CENTRO UNIVERSITÁRIO FEEVALE

EDISON LUCIANO DE MELLO

PROPOSTA DE IMPLANTAÇÃO DE UM MODELO DE MELHORIA DE PROCESSO DE MANUTENÇÃO DE SOFTWARE

Novo Hamburgo, novembro de 2008.

EDISON LUCIANO DE MELLO

PROPOSTA DE IMPLANTAÇÃO DE UM MODELO DE MELHORIA DE PROCESSO DE MANUTENÇÃO DE SOFTWARE

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso

Professor Orientador: Ms. Eduardo Pretz

Novo Hamburgo, novembro de 2008.

RESUMO

Com o crescimento acelerado da tecnologia, o software tem se tornado uma ferramenta indispensável para praticamente todas as áreas da sociedade. Essa grande demanda tem feito com que os softwares aumentem de tamanho e complexidade, fazendo com que o processo de manutenção dos mesmos se torne uma tarefa muito desafiadora e complicada. A manutenção de software tem como principal objetivo contribuir para o melhoramento e evolução do mesmo, bem como aumentar o seu ciclo de vida. As melhorias nos processos têm sido criadas e implantadas com o intuito de resolver estes problemas que em sua grande maioria são conseqüência da não existência ou má definição de algum processo. Assim, este trabalho tem como objetivo analisar o atual processo de manutenção de software de uma empresa e através de estudos de conceitos de melhores práticas de manutenção de software, propor um novo processo através de melhorias contínuas. Espera-se com isso, gerar um produto final de melhor qualidade, diminuir custos com a manutenção e conseqüentemente garantir a maior satisfação do cliente.

Palavras-chave: Manutenção de Software, Melhoria de Processo de Software.

ABSTRACT

With the accelerated growth of technology, software has become an indispensable tool to practically all the areas of society. This great demand has made software to increase their size and complexity, turning their maintenance process into a very complicated and challenging task. Software maintenance aims, mainly, to contribute to their improvement and evolution, as well as increase their life cycle. Improvements on processes have been created and deployed to solve the problems which are, in their great majority, the consequences of non existence or bad definition of some processes. Thus, this work aims to analyze the current software maintenance process of a company and, through studies of the concepts of better software maintenance practices, propose a new process through continuous improvement. It is expected, this way, to generate a final product of better quality, to lower costs with maintenance and consequently, to guarantee a greater satisfaction of costumers.

Key words: Software Maintenance, Improvement on the Software Process.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MR	Requisição de Mudança
PR	Relatório de Problemas
JTC	Joint Technical Committee
SM	Solicitação de Mudança
CMMI	Capability Maturity Model Integration - Integração dos Modelos de Maturidade da Capacidade
SLA	Service Level Agreements
DP	Documentação de Projeto
SM^{mm}	Software Maintenance Maturity Model – Modelo de Maturidade de Manutenção de Software
ERP	Enterprise Resource Planning
SEI	Software Engineering Institute – Instituto de Engenharia de Software

SUMÁRIO

INTRODUÇÃO

A manutenção de software é hoje o principal desafio das empresas que dependem das receitas do desenvolvimento de software, devido ao elevado esforço envolvido para sua realização. Hoje em um mercado globalizado e muito competitivo com clientes cada vez mais exigentes, os produtos e serviços oferecidos necessitam de qualidade e principalmente de um preço atrativo. Diante disto, as organizações lutam por melhores resultados do seu investimento em desenvolvimento, mantendo o software operacional durante o maior período possível, aumentando conseqüentemente a fase de manutenção.

A manutenção é necessária para assegurar que o sistema continue a satisfazer as necessidades dos utilizadores (IEEE, 1998). Contudo, é uma atividade muito dispendiosa, onde pode ser responsável por mais de 70% de todo o esforço despendido por uma organização de software. (PRESSMAN, Roger S., 2007, p876). Esse custo é aceitável, visto que abrange a maior parte do ciclo de vida do software, existindo sempre o objetivo de diminuí-lo, melhorando o processo utilizado para a manutenção de software.

Segundo Pigoski (1997), o maior custo ocorre nos três primeiros anos de vida do software, sendo os gastos com recursos humanos os mais significativos nesse período. Além do custo monetário, que é o mais óbvio, existem outros menos tangíveis que também estão associados à atividade de manutenção. Um desses custos é a perda ou o adiamento de oportunidades de desenvolvimento, causados pelo fato dos recursos disponíveis para essa atividade serem canalizados para tarefas de manutenção. Outros custos desta natureza incluem a insatisfação do cliente, quando pedidos não podem ser atendidos em tempo hábil, a redução da qualidade global do sistema em virtude da introdução de erros causados pelas alterações e, uma grande diminuição na produtividade dos programadores.

Por esse e outros motivos, as melhorias para o Processo de Manutenção de Software vem ganhando espaço e cada vez mais estudiosos da área se dedicam em propor melhorias para essa fase do desenvolvimento de software. Entre as principais normas internacionais que abordam processos de manutenção de software, podemos destacar: ISO/IEC 12207,

ISO/14764-1998, que é uma evolução do processo de manutenção da ISO/IEC 12207, e podemos ainda citar o SM^{mm} *(Software Maintanance Maturity Model),* que é análogo ao CMMI *(Capabilitty Maturity Model Integration).*

A proposta deste trabalho é conhecer as melhores práticas para os processos de manutenção de software e avaliar o processo atual de uma empresa de desenvolvimento de software, a fim de propor melhoria na capacidade dos processos dos mesmos, providenciando assim um ponto de partida para um processo de manutenção ideal.

A partir disso, este trabalho está organizado em 3 capítulos teóricos. O capítulo 1 define Manutenção de Software, estando dividido em 5 subtítulos, a saber: definição de manutenção de software, definição de manutenibilidade, tipos de manutenção de software, problemas identificados na manutenção de software e o contexto da manutenção de software. O capítulo 2 descreve alguns modelos de qualidade voltados para a Manutenção de Software, estando dividido em 3 subtítulos, a saber: A origem dos principais modelos da ISO/IEC, a ISO/IEC 14764 e o Modelo de Taute. O capítulo 3 descreve o histórico da empresa analisada e seu processo atual de manutenção de software.

Tal conhecimento irá fundamentar a parte prática, permitindo desenvolver uma proposta de melhoria nos processos de manutenção de software na empresa descrita.

1 MANUTENÇÃO DE SOFTWARE

Este capítulo consistirá na revisão bibliográfica de manutenção de software, descrevendo seu contexto geral, definições, tipos e principais problemas encontrados.

1.1 Definição de Manutenção de Software

Grande parte dos sistemas sofre alterações ao longo do tempo em que estão sendo utilizados. Seus requisitos originais são modificados para refletir algum tipo de mudança nas necessidades do usuário, mudanças fiscais, correção de erros e outros. Este processo de mudança do software após sua implantação é chamado de Manutenção de Software. Podemos dizer também que a manutenção de software é um processo para garantir a sobrevivência do mesmo.

Dentro dos conceitos de manutenção de software, SOMMERVILLE, 2003 define:

> "É o processo geral de modificação de um sistema depois que ele foi colocado em operação. As modificações podem ser simples, destinadas a corrigir erros de código, mais extensas, a fim de corrigir os erros de projetos, ou significativas, com a finalidade de corrigir erros de especificação".

Segundo Penny, et al., 2003, Manutenção de Software é definido como:

> "A modificação de um produto de software após entrega, para corrigir falhas, para melhorar a performance ou outros atributos, ou para adaptar o produto de software a um ambiente modificado".

A ISO/IEC 12207, 1995, p17 define:

> "A modificação do código e da documentação associada devido a um problema ou a necessidade de melhoria. O objetivo é o de modificar o produto de software existente enquanto preserva a sua integridade".

A manutenção de software foi caracterizada por vários autores, como um "iceberg". Ansiosamente esperamos que aquilo que é imediatamente visível esteja lá de verdade. Realisticamente, sabemos que uma massa enorme de problemas e custos potenciais esconde-se sob a superfície. No horizonte podemos pré ver uma organização de software "baseada em

manutenção” que não mais pode produzir novo software, porque está gastando todos os seus recursos disponíveis mantendo um software antigo. (PRESSMAN, Roger S., 2007, p876).

A manutenção de software deve ser encarada como uma totalidade de atividades para que se mantenha e evolua um produto de software, controlando prioridades, melhorias, treinamentos, suportabilidade e principalmente controlando custos.

1.2 Definição de Manutenibilidade

Podemos dizer que a manutenibilidade é sem dúvida uma área de grande interesse da Engenharia de software. Segundo GLASS (2003), se os softwares forem manuteníveis, diminui a demanda por desenvolvimento na medida que os softwares atuais evoluem atendendo as novas necessidades.

Manutenibilidade de software diz respeito à facilidade com o que o mesmo pode ser modificado para satisfazer requisitos do usuário ou ser corrigido quando deficiências são detectadas (PIGOSKI, 1996).

Segundo Pressman, a Manutenibilidade pode ser definida qualitativamente como a facilidade com que o software pode ser entendido, corrigido, adaptado e/ou aumentado. A manutenibilidade é a meta primordial que orienta os passos de um processo de engenharia de software.

O IEEE (1993), estabelece que manutenibilidade é a facilidade com que um sistema de software ou componente pode ser modificado para corrigir falhas, melhorar performance ou outros atributos, ou adaptado para uma mudança de ambiente.

