Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica

Mateus Maida Paduelli

Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica

Mateus Maida Paduelli

Orientadora: ***Profa. Dra. Rosely Sanches***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos necessários para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	21/05/2007
Visto do Orientador:	

USP – São Carlos
Junho/2007

Dedico este trabalho:

A meus pais, Sérgio e Diva, pelo
carinho e exemplo de vida,
honestidade e perseverança.

AGRADECIMENTOS

A Deus, em primeiro lugar, pela saúde, inteligência e por ter me colocado ao lado das pessoas certas para ajudarem-me neste trabalho.

Aos meus pais, Sérgio e Diva, exemplos de persistência, trabalho e honestidade, por todo o carinho e atenção que sempre me dedicaram. Eles representam o foco de minha inspiração conduzindo-me por todas as conquistas e realizações. Aos meus irmãos, verdadeiros amigos, por todo o apoio nos momentos em que necessitei. Aos meus avôs e avós pelo exemplo de vida e perseverança.

A Rosely Sanches, professora, orientadora e acima de tudo amiga pela confiança depositada em mim e por fazer-me acreditar em minha capacidade. Também pelas suas assistências, orientações e contribuições durante todas as atividades desta pesquisa.

E a todos que, de alguma forma, contribuíram para essa minha vitória.

Muito Obrigado!

"Não sei como o mundo me vê; mas eu me sinto como um garoto brincando na praia, contente em achar aqui e ali uma pedrinha mais lisa ou uma concha mais bonita, tendo sempre diante de mim, ainda por descobrir, o grande oceano da verdade."

Isaac Newton (1642-1727)

RESUMO

PADUELLI, M. M. **Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica**. 2007. 144 p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, 2007.

O volume crescente de software em funcionamento em todo tipo de organização vem despertando atenção para uma fase do ciclo de vida de software, até então considerada sempre de maneira secundária, a manutenção de software. O fato de geralmente não ser viável substituir os produtos de software de uma organização por outros baseados em tecnologias mais recentes, torna a manutenção daqueles sistemas legados um desafio adicional para a busca de técnicas e métodos para a manutenção de software. Os problemas oriundos dessa atividade precisam ser melhor compreendidos, e é justamente na definição e estudo dessas dificuldades que este trabalho se dedica. O confronto da teoria de engenharia de software com observações práticas conduz para a melhor definição de quais são os problemas típicos de manutenção de software e do que se dispõe para abordá-los. Finalmente, com base no entendimento formado sobre os problemas, neste trabalho são apresentadas diretrizes para guiar a elaboração de uma disciplina específica de manutenção de software para cursos de graduação na área de computação.

Palavras-chave: *Manutenção de Software, Problemas de Manutenção de Software, Ensino de Manutenção de Software.*

ABSTRACT

PADUELLI, M. M. **Software Maintenance: typical problems and guidelines for a specific discipline**. 2007. 144 p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, 2007.

The increasing volume of software being used in all types of organizations has been calling attention for a phase of the software life cycle, until now considered in a secondary way, the software maintenance. Since it is generally not possible to replace all software products used in an organization by others based on more recent technologies, the maintenance of those legacy systems becomes one more challenge for the search of techniques and methods to handle the software maintenance efficiently. The problems arising from this activity need to be better understood, and it is precisely on the definition and study of these difficulties that this work is devoted. The confrontation between the theory of software engineering and practice observations drives to the definition of typical problems of software maintenance and what exists to solve them. Besides, based on the understanding about these problems, this work also presents guidelines to drive the elaboration of a specific discipline of software maintenance for undergraduate courses in computing area.

Keywords: *Software Maintenance, Software Maintenance Problems, Software Maintenance Teaching*.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

CAPÍTULO

1

Introdução

1.1 Contexto e Motivação

O termo engenharia de software foi criado em uma conferência no final da década de 60 (Naur & Randell, 1968), sediada em Garmisch, Alemanha, que envolveu usuários, fabricantes e pesquisadores, interessados em tratar dos constantes problemas no desenvolvimento de software.

