CENTRO TECNOLÓGICO DA ZONA LESTE
FACULDADE DE TECNOLOGIA DA ZONA LESTE

THIAGO CEZAR PEREZ

PROPOSTA DE UM MODELO PARA DESENVOLVIMENTO DE SOFTWARE COM QUALIDADE

São Paulo, 2005.

CENTRO TECNOLÓGICO DA ZONA LESTE

FACULDADE DE TECNOLOGIA DA ZONA LESTE

THIAGO CEZAR PEREZ

PROPOSTA DE UM MODELO PARA DESENVOLVIMENTO DE SOFTWARE COM QUALIDADE

Monografia apresentada no curso de Tecnologia em Informática com ênfase em Gestão de Negócios na FATEC ZL como requerido parcial para obter o Título de Tecnólogo em Informática com ênfase em Gestão de Negócios.

Orientador: Prof. Ricardo Satoshi Oyakawa

São Paulo, 2005

CENTRO TECNOLÓGICO DA ZONA LESTE

FACULDADE DE TECNOLOGIA DA ZONA LESTE

THIAGO CEZAR PEREZ

PROPOSTA DE UM MODELO PARA DESENVOLVIMENTO DE SOFTWARE COM QUALIDADE

Monografia apresentada no curso de Tecnologia em Informática com ênfase em Gestão de Negócios na FATEC ZL como requerido parcial para obter o Título de Tecnólogo em Informática com ênfase em Gestão de Negócios.

COMISSÃO EXAMINADORA

_____

Prof. Ricardo Satoshi Oyakawa

Faculdade de Tecnologia da Zona Leste

_____

Prof. Joel Valentino Candido

Faculdade de Tecnologia da Zona Leste

_____

Profª. Ana Lúcia Calaça

ETE Prof. Aprígio Gonzaga

São Paulo, 14 de dezembro de 2005.

Aos meus pais, Cezar e Silvia e às minhas irmãs, Mariana e Gabriella.

AGRADECIMENTOS

Ao meu orientador, Prof. Ricardo Satoshi, grande conselheiro em todas as etapas deste trabalho.

A minha família, pelo apoio e amor.

Aos amigos e colegas, pela alegria e conversas que me proporcionam.

Aos professores e colegas de curso, pois juntos trilhamos uma etapa importante de nossas vidas.

A todos que, com boa intenção, colaboraram para a realização e finalização deste trabalho.

Todo o erro humano é a impaciência, uma renúncia prematura ao método, uma ilusória fixação a uma ilusão.

Franz Kafka

RESUMO

Este trabalho apresenta de maneira resumida e introdutória os principais conceitos ligados à Engenharia e Qualidade de Software. A seguir, é realizado um levantamento com 20 pequenas empresas de desenvolvimento de software, com o objetivo de compreender seus hábitos de desenvolvimento de software e a adequação desses hábitos aos princípios da Engenharia de Software. Por fim, será proposta uma metodologia para o desenvolvimento de softwares com qualidade e viabilidade econômica, que atenda às demandas e respeite as restrições mínimas das pequenas empresas de desenvolvimento de software.

Palavras-chave: engenharia de software; qualidade; pequenas empresas; metodologia.

ABSTRACT

This paper presents in a brief and introductory way the main concepts linked to Software Engineering and Software Quality. Later, it is produced a research with 20 small software development companies, in order to understand their habits of developing software and the conformity of these habits to the Software Engineering principles. At last, it is going to be proposed a methodology for developing software with quality and economical viability, which attends and respects the minimum demands and constraints of small software development companies.

Key-words: software engineering; quality; small companies; methodology.

SUMÁRIO

1. INTRODUÇÃO..... 9
1.1 Justificativa..... 11
1.2 Objetivo..... 11
1.3 Metodologia..... 11
2 ENGENHARIA DE SOFTWARE..... 13
2.1 Fases Genéricas do Desenvolvimento de Software..... 15
2.1.1 Primeira Fase: Definição..... 15
2.1.2 Segunda Fase: Desenvolvimento..... 17
2.1.3 Terceira Fase: Manutenção..... 18
3 QUALIDADE DE SOFTWARE..... 19
3.1 Qualidade..... 19
3.1.1 Necessidades dos Clientes..... 20
3.1.2 Estabelecimento de Métricas..... 21
3.1.3 Características das Métricas..... 24
3.2 Qualidade de Software..... 25
3.2.1 Fatores da Qualidade de Software..... 26
3.2.2 Garantia da Qualidade de Software..... 28
3.2.3 Atividades da Garantia da Qualidade de Software..... 29
4 ESTUDO DE CASO E PROPOSTA DE MODELO..... 31
4.1 Apresentação..... 31
4.2 Apresentação e Análise dos Resultados..... 31
4.3 Proposta de Metodologia..... 43
4.3.1 Definição..... 43
4.3.2 Desenvolvimento..... 45
4.3.3 Manutenção..... 47
5 CONSIDERAÇÕES FINAIS..... 49
6 REFERÊNCIAS BIBLIOGRÁFICAS..... 50
ANEXO..... 51

ÍNDICE DE FIGURAS

Figura 1: Pontos Fortes e Fracos da Fase de Definição..... 32
Figura 2: Pontos Fortes e Fracos da Fase de Desenvolvimento..... 33
Figura 3: Pontos Fortes e Fracos da Fase de Manutenção..... 33
Figura 4: Conhecimento das Metodologias de Software pelos Desenvolvedores..... 34
Figura 5: Metodologias Utilizadas pela Empresa..... 35
Figura 6: Demanda por Manutenção..... 37
Figura 7: Obstáculos à Implantação de uma Metodologia de Desenvolvimento..... 38
Figura 8: Freqüência de Atrasos em Projetos..... 39
Figura 9: Freqüência de Extrapolação de Orçamentos..... 40
Figura 10: Precisão no Estabelecimento de Métricas de Custo Ideal e Prazo..... 41
Figura 11: Adoção de Métricas para o Cálculo de Custo Ideal e Prazo..... 41
Figura 12: Eficácia das Metodologias Adotadas..... 42
Figura 13: Interesse na Adoção de uma Metodologia..... 42

1 INTRODUÇÃO

No contexto econômico atual, a concorrência entre as empresas alcançou, em quase todos os setores, proporções internacionais. Seja qual for o ramo de atividade ou o porte da organização em questão (micro, pequena, média ou grande), a disputa pela preferência do cliente é acirrada. Na busca por novos mercados, as empresas investem continuamente na melhoria de seus processos, na pesquisa de mercado e no desenvolvimento de produtos de maior valor agregado e menor custo.

Os clientes estão cada vez mais críticos e atentos ao desempenho dos produtos que adquirem. Somando-se a isso, a fidelização de um cliente possivelmente nunca foi uma tarefa tão árdua como é atualmente. Afinal, a já mencionada concorrência entre as empresas favorece a oferta de uma grande variedade de produtos com preços e atributos similares. Nesse sentido, buscar diferenciais competitivos é um objetivo fundamental de uma organização que deseja se manter atrativa a novos clientes e investidores. O desenvolvimento de produtos de qualidade é uma preocupação essencial para todas essas empresas que buscam se destacar entre seus concorrentes.