Todos os autores concordam que a manutenibilidade, é a facilidade de alterar um determinado software. Então, podemos concluir que um bom processo de manutenção de software, começa na faze de desenvolvimento do produto. Nesse momento é muito importante que já se pense na manutenção, desenvolvendo códigos que facilitam depois a sua manutenção. Devemos pensar que as horas a mais gastas com o desenvolvimento, resulta sem dúvida alguma, em redução de custos no futuro na fase de manutenção.

1.3 Tipos de Manutenção de Software

Lientz & Swanson inicialmente identificaram três categorias de manutenção: Corretiva, Adaptável e Perfectiva. Pesquisas realizadas por Lientz & Swanson (1980) e Nosek e Palvia (1990) sugerem que aproximadamente 65% das manutenções são relacionnadas à

implementação de novos requisitos, 18% a mudanças do sistema para adaptá-lo a um novo ambiente e 17% para corrigir defeitos do sistema. Já Pigoski (1996), sugere que 55% das manutenções são relacionadas a perfectiva, 20% a corretiva e 25% a adaptativa. As figura 1.1 e 1.2 ilustram melhor essa distribuição de esforços.

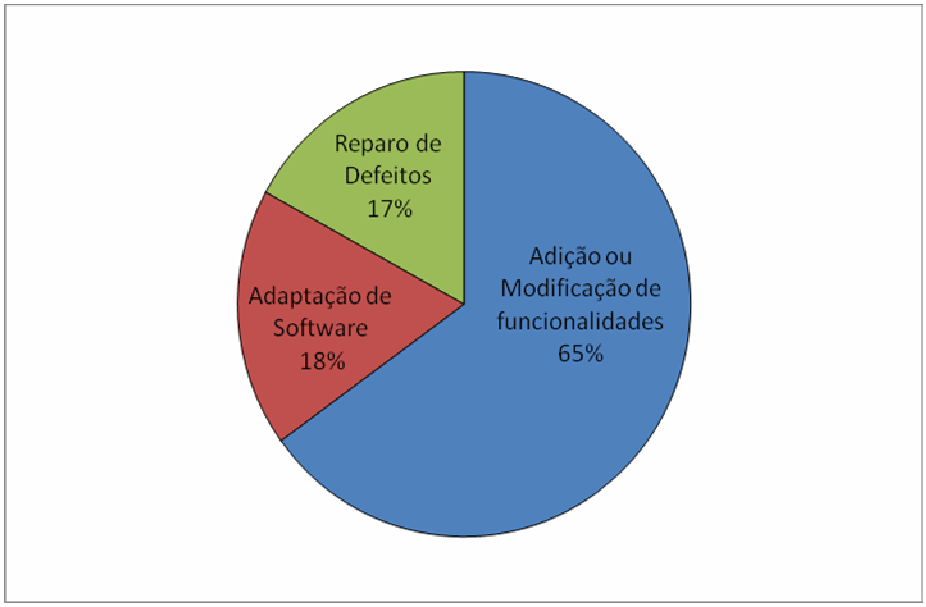

Figura 1.1 – Distribuição de esforços de manutenção
Fonte: Lientz e Sawanson (1980) e Nosek e Palvia (1990)

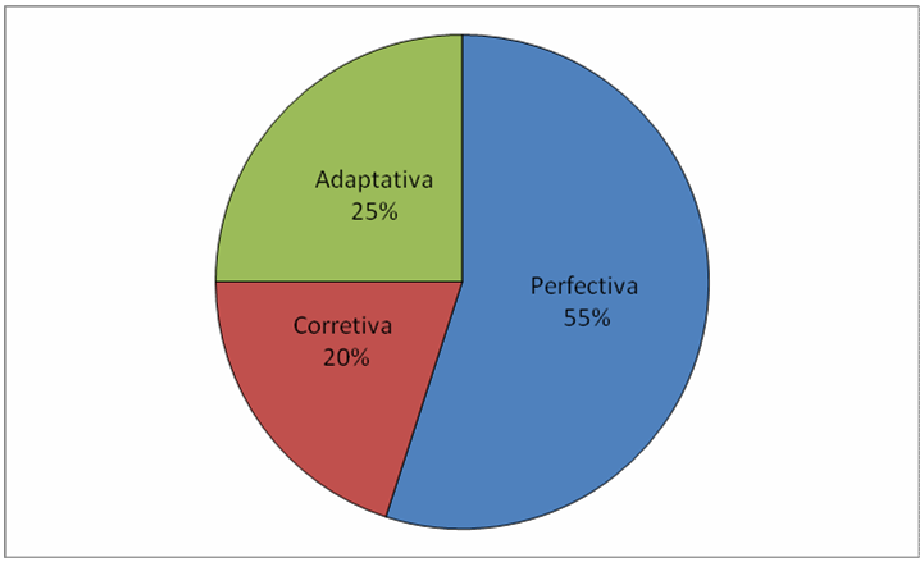

Figura 1.2 – Distribuição de esforços de manutenção
Fonte: Pigoski, 1996)

Percebe-se que ambos os autores concordam que o maior esforço da manutenção está relacionado com a manutenção perfectiva, ou seja, mudanças são realizadas para melhorar alguns aspectos do sistema.

Estes valores foram atualizados, e a ISO14764 definiu as seguintes categorias para a Manutenção de Software: (Manutenção Corretiva, Manutenção Adaptativa, Manutenção Perfectiva, Manutenção Preventiva). A figura 1.3, mostra como está dividida essa estrutura.

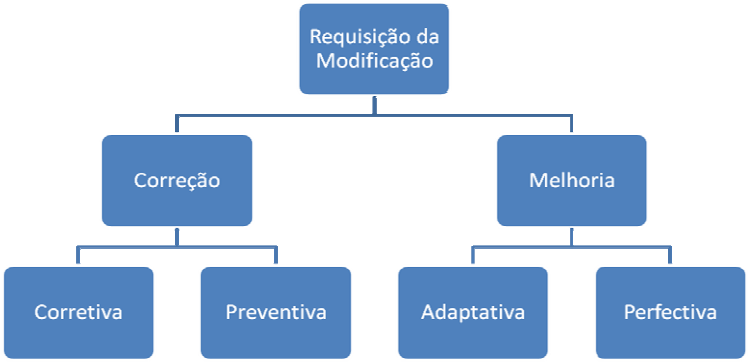

Figura 1.3 – Tipos de Manutenção
Fonte: ISO 14764 (1998).

1.3.1 Manutenção Corretiva

É a modificação de um produto de software, realizada após a entrega para corrigir falhas descobertas. (ISO14764, 1998).

À medida que as falhas ocorrem, elas são relatadas à equipe de manutenção, que encontra a causa da falha e faz as devidas correções e mudanças nos requisitos, projetos, no código, no conjunto de testes do sistema e na documentação conforme for necessário. (Pfleeger, 2006. p386).

1.3.2 Manutenção Adaptativa

É a modificação de um produto de software, realizada após a entrega para se adaptar a um ambiente variável. (ISO14764, 1998).

1.3.3 Manutenção Perfectiva

É a modificação de um produto de software, realizada após a entrega para melhorar o desempenho ou sustentabilidade. (ISO14764, 1998).

1.3.4 Manutenção Preventiva

É a modificação de um produto de software, realizada após a entrega para detectar e corrigir falhas antes que se tornem falhas eficazes. (ISO14764, 1998).

Já Sommerville, [SOMMERVILLE, 2007, p326] sugere somente as três primeiras, ou seja, une a terceira e a quarta classificação em uma única, pois classifica como evolutiva a manutenção que aperfeiçoa o software do sistema por meio de implementação de novas funcionalidades, mas em outras situações pode significar manter a funcionalidade do sistema, melhorando sua estrutura e seu desempenho.

1.4 Problemas identificados na manutenção de software

A manutenção de software é sem dúvida uma atividade à qual sempre se atribui uma série de problemas. (SOMMERVILLE, 2003). Em geral, a manutenção de um sistema é uma tarefa bastante complexa. Como o sistema já está em funcionamento, a equipe de manutenção precisa equilibrar a necessidade de modificação com a necessidade de mantê-lo acessível aos usuários com todas as funcionalidades existentes. Esse trabalho muitas vezes leva tempo, principalmente devido ao pouco ou nenhum envolvimento da equipe de manutenção no desenvolvimento do produto de software. Em muitos casos a equipe de manutenção só passa a conhecer o software após sua implantação. (PFLEEGER, 2004).

Uma série de problemas foi percebido por Pressman (PRESSMAN, 2007) na prática de manutenção. Dentre os principais problemas ele citou:

- Há falta de métodos e ferramentas eficazes para auxiliarem em como determinar uma manutenção afetará os sistemas;
- A maioria dos sistemas quando projetados não levam em consideração o fator manutenibilidade em seus desenhos;
- Novas funcionalidades criadas no software durante a fase de manutenção, na maioria das vezes não estão inteiramente integradas à arquitetura do software;
- Falta de recursos com experiência em manutenção;
- Em geral novos requisitos são adicionados ao invés de requisitos já existentes serem reavaliados;
- A documentação não existe ou é muito ruim.

- Muitas vezes é extremamente difícil de entender o programa de “outra pessoa”;

Segundo Pressman, os problemas encontrados podem de certa forma, ser atribuídos ao grande número de programas atualmente existente que foi desenvolvido sem levar em consideração a engenharia de software. Uma metodologia disciplinada não deve ser vista como um obstáculo, mas sim, como uma forma elegante de resolver os problemas.