Os problemas enfatizados na época abrangiam atrasos na entrega, orçamentos irreais, falta de resposta correta aos anseios dos usuários e dificuldades diversas para criar, usar, manter e melhorar software. Desde aqueles primeiros momentos, a engenharia de software iniciou seu desenvolvimento, passando a criar e aperfeiçoar continuamente métodos, procedimentos e ferramentas para tornar a atividade de desenvolver e manter software uma tarefa que pudesse ser medida, controlada e avaliada. De uma forma simples, pode-se dizer que ao longo das últimas décadas houve um amadurecimento do conceito de software e de suas características e processos atrelados.

Todo esse desenvolvimento culmina na atualidade com o aumento da atenção destinada à atividade de manutenção de software. Essa importância decorre, em parte, da crescente quantidade de software em funcionamento nas organizações ao redor de todo globo, que por representarem investimentos significativos, precisam continuar em funcionamento através dos anos, e não serem substituídos, momento no qual surge a necessidade de

manutenção de software. Essa necessidade traz consigo problemas originados de diversas fontes, como a própria administração da organização, o perfil dos clientes ou as deficientes técnicas utilizadas na construção do software original.

Considerando-se toda a importância prática da atividade de manutenção de software, é esperado que as boas universidades preparem seus alunos para oferecerem mecanismos acadêmicos apoiados na prática, que facilitem a minimização dos efeitos negativos que a falta de controle e entendimento da atividade de manutenção pode trazer para os negócios.

Esse entendimento é evidenciado pelo fato de que a demanda por profissionais capazes de tratar eficientemente problemas oriundos da manutenção de software possui crescimento visível, uma vez que a cada ano mais softwares são conduzidos ao perfil de recursos indispensáveis e construídos com técnicas não mais recomendáveis.

Considerando que a engenharia de software é um campo de estudo em evolução e adaptação contínua, uma vez que seu objeto de trabalho (software) apresenta características dinâmicas, é de se esperar que a forma de replicar todo o conhecimento acumulado também passe por atualizações e adaptações. Esse ajuste natural visa formar novos pesquisadores ou preparar futuros profissionais, para atuarem de forma mais produtiva e eficiente nas organizações que utilizem software como atividade meio ou atividade fim.

Partindo desse pressuposto, este trabalho contribui tanto no aspecto de entender melhor as dificuldades de uma das mais importantes atividades dentro do ciclo de vida de software, a atividade de manutenção de software, como também oferece diretrizes para o ensino acadêmico dessa atividade. O enfoque em ensino justifica-se pela carência ainda existente na obtenção de informações que estabeleçam uma base sólida para que as universidades possam propor uma forma não puramente acadêmica de ensinar manutenção de software, mas sim uma maneira que ofereça também aos alunos uma visão mais ampla e clara de todos os fatores envolvidos nessa tarefa. Essa visão deverá estar ligada à realidade das organizações e aos seus problemas diários quando precisam manter software dentro de inúmeras limitações, como tempo, recursos e falta de treinamento. Essas considerações vão ao encontro da influência internacional da *Association for Computing Machinery* (ACM), que de acordo com o que será exposto no capítulo de *Características do Ensino de Manutenção de Software*, já considera uma disciplina específica de manutenção de software nos seus currículos de referência para cursos de computação.

Existem iniciativas em algumas universidades brasileiras no sentido de estabelecer uma disciplina específica para manutenção de software em cursos de graduação na área de computação. Essa postura mostra a conscientização crescente sobre a importância de ensinar

manutenção de software ainda no meio acadêmico, antes que os alunos se tornem profissionais.

No entanto, ainda não existe um consenso sobre o conteúdo que essa disciplina deve abordar. Alguns relatos descrevem os resultados com a implantação experimental de disciplinas de manutenção de software em cursos de graduação, e alguns até de pós-graduação, sendo que em todos os casos o conteúdo ministrado é baseado em experiências pessoais dos professores. A falta de dados que apresentem números atualizados sobre os problemas mais comuns é um dos motivos dessa incerteza sobre o conteúdo a ser abordado.