Tal preocupação é justificada por Juran (1990) em três argumentos:

- Perdas nas vendas: empresas que não oferecem produtos que primem pela qualidade, fatalmente perdem mercados para aquelas empresas que contam com este atributo como valor agregado;
- Custos da baixa qualidade: deixar de investir em qualidade tem se revelado um grave erro estratégico por representar no futuro um acréscimo sensível nos

custos, por exemplo: reclamação de clientes, desperdício de materiais, processos movidos por órgãos de defesa do consumidor.

- Ameaças à sociedade: a evolução tecnológica criou em um alto grau de dependência da sociedade por computadores, eletrodomésticos, automóveis, alimentos pré-preparados. Quando uma empresa não atenta devidamente para a qualidade daquilo que produz pode pôr em risco a vida das pessoas que utilizam esses produtos.

As empresas de desenvolvimento de software, evidentemente, não podem se furtar a participar desse movimento em busca da qualidade. Desenvolver sistemas com qualidade e custo baixo é um dos objetivos essenciais da empresa de desenvolvimento de software. Para auxiliar tais empresas nesta tarefa, a Engenharia de Software oferece uma vasta literatura sobre o assunto.

No entanto, no caso de empresas pequenas, objeto deste trabalho, o caminho para se produzir um software com qualidade não é simples. A realidade de prazos, recursos humanos e materiais dessas empresas torna difícil a passagem por todas as fases de um projeto de sistema, seja qual for o paradigma adotado, nos moldes recomendados pela Engenharia de Software.

Mesmo com essas dificuldades, o desafio de desenvolver um software com qualidade não pode ser abandonado. E é nesses termos e para essas pequenas empresas de desenvolvimento de software que se apresenta este trabalho: propor um modelo para o desenvolvimento de software com qualidade e custos previsíveis.

1.1 JUSTIFICATIVA

A necessidade que as empresas, especialmente as de pequeno porte que desenvolvem produtos de software, que são objeto deste trabalho.

A qualidade do produto que deve ser contemplada de forma eficiente para se manter competitiva no mercado globalizado.

As dificuldades apresentadas por essas empresas em desenvolver softwares e identificar suas deficiências nas fases de produção foram abordadas e analisadas, tendo como objetivo final deste projeto propor uma metodologia bem definida para desenvolver um produto final de alta qualidade.

1.2 OBJETIVO

O objetivo deste trabalho é propor um modelo de desenvolvimento de software com qualidade para as pequenas empresas de desenvolvimento de software, tendo em vista suas realidades de prazos, orçamentos, recursos materiais e humanos.

1.3 METODOLOGIA

Primeiramente serão apresentados, de maneira resumida, os conceitos da Engenharia de Software que darão subsídios importantes e servirão de base para a construção do modelo anteriormente referido.

Em um segundo momento, será apresentado um estudo de caso envolvendo vinte pequenas empresas de desenvolvimento de software. Neste estudo de caso, será apresentado um breve histórico das empresas, um questionário aplicado às empresas com o objetivo de entender seus modelos de desenvolvimento de software e, por fim, avaliar criticamente a necessidade da adoção de um novo modelo de desenvolvimento melhor organizado e que vise a qualidade do produto final – o software.

Finalmente, um modelo de desenvolvimento de software será descrito e proposto, de acordo com os fundamentos da Engenharia de Software, às pequenas empresas de desenvolvimento de software com o objetivo de auxiliar tais empresas a desenvolver produtos com maior qualidade.

2 ENGENHARIA DE SOFTWARE

A Engenharia de Software é fruto da necessidade observada já em décadas passadas de se desenvolver com qualidade e viabilidade econômica sistemas cada vez mais complexos que atendessem a situações reais e ambientes cada vez mais sofisticados. Também restrições como cronogramas e orçamentos exigem a adoção de uma disciplina que oriente o desenvolvimento de softwares.

A definição para Engenharia de Software dada pelo Institute of Eletrical and Eletronics Engineers (IEEE) é a seguinte:

> "A Engenharia de Software é a aplicação de um processo sistemático, disciplinado e quantificado ao desenvolvimento, operação e manutenção de software, ou seja, a Engenharia de Software é a aplicação de técnicas de engenharia ao software" (INSTITUTE OF ELETRICAL AND ELETRONICS ENGINEERS, 1993. Apud INSTITUTO BRASILEIRO DE TECNOLOGIA AVANÇADA (IBTA), 2005, p. 8).

Um outro ponto de vista é abordado na definição para Engenharia de Software apresentada por Yourdon e Coad:

> “O termo engenharia de software apresenta uma curiosa ironia: embora ele possa dar a impressão de fórmulas, algoritmos e tratamentos científicos complicados, a Engenharia de Software na verdade é baseada fortemente em pessoas” (COAD, Peter e YOURDON, Edward, 1991, p. 9).

Segundo Pressman (1995), a Engenharia de Software é constituída por três elementos básicos: métodos, ferramentas e procedimentos.

Os métodos indicam os caminhos de como se desenvolver um software. Pressman exemplifica da seguinte maneira:

> “Os métodos envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto de estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção” (PRESSMAN, Roger S., 1995, p. 31).

As ferramentas podem ser entendidas como suportes dados aos métodos. Esse suporte pode ser automatizado ou semi-automatizado. Citando novamente Pressman:

> “Quando as ferramentas são integradas de forma que a informação criada por uma ferramenta possa ser utilizada por outra, é estabelecido um sistema de suporte ao desenvolvimento de software chamado engenharia de software auxiliada por computador (CASE – Computer-Aided Software Engineering)” (PRESSMAN, Roger S., 1995, p. 32).

Por sua vez, os procedimentos ligam as ferramentas e os métodos. Procedimentos são organizados em fases e tarefas em que é estipulada uma série de atividades bem definidas bem como os produtos (conseguidos com a aplicação de

ferramentas CASE) dessas atividades, como relatórios, por exemplo. Os procedimentos também indicam a seqüência em que os métodos serão adotados.

Uma determinada combinação de métodos, ferramentas e procedimentos pode ser entendida como um paradigma de Engenharia de software. Os paradigmas mais conhecidos são o Modelo Cascata, o Modelo por Prototipação, o Modelo Espiral e as Técnicas de Quarta Geração.

2.1 FASES GENÉRICAS DO DESENVOLVIMENTO DE SOFTWARE

Para Pressman (1995), independentemente do paradigma adotado, todo projeto de desenvolvimento de software contará com três etapas comuns: a definição, o desenvolvimento e a manutenção.

2.1.1 Primeira Fase: Definição

Na fase de definição, o objetivo do sistema é focalizado. Neste ponto, o essencial é entender qual o problema ou problemas específicos que o sistema busca solucionar. É na fase da definição que todas as informações, os processos, as atividades relevantes pertencentes ao ambiente estudado devem ser analisadas para que o software atenda essas necessidades da melhor maneira possível.

