modelo para a quantificação da influência que uma metodologia de desenvolvimento exerce sobre a qualidade do software.

Como base para a apresentação do modelo, foram descritos os principais fatores, que interagem num ambiente de desenvolvimento de software: Recursos Humanos, Gerência e Metodologia.

Os fatores descritos exercem, tanto separadamente quanto como resultado da interação entre eles, influência sobre a qualidade do software. Apesar disso, o modelo proposto procurou quantificar apenas a influência exercida pela metodologia utilizada.

Para que isso seja possível, o modelo propõe o cálculo da qualidade média dos softwares desenvolvidos e da metodologia utilizada, para cada vez que se implantar uma nova versão da metodologia.

Quando uma nova versão da metodologia for implementada, serão produzidos softwares com um dado perfil de qualidade.

Através da comparação dos dois perfis ou da média de qualidade, é possível quantificar a relação existente entre qualidade da metodologia e qualidade de software.

A partir dos dados obtidos em pesquisa de campo, junto a analistas e usuários, pode-se concluir que o modelo proposto atinge o objetivo estabelecido. Significa que, através da aplicação dos passos sugeridos pelo modelo é possível quantificar a relação entre metodologia e qualidade de software.

A condição para que isto seja possível, é que os demais fatores, Recursos Humanos e Gerência, permaneçam constantes. Em condições normais, as empresas não vivenciam grandes transformações no perfil de seus Recursos Humanos ou de sua Gerência.

Portanto, qualquer modificação no perfil das médias de qualidade dos softwares desenvolvidos, terá sido provocado, principalmente, pelas mudanças realizadas na metodologia.

Além de atingir o objetivo estabelecido, o modelo proporcionou vantagens inegáveis, como resultado de sua implantação.

Entre essas vantagens, destacam-se:

– Melhoria da Comunicação Analista/Usuário

Durante a apresentação do trabalho, destacou-se a grande dificuldade de comunicação entre analistas e usuários. Através da aplicação do modelo, foram obtidas informações que possibilitam a diminuição deste problema.

– Menor Custo na Implantação de Novas Versões da Metodologia

Conforme ressaltado no capítulo 4, a aplicação do modelo fornece um "mapa" da qualidade da metodologia. Através deste mapa, identifica-se os pontos deficientes da metodologia. Em função disso, o trabalho de diagnóstico da metodologia já estará pronto. Com isso, o tempo necessário à modificação da metodologia, e consequentemente o custo, será menor.

– Identificação do Grau de Satisfação dos Clientes

Todo programa de melhoria da qualidade do produto desenvolvido, está baseado no grau de satisfação do cliente,com relação a este produto.

O modelo proposto, indica os passos que devem ser seguidos para se avaliar a qualidade do software desenvolvido, sob o ponto de vista do cliente.

Portanto, caso se decida implementar um programa deste tipo, um passo muito importante já terá sido dado.

– Motivação para a Qualidade

A aplicação dos questionários para a avaliação da qualidade, tanto do software quanto da metodologia, induziu os entrevistados a se despertarem para a qualidade, conforme verificado na prática.

Como a qualidade está fundamentalmente nas pessoas, conclui-se que a contribuição do modelo é bastante grande.

O modelo proposto, portanto, deve ser encarado, não só como um método para se quantificar a influência de uma metodologia sobre a qualidade do software, mas também, e principalmente, como um primeiro passo decisivo para a implantação de uma verdadeira cultura de qualidade na empresa.

6.2. RECOMENDAÇÕES

Com a conclusão do presente trabalho, abrem-se novas possibilidades de pesquisas.

– Realizar várias aplicações do modelo proposto, a fim de se quantificar, efetivamente, a influência de uma metodologia sobre a qualidade do software.

– Em função das características peculiares apresentadas, pode-se desenvolver um programa para a melhoria da qualidade do software desenvolvido.

– Devido às deficiências apresentadas, sugere-se desenvolver um software que possibilite o planejamento e o controle das atividades, relacionadas ao desenvolvimento de softwares.

7. BIBLIOGRAFIA

1 - [AAKER, 86] - AAKER, David A. Marketing Research. J. Wiley, 3 ed., 1986.