Pfleeger (2004) também identificou alguns problemas na manutenção e classificou-os basicamente em quatro grupos como podemos ver a seguir:

- **Problemas com pessoal**: o pessoal deve agir como um intermediário entre o problema e a sua solução, corrigindo e ajustando o software, a fim de garantir que a solução siga o curso do problema à medida que ele se modifica. Entre os principais problemas com os recursos humanos podemos citar:
  - **Entendimento limitado:** o entendimento do usuário também apresenta problemas. Por exemplo, se os usuários não entendem como o sistema funciona, eles podem fornecer dados incompletos ou errados quando relatarem os efeitos de um problema aos mantenedores.
  - **Prioridades de Gerenciamento*:*** os gerentes consideram a manutenção e o aprimoramento como mais importante do que desenvolver novas aplicações. Em outras palavras, enquanto os gerentes estimulam os responsáveis pela manutenção a reparar um sistema antigo, os usuários pedem novas funções ou, até mesmo, um novo sistema. Desta forma, a pressa em disponibilizar um novo produto, pode acarretar uma solução ineficiente e de difícil manutenção.
  - **Ânimo*:*** Falta de ânimo da equipe de manutenção. Um dos principais motivos para o pouco ânimo é o status de trabalho “de segunda classe”, freqüentemente relacionado com a equipe de manutenção.
- **Problemas técnicos:** os problemas técnicos também afetam a produtividade da manutenção. Algumas vezes, eles são conseqüência do que os desenvolvedores e mantenedores fizeram anteriormente. Outras vezes, eles resultam de paradigmas ou processos específicos, adotados na manutenção, conforme é listado abaixo:
  - **Recursos e paradigmas*:*** projetos inconsistentes ou inflexíveis podem exigir tempo extra para que seja entendido, modificado e testado.

- **Dificuldade para a realização dos testes:** os testes podem ser um problema quando não se dispõe do tempo necessário para a sua realização.

- **A necessidade de conciliação:** muitas vezes os princípios de engenharia de software competem com as questões de conveniência e de custo. Freqüentemente, um problema pode ser resolvido conforme uma de duas maneiras: uma solução rápida mais inapropriada, que funciona mas não se adapta ao projeto do sistema ou à estratégia de codificação; ou uma solução mais demorada e elegante, consistente com os princípios utilizados para gerar o restante do sistema. Em muitos casos os programadores e analistas da equipe de manutenção, são obrigados a concentrar seus recursos em uma solução rápida e inadequada, ocasionando na maioria dos casos problemas posteriores e conseqüentemente mais manutenção.

- **Custo da manutenção:** Descobrir qual a real quantidade de recursos que deve ser destinada a uma manutenção, que tipo de erro deve ser corrigido imediatamente e que tipo deve ser analisado de forma a ser aproveitado para uma melhoria, além de incorporar uma manutenção ao software com o mínimo de risco possível, tem sido questões citadas por Pfleeger [PFLEEGER,2004] e que tem preocupado muito vários especialistas.

Além dos problemas citados acima pelos respectivos autores, podemos enumerar outros tantos fatores, que dificultam o trabalho de manutenção de software extraídos da prática. Muitos deles advêm do processo de desenvolvimento, outros ocorrem durante o próprio processo de manutenção. Dentre muitos, podemos citar:

- **Falta de conhecimento do sistema**: A equipe que dá manutenção ao software, por muitas vezes não conhece bem o domínio da aplicação. É comum que os profissionais que trabalham em manutenção de software não tenham a oportunidade de participar do desenvolvimento dos sistemas pelos quais serão futuramente responsáveis. Tal fato não geraria maiores problemas, se as documentações vindas do processo de desenvolvimento fossem bastante completas e de fácil utilização, e que, aliado a isso, fosse dado a estes profissionais algum tipo de treinamento sobre os novos sistemas que estão sendo implantados. Estas iniciativas poderiam amenizar muito dos problemas

encontrados atualmente na manutenção de software e de algum modo garantir a qualidade do software durante toda sua vida útil

- **Alterações podem causar novos erros**: Algumas mudanças, quando introduzidas num software já existente, podem inserir novos erros. Segundo Pfleeger (2004), a uma tendência muito grande de novos erros após uma determinada manutenção, principalmente na programação orientada a objetos. As técnicas orientadas a objetos podem tornar os programas mais difíceis de serem entendidos, devido à profusão de partes de programas.
- **Degradação das estruturas**: é muito comum alterações sem que haja uma análise de todo o software em si. Muitas vezes criam-se funcionalidades que não se interligam de forma adequada ao software.
- **Falta de documentação**: a falta de uma documentação adequada direcionada para a manutenção de software, é outro problema muito sério. Grande parte das documentações geradas durante a fase de projetos e de desenvolvimento são arquivadas após sua implantação. Somente alguns desses documentos são realmente úteis para a fase de manutenção de software.
- **Falta de métricas**: há dificuldade em obtenção de métricas para gerenciamento e aumento de visibilidade das manutenções.
- **Custo real desconhecido**: O custo do processo de manutenção de software continua sendo uma grande preocupação para as organizações. Descobrir o custo efetivo de uma manutenção e identificar seu benefício ainda é uma tarefa bastante complexa. Hoje muitas empresas se preocupam apenas em estimar e controlar horas, desconsiderando totalmente qual o real valor gasto para uma determinada manutenção.
- **Falta de treinamento**: falta de treinamento continuado para a equipe de manutenção. Este treinamento de contemplar além dos já habituais cursos de linguagem e técnicas de programação e análise, a obtenção de conhecimento sobre os sistemas já existentes.
- **Falta de um modelo de qualidade**: outro problema muito comum, é a falta de um modelo de qualidade para o processo de manutenção de software, principalmente em empresas orientadas a produto.

Este e outros problemas associados à manutenção de software, fazem com que o software vá degradando sua qualidade durante o período de manutenção e conseqüentemente, tornando-se cada vez mais difícil e custoso de ser mantido, levando-o à sua descontinuação mais rápido do que o previsto.

1.5 Contexto geral da Manutenção de Software

É muito comum empresas de desenvolvimento de software, não terem um processo bem definido para a manutenção de software. Contudo, é importante salientar que para criar um processo de manutenção, é necessário entender as atividades referentes à manutenção e principalmente o contexto que o mantenedor opera no seu dia a dia. A figura 1.4 resume as interfaces existentes na manutenção de software num contexto organizacional típico. (APRIL, et al., 2005).

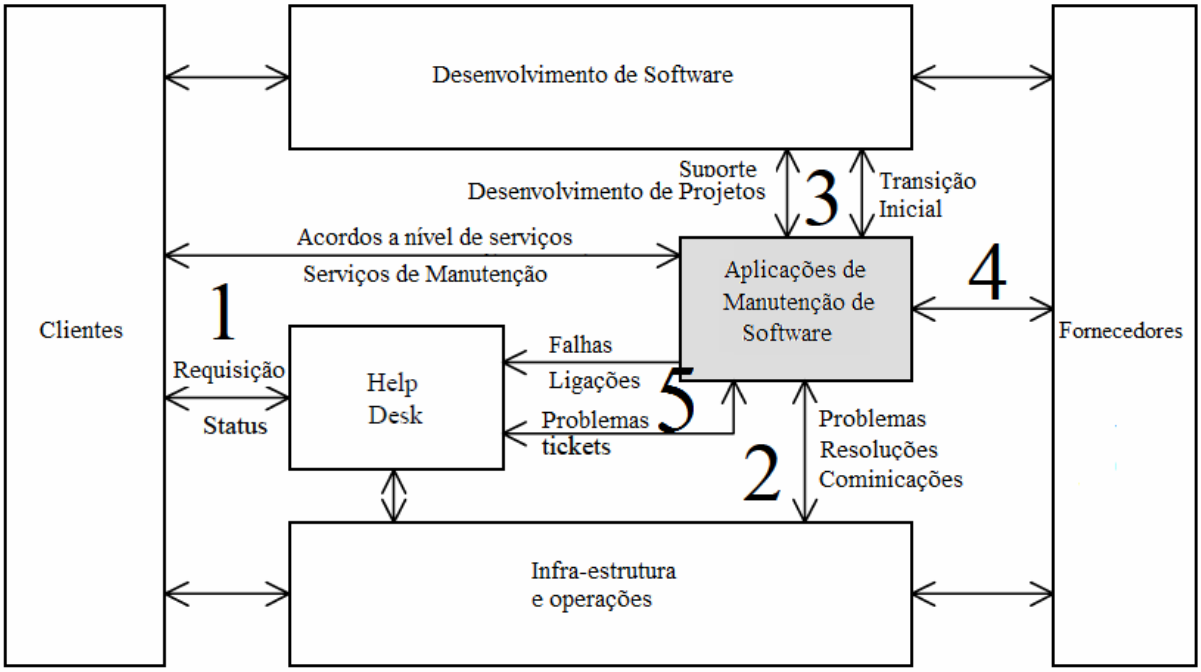

Figura 1.4 – Contexto de Manutenção de Software
Fonte: (April, et al., 2005).

As interfaces existentes com a Manutenção de Software estão divididas da seguinte forma:

- Interface 1: clientes e utilizadores de manutenção de software.
- Interface 2: departamento de infra-estrutura e operações;

- Interface 3: programadores e desenvolvedores;
- Interface 4: fornecedores;
- Interface 5: manutenção de primeira linha e help-desk;

Analisando estas interfaces, vimos que requerem serviços diários, onde a gestão da manutenção precisa manter as aplicações de modo a correrem suavemente, a reagir rapidamente de modo a restaurar o serviço, a corresponder ou exceder o nível de serviço acordado e ainda manter os usuários confiante que existe uma equipe competente e dedicada à sua disposição que atua dentro do contexto de manutenção. (APRIL, et al., 2005).