Na definição de um sistema, três etapas ocorrerão de alguma forma (Pressman, 1995):

- Análise do Sistema: nesta etapa será feita a determinação dos papéis desempenhados pelos componentes do sistema. Existem vários métodos de se realizar uma Análise do Sistema, tais como, a Análise Estruturada, a Análise Essencial ou a Análise Orientada a Objetos. Cada método de Análise do Sistema tem a preocupação de entender o sistema de acordo com a sua abordagem específica. No entanto, o mais importante a ressaltar neste tópico é que seja qual for o tipo de análise, o fundamental é entender que:

> "Os analistas têm que considerar o domínio do problema no qual trabalham. Por exemplo, considere o problema do controle de tráfego aéreo. O analista precisa mergulhar no domínio do problema, e mergulhar tão profundamente que começará a descobrir nuanças que mesmo aqueles que lidam com controle de tráfego aéreo todos os dias não tinham considerado completamente" (COAD, Peter e YOURDON, Edward, 1991, p.8).

- Planejamento do Projeto de Software: neste ponto, após ter sido realizada a Análise do Sistema em questão, os recursos humanos serão alocados, serão elaborados os cronogramas, o orçamento, as tarefas serão programadas e agendadas. Um ponto interessante dessa fase, muitas vezes negligenciado pelos projetos de software, é a Análise e Gestão de Risco. Tal atividade prevê uma série de passos a serem seguidos com o intuito de se verificar os riscos potenciais ao projeto de software. E, além da Análise, elaborar um plano de contingência para amenizar os impactos das possíveis perdas, se não for possível neutralizá-las.

- Análise de Requisitos: neste momento, a delimitação do problema a ser resolvido será realizada e analisada por usuários e analistas. O ponto central é que

as partes envolvidas na elaboração do sistema tenham a mesma visão do que será desenvolvido e quais situações o sistema em questão deverá resolver e de que forma. A aplicação de questionários, visitas ao local de trabalho do cliente, conversas com os futuros usuários do sistema, acesso a documentação dos procedimentos dos clientes e reuniões são alguns exemplos de técnicas de Análise de Requisitos. No caso das reuniões, há a possibilidade de se utilizar uma técnica conhecida como Joint Application Design (JAD). A JAD é uma técnica de Análise de Requisitos que foi desenvolvida no sentido de tornar as reuniões mais produtivas, acelerando assim o desenvolvimento de um software. A metodologia JAD está dividida basicamente em três momentos: a reunião inicial, em que os objetivos do projeto são apresentados; a reunião de revisão, na qual os pontos estabelecidos na reunião inicial são discutidos, tais como a situação das tarefas programadas, os problemas detectados ou previstos e suas correções; e, por fim, a sessão de design, em que outras reuniões são feitas para programar e acompanhar o desenvolvimento do projeto de software em questão.

2.1.2 Segunda Fase: Desenvolvimento

- Projeto de Software: o ponto central de um Projeto de Software é representar o produto da Análise de Requisitos em termos de representações como gráficos ou modelos. Durante a fase de Análise de Requisitos não interessa ao analista envolver as restrições de tecnologia na sua análise, no entanto, essas restrições são muito importantes e são consideradas durante a elaboração do Projeto de Software, em que aspectos como arquitetura do sistema, estrutura de dados, linguagem de

programação e padrão de interface gráfica são escolhidos, com o objetivo de atender da melhor maneira possível aos requisitos do sistema.

Em outras palavras:

> “Esta fase produz uma descrição computacional do que o software deve fazer e deve ser coerente com a descrição feita na Análise. Em alguns casos, algumas restrições da tecnologia a ser utilizada já foram amarradas no levantamento de requisitos. Em outros casos, essas restrições devem ser especificadas. Mas, em todos os casos, a fase de projeto do sistema é direcionada pelos modelos constituídos na fase de análise e pelo planejamento do sistema” (BEZERRA, Eduardo, 2002, p. 25).

- Codificação: é o estágio em que as representações constantes no Projeto de Software são convertidas em uma linguagem de programação que melhor atenda às necessidades do projeto.

- Testes do software: durante os Testes de Software são verificados possíveis erros de lógica ou implementação. Um aspecto interessante a ser lembrado é que um Relatório de Testes pode ser elaborado contendo informações sobre os erros encontrados durante os testes.

2.1.3 Terceira Fase: Manutenção

- Correção: a Manutenção Corretiva se presta a corrigir os defeitos encontrados no software, normalmente encontrados pelos clientes.

- Adaptação: a evolução do ambiente (Sistema Operacional, Hardware) para o qual o software foi desenvolvido pode exigir uma Adaptação, que consiste em adequar novamente o software ao seu ambiente.
- Melhoramento Funcional: o uso diário do software por parte dos usuários pode resultar na sugestão de implantação de novas funcionalidades, adição ou eliminação de ferramentas.

3 QUALIDADE DE SOFTWARE

A seguir serão apresentados resumidamente os principais conceitos ligados à Qualidade, em termos de ramo do conhecimento administrativo, bem como os conceitos básicos ligados à Qualidade de Software.

3.1 QUALIDADE

A partir da década de oitenta, as empresas passaram a ser preocupar progressivamente em desenvolver produtos de qualidade. Entretanto, neste momento, a concepção predominante entre as empresas estabelecia que bastava atender às especificações de um produto para se dissesse que tal produto tinha qualidade. Posteriormente, entendeu-se que Qualidade é atingida principalmente quando se satisfaz as necessidades dos clientes. De qualquer modo, não é tarefa simples definir Qualidade.

Das várias definições possíveis e plausíveis para Qualidade, Juran (1990) aponta dois aspectos elementares para o entendimento do que vem a ser

Qualidade. Estes aspectos interferem diretamente no Planejamento da Qualidade e no Planejamento Estratégico de qualquer empresa: o desempenho do produto e a ausência de defeitos.

A Qualidade, observada através do prisma do desempenho de um produto, se refere a fatores como rapidez atendimento às solicitações do cliente, velocidade de processamento de dados de um computador, consumo de combustível de um automóvel. Dessa maneira, é possível afirmar que é influenciado por esses diferenciais de desempenho que um cliente decide comprar um determinado produto e não outro. E é por isso que as empresas se esforçam para que o desempenho de seus produtos tenha Qualidade igual ou superior ao de seus concorrentes.

Analisada sob o crivo da ausência de defeitos, a Qualidade pode ser entendida como um esforço planejado para se reduzir, em alguns casos evitar, entregas atrasadas, travamento de sistemas, faturas incorretas, atrasos de cronogramas ou extrapolação de orçamentos. Portanto, a organização que não atenta devidamente para a busca da ausência de defeitos, corre o risco de ver suas vendas em longo prazo tomarem uma trajetória descendente. Vale lembrar que buscar a ausência de defeitos é um esforço só realizável em longo prazo.