2 - [AGRESTI, 81] - AGRESTI, Willian W. Applying Industrial Engineering to the Software Development Process. In: Proceedings Productivity: an urgent priority, 1981.

3 - [ANSI/IEEE, 82] AMERICAN NATIONAL STANDART INSTITUTE/ INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERING. Definitions on Software Engineering. 1982.

4 - [ARTHUR, 85] - ARTHUR, Lowell Jay. Produtividade do ProgramadorTradução: Ricardo Reinprecht. Editora Campus. 1985.

5 - [AWAD, 77] - AWAD, Elias M. Prediction of Sactisfation of Systems Analists, Programmers. Data Management Journal, 1977.

6 - [BAILLIE, 86] - BAILLIE, Allan S. Deming Approach: Being Better than the Best . SAM Advanced Management Journal, 1986.

7 - [BASILI, 75] - BASILI, Victor R. e TURNER, Albert J. Iterative Enhancement: A practical Technique for Software Development.IEEE Transactions on Software Engineering.v.Se-1, n. 4, Dezembro, 1975.

8 - [BASILI, 85] - BASILI, Victor R. Evaluation of Software Methodology. Computer Science, Technical Report Series. University of Maryland. 1985.

9 - [BISWAS, 88] - BISWAS, Tilak. The End-User Interview. Auerbach-Publishers. EDP Auditing. 1988.

10 - [BOEHM, 80] - BOEHM, Barry W. An Experiment in Small-Scale Application Software Engineering . Proceedings IFIP Congress. 1980

11 - [BOEHM, 81] - _____. Improving Software Productivity. In Proceedings Productivity: an urgent prioryt. Los Angeles. 1981.

12 - [BOEHM, 89] - _____. Theory-W Software Project Management: Principles and Examples . IEEE Transactions on Software Engineering. v. 15, n. 7 , Julho, 1984.

13 - [BOYD, 78] - BOYD, Donald L e PIZZARELLO, Antonio. Introduction to the WELLMADE Design Methodology. IEEE Transactions on Software Engineering, v. Se - 4, n. 4, Julho, 1978.

14 - [BREWER, 80] - BREWER, J. A. Improving Systems Productivity . Data Systems. Junho, 1980.

15 - [BROOKS, 75] - BROOKS, F. P. . The Mithical Man-Month . Addison-Wesley . 1975

16 - [BROWN, 85] - BROWN, Marlene . New Tecnology but Old Methods . Data Processing. v. 27, n. 6, Julho/Agosto. 1985.

17 -[BRUGGERE, 79] - BRUGGERE, Thomas H . Software Engineering : Management, Personel and Methodology . In

Tutorial : Software Management. Donald Reifer. 1979.

18 - [C., 81] - C., Barbara e KISER, Stewart . Software Management Productivity - Understanding the Software Development Process . In Proceedings Productivity: an urgent priority, Los Angeles, 1981.

19 - [CARD, 87] - CARD, et all . Improving Software Quality and Productivity. Information and Software Tecnology. v. 29, n. 5, Junho, 1987.

20 - [CATCHPOLE, 86] - CATCHPOLE, Paul .Requirements for a Successfull Methodology in Information Systems Design . Data Processing . v. 28 , n. 4, Maio, 1986.

21 - [CHEN, 81] - CHEN, Eric e ZELKOWITZ, Marvin V. Use of Cluster Analisys to Evaluate Software Engineering Methodologies . In International Conference os Software Engineering, San Diego, 1981.

22 - [CHIAVENATTO,80] - CHIAVENATTO, Idalberto . Administração de Recursos Humanos. Editora Atlas. São Paulo . 1980

23 - [CONELL, 85] - CONELL, John e BRICE, Linda .Practical Quality Assurance . Datamation. Março, 1985.

24 - [COWARD, 88] - COWARD, P. David . A Review of Software Tecnology . v. 30, n. 3, Abril, 1988.

25 - [DALY, 77] - DALY, E. Management of Software Engineering . IEEE Transactions on Software Engineering . v. Se-3, 3, Maio, 1977.

26 - [DALY, 79] - DALY, E. Organizing for Successfull Software Development . Datamation. Dezembro, 1979.