Segundo April, Cada interface designada tem um objetivo, conforme estão descritos abaixo:

A primeira interface é de fundamental importância, pois trabalha com os clientes e usuários. É nesta fase que consiste a negociação e discussão de: pedidos individuais de prioridades, acordos de níveis de serviço (SLA), planejamento, orçamentos, serviço ao cliente e atividades relacionadas com a satisfação dos clientes e usuários. A interface com os usuários é chave fundamental para a manutenção, onde esses usuários deverão ser freqüentemente envolvidos em comunicações diárias que requerem:

- Resposta operacional rápida a relatórios de problemas;
- Responsabilidade para investigação para uma regra de negócio específica, tela ou relatório;
- Relatório de progresso em um grande número de pedidos de modificações;

A segunda interface trabalha basicamente com a comunicação com o *Help-Desk*, com a infra-estrutura e com a organização das operações. Nesta fase, há uma troca muito grande de informações entre os programadores de software e as operações, onde o usuário é raramente comunicado ou envolvido. Esta interface também inclui atividades menos freqüentes, como a coordenação e recuperação de serviços após falhas ou desastres, com o objetivo de restaurar o acesso a serviços acordados nos termos e condições dos SLA.

A terceira interface está localizada entre o desenvolvimento e a manutenção, e é tipicamente iniciada durante o desenvolvimento de um novo software. Sabemos que a causa inicial de vários problemas de manutenção pode estar no processo de desenvolvimento, e é reconhecido que os programadores necessitam estar envolvidos ou exercer alguma forma de controle durante a pré-entrega e a transição do desenvolvimento para a manutenção.

(PIGOSKI, 1997). Diante disso, esta interface reforça a idéia que os mantenedores devem ajudar e apoiar projetos de desenvolvimento, ajudando desta forma a criar um sistema com alto nível de manutenibilidade.

É nesta interface de desenvolvimento-manutenção, que deve ocorrer as contribuições feitas pelos programadores de manutenção que de modo concorrente dão suporte, e por vezes estão envolvidos também em grande número de projetos em desenvolvimento. O conhecimento do programador da manutenção é de grande valor para os desenvolvedores que precisam criar um sistema novo ou alterar a interface em um software obsoleto.

A quarta interface mostra a relação com um número crescente de fornecedores, fornecedores de *outsourcing*, e vendedores de *Enterprise Resource Planning* (ERP) (April, et al., 2005). Os programadores têm diferentes tipos de relações com os fornecedores. Por exemplo: a) com fornecedores que desenvolvem um novo software ou configuração de um ERP; b) com sub-contratações para fazer parte do time de manutenção, para ajudar com perícias específicas.

Para assegurar um serviço de qualidade aos usuários, os programadores têm de desenvolver alguma compreensão dos variados tipos de contratos e geri-los de forma eficiente de modo a assegurar a performance do fornecedor, que normalmente tem impacto nos resultados dos SLA (Service Level Agreements).

A última interface pode ser representada de diversas formas, de acordo com as diferentes estruturas organizacionais. O *help-desk* é fundado, por vezes, como parte da organização de manutenção, ou parte das operações da organização e pode ser localizado num produto independente do suporte à organização.

Na figura 1.4, observa-se que os utilizadores podem passar a primeira linha de suporte e reportam diretamente ao pessoal da manutenção. Para ser eficaz, é usado um processo mecânico de resolução de problemas que assegura a comunicação para uma rápida resolução de falhas. Um pedido específico de um determinado usuário pode ser chamado de *ticket*, e tipicamente circula entre o *help-desk*, a manutenção e o nível operacional que pode a isolar o problema.

2 MODELOS DE QUALIDADE DE MANUTENÇÃO DE SOFTWARE

Este capítulo descreve um pouco da história dos primeiros modelos de manutenção de software que surgiram, bem como detalha o modelo da ISO/IEC 14764 e o modelo de Taute.

2.1 A origem dos principais modelos da ISO/IEC

Os processos de manutenção preocupam-se em investigar as questões relativas a quem, o que, onde, quando ocorre uma solicitação de mudança. O padrão provisório IEEE 1219-1992 descreve um processo interativo para gestão e execução das atividades de manutenção de software. Esse padrão identifica sete principais atividades inerentes a um processo de manutenção de software, que é disparado por uma Solicitação de Mudança (SM). A estrutura básica desse modelo de processo é apresentada na Figura 1.5. O modelo associa os mecanismos de entrada, saída e controle relativos a cada fase do processo de manutenção. Esse modelo não pressupõe nenhum modelo de processo de software específico. As SMs, a Documentação de Projeto (DP), o código-fonte, os bancos de dados e repositórios são fontes de entrada. (PETERS, James F. e PEDRYES Witold, 2001).

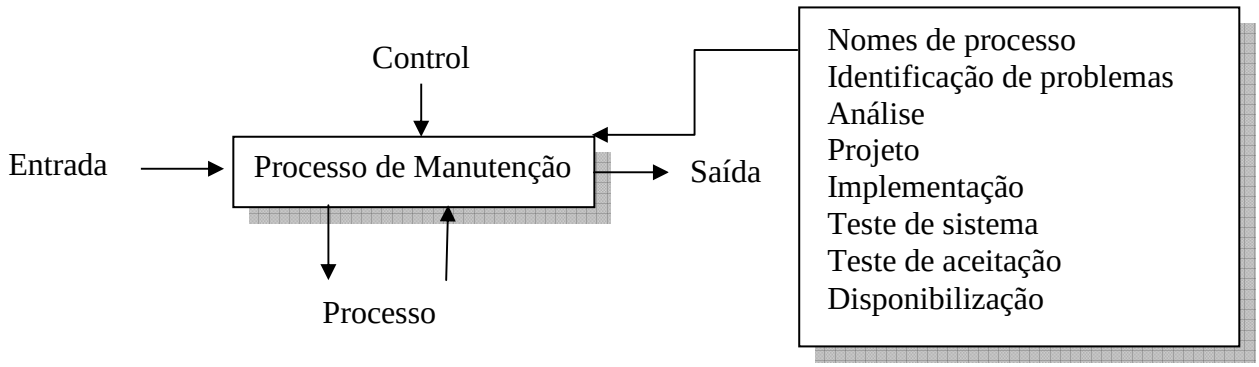

Figura 1.5 – Ciclo de Manutenção de Software IEEE 1219
Fonte: IEEE 1219

Em 1987 com uma parceria entre a ISO e a IEC, foi criado um Comitê Técnico Conjunto – JTC (Joint Technical Committee) sobre tecnologia da informação. O principal objetivo desse comitê era estabelecer um padrão para os processos do ciclo de vida do software, que culminou com a norma ISO/IEC 12207.

Assim, a ISO/IEC 12207 teve sua primeira publicação em 01 de agosto de 1995, estando desde então em constante evolução.

Esta norma estabelece uma arquitetura de alto nível prevendo um conjunto de processos de engenharia de software que uma organização deve utilizar para adquirir, fornecer, desenvolver ou manter software, ou seja, ela documenta os processos do ciclo de vida de software em modelo de referência de processos. Ela descreve a arquitetura dos processos do ciclo de vida do software, mas não especifica os detalhes de como implementar ou executar as atividades e tarefas incluídas nos processos.

Os processos são agrupados, por uma questão de organização, de acordo com sua natureza, ou seja, seu objetivo principal no ciclo de vida de software. A tabela 1.1 ilustra bem essa estrutura.

Tabela 1.1: Processos do ciclo de vida do software segundo a ISO/IEC 12207:

Grupos de Processos	Processos
Processos fundamentais	Aquisição
	Fornecimento
	Desenvolvimento
	Operação
	Manutenção
Processos de Apoio	Documentação
	Gerência de configuração
	Garantia da qualidade
	Verificação
	Validação
	Revisão conjunta
	Gerência de mudança
	Auditoria
	Resolução de Problema

Cont.

Processos Organizacionais	Gerência
	Melhoria
	Infra-estrutura
	Reutilização
	Riscos
	Recursos Humanos

Fonte: (NBR ISO/IEC 12207, 1998, p. 6)

Analisando a tabela acima, percebe-se que o agrupamento resultou em três diferentes classes de processo que são:

- Processos fundamentais;
- Processos de apoio;
- Processos organizacionais;

O processo de manutenção desse modelo é localizado dentro dos processos fundamentais, mais precisamente no capítulo 5.5, relatando todos os processos do mantenedor. Este processo é ativado quando o produto de software é submetido a modificações no código e na documentação associada devido a um problema, ou à necessidade de melhoria ou adaptação

É importante salientar que esta norma não representa um modelo fixo ao qual uma organização que adote se submete. Ao contrário disso, ela funciona como uma estrutura de apoio, devendo a organização que adotar proceder com adaptações nas recomendações para sua realidade.

2.2 Modelo ISO/IEC 14764

Esta seção apresenta a norma ISO/IEC 14764 – Ciclo de vida de Manutenção de Software. É a mais atual norma que descreve os processos de manutenção de software. Este modelo é uma explosão do processo de manutenção da ISO/IEC 12207. Contém em detalhes as atividades e tarefas necessárias para modificar um produto de software existente preservando sua integridade.