3.1.1 Necessidades dos Clientes

Seja qual for o entendimento que se tenha do que é Qualidade, há que se admitir que para atingir a Qualidade é essencial conhecer as necessidades dos clientes. Assim, estabelecer métodos de coleta, tabulação e tradução de informações a respeito das necessidades dos clientes é uma tarefa muito importante.

A coleta de dados a respeito das necessidades dos clientes é um trabalho que exige tempo e dedicação da empresa. Essa coleta de dados pode ser realizada através de muitas maneiras, tais como questionários por correio, visitas a clientes ou telefonemas.

A tabulação pode contar com o auxílio de ferramentas como planilhas eletrônicas e processadores de texto.

E, por fim, a tradução dessas informações é um ponto crucial para que essas informações sejam compreendidas pelos responsáveis estratégicos da organização. A maneira mais eficaz de traduzir essas informações é a utilização de números, medições e sensores que sejam universalmente entendidos.

3.1.2 Estabelecimento de Métricas

No planejamento da Qualidade é essencial trabalhar com informações precisas. Essa precisão se faz necessária porque, como já foi visto, a Qualidade só pode ser alcançada mediante o bom atendimento às necessidades dos clientes. Na comunicação entre clientes e fornecedores, muitos termos podem ser compreendidos facilmente somente com o uso de palavras, no entanto, muitos outros aspectos relacionados ao planejamento, desenvolvimento, controle e produção de um bem ou serviço requerem uma comunicação mais precisa. A compreensão das necessidades dos clientes sob tais aspectos se torna mais simples com se utilizam números para expressá-los.

Uma unidade de medida é “uma quantidade definida de alguma característica de Qualidade” (Juran, J. M., 2003, p. 118), ou seja, a representação de

uma característica de qualidade em números. Exemplos de unidades de medida podem ser horas para serviços de entrega.

Para extrair ou analisar as unidades de medida, se utilizam sensores. Assim, um sensor é um método ou um instrumento que auxilia a avaliação de uma determinada unidade de medida, como, por exemplo, uma balança para se medir a massa de um corpo em quilos e gramas.

Anteriormente, foi dito que a Qualidade pode ser compreendia sob dois aspectos principais: o desempenho do produto e a ausência de defeitos. Com relação a esta última, Juran (2003) apresenta uma fórmula básica para se medir a qualidade de um produto. Tal fórmula é atribuível às características maioria das deficiências de bens ou serviços.

Qualidade = Freqüência de Deficiências ÷ Oportunidade para Deficiências

Em que a Freqüência das Deficiências corresponde ao número de ocorrências de defeitos na prestação, utilização ou produção do produto, como número de peças com defeito, número de entregas atrasadas. Já a Oportunidade para Deficiências representa o número total de prestação, utilização ou produção do produto, por exemplo, número total de peças produzidas ou número total de entregas realizadas. Os resultados assumem a forma de percentuais ou proporções, como, percentual de erros, proporção entre entregas realizadas com atraso e entregas realizadas no prazo.

Mesmo com essa fórmula básica de se medir a Qualidade de um produto quanto à ausência de deficiências é importante traduzir existem situações em que importante utilizar uma unidade de medida única para avaliar o impacto de todas as

deficiências de um produto. A unidade geralmente utilizada é o dinheiro. Dá-se a essa conversão de medidas de Qualidade em medida de custo o nome de Custo da Má Qualidade - CDMQ (Juran, 2003). Em linhas gerais, CDMQ é a soma de todos os custos que não haveriam na produção se os produtos e os processos fossem perfeitos, como o custo de horas extras para trabalhadores, custos com manutenção de produtos que contêm defeitos.

Apesar das poucas informações contábeis que as empresas, principalmente as pequenas, têm sobre esses custos, é possível se concentrar nas estimativas aproximadas dessas cifras e em que áreas da empresa esses custos se concentram. Ademais, perguntar-se quais custos presentes desapareceriam se os produtos e processos fossem perfeitos é uma outra ferramenta importante na determinação do CDMQ.

Quando a qualidade é avaliada levando-se em consideração o desempenho do produto, o ambiente para o estabelecimento de métricas oriundas de um método unificado de cálculo se torna mais adverso. Afinal, não há como se estabelecer uma única unidade de medida capaz de representar tantas características inerentes a tantos produtos. Mesmo assim, quando a empresa busca entender de que maneira o seu mercado mede as características de um determinado produto.

Quando se avalia a Qualidade de um bem quanto ao seu desempenho, o uso de unidades tecnológicas (tempo em horas ou temperatura em graus, por exemplo) é muito freqüente e recomendável por serem unidades de medida objetivas.

Ao avaliar a Qualidade o desempenho de um serviço, porém, não é possível utilizar muitas métricas objetivas, como a presteza que é medida por horas ou minutos. A maior parte das características dos serviços são naturalmente subjetivos,

como a cortesia ou a legibilidade de relatórios, e, portanto, são medidas pela opinião humana.

3.1.3 Características das Métricas

Em primeiro lugar, uma unidade de medida deve ter uma definição precisa, ou seja, precisam assegurar uma comunicação com o menor nível de incerteza possível ou mesmo nenhuma incerteza. Em muitos casos, as métricas são subdivididas. Assim, um erro pode ser crítico, grande ou pequeno.

A segunda característica a ser ressaltada é a diferenciação entre medições por atributos e medições por variáveis. As primeiras avaliam a existência ou não de uma característica, como, se sim ou não, contrato fechado ou não. Já as medições por variáveis analisam uma condição por escala graduada, por exemplo, prazo de entrega por dias.

Por fim, algumas características de Qualidade se caracterizam por serem abstratas. Nesses casos, a métricas geralmente adotadas verificam a falta dessas características. Como a falta de cortesia.

Dessa maneira, é possível apontar os parâmetros que definem uma unidade de medida ideal:

- Seus resultados são compreensíveis a todos os setores da empresa da mesma maneira;

- Dá subsídios para a tomada de decisões;
- Pode ser utilizada em larga escala pela organização;
- Seus resultados são interpretados de uma forma uniformizada;
- O custo de ter tal medida apresenta uma boa relação com o valor que ela acrescenta à empresa;
- Há compatibilidade entre a unidade de medida e os sensores existentes na empresa;

Em sintonia com esses parâmetros, os caminhos para se criar uma nova unidade de medida são sintetizados por Juran (2003) da seguinte forma:

- Elaborar uma missão de criar meios de medir a Qualidade de um determinado produto;
- Delegar essa missão a uma equipe;
- Elaborar relatórios de desempenho segundo as unidades sugeridas pela equipe designada;
- Avaliar tais métricas mediante seu uso, efetuando revisões e colhendo as opiniões dos usuários. Este ponto é nuclear para o sucesso das novas métricas, pois, por se tratar de um processo de evolução, a experiência dos usuários é fundamental;

3.2 QUALIDADE DE SOFTWARE

A Qualidade de Software é um importante objetivo a ser alcançado no desenvolvimento profissional de softwares. No entanto, observa-se uma visão equivocada por parte de alguns desenvolvedores que põem ênfase na Qualidade

somente após a escrita do código. Este comportamento é extremamente equivocado (Pressman, 1995).