27 - [DAVIS, 88] - DAVIS, Alan M. et all . A Strategy for Comparing Alternative Software Development Life Cycle Models . IEEE Transactions on Software Engineering . v. 14, n. 10, Outubro, 1988.

28 - [DeMARCO, 89] - DeMARCO, Tom . Análise Estruturada e Especificação de Sistema . Tradução de Beatriz de Carvalho. Editora Campus. Rio de Janeiro, 1989.

29 - [DeMARCO, 90] - DeMARCO, Tom e LISTER, Timothy . Peopleware . Tradução: Kátia Aparecida Rocha. Editora McGraw-Hill. São Paulo, 1990.

30 - [DeMILLO, 80] DeMILLO, R. e LIPTON, R. Software Project Forecasting. Georgia Institute of tecnology. Atlanta. G.A.. 1980.

31- [DIAS, 86] DIAS, Donaldo e GAZZANEO, Giosafatte . Projeto de Sistemas de Processamento de Dados . Editora Livros Técnicos e Científicos S.A. Rio de Janeiro. 1986.

32 - [DUARTE, 89] - DUARTE, Anibal A. C. Qualidade no Projeto:Começo do Êxito . Qualitech Brasil - Conferência Nacional da Qualidade. São Paulo. 1989.

33 - [DUCLÓS, 82] - DUCLÓS, Luiz Carlos . Simulation Cost Model for the Software Product: A Quality Assurance Approach . Dissertation (Phd). University of Southern California. Los Angeles.

34 - [DYER, 87] - DYER, Michael . An Approach to Software Reliability Measurement . Information and Software Tecnology . v. 29, n.8. Outubro. 1987.

35 - [FAGAN, 86] - FAGAN, Michael E. Advances in Software Inspections . IEEE Transactions on Software Engineering. v. Se-12, n. 7, Julho, 1986

36 - [FAIRLEY, 85] - FAIRLEY, R. Software Engineering Concepts . McGraw-hill . 1985.

37 - [FASTIE, 88] - FASTIE, Will - "Futere of Software Engineering", PC Tech Journal, 6(10),Nova York, Outubro, 1988.

38 - [FENTON, 87] - FENTON, Norman E. e KAPOSI, Agnes . Metrics and Software Structure . Information and Software Tecnology. v. 29, n.6, Julho/Agosto, 1987.

39 - [GERHART, 76] - GERHART, Susan e YELOWITZ, Lawrence . Observations of Fallibility in Applications of Modern Programming Methodologies . IEEE Transactions on Software Engineering . v. Se-2, n. 3, Setembro, 1976.

40 - [GOLDBERG, 86] - GOLDBERG, R. Software Engineering: An Emerging Discipline . IBM Systems Journal . v. 25, n. 3/4, 1986.

41- [HETZEL, 87] - HETZEL, Willian . Guia Completo ao Teste do Software . Editora Campus. 1987.

42 - [HOWES, 84] - HOWES, Norman R. Managing Software Development Projects for Maximum Productivity . IEEE Transactions on Software Engineering . v. Se-10, n. 1, Janeiro, 1984.

43 - [KAMOJI, 86] - KAMOJI, Dan . Management Controls for Informations Systems. Data Processing v. 28, n. 4, Maio, 1986.

[KEIDER, 74] - KEIDER, Stephen P. Why Projects Fail . Datamation . Dezembro. 1974.

[LINGER, 81] - LINGER, R. C. Human Productivity in Software Development . In Proceedings Productivity: an urgent priority. Los Angeles. 1981.

[LIPCHIN, 81] - LIPCHIN, Leonid . The Role of Strategic Planning for Productivity Improvement through Computer-Aided Engineering . In Proceedings Productivity: an urgent prioryt . Los Angeles . 1981.

[LOBOS,91] - LOBOS, Júlio . Qualidade Através das Pessoas . Instituto da Qualidade. 1991.

[MILLS, 89] - MILLS, Harland D. et all . Mathematical Principles for a First Course in Software Engineering . IEEE Transactions on Software Engineering . v. 15, n. 5, Maio, 1989.

[MILNER, 85] - MILNER, David . An Integrated Approach to Systems Development . Data Processing . v. 27, n. 3, Abril, 1985.