Partindo-se dessas considerações, fica mais claro entender a necessidade de pesquisar e relacionar mais informações sobre a atividade de manutenção dentro de organizações, de maneira a evidenciar quais são as principais dificuldades dessa atividade, para que possam então ser estabelecidos programas de ensino sustentados também pela prática e não apenas fechados dentro da realidade do meio acadêmico. Entender quais são os problemas é um passo muito importante para que seja possível estudar as formas de abordagem possíveis para tais problemas, ou para evitar que eles surjam, organizando por fim um conteúdo a ser ensinado que esteja adequado à realidade organizacional.

1.2 Objetivos

Os objetivos desta pesquisa, bem como os passos realizados para alcançá-los, podem ser visualizados de uma maneira global pelo esquema mostrado na figura 1.1.

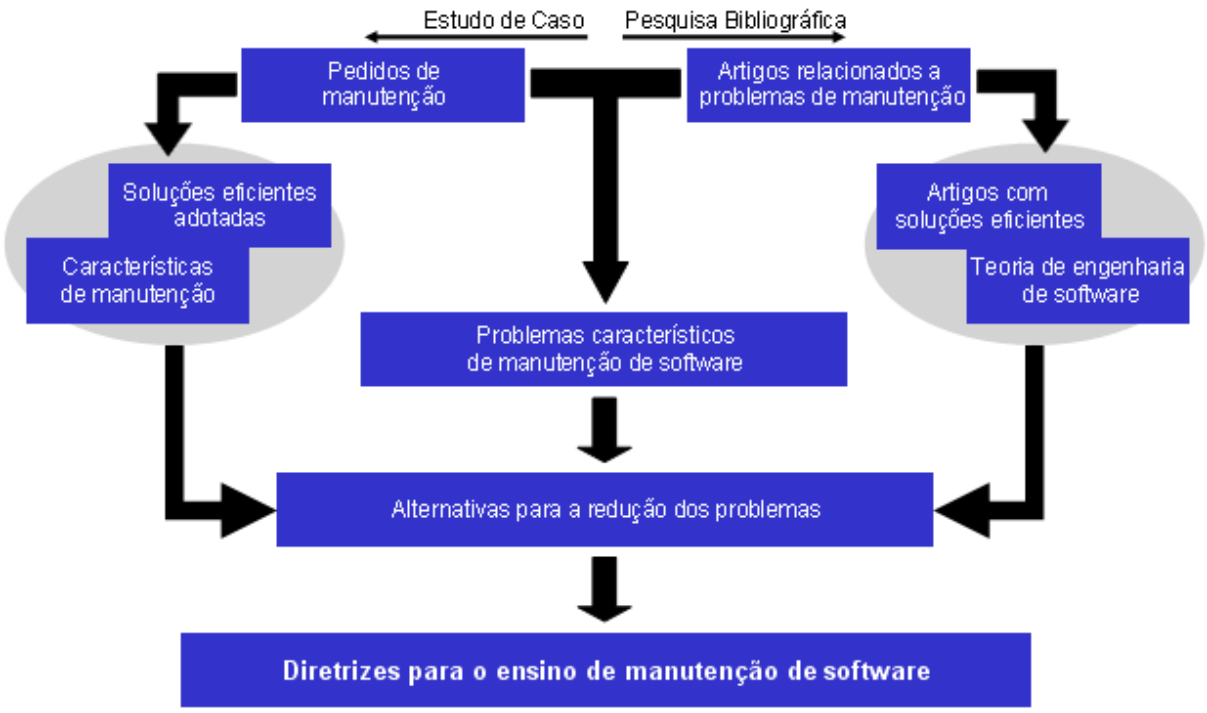

FIGURA 1.1: MODELO GERAL DO DESENVOLVIMENTO DESTE TRABALHO

Em um primeiro momento, por meio de um estudo de caso em uma organização do setor de software, foram levantados totais a respeito de pedidos de manutenção sobre diferentes características (data, tipo de manutenção, complexidade etc.), bem como sobre a maneira como a organização aborda a sua atividade de manutenção (fluxo de informações, sistemática de atualizações etc.), sendo possível inferir sobre os principais problemas afetos à abordagem que a organização utiliza para a atividade.