Um outro problema encontrado na busca por desenvolver softwares com Qualidade é a questão primária de se definir o que é Qualidade de Software. Das várias definições disponíveis, a definição apresentada por Pressman se mostra mais apropriada para o alcance do objetivo deste trabalho. Em seu livro, Pressman define Qualidade de Software como:

> "Conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento explicitados declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido" (PRESSMAN, Roger S. 1995, p. 724).

A definição acima fornece insumos para compreender a Qualidade de Software de uma maneira mais completa. A saber:

- Quando se fala que a Qualidade de Software é traduzida em adequação aos requisitos levantados, se cria uma base sobre a qual a Qualidade pode ser medida, isto é, a não conformidade com os requisitos representa falta de Qualidade;
- Os padrões de desenvolvimento estabelecidos durante o Projeto de Software têm que definir parâmetros que indiquem claramente como o software passará pelo trabalho de Engenharia. Assim, o desrespeito aos padrões de desenvolvimento poderá resultar em falta de Qualidade;
- Os requisitos implícitos (requisitos que não são definidos durante a Análise de Requisitos, mas que são desejados, como a boa manutenibilidade do sistema) devem ser atendidos assim como os requisitos levantados por Análise, sob pena de não se atingir a Qualidade desejada;

3.2.1 Fatores da Qualidade de Software

A necessidade de medir a Qualidade de um software é condição imprescindível para que a empresa de desenvolvimento de software avalie os impactos que o seu produto final – o software – tem sobre o cliente. Dessa maneira, é preciso apontar os fatores que devem ser considerados para uma avaliação eficaz de um sistema. Esses Fatores da Qualidade de Software, segundo a proposição de McCall (Pressman, 1995), se detêm em três aspectos principais: os aspectos operacionais, a manutenibilidade do software e sua adaptabilidade a novos ambientes.

Sobre os aspectos operacionais do sistema, os Fatores da Qualidade do Software são:

- Corretitude: a medida de cumprimento das especificações feitas e de atendimento das necessidades do cliente;
- Confiabilidade: em linhas gerais, o nível de confiança que se tem de que uma determinada função do sistema será realizada com a precisão exigida pelo cliente;
- Eficiência: o nível de recursos de máquina, de tempo, de código utilizados pelo software para executar suas funções;
- Integridade: a medida de controle de acesso oferecida pelo sistema para impedir sua utilização por pessoas não autorizadas;
- Usabilidade: o grau de facilidade que os usuários terão para aprender, operar, entrar, com, extrair e analisar informações do sistema;

Quanto a manutenibilidade do sistema, os Fatores da Qualidade do Software importantes são:

- Manutenibilidade: de modo genérico, o nível de complexidade com que erros podem ser encontrados e corrigidos;
- Flexibilidade: quão difícil será alterar o sistema;
- Testabilidade: o esforço necessário para efetuar testes em funções do software de modo a garantir uma execução melhor das funções;

E, finalmente, a adaptabilidade do sistema a novos ambientes é caracterizada por Fatores como:

- Portabilidade: o nível de facilidade na mudança requerida para migrar entre diversas plataformas de hardware ou software;
- Reusabilidade: medida que informa como e com que facilidade partes de um sistema podem ser utilizadas em um outro sistema;
- Interoperabilidade: grau de esforço de se juntar um sistema à outro;

Infelizmente, muitos desses fatores não podem ser mensurados objetiva ou diretamente. No entanto, como já foi descrito anteriormente, é possível estabelecer unidades de medida de modo a garantir um melhor atendimento às necessidades dos clientes. Os Fatores da Qualidade de Software indicam, de qualquer modo, apontamentos interessantes para se quantificar, ou seja, transformar em números, a Qualidade de um sistema.

3.2.2 Garantia da Qualidade de Software

A Garantia da Qualidade de Software é uma atividade vital para qualquer empresa de desenvolvimento de software. Em linhas gerais, a Garantia da Qualidade de Software (GQS), se destina a avaliar se o software está sendo desenvolvido de acordo com o Processo de Desenvolvimento de Software estabelecido e utilizado pela empresa e por verificar se o produto (o software) atende às necessidades dos clientes aos quais se destina. Necessidades que foram levantadas e analisadas durante a Análise de Requisitos.

É necessário que essa atividade de caráter essencialmente verificador e validador de processos e produtos conte com a participação de membros de várias áreas da empresa, desde o pessoal de vendas até o gerente do projeto (Pressman, 1995) formando, assim, um grupo cujo fim é responder a questionamentos sobre a adequação do desenvolvimento de software aos padrões estabelecidos e o atendimento aos Fatores da Qualidade de Software, por exemplo. Esses grupos se assemelham aos Círculos de Qualidade, adotados inicialmente no Japão com o objetivo de alcançar um aperfeiçoamento da Qualidade do produto e um auto-aperfeiçoamento (Baker, 1991).

3.2.3 Atividades da Garantia da Qualidade de Software

As variadas tarefas atribuíveis a GQS podem ser agrupadas em sete grandes atividades (Pressman, 1995):

- Aplicação de Métodos Técnicos: que garante a realização de especificações e desenvolvimento de projetos de alta Qualidade mediante a utilização de métodos e ferramentas técnicas;

- Revisões Técnicas Formais: que possibilita a detecção de falhas e inconformidades em especificações (protótipos) e projetos através de reuniões estilizadas;
- Testes de Software: que colabora em uma detecção mais efetiva de erros conseguida através de uma combinação entre passos estabelecidos estrategicamente com aplicação de métodos de teste;
- Aplicação de Padrões: que consiste em verificar a adequação dos padrões especificados para o desenvolvimento do produto à atividade de desenvolvimento propriamente dita. Ou seja, uma verificação sobre o cumprimento dos padrões, padrões que podem ser estabelecidos pela empresa desenvolvedora de software, pelo cliente, ou por uma instituição externa ou governo;
- Controle de Mudanças: mudanças no software podem resultar em efeitos indesejados e imprevisíveis ou de difícil detecção, portanto, sempre que forem solicitadas, as mudanças devem passar por um processo de Controle de Mudanças que favoreça a Qualidade do produto final através de avaliações da natureza das mudanças e do controle de seus impactos sobre o sistema. O Controle de Mudanças pode ser necessário tanto no desenvolvimento do software quanto na sua manutenção;
- Medição: que é inerente a qualquer disciplina de Engenharia e no contexto da Garantia da Qualidade de Software se faz necessária para quantificar a Qualidade do software e avaliar impactos de mudanças metodológicas e procedimentais;
- Manutenção de Registros e Reportagem: que é o esforço sistemático e contínuo de documentação de procedimentos de coleta de dados, resultados de revisões, auditorias, controles de mudança, testes e outras atividades da

Garantia da Qualidade de Software. O objetivo central dessa atividade é preservar e disseminar o conhecimento que se tem de um projeto específico;

4 ESTUDO DE CASO E PROPOSTA DE MODELO

4.1 APRESENTAÇÃO

Para propor uma metodologia de desenvolvimento de software, buscou-se compreender, através de um questionário, os comportamentos de desenvolvimento de 20 pequenas empresas de desenvolvimento de software.