A figura 1.6 ilustra melhor a comparação entre as normas ISO/IEC 12207 e 14764.

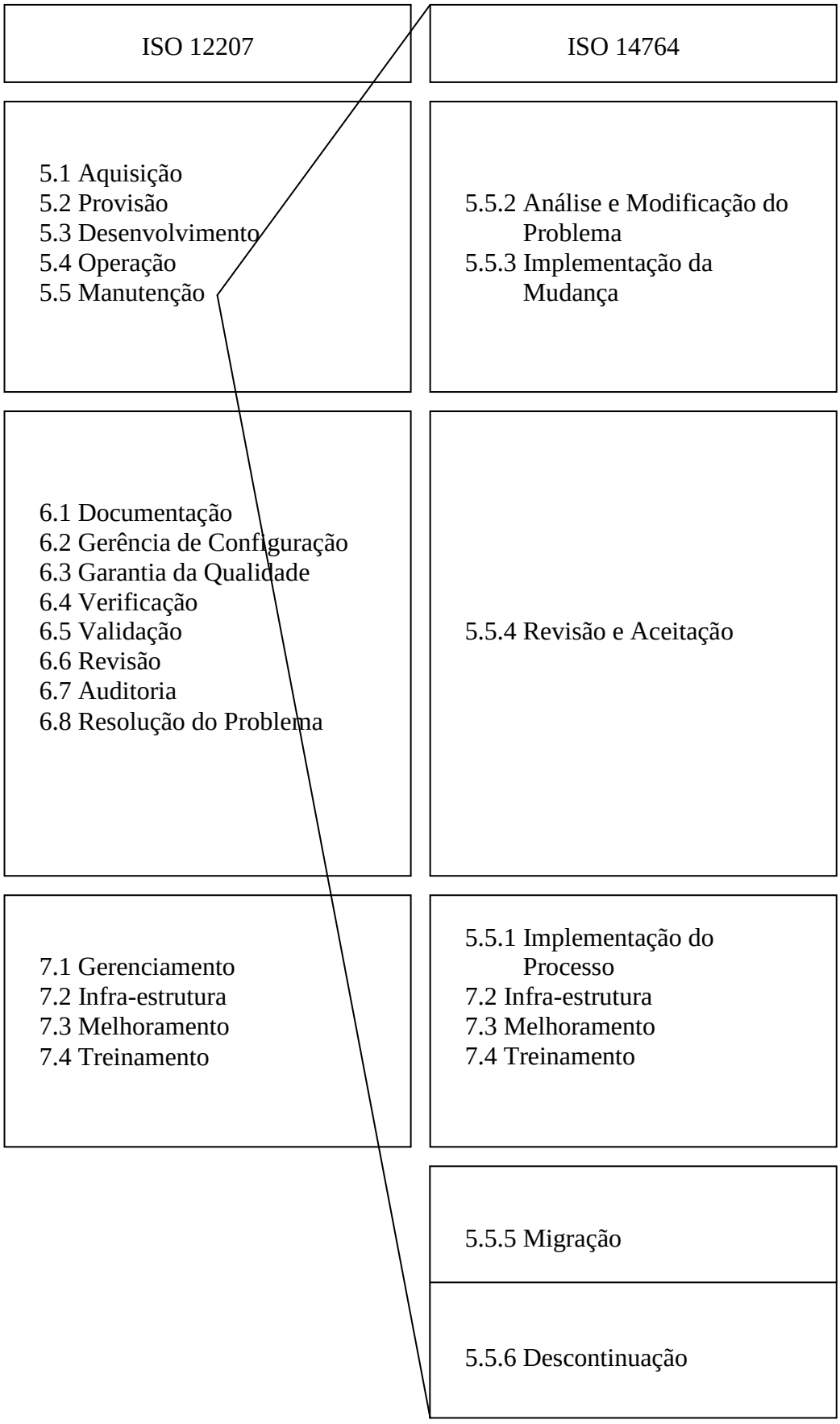

Figura 1.6 – Processo do ciclo de vida de um software
Fonte: (April, et al., 2005).

A lista de atividades da ISO/IEC 14764 é a mesma descrita na ISO/IEC 12207, porém com mais detalhes. São eles:

- Implementação do processo;
- Análise do Problema e modificação
- Implementação da modificação;
- Revisão/aceitação da manutenção;
- Migração;
- Descontinuação;

Na figura 1.7 é ilustrado como é o clico dessas atividades.

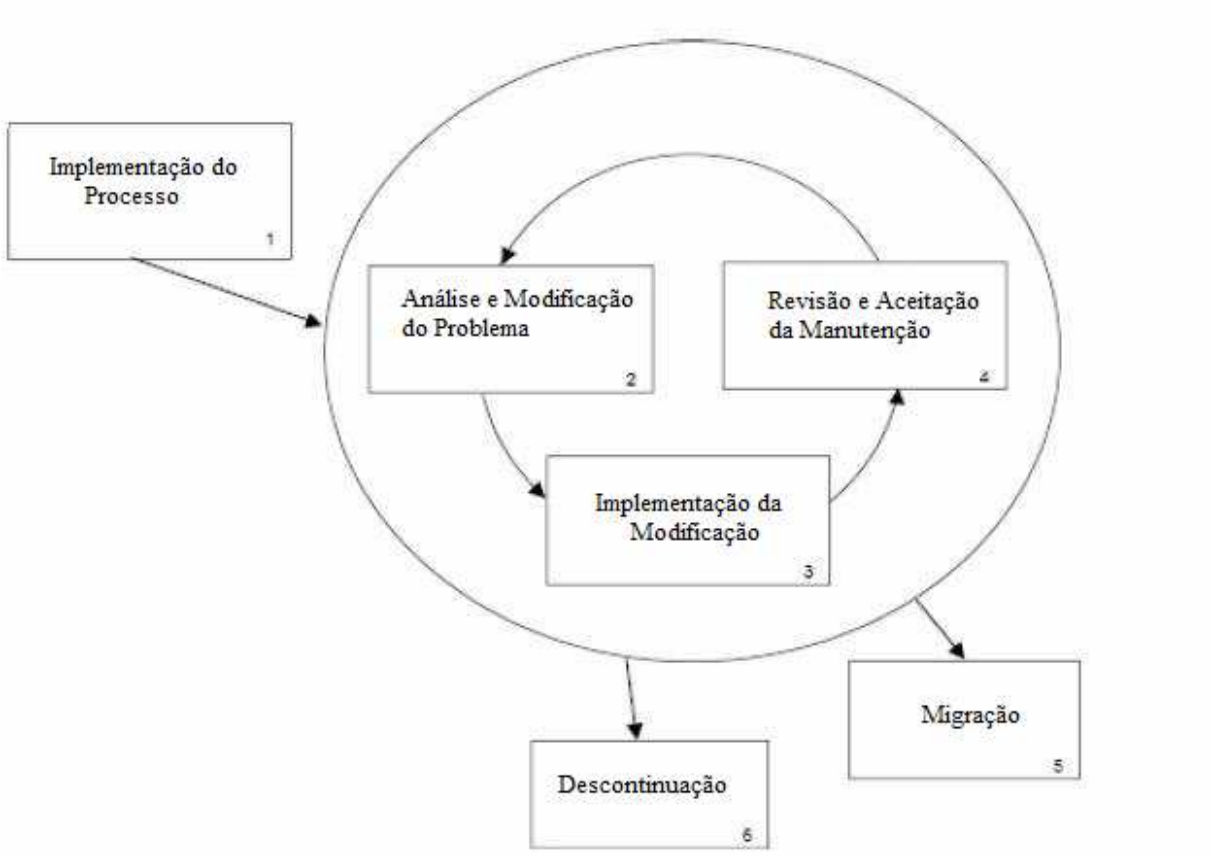

Figura 1.7 – Ciclo de Manutenção de Software da ISO 14764
Fonte: ISO/IEC 14764 (1998)

De acordo com essa norma, estas tarefas e atividades são de responsabilidade do mantenedor de software. Cabe ao mantenedor certificar-se de que o processo de manutenção de software existente é funcional antes de iniciar-se qualquer desenvolvimento de software.

Após a entrega do produto de software, o mantenedor deve modificar o código e a documentação associada em resposta às requisições de modificação ou relatórios de problemas. Esse processo oferece suporte ao produto desde sua concepção, sua migração para novos ambientes e até o software ser descontinuado. Deve ser ativado quando existir um requisito para alterar um produto de software. Assim que o processo for ativado, planos e procedimentos devem ser desenvolvidos e recursos devem ser alocados especificamente para a manutenção. O processo é encerrado quando o produto for finalmente descontinuado ou migrado. Nas seções seguintes, serão descritas as atividades que compões este processo.

2.2.1 Implementação do processo

Durante essa atividade, o mantenedor estabelece os planos e procedimentos que são executados durante o processo de manutenção.

São relacionadas como necessárias para as atividades de implementação as seguintes entradas:

- Baselines relevantes;
- A documentação do sistema;
- Requisição de Mudança / Relatório de Problemas (MR/PR);

Para desenvolver e documentar a estratégia que será usada para conduzir o processo de manutenção, as seguintes tarefas devem ser executadas:

- Desenvolver os planos e procedimentos de manutenção;
- Estabelecer procedimentos para tratar as Requisição de Mudança / Relatório de Problemas (MR/PR);
- Implementar o processo de gerência de configuração.