Conforme sugere a figura anterior, ao resultado das informações obtidas pelos *pedidos de manutenção* são somados os itens referidos em *artigos relacionados a problemas manutenção de software*, para finalmente estabelecer uma relação dos problemas típicos de manutenção.

No passo seguinte, após a elaboração da relação de problemas, são consideradas, por um lado, as *soluções eficientes adotadas* pela organização estudada para resolver ou reduzir os seus problemas de manutenção, sendo as *características de manutenção* a forma encontrada para oferecer respostas aos seus problemas. Por outro lado, são verificados *artigos com soluções eficientes* apontadas para problemas correlatos, finalmente considerando o que a *teoria de engenharia de software* disponibiliza em termos de recursos para investir contra os problemas de manutenção. Aqui, a abordagem da engenharia de software foi fortemente baseada na norma ISO/IEC 12207[1], conforme será apresentado nos capítulos seguintes. Essa fusão de informações, conforme mostram as setas na figura anterior, produz uma relação de *alternativas para a redução dos problemas* de manutenção de software.

Finalmente, sabendo-se quais são os problemas, e o que pode ser feito para auxiliar seu controle ou prevenção, são propostas *diretrizes para o ensino de manutenção de software*. Essas diretrizes levam a uma proposta de assuntos considerados essenciais a uma disciplina de manutenção de software, não sendo, no entanto, uma especificação final de ementa.

Essa maneira de propor os temas essenciais a uma disciplina, por meio da verificação dos problemas práticos existentes, está de acordo com a idéia de proposta pedagógica *orientada a problemas*, conforme será discutido no item 7.2.

Pode-se resumir o propósito deste trabalho como uma contribuição ao estudo de problemas de manutenção de software e também à sistematização do ensino de manutenção de software, por meio de um levantamento de dados reais e seu relacionamento com boas práticas em engenharia de software, de forma a estabelecer uma base no qual o conteúdo da disciplina possa se apoiar.

[1] International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC).

1.3 Organização do Trabalho

Esta dissertação está dividida em oito capítulos. No presente capítulo foi apresentado o contexto em que se insere o trabalho, as motivações para o seu desenvolvimento e os seus objetivos. No *capítulo 2* é apresentada a revisão bibliográfica relativa à manutenção de software, expondo os conceitos importantes que serão utilizados ao longo do texto. No *capítulo 3* é descrito o estudo de caso realizado, incluindo a metodologia empregada, o levantamento de dados e os resultados obtidos.

No *capítulo 4* é feita uma comparação entre os problemas de manutenção de software do passado com os atualmente verificados no estudo de caso, além do estabelecimento de um rol de problemas típicos de manutenção. Esse rol é retomado no *capítulo 5*, que apresenta alternativas para a redução dos problemas, com base na extensão do estudo de caso, na norma ISO/IEC 12207 e na literatura.

Nos capítulos seguintes são tratados assuntos referentes ao ensino de manutenção de software. No *capítulo 6* é feita uma revisão bibliográfica das características pertinentes a esse ensino, com a exposição de padrões de currículos para cursos de computação e resultados com disciplinas de manutenção de software experimentais.

No *capítulo 7* é apresentada a proposta de assuntos que se mostraram essenciais a uma futura ementa de disciplina de manutenção de software, expondo a fundamentação de cada assunto, bem como a proposta de estruturação.

Finalmente, no *capítulo 8*, as conclusões gerais deste trabalho são apresentadas, incluindo suas contribuições e idéias para trabalhos futuros.

CAPÍTULO

2

Manutenção de Software

2.1 Considerações Iniciais

O desenvolvimento de qualquer software, exceto programas muito simples, é uma tarefa bastante complexa. Esse fato se tornou evidente com a "crise de software", o que originou o conceito de engenharia de software como uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, manutenção e descarte de software.