Como muitas empresas e profissionais não quiseram ser identificados, estas informações foram omitidas no questionário.

Todas as empresas pesquisadas situam-se no estado de São Paulo, cinco empresas, localizadas na cidade de São Carlos, participam do Projeto Empreender, cujo objetivo principal é:

> "Atender às Associações Comerciais e Empresariais organizando núcleos setoriais e valorizando o aspecto associativo. Em um núcleo setorial, que reúne empresas do mesmo segmento, para entrar em contato direto com outras empresas, buscar soluções comuns para questões que, sozinho, o empresário teria dificuldades em resolver". (SEBRAE)

4.2 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

O levantamento realizado com 20 pequenas empresas de desenvolvimento de software teve como preocupação entender seus hábitos de desenvolvimento de software.

Mais que meros dados estatísticos, é possível analisá-los e extrair deles informações interessantes e que ajudam a traçar um perfil dessas empresas e, com

base nesse perfil, é possível propor uma metodologia que atenda às demandas do processo de desenvolvimento.

A questão 6 (das fases da metodologia básica de desenvolvimento de software descrito por Roger S. Pressman, quais são pontos fracos e quais são os pontos fortes no processo de desenvolvimento de softwares da empresa, ainda que essas fases não ocorram nesta seqüência ou de uma maneira metódica) apresentou um leque muito interessante de características existentes no processo de desenvolvimento das empresas pesquisadas. Os resultados seguem abaixo:

Error! Not a valid link.

Figura 1: Pontos Fortes e Fracos da Fase de Definição

Error! Not a valid link.

Figura 2: Pontos Fortes e Fracos da Fase de Desenvolvimento

Error! Not a valid link.

Figura 3: Pontos Fortes e Fracos da Fase de Manutenção

Esses gráficos fornecem uma importante fonte de informações a respeito da situação das empresas pesquisadas. Na Figura 1, correspondente à fase de Definição do que precisa ser desenvolvido, as empresas pesquisadas apresentam um perfil bastante favorável, especialmente na etapa de Análise de Requisitos, em que 85% das empresas consideram esta etapa como um ponto forte no seu processo de desenvolvimento. Mesmo a Análise de Sistema foi apontada por 60% das empresas

como um ponto forte. Este bom desempenho é compreendido melhor quando se analisam outros gráficos extraídos deste levantamento. Para a primeira pergunta (As pessoas envolvidas com o desenvolvimento de software da sua empresa têm conhecimentos sobre as principais metodologias de desenvolvimento de software?), as respostas foram:

Error! Not a valid link.

Figura 4: Conhecimento das Metodologias de Software pelos Desenvolvedores

A segunda pergunta (A empresa adota alguma metodologia para o desenvolvimento de softwares? Qual?) apresentou os seguintes resultados:

Error! Not a valid link.

Figura 5: Metodologias Utilizadas pela Empresa

Assim, como a maioria das pessoas tem conhecimento sobre as principais metodologias e a maioria das empresas (80%) tem alguma metodologia, seja RUP (Rational Unified Process), Análise Estruturada, Espiral ou Evolutiva, uma metodologia própria ou outra metodologia. Todas essas metodologias apontam a importância de uma boa Definição do sistema antes de começar a fase de Desenvolvimento.

Alguns dados parecem influenciar outros como é o caso da etapa de Projeto de Software. Afinal, 75% das empresas disseram ser essa etapa um ponto forte do seu processo de desenvolvimento e, não por acaso, um número muito próximo, 70%

das empresas, apontaram o Planejamento do Projeto de Software como um ponto forte também. Ou seja, se conclui que quando é feito um bom Planejamento, consegue-se um bom Projeto de Software.

Outro exemplo disso é a Análise de Requisitos. Um bom levantamento dos requisitos produz uma Codificação mais simples (considerada por 75% das empresas como um ponto forte). Ora, conhecendo bem o que determinado cliente precisa, conhecendo a fundo os requisitos funcionais (as funcionalidades de um sistemas como o que um usuário pode fazer ou não, quais informações devem estar disponíveis a determinado usuário) e os não-funcionais (como o desempenho do software, a portabilidade em diversas plataformas ou a confiabilidade da informação) é possível codificar com mais eficácia.

O inverso também é verdadeiro, já que, por exemplo, 80% das empresas consideram a Realização de Testes de Software um ponto fraco. Isto implica na taxa de 75% de empresas que consideram a Manutenção Corretiva como um ponto forte no seu processo de desenvolvimento. Pode-se supor que com tamanha demanda por manutenção, as empresas fazem da Correção uma atividade rotineira e que precisa ter a máxima eficiência.

A grande demanda por manutenção foi constatada na pergunta 11 (Há uma grande demanda de solicitação por manutenção?) e revelou o seguinte:

Error! Not a valid link.

Figura 6: Demanda por Manutenção

Mesmo a demanda por Manutenção Adaptativa, colocada como ponto fraco por 80% das empresas, tem esse resultado condicionado pelo tempo dispensado com Manutenção Corretiva. O Melhoramento Funcional também revela um resultado negativo com somente 30% das empresas considerando esta etapa como um ponto forte do seu processo de desenvolvimento.

O gráfico abaixo dá indícios de onde se origina essa carência de uma etapa de realização de testes bem estruturada e da grande demanda por manutenção. Ele foi extraído das respostas dadas à pergunta 3 (Qual o maior obstáculo para a implantação de uma metodologia de desenvolvimento?).

Error! Not a valid link.

Figura 7: Obstáculos à Implantação de uma Metodologia de Desenvolvimento

O que revela que mais para 70% das empresas, os maiores obstáculos à implantação de uma metodologia de desenvolvimento poderiam ser solucionados com uma mudança de postura da cultura empresa e do pessoal ligado à área de desenvolvimento.

Um outro motivo para tal deficiência na área de Testes e Manutenção é respondido quando é analisado o gráfico realizado com as respostas dadas às perguntas 8 (Se for possível estimar o prazo dos projetos, com que freqüência o prazo estimado é excedido?) e 9 (Se for possível estimar o custo ideal dos projetos, com que freqüência o custo ideal estimado é excedido?).

Error! Not a valid link.

Figura 8: Freqüência de Atrasos em Projetos

Error! Not a valid link.

Figura 9: Freqüência de Extrapolação de Orçamentos

Os dados revelam um número muito alto de atrasos em projetos (65% atrasam freqüentemente e 20% atrasam sempre) e que pode ter várias causas como falhas na definição do sistema, alto número de alterações nos requisitos do projeto original, inexperiência da equipe de desenvolvimento com o paradigma utilizado ou uma Análise de Risco feita sem a devida atenção.

O fator empírico na estipulação dessas estimativas deve ser levado em consideração também, já que 60% das empresas dizem calcular prazo e custo ideal com precisão (pergunta 7), mas 70% alegam não ter uma métrica para fazer tais estimativas (pergunta 10). As figuras 10 e 11 apresentam esses dados:

Error! Not a valid link.