[MORREALE, 85] - MORREALE, Richard .Project Planning and Control . Data Processing. v. 27, n. 3, Abril, 1985.

[MUSA, 75] - MUSA, John D. A Theory of Software Reliability and its Aplication . IEEE Transactions on Software Engineering . v. Se-1, n. 3, Dezembro, 1975.

[NETO, 87] - NETO, Edgar P. C. Metodologia para Implantação de um Sistema de Garantia/Controle da Qualidade de Produtos . Revista de Administração . v. 22, n. 2, Abril/Junho, 1987.

62 - [RIDJANOVIC, 81] - RIDJANOVIC, Dzeman e BRODIE, Michael L. . Specification versus Implementation . In Proceedings Productivity: an urgent priority. Los Angeles. 1981.

63 - [ROCHA,87] - ROCHA, Ana R. C. da . Análise e Projeto Estruturado de Sistemas . Editora Campus . 1987.

64 - [ROCHA,92] - ROCHA, Carlos A. M. da . Dissertação em Elaboração . Departamento de Engenharia de Produção e Sistemas. Universidade Federal de Santa Catarina - UFSC. 1992.

65 - [STAA, 83] - STAA, A. V. . Engenharia de Programas . Livros Técnicos e Científicos . 1983.

66 - [STAA, 80] - _____ . The Development Proposal : The First Step in Software System Constructions . The Journal of Systems and Software . n. 1. 1980.

67 - [STEVENS, 77] - STEVENS, Howard P. e KROCHMAL, Jerome J. . Engineering Motivators and Demotivators . IEEE Transactions on Software Engineering. 1977.

68 - [TAYLOR, 71] - TAYLOR, Frederick W. . Princípios da Administração Científica . Tradução: Arlindo Vieira Ramos. Editora Atlas. 7ª Edição. 1971.

69 - [TEIXEIRA, 79] - TEIXEIRA, Sérgio R. P. . Engenharia de Software : Experiência e Recomendações . Editora Edigard Blücher. 1979.

70 - [THAYER, 81] - - THAYER, Richard H. et all . Major Issues in Software Engineering Project Management . IEEE Transactions on Software Engineering . v. Se-7, n. 4, 1981.

71 - [VELASCO, 88] - VELASCO, FLÁVIO R. D. . Garantia da Qualidade de Software: Um Enfoque Prático . Apostila do Curso Controle de Qualidade de Software. CDQI - INTERNACIONAL .1988.

72 - [VELASQUEZ, 87] - VELASQUEZ, Concepción U. V. . Uma Abordagem Conceitual do Sistema de Informação da Qualidade . Dissertação de Mestrado. UFSC. Florianópolis. 1987.

73 - [VERYARD - 85] - VERYARD, Richard . Methods of Maintenance . Data Processing. v. 27, n. 9, Novembro, 1985.

74 - [VERYARD - 85] - _____ . What are Methodologies Good for? . Data Processing. v. 27, n. 6 , Julho. 1985.

75 - [VERYARD - 87] - _____ . Future of Information Systems Design Methodologies .Information and Software Tecnology . v. 29, n. 1. Janeiro/Fevereiro. 1987.

76 - [VERYARD, 87] - _____ . Implementing a Metodology . Information and Software Tecnology. v. 29, n. 9, Novembro, 1987.

77 - [YAU, 88] - YAU, Stephen S. et all . An Integrated Life-Cycle Model for Software Maintenance . IEEE Transactions on Software Engineering. v. 14, n. 8, Nova York, 1988.

78 - [YOURDON,89] - YOURDON, Edward . Administrando o Ciclo de Vida do Sistema. Editora Campus.1989.