A engenharia de software surgiu inicialmente mais como uma promessa do que uma realidade. De fato, muitos dos problemas ligados ao desenvolvimento e manutenção de software continuam sem solução, apesar de muitos conceitos terem evoluído.

Entender o que significa manutenção de software, e principalmente a abrangência do significado do termo, constitui passo fundamental para o estudo e aprofundamento de soluções para os problemas atrelados a essa tarefa.

Faz-se importante destacar nesse ponto que o autor deste trabalho vem trabalhando diretamente com manutenção de software no meio organizacional há alguns anos, desde o momento em que concluiu seu curso de graduação na área de computação. Esta experiência acaba por influenciar a percepção dos fatores mais importantes envolvidos na prática com a manutenção de software, contribuindo para a concretização deste trabalho.

2.2 Atividade de Manutenção de Software

Este tópico tem por objetivo destacar as características, tipos e desafios para a manutenção de

software. Trata-se de um tópico importante para esclarecer alguns pontos pertinentes ao assunto que serão considerados ao longo deste trabalho.

2.2.1 Definições

A atividade de manutenção de software é caracterizada pela modificação de um produto de software já entregue ao cliente, para a correção de eventuais erros, melhora em seu desempenho, ou qualquer outro atributo, ou ainda para adaptação desse produto a um ambiente modificado (IEEE, 1998).

Embora a definição trate genericamente qualquer produto de software, existem diferenças entre a manutenção de softwares com propósitos distintos. Essa distinção é explicada por Pfleeger (2001), que estabelece três categorias de sistemas.

Uma primeira classificação[2] representa aqueles softwares construídos com base em uma especificação rígida e bem definida, cujos resultados esperados são bem conhecidos. Por exemplo, um software construído para realizar operações com matrizes (adição, multiplicação e inversão). Nesse tipo de software, uma vez que tenha sido construído considerando a correta implementação do método, dificilmente haverá a necessidade de manutenções.

Já em uma segunda classificação[3], são agrupados os softwares que constituem implementações de soluções aproximadas para problemas do mundo real, uma vez que soluções completas somente são conseguidas na teoria nesses casos. Como exemplo, pode-se citar um jogo de xadrez. Embora suas regras sejam bem definidas, não é possível construir um software que calcule a cada passo todos os possíveis movimentos de peças do tabuleiro, de forma a determinar o melhor movimento. Isso porque o número de movimentos possíveis é muito grande para ser calculado em um intervalo de tempo relativamente curto. A técnica utilizada para desenvolver esse tipo de solução baseia-se em descrever o problema de forma abstrata e então definir os requisitos de software a partir dessa abstração.

Percebe-se que esse tipo de sistema já abre espaço para diferentes interpretações por parte do desenvolvedor, o que tende a produzir software com maior necessidade de manutenção do que quando comparado aos da classificação anterior. Por considerar uma abstração para especificação de requisitos, a necessidade de mudança pode aparecer caso a abstração mude, na medida em que um maior entendimento do problema seja alcançado.

Finalmente, uma terceira e última classe de softwares[4] considera mudanças no ambiente onde o software vai ser utilizado, característica não existente nas duas classificações anteriores.

---

[2] *S-systems*, na nomenclatura adotada pelo autor.
[3] *P-systems*, idem.
[4] *E-systems*, idem.

Um software integrante da terceira classificação corresponde àquele criado com base em um modelo dos processos abstratos envolvidos no sistema, e precisará mudar sempre que ocorrer mudanças nesses modelos, sendo, portanto, parte do ambiente que ele modela. Um exemplo desse tipo de software seria aquele que apresenta informações da economia de um país. À medida que a economia passa a ser mais bem compreendida, o modelo muda e com ele a abstração do problema, causando uma necessidade inevitável de manutenção no software. Esse tipo de software, comumente encontrado no dia-a-dia das organizações, tem interesse particular neste trabalho.