Figura 10: Precisão no Estabelecimento de Métricas de Custo Ideal e Prazo

Error! Not a valid link.

Figura 11: Adoção de Métricas para o Cálculo de Custo Ideal e Prazo

Assim, a partir do momento em que a empresa passa a adotar alguma métrica para calcular o prazo e o custo ideal do projeto, estas estimativas passam a ter mais chances de serem cumpridas.

Por fim, 62,50% das empresas que adotam uma metodologia, consideram-na eficaz (pergunta 5), não obstante os altos índices de atrasos em prazos e em extrapolação de orçamentos. E, confirmando os dados extraídos na pergunta 3, 80% das empresas que não adotam uma metodologia de desenvolvimento, não têm interesse em adotar (pergunta 4), em uma demonstração de barreira cultural da empresa.

Error! Not a valid link.

Figura 12: Eficácia das Metodologias Adotadas

Error! Not a valid link.

Figura 13: Interesse na Adoção de uma Metodologia

4.3 PROPOSTA DE METODOLOGIA

Com base nos dados e nos gráficos estudados anteriormente, é possível propor uma metodologia para o desenvolvimento de software que diminua as deficiências (os pontos fracos) encontradas no processo de desenvolvimento de software das empresas pesquisadas.

Cabe ressaltar que, como já foi explicitado no Objetivo deste trabalho, a proposta de Metodologia que será descrita aqui terá como preocupação dar sugestões para cobrir as principais deficiências detectadas pela análise dos dados anterior.

4.3.1 Definição

Apesar dos bons índices apresentados quanto à fase de Definição, muitas lacunas existem, tendo em vista a alta freqüência de atrasos, por exemplo. Por isso, a identificação das necessidades dos clientes é um ponto tão importante para o sucesso de um projeto. Assim, a Análise de Sistema deve ser feita com a maior atenção, com toda a documentação necessária. É muito comum que as empresas não dêem a devida atenção à documentação, geralmente por causa do custo que isso tem e do tempo que demanda, no entanto, é uma espécie de economia que futuramente se revela inútil e equivocada, gerando retrabalho, impedindo a reutilização de análises realizadas em projetos anteriores e impossibilitando muitas vezes uma definição apropriada do sistema em questão.

Admitindo que todas as fases do desenvolvimento de software têm uma etapa que é o cerne de todas as atividades a serem realizadas em determinada fase, é lícito aceitar a Análise de Requisitos como o núcleo da fase de Definição.

Para tal etapa, a técnica JAD. Evidentemente, por este trabalho se dirigir a pequenas empresas, nem todas as determinações do JAD podem ser exercidos por tais companhias. Portanto, se propõe que ao realizar a Análise de Requisitos, além do que já foi dito anteriormente, o JAD seja aplicado da seguinte maneira:

- Para as reuniões, a presença de um moderador (que pode ser algum representante do cliente ou alguém empresa desenvolvedora), cujo principal papel seria conduzir as discussões, o fechamento de acordos, a resolução de conflitos e o controle do tempo de reunião; um secretário que deve tomar nota do que é discutido

e acertado entre a empresa de desenvolvimento e o cliente; um ou mais representante dos clientes, para expor as necessidades que têm e que o sistema deverá suprir; e um ou mais representante dos desenvolvedores que estabeleceria os limites da equipe de desenvolvimento e acompanharia com o cliente as restrições e riscos do projeto. O contato pessoal, em reuniões de duração não muito longas e bem conduzidas é fundamental para um bom entendimento do que precisa ser desenvolvido para atender às necessidades dos clientes;

- A fim de orientar as reuniões, alguns pontos não podem deixar de constar em uma reunião estruturada como o JAD: abertura dos trabalhos; introdução à computação; apresentação do organograma da empresa; recordação do fluxo atual das atividades; problemas, defeitos e empecilhos do fluxo atual; definição dos objetivos; racionalização do fluxo atual; proposição do novo fluxo; análise dos dados de entradas; relatórios e telas; definição do conteúdo dos arquivos; cronograma (prazos e responsáveis); e encerramento.

A Análise de Risco também é um fator muito negligenciado no processo de desenvolvimento das empresas, razão pela qual há tão altos índices de atrasos em cronogramas e gastos maiores que os previstos.

Assim como a Análise do Sistema, a Análise de Risco deve ser realizada pelo analista envolvido com a maior dedicação possível, buscando encontrar o maior número possível de riscos, do mais remoto ao mais provável. E não somente isso, mas também, e principalmente, preparar um plano de contingência para cada risco identificado durante o processo de Análise.

Essa Análise pode ser feita de maneira muito simples, com uma tabela feita em qualquer processador eletrônico de textos. Uma tabela com, no mínimo três

colunas: a primeira com a descrição do risco, a segunda com a classificação (em escalas) dos efeitos (como remoto, ignorável, catastrófico, ou qualquer outra escala de valores ou nominal que mais conviesse à empresa) e uma terceira que contivesse o plano de contingência para enfrentar o risco detectado. Tal tarefa, no que pese a dedicação necessária à Análise de Risco, é muito simples de ser executada, mas de relevância crucial.

4.3.2 Desenvolvimento

Neste ponto, a Codificação e a Realização de Testes de Software parece ser ponto crítico a ser enfrentado.

A reusabilidade de códigos, e a robustez devem ser consideradas linhas mestras da Codificação de sistemas.

- Reusabilidade: a reutilização de códigos precisa respeitar algumas normas como a coerência dos métodos ou funções (cada método ou função deve realizar uma única tarefa facilmente reconhecível e bem definida, se um método ou função realiza mais de uma tarefa, é aconselhável dividi-lo em dois ou mais outros métodos ou funções); os métodos ou funções devem ser pequenos; os métodos ou funções devem ser consistentes (manter mesmos estilos de nome, condições, ordem de argumentos, condições de erro); os tipos de argumentos, precondições devem ser o mais genéricas possível;
- Robustez: a robustez de um método ou de uma função é a sua capacidade de lidar sem falhas com parâmetros inválidos. Quando a eficiência do software conflita com a sua robustez, esta pode ser preterida por aquela, mas nunca se

deve sacrificar a robustez com relação aos erros dos usuários. Assim, uma série de medidas como a validação das entradas de usuários, a validação de argumentos e evitar limites predefinidos (utilizando a alocação dinâmica de memória) podem ser tomadas em busca pela robustez do código;

A Realização de Testes de Software também foi apontada como uma séria deficiência no processo de desenvolvimento de software das empresas pesquisadas. Esta etapa é considerada por muitos desenvolvedores como muito difícil de ser realizada já que estes são pessoas acostumadas a construir sistemas e não a destruí-los (Pressman, 1995). No entanto, é fundamental que se realize testes de software antes de colocá-lo à disposição dos clientes, ou mesmo colocar o software à disposição do cliente, mas em caráter de testes, como um protótipo. Mas é importante ter sempre em mente que uma boa Realização de Teste irá revelar, no mínimo, uma falha. Por mais fatalista que pareça essa afirmação, ela deve ser encarada com serenidade pelos desenvolvedores. È possível ainda afirmar que se, ao fim dos testes não for encontrado nenhum defeito, a Realização dos Testes não foi bem executada. Afinal, é muito mais interessante para a empresa descobrir os defeitos de seu software antes de entregá-lo ao cliente, o que, do contrário, provocaria retrabalho, gastos inesperados, perda de qualidade e insatisfação do cliente.