79 - [YOURDON, 90] - YOURDON, Edward . Revisões Estruturadas . Editora Campus. 1990

APÊNDICE 1

C – ( ) Inadequada

D – ( ) Não se aplica

4 – A organização das informações na documentação apresentada pela empresa, pode ser considerada:

A –) Adequada

B –) Não muito adequada

C –) Inadequada

D –) Não se aplica

5 – A organização das informações contidas no Manual do Usuário pode ser considerada:

A –) Adequada

B –) Não muito adequada

C –) Inadequada

D –) Não se aplica

6 – Quanto à facilidade de compreensão, o manual do usuário pode ser considerado:

A –) Fácil

B –) Médio

C –) Difícil

D –) Não se aplica

7 – Quanto melhor obteve as informações desejadas, quando consultou o Manual do Usuário:

A –) Totalmente

B –) Parcialmente

D - ( ) Não se aplica

12 - O tempo dedicado ao treinamentos, pode ser classificado como:

A - ( ) Suficiente

B - ( ) Excessivo

C - ( ) Insuficiente

D - ( ) Não se aplica

13 - O software desenvolvido satisfez suas necessidades:

A - ( ) Totalmente

B - ( ) Parcialmente

C - ( ) Não atendeu

D - ( ) Não se aplica

14 - As informações fornecidas pelo software são:

A - ( ) Totalmente confiáveis

B - ( ) Parcialmente confiáveis

C - ( ) Não confiáveis

D - ( ) Não se aplica

15 - Quanto à facilidade de aprendizado e operação, o software pode ser classificado como:

A - ( ) Fácil

B - ( ) Médio

C - ( ) Difícil

D - ( ) Não se aplica

16 - A nessidade de manutenção é:

APÊNDICE 2

QUESTIONÁRIO ENVIADO AOS ANALISTAS

VISIBILIDADE

– A teoria contida na metodologia é de fácil compreensão? ( )

– Os mecanismos de controle, a serem utilizados durante o desenvolvimento, são claramente definidos ?( )

– A metodologia é simples de ser aplicada ?( )

SIMPLICIDADE

– A documentação gerada durante o desenvolvimento, e entregue ao usuário, é fácil de ser entendida ?( )

– O conteúdo da documentação entregue é suficiente para fornecer ao usuário um perfeito entendimento do sistema ?( )

– A estrutura de apresentação da documentação é adequada ?( )

FACILIDADE DE TREINAMENTO

– Os conceitos da metodologia são fáceis de serem assimilados? ( )

– O treinamento para utilização da metodologia, oferecido atualmente, é suficiente para a perfeita utilização da mesma?( )

ABRANGÊNCIA

– A metodologia engloba todo o ciclo de vida do software? ( )

– Todas as fases do ciclo de vida são abordadas na mesma profundidade? ( )

UTILIZABILIDADE

– A metodologia pode ser utilizada, com a mesma eficiência, em projetos com características diferentes? ( )

– A metodologia se aplica tanto a grande quanto a pequenos projetos ? ( )

SER HIERÁRQUICA

– As ferramentas utilizadas pela metodologia, incentivam a decomposição sucessiva, de uma visão macroscópica até uma visão microscópica ?( )

– As ferramentas utilizadas pela metodologia, favorecem a correta vinculação entre as visões mais abrangentes e as visões mais detalhadas? ( )

POSSUIR APOIO

– A metodologia possui técnicas que auxiliam na criação de especificações?( )

– O apoio automatizado, que auxilia a aplicação da metodologia é suficiente?( )

– A metodologia oferece apoio para modificação do software ?( )

– A metodologia contém uma metodologia de testes para o controle de qualidade do software?( )

CORREÇÃO

– A linguagem utilizada pela metodologia é fácil de ser entendida?( )

CONCISÃO

– O volume de texto da metodologia, em comparação com a quantidade de informação fornecida, é pequeno?( )

– O volume de texto da documentação entregue ao usuário, em comparação com a quantidade de informação fornecida, é pequeno?( )

UNIFORMIDADE DE TERMINOLOGIA

– Os termos utilizados pela metodologia possuem o mesmosignificado ao longo de todo o documento?( )

– Os termos utilizados na documentação entregue ao usuário, possuem o mesmo significado ao longo de todo documento?( )

DISPONIBILIDADE

– O acesso à versão mais recente da metodologia, é fácil?( )

ESTRUTURA

– A estrutura de apresentação da metodologia, facilita sua manipulação?( )

– Dentro de cada fase, a estrutura de apresentação da metodologia segue a ordem natural do desenvolvimento?( )

– As tabelas, fluxogramas e exemplos, contidos na metodologia, facilitam seu manuseio?( )

RASTREABILIDADE

– É possível rastrear um determinado aspecto do software, ao longo dos documentos produzidos?( )