O plano de manutenção deve ser desenvolvido em paralelo com o plano de desenvolvimento. Deve-se estabelecer procedimentos para receber, armazenar, acompanhar os pedidos de modificação e os relatórios de problemas dos usuários e fornecer um retorno para os mesmos. Além disto, uma interface organizacional com o processo de gerência de configuração também deve ser feita para as modificações realizadas no sistema ou produto de software possam ser gerenciadas. São geradas as seguintes saídas dessa atividade:

- Plano de manutenção;

- Plano de treinamento;
- Procedimentos de manutenção;
- Gerenciamento do plano de projeto;
- Procedimentos para resolução do problema;
- Planejamento de medidas;
- Manual da manutenção;
- Planos para retorno do usuário;
- Plano de transição;
- Plano de Gerência de configuração;
- Plano de avaliação da manutenibilidade;

2.2.2 Análise do problema e da modificação

Antes de modificar o produto o mantenedor deve analisar a Analisar as Requisição de Mudança / Relatório de Problemas (MR/PR), para avaliar o impacto que a manutenção terá no sistema existente, na organização e nos sistemas que os quais o produto possui interfaces. Nesta análise o mantenedor também deve desenvolver e documentar as possíveis soluções e obter a aprovação necessária para iniciar a implementação da solução escolhida. Esta análise deve considerar os seguintes aspectos:

- Tipo: Deve verificar se a manutenção é corretiva, adaptativa, perfectiva, ou preventiva;
- Escopo: Deve verificar o tamanho, os custos envolvidos e o esforço necessário para efetuar a manutenção;
- Criticidade: Deve-se verificar a importância da manutenção e seu impacto em aspectos como segurança e performance do sistema

Devem ser realizadas os seguintes tarefas:

- Analisar as Requisição de Mudança / Relatório de Problemas (MR/PR);
- Verificar o problema;
- Desenvolver opções para implementar a solução;

- Documentar a análise das Requisição de Mudança / Relatório de Problemas (MR/PR), os resultados e as opções de implementação;
- Obter aprovação para a manutenção;

São necessários para esta atividade as seguintes entradas:

- Baseline atual;
- Requisição de Mudança / Relatório de Problemas (MR/PR);
- O repositório do software;
- A documentação do sistema;

Sendo que a documentação do sistema inclui:

- Informações sobre a configuração;
- Requisitos funcionais;
- Requisitos de interface;
- Dados sobre o planejamento do projeto;
- As saídas geradas pela atividade de implementação do processo:

São geradas as seguintes saídas desta atividade:

- Análise de impacto;
- Opções recomendadas;
- Modificações aprovadas;
- Documentação atualizada

Sendo a análise de impacto deve incluir as seguintes informações:

- Definição do problema ou dos novos requisitos;
- Avaliação do problema ou dos requisitos;
- Classificação da manutenção necessária;
- Prioridades;
- Dados para a verificação em caso de manutenção corretiva;

- Estimativa inicial dos recursos necessários para realizar a alteração do sistema existente.

Ao final dessa atividade uma análise de riscos deve ser realizada. Usando as saídas dessa atividade as estimativas iniciais devem ser reavaliadas e uma decisão deve ser tomada sobre prosseguir ou não para a atividade de implementação da modificação.

2.2.3 Implementação da Modificação

Durante a atividade de implementação da modificação o mantenedor deve desenvolver e testar a modificação do produto de software, que foi especificada e aprovada na atividade de análise do problema e da modificação.

Para essa atividade é utilizada as seguintes entradas:

- A baseline;
- A Requisição de Mudança / Relatório de Problemas (MR/PR);
- A documentação atualizada;

A baseline deve incluir;

- Definições da arquitetura do sistema;
- Os registros das requisições de modificação;
- O código fonte;

O mantenedor deverá analisar a modificação solicitada e determinar quais unidades de software, programas e documentos devem ser alterados e a seguir evocar o mesmo processo de desenvolvimento definido na norma ISO/IEC 12207. Devem ser realizadas as mesmas atividades e empregadas as mesmas técnicas de engenharia de software que são usadas em projetos de desenvolvimento de software para a atividade de manutenção de software sempre que necessário.

São geradas as seguintes saídas para essa atividade:

- Planos e procedimentos de teste realizado;
- Modificação do código fonte;
- Relatório de teste;
- Métricas;

- Documentação atualizada

Sendo que a documentação deve incluir:

- Registros de modificação atualizados;
- Relatório detalhado de análise;
- Requisitos atualizados;
- Material de treinamento atualizado;
- Planos de testes, procedimentos de testes e relatórios de testes atualizados.

2.2.4 Revisão e Aceitação da Modificação

Esta atividade visa garantir se as modificações estão corretas e que foram realizadas em conformidade com os padrões estabelecidos usando todas as metodologias de forma adequada. É muito importante que o mantenedor conduza a revisão com quem autorizou a modificação para averiguar a integridade do sistema modificado. Em seguida o mantenedor deve obter a aprovação formal de que a modificação foi realizada como especificado, com ou sem contrato.

As entradas necessárias para esta atividade são as seguintes:

- O software modificado;
- Resultados dos testes da modificação;

São geradas as seguintes saídas para essa atividade:

- Nova Baseline incorporando as modificações realizadas;
- Modificações rejeitadas;
- Relatório de aprovação;
- Relatório de auditoria e revisão;
- Relatório de testes para qualificação do produto;

2.2.5 Migração

Durante a vida de um software , modificações podem ser necessárias para que um software rode em um determinado ambiente. Para realizar esta migração o mantenedor deverá determinar, desenvolver e documentar todas as ações necessárias.

São necessárias as seguintes entradas para essa atividade:

- O ambiente antigo;
- O novo ambiente;
- A baseline antiga;
- A nova baseline;

Para realizar a migração em conformidade com a norma ISO/IEC 12007, o mantenedor deve planejar a migração, avaliar o impacto, notificar os usuários a respeito da migração, fornecer-lhes treinamento e notificá-los do término da migração, além de arquivar toas as informações a respeito da migração.

São geradas as seguintes saídas com essa atividade:

- Plano de migração;
- Ferramentas de migração;
- Notificações dos usuários;
- Produto de software migrado;
- Métricas;
- Dados arquivados;

2.2.6 Descontinuação de Software

Uma vez que um produto de software atinge o final de sua vida útil, ele deve ser descontinuado. Uma análise deve ser realizada para apoiar a decisão de descontinuar o software. Esta análise é freqüentemente econômica e pode ser incluída no plano de descontinuidade. O software pode ser substituído por um novo software, mas em alguns casos não. Mesmo assim, para descontinuá-lo o mantenedor deve identificar, desenvolver e documentar as ações necessárias.

São necessárias as seguintes entradas para esta atividade:

- O produto de software antigo que será descontinuado;
- O novo produto de software;
- O ambiente antigo;

Para descontinuar um produto de software em conformidade com essa norma, o mantenedor deve desenvolver um plano de descontinuidade, notificar os usuários a respeito da descontinuidade, fornecer-lhes treinamento e notificá-los do término da descontinuidade, além de arquivos todos os dados referente a mesma.

2.3 Modelo de Taute

Vários modelos de manutenção foram desenvolvidos desde a década de 1970. os primeiros modelos tinham a manutenção corretiva como ponto principal. Um exemplo de modelo corretivo é fornecido em Sharpley (1977). O modelo de Sharpley concentrava-se na verificação de problemas, no diagnóstico de problemas, na reprogramação e em verificar se o código corrigido satisfazia os requisitos do documento baseline. Os modelos de processo preponderam no pensamento de manutenção de software recente. Um exemplo é o modelo de manutenção de software de Taute. (PETERS, James F. e PEDRYES Witold. 2001).

O modelo de manutenção de Taute, conforme a Figura 1.6 é direto, prático, e de fácil compreensão. Foi introduzido por Taute (1983) e elaborado por Parikh (1986). O modelo possui diversas características especiais. Em primeiro lugar, é específico para manutenção, muito simples de se implantar e muito recomendado para empresas de pequeno porte. Em outras palavras, ele não é projetado para fazer parte das outras fases do ciclo de vida de um software. Em segundo lugar, a manutenção de software é vista como cíclica. (PETERS, James F. e PEDRYES Witold, 2001).

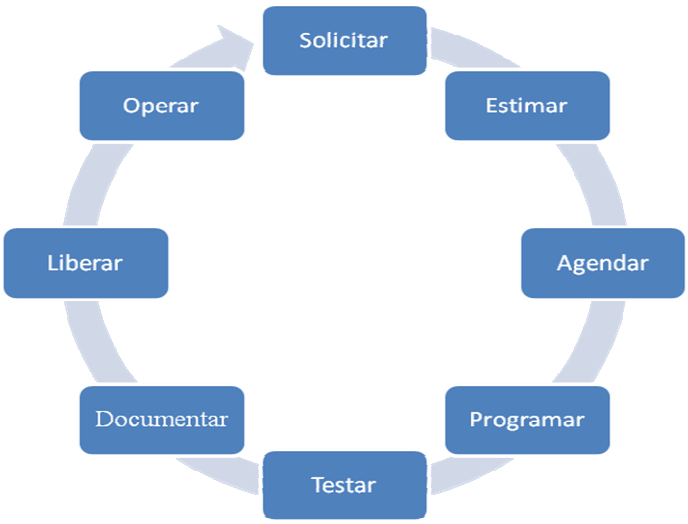

Figura 1.8 – Ciclo de Manutenção de Software de Taute
Fonte: Autor da Monografia

O ciclo inicia-se com uma solicitação de mudança, e termina com a operação bem-sucedida de um produto de software modificado. Finalmente, esse modelo destaca-se dos demais por enfatizar as versões programadas dos produtos de software resultantes das solicitações de mudanças. Essa característica do modelo de Taute é bastante atraente, pois é excelente para a gestão da manutenção de software. (PETERS, James F. e PEDRYES Witold, 2001).