Resumidamente, uma boa política de testes deve conter três linhas mestras: encarar a Realização de Testes como um processo em se executa um programa na intenção de se descobrir erros; entender que um bom teste é aquele que tem uma alta probabilidade de apresentar erros ainda não descobertos e que um teste só pode ser considerado eficaz, se houver encontrado, no mínimo uma falha; se não, há alguma lacuna na Realização dos Testes.

4.3.3 Manutenção

A Manutenção não deve ser encarada com reservas. Tão estranho e tão verdadeiro quanto afirmar que uma eficaz Realização de Testes fatalmente encontrará defeitos, é afirmar que mesmo que sistemas sejam bem projetados deverão passar por Manutenção. Evidentemente que o tempo gasto com Manutenção só não deveria ocupar mais tempo e recursos que o desenvolvimento dos softwares.

De acordo com a defasagem demonstrada pelo levantamento realizado, a Manutenção Adaptativa merecerá atenção especial deste trabalho.

No ambiente globalizado, em que as evoluções em hardware e software ocorrem rapidamente, é necessário manter um programa sólido de Manutenção Adaptativa para que os sistemas desenvolvidos possam ser utilizados pelos diferentes ambientes e tecnologias desenvolvidos.

Neste sentido, investir na portabilidade também é uma opção válida para que um sistema não fique tão dependente de um ambiente específico. Além de uma solução que diminui o tempo gasto pela empresa com Manutenção, tal medida pode ser trabalhada como um diferencial competitivo que fará um cliente optar por um sistema ou por outro concorrente.

5 CONSIDERAÇÕES FINAIS

Deste trabalho conclui-se que a ênfase na Qualidade é uma tendência entre as empresas de todos os ramos de atividade econômica e as empresas de desenvolvimento de software não podem se furtar a desenvolver sistemas com Qualidade sob pena de perderem seus clientes e não resistir em um cenário altamente competitivo e globalizado.

Na busca pela Qualidade, a Engenharia de Software fornece importantes métodos, procedimentos e ferramentas que auxiliam as empresas de desenvolvimento de software. Um bom Projeto de Software deve contemplar a Análise do Sistema, Planejamento do Projeto de Software, Análise de Requisitos, Projeto de Software, Codificação, Testes de Software, Manutenções Corretiva, Adaptativa e para Melhoramento Funcional.

Através de levantamento realizado com 20 pequenas empresas de desenvolvimento foi possível constatar bons resultados na Definição de sistemas, mas uma carência muita grande na Realização de Testes, o que reflete em uma alta demanda por Manutenção e uma grande freqüência de atrasos em cronogramas e custos finais mais altos que os previstos.

Com base nos resultados extraídos desta pesquisa foi possível propor linhas gerais de uma metodologia que oriente as empresas na solução dessas deficiências. A eficácia dessa metodologia poderá ser avaliada em trabalhos posteriores.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML – São Paulo, 2002. Editora Campus.

COAD, Peter e YOURDON, Edward, Análise Baseada em Objetos – São Paulo, 1991. Editora Makron Books.

IBTA, Instituto Brasileiro de Tecnologia Avançada. Apostila Fundamentos Engenharia de Software – São Paulo, 2005.

JURAN, J. M. A Qualidade desde o Projeto – São Paulo, 2003. Editora Thomson Learning.

JURAN, J. M. Controle da Qualidade: Componentes Básicos da Qualidade – São Paulo, 199. Editora Makron Books. (pgs 54~147) (Autor do Capítulo: Edward M. Backer).

PORTAL SEBRAE – São Paulo. Projeto Empeender Disponível em: http://www.sebrae.com.br/br/programaseprojetos/programaseprojetos_1942.asp. Último acesso em 16 de novembro de 2005.

PRESSMAN, Roger S. Engenharia de Software – São Paulo. Editora Makron Books, 1995.

ANEXO

O questionário abaixo foi aplicado a 20 pequenas empresas pesquisadas neste estudo de caso:

Primeira Parte – Engenharia de Software

1. As pessoas envolvidas com o desenvolvimento de software da sua empresa têm conhecimentos sobre as principais metodologias de desenvolvimento de software?

(_____) Sim (_____) Não

2. A empresa adota alguma metodologia para o desenvolvimento de softwares? Qual?

(_____) Sim, RUP (_____) Sim – Uma metodologia própria.

(_____) Sim, Análise Estruturada (_____) Sim, Análise Espiral/Evolutiva

(_____) Sim, Outra _____.

(_____) Não

3. Qual o maior obstáculo para a implantação de uma metodologia de desenvolvimento? Por favor, escolha somente um.

(_____) Investimento Necessário (_____) Desconhecimento

(_____) Cultura da Empresa (_____) Costumes do Pessoal ligado ao Desenvolvimento

4. Se a empresa não adota nenhuma metodologia de desenvolvimento, existe interesse em adotar uma metodologia de desenvolvimento, ainda que própria?

(_____) Sim (_____) Não

5. Se a empresa adota uma metodologia de desenvolvimento, ela é considerada eficaz?

(_____) Sim, é eficaz (_____) Não, não é eficaz

6. A Engenharia de Software sugere, segundo o modelo proposto por Roger S. Pressman, uma metodologia de desenvolvimento básica. Das fases dessa metodologia, listadas abaixo, quais são pontos fracos e quais são os pontos fortes no processo de desenvolvimento de softwares da empresa, ainda que essas fases não ocorram nesta seqüência ou de uma maneira metódica.

Fase	*Ponto Forte*	*Ponto Fraco*
Análise do Sistema		
Planejamento do Projeto de Software		
Análise (ou Levantamento) de Requisitos		
Projeto de Software		
Codificação		
Realização de Testes de Software		
Correção		
Adaptação		
Melhoramento Funcional		

Segunda Parte – Qualidade de Software

7. Tendo em vista o modo como a empresa desenvolve softwares, é possível estimar com precisão o prazo e o custo ideal do projeto?

(_____) Sim (_____) Não

8. Se for possível estimar o prazo dos projetos, com que freqüência o prazo estimado é excedido?

(_____) Sempre. (_____) Freqüentemente

(_____) Raramente (_____) Nunca

9. Se for possível estimar o custo ideal dos projetos, com que freqüência o custo ideal estimado é excedido?

(_____) Sempre (_____) Freqüentemente

(_____) Nunca (_____) Raramente

10. Há uma métrica para calcular esta estimativa?

(_____) Sim (_____) Não

11. Há uma grande demanda de solicitação por manutenção?

(_____) Sim (_____) Não