O modelo de Taute possui oito fases:

- ***Fase de Solicitação***: Cada vez que uma solicitação de mudança é submetida, as técnicas de gestão de configuração são utilizadas para processá-la . Seu tipo de manutenção é identificado através da utilização do esquema de Swanson (correção, adaptação e aperfeiçoamento) recebe um número de identificação exclusivo, sendo em seguida encaminhado para um gerenciador de solicitação de mudanças, para que seja armazenado e monitorado.
- ***Fase de estimativa***: Essencialmente o mesmo que a fase de análise na gestão de configuração. O tempo, custos, recursos e o impacto de uma solicitação de mudança são determinados.
- ***Fase de agendamento***: São determinadas as solicitações de mudanças para o lançamento da próxima versão de um produto de software. Nessa fase, os documentos de planejamento são preparados.
- ***Fase de programação***: Uma cópia da versão de produção do software a ser modificado é liberada, e começa a criação de uma versão de teste de uma nova versão.
- ***Fase de testes***: Uma cópia recém-modificada do software de produção é testada.
- ***Fase de documentação***: Depois da conclusão bem-sucedida dos testes de um novo produto, vem a preparação da documentação do usuário e do sistema. A documentação existente é atualizada.
- ***Fase de lançamento***: O novo sistema (versão beta) e sua documentação atualizada são disponibilizados. Os usuários começam os testes de aceitação do software.
- ***Fase operacional***: A disponibilização e a operação da nova versão começa quando os testes de aceitação são bem-sucedidos e o programa liberado.

3 A EMPRESA ANALISADA

Este capítulo descreve um pouco da história da empresa analisada, bem como seu atual processo de manutenção de software.

3.1 Histórico da Empresa

Fundada em 1981 e atualmente localizada em Montenegro, no RS, onde concentram-se as atividades de Desenvolvimento de Sistemas, Marketing, Vendas, Suporte Técnico, Administração Geral, Finanças e Recursos Humanos.

A primeira versão do produto para o segmento de Consórcio foi concluída em 1983. Em 1987, a empresa foi homologada pela Ford Brasil S/A para fornecer seu Sistema de Gestão de Consórcio às Administradoras ligadas aos Distribuidores Ford. A experiência adquirida nesse mercado possibilitou que a empresa conquistasse a preferência das Administradoras nacionais e ingressasse no mercado internacional conquistando clientes que atuam na Argentina, Colômbia, Equador, México e Peru.

No segmento de Concessionárias de Veículos, a primeira versão do produto foi concluída em 1986. Desde então, a empresa vem ampliando mercado e obtendo homologações, como nos casos dos projetos DATADIF (rede Ford e Autolatina, 1989 e 1990), projeto SINCRO (rede Volkswagen, 1991) e projeto de informatização das concessionárias GM (1995).

Nesses 27 anos de atuação, a empresa investiu na inovação tecnológica dos produtos e no constante aperfeiçoamento dos seus serviços e processos internos, sempre com o intuito de prestar um atendimento diferenciado aos seus clientes.

A empresa já possui a certificação ISO 9001:2000 e em 2006, com objetivo de melhorar seus processos de desenvolvimento, iniciou o projeto CMMI (*Capabilitty Maturity Model Integration*), buscando o nível 2 de maturidade. Como o modelo CMMI não contempla processos de manutenção de software, a empresa está em busca de um modelo que possa

preencher as lacunas no processo de manutenção deixadas pelo modelo CMMI, integrando tanto as melhores práticas bem como as normas e conceitos da empresa, providenciando assim um ponto de partida para um processo de manutenção ideal.

3.2 Processo de Manutenção atual da empresa

O processo de manutenção da empresa é um processo já bem definido, mas que não seguiu nenhum modelo em especial. Ele foi desenvolvido de acordo com a necessidade e experiência que a empresa foi adquirindo do decorrer dos anos de sua atuação no mercado.

A figura 1.7 mostra um fluxograma do processo de manutenção atual da empresa.

Fluxograma do Procedimento de Manutenção

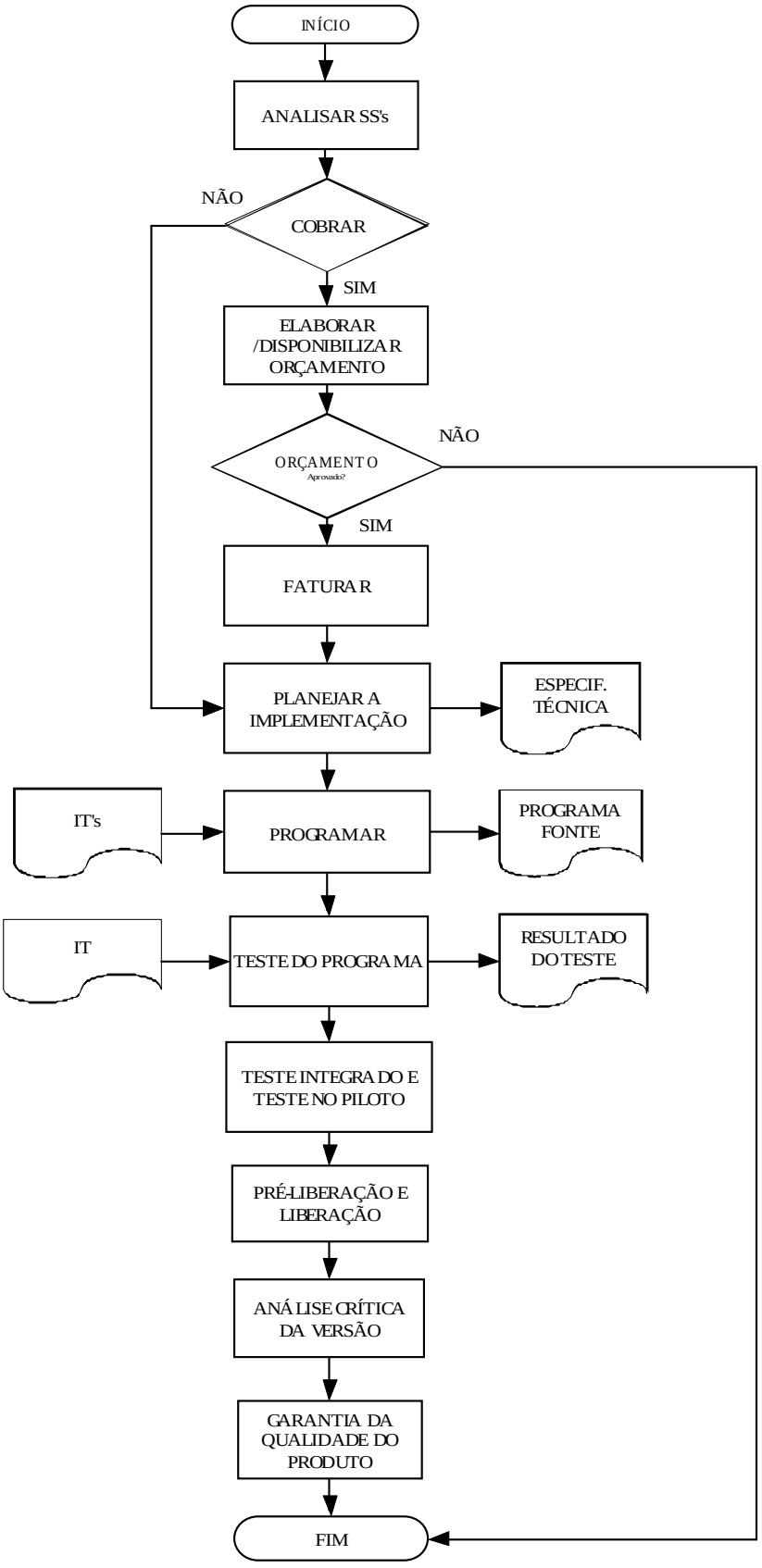

Figura 1.9 – Fluxograma de Manutenção de Software da Empresa analisada
Fonte: Auto da monografia

3.2.1 Análises das Solicitações

A análise das solicitações começa pelo analista de negócio, que verifica todos os pedidos, exceto pedidos de correção que são encaminhados diretamente ao setor de manutenção. É de responsabilidade dele, fazer a análise da regra de negócio e somente depois encaminhar para o setor de manutenção.

É de responsabilidade do coordenador da manutenção selecionar e analisar todas as solicitações que são encaminhadas ao setor de manutenção pela equipe de atendimento. É ele quem reclassifica caso necessário, agenda prazos e colocada a solicitação na fila para ser atendida.

3.2.2 Elaboração de Orçamento

No caso de solicitações específicas de um determinado cliente, no qual a mesma será cobrada, é o coordenador de manutenção quem elabora o orçamento para o atendimento da solicitação, estimando o número de horas previstas conforme e o valor da hora técnica que será cobrada do cliente. Determina uma validade para o orçamento e o encaminha para o cliente.

3.2.2.1 Aprovação do Orçamento

A aprovação do orçamento deve ser recebida por documento devidamente assinado pelo cliente e enviado por fax ou correspondência. No caso da não aprovação ou expiração da data de validade, a solicitação do cliente é encerrada. Se o orçamento for aprovado pelo cliente solicitante, o coordenador da manutenção colocada a solicitação na fila para ser atendido.

3.2.2.2 Faturamento do orçamento

O setor financeiro da empresa recebe a aprovação do orçamento e providencia o faturamento do mesmo ao cliente.

3.2.3 Planejamento da implementação

O planejamento da implementação será realizado obrigatoriamente por dois analistas, onde um fará a especificação técnica e a implementação, e o outro será responsável pela aprovação da análise e especificação técnica realizada pelo outro analista.

3.2.3.1 Analista responsável pela especificação técnica

Identifica as solicitações a serem atendidas que estão disponíveis nas filas, elabora a especificação técnica e o procedimento de teste, identificando o que será feito e a abrangência do projeto. Define as horas previstas conforme planilha de estimativas para o atendimento da solicitação e encaminha para o analista responsável pelo módulo.

3.2.3.2 Analista responsável pelo módulo

A empresa criou a figura do analista especialista responsável por determinados módulos do sistema. Por exemplo, especialista no módulo contábil, financeiro, estoque, oficina, comunicação, etc. Desta forma, é esse analista que é o responsável pela análise de impacto dessas solicitações.

Identifica as solicitações que estão encaminhadas para ele, faz a análise crítica da especificação técnica e do procedimento de teste e, se necessário faz os devidos ajustes. Versiona a especificação técnica e encaminha a solicitação ao analista programador.

3.2.4 Programação

O analista responsável pela programação identifica as solicitações a serem atendidas que estão encaminhadas para ele e realiza as alterações conforme a especificação técnica aprovada. Versiona o programa e atualiza o repositório. Encaminha a solicitação para o setor da qualidade de software para ser realizado os devidos testes.

3.2.5 Realização dos testes do programa

É de responsabilidade do coordenador da qualidade do software, verificar as solicitações encaminhadas para testes e as devidas prioridades e encaminhar a solicitação para um analista responsável pelo teste.

3.2.5.1 Analista responsável pelo teste

Verifica as solicitações encaminhadas para o seu usuário, e realiza os testes objetivando identificar falhas no programa. Caso encontre falhas, devolve a solicitação ao setor de manutenção. Registra os resultados encontrados, e encaminha a solicitação para o documentador.

3.2.5.2 Documentador

Identifica as solicitações encaminhadas para o seu usuário e se existe documentos anexados. Revisa todos os documentos existentes e caso necessário faz as devidas correções. Quando a alteração envolve alguma modificação de manual, também providencia as devidas

alterações. É de responsabilidade também do documentador, a geração do “Leia-me” da versão que será liberada. Este documento é gerado em todas as versões que são liberadas, sendo que contém nele, uma breve explicação de tudo o que foi alterado no sistema..

3.2.6 Teste integrado

Depois de todas as solicitações que serão liberadas terem sido testadas individualmente, é realizado pela qualidade de software teste integrado de todos os pontos críticos do sistema. Esse teste tem como principal objetivo, garantir o funcionamento de tudo o que já estava executando no cliente. São registrados todos os resultados, e os problemas são reportados ao setor de manutenção para as devidas correções.

3.2.7 Teste acompanhado do cliente Piloto

Após a conclusão do teste integrado, o gerente do desenvolvimento juntamente com gerente de serviço, define um cliente que será o Piloto da nova versão. Esse cliente ficará em teste com a nova versão durante o período de uma semana, acompanhado de um analista de atendimento da empresa, e todos os problemas encontrados são reportados diretamente ao setor da qualidade, que por sua vez, encaminha para o setor de manutenção para as devidas correções.

3.2.8 Pré-liberação e Liberação

Após a conclusão de todos os testes, o gerente de desenvolvimento juntamente com o coordenador do suporte nacional, define alguns clientes levando em consideração porte, localização e urgência, para receberem a versão antes da liberação oficial.

3.2.8.1 Análise crítica das melhorias da versão

Antes da liberação oficial da versão, são agendas reuniões com toda a equipe para realizar a análise crítica (por módulo), com o objetivo de validar as melhorias e atendimentos realizados na versão que será distribuída. Se identificados problemas, os mesmos são registrados em Ata, onde as ações a serem tomadas são definidas, com prazo e responsável. Depois disso, a versão é liberada para todos os clientes.

3.2.8.2 Garantia da qualidade do processo.

Após a liberação da versão, é realizada uma avaliação amostral de 10% de todas as solicitações atendidas, para analisar os respectivos códigos fontes das alterações se estão de acordo com o Guia de Desenvolvimento da empresa.

CONCLUSÃO

A partir do estudo realizado, pode-se afirmar que a manutenção de software é uma tarefa bastante complexa, onde a não existência de um processo de manutenção eficiente, gera custos elevados, um produto de software com baixa qualidade e principalmente clientes insatisfeitos. Diante disso, surgem os modelos de qualidade de software, com a finalidade de padronizar e profissionalizar o processo de manutenção de software.

Nesse sentido, a primeira parte desse trabalho, foi realizado um estudo teórico que aborda o contexto geral da manutenção de software, bem como os conceitos, tipos e os principais problemas. Com esse estudo foi possível identificar vários processos e os resultados que se precisa obter para atender o modelo de manutenção satisfatoriamente.

Dando continuidade será realizado o mapeamento dos atuais problemas no atual processo da empresa analisada, levantando-se as partes que necessitam serem melhoradas, e assim propor melhorias no processo de manutenção existente, aderentes às melhores práticas.

REFERÊNCIAS BIBLIOGRÁFICAS

GRUBB, Penny; TAKANG, Armstrong A. **Software Maintenance, Concepts and Practices – Second Edition.** New Jersey: World Scientific Publishing Co. Pte. Ltd., 2003. 350p.

LIENTZ, B. SWANSON, E. **Software Maintenance Management**, Reading, Mass., Addison Wesley, 1980. 214p.

PRESSMAN, Roger S. **Software Engineering: A Practitioner's Approach – third edition.** São Paulo: Pearson Makron Books, 2007. 1056p.

SOMMERVILLE, Ian. **Engenharia de Software** – 6a Edição, São Paulo: Addison Wesley, 2003. 592p.

WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTO, Celia Joseli do. **A Qualidade de software – Teoria e Prática.** São Paulo: Prentice Hall, 2001.

PIGOSKI, T.M. Practical software maintenance: Best practice for managing your software investment. New York: John Wiley & Sons, 1997. 384p.

SEI – Software Engineering Institute. Capability Maturity Model® Integration (CMMI), Version 1.2, Carnegie Mellon University: agosto de 2006. Disponível em: <http://www.sei.cmu.edu/cmmi>. Acesso em: 10 de Setembro de 2008.

APRIL, Alain A.; DUMKE, Reiner R.; ABRAN, Alain ***SM***^{mm} **Model to Evaluate and Improve the Quality of the Software Maintenance Process**. École de Technologie Supérieure, Montréal, Canada. Disponível em: <http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/paper/AprilPreprint.pdf>. Acesso em 01 de Setembro de 2008.

APRIL, Alain A.; DUMKE, Reiner R.; ABRAN, Alain ***SM***^{mm} **The Software Maintenance Process Model**. École de Technologie Supérieure, Montréal, Canada. Disponível em: <http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/paper/AprilHuffmanAbranDumkeJournal2005.pdf>. Acesso em 01 de Setembro de 2008.

ISO/IEC 12207, (1995). Tecnologia da Informação – Processos de ciclo de vida de Software, ISO/IEC: Outubro de 1998.

ISO/IEC 14764, (1998). Software Engineering-Software Maintenance, ISO and IEC.

IEEE Std 1219, (1998). Standard for Software Maintenance, IEEE Computer Society Press.

PRODANOV, Cleber Cristiano. **Manual da Metodologia Científica.** Novo Hamburgo: Feevale, 2003. 77p.

IEEE - Institute of Eletrical and Eletronics Engineers. **Recommended Practice for Software Requirements Specifications:** Std 830, N.Y.,1993. 37p. Disponível em <http://www.eee.metu.edu.tr/~bilgen/IEEE830.pdf>. Acesso em 06 set. 2007.

IEEE - Institute of Eletrical and Eletronics Engineers. **Standards Glossary of Software Engineering Terminology:** Std 610.12, N.Y.,1990. 84p.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software – Fundamentos, Métodos e Padrões.** Rio de Janeiro: LCT Editora, 2003. 602p.

SEI - SOFTWARE ENGINEERING INSTITUTE.**CMMI for Development version 1.2**. Pittsburg, PA: Carnegie Mello University, CMU/SEI, 2006. 573p. Disponível em <http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html> Acesso em 06 setembo. 2008.

SEI - SOFTWARE ENGINEERING INSTITUTE.**Standard MMI Appraisal Method for Process Improvement version 1.2: Method Definition Document** . Pittsburg, PA: Carnegie Mello University, CMU/SEI, 2006. 245p. Disponível em <http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html> Acesso em 06 setembo. 2008.

PFLEEGER, Shari Lawrence. **Engenharia de software:** teoria e prática. 2. ed. São Paulo, SP: Prentice Hall, 2004. 537 p.

PETERS, James F.; PEDRYCZ Witold. **Engenharia de software: Teoria e Prática.** Rio de Janeiro, RJ: Editora Campus Ltda. 2001. 602p.

GLASS, Robert L. Editor's Corner – Which do you think? Modern Methods Will Lead to Less Maintenance, or More? J Systems Software. 1993.