Além de considerar tipos diferentes de software, o processo de manutenção não corresponde a uma atividade isolada. Durante a sua execução, diferentes partes precisam interagir de forma que os objetivos da manutenção sejam entendidos e os resultados esperados alcançados. Essas partes são apresentadas em Polo *et al.* (1999), que as define como:

- *Organização Cliente:* essa organização corresponde ao *adquirente* do software, conforme definido pela norma ISO/IEC 12207. Isso significa a organização que possui o software e requisita o serviço de manutenção.
- *Mantenedor:* trata-se da organização que oferece o serviço de manutenção.
- *Usuário:* representa a organização ou pessoa que utiliza o software em questão, valendo-se de suas funcionalidades para automatizar e facilitar tarefas.

Ainda segundo esse autor, existe um relacionamento entre essas partes e a constituição de um fluxo para os pedidos de manutenção, conforme é esquematizado na figura 2.1.

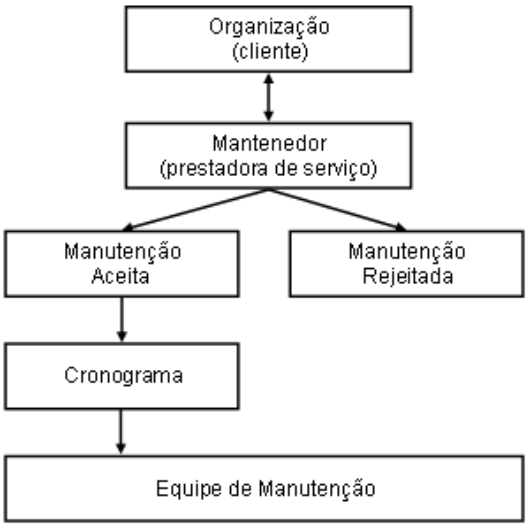

FIGURA 2.1: FLUXO ENVOLVIDO NA MANUTENÇÃO DE SOFTWARE (*ADAPTAÇÃO POLO ET AL. 1999*)

A organização solicitante deve possuir algum responsável pela solicitação de manutenção, que será encarregado de verificar os requisitos e informar o mantenedor. Esse

responsável deverá considerar os erros e dificuldades apontadas pelo *help-desk*, que corresponde ao departamento diretamente encarregado do diálogo com os usuários.

Do lado do mantenedor, a organização que presta o serviço deve possuir alguém responsável por avaliar as requisições, julgando-as apropriadas ou não para os objetivos do software e da organização solicitante. Uma vez que os pedidos de manutenção são aceitos, deve existir alguém responsável por estabelecer um cronograma de entrega, que deverá considerar as prioridades e interesses de ambas as partes. Esse cronograma deverá ser seguido pela equipe de manutenção, que compreende o pessoal envolvido diretamente em atender às solicitações.

Finalmente, o papel do usuário consiste em utilizar o software reportando problemas para o *help-desk*, que por sua vez informará o responsável pelas solicitações de manutenção, fechando o ciclo.

Uma observação faz-se necessária com relação ao termo *falha* e *defeito* de software. Conforme explica Pressman (2005), a IEEE[5] define para o contexto de hardware a explicação de que um defeito constitui uma *anomalia do produto*. Já uma *falha* pode significar um defeito em um dispositivo ou componente, ou uma definição incorreta de passo, processo ou de dados em um programa de computador.

No contexto de manutenção de software, os termos falha e defeito são sinônimos, e ambos se referem a algum problema de qualidade, identificado após o software ter sido entregue ao usuário.

2.2.2 Evolução de Software

As mudanças que ocorrerão em um software para deixá-lo mais completo, livre de erros, ou adaptado ao seu ambiente, podem ser definidas como atividades de evolução de software, conforme explica Sommerville (2003).

A decisão de investir na evolução de um software, ou abandoná-lo para iniciar um projeto novo, construído com base nos requisitos atuais, não é uma tarefa trivial (Pfleeger, 2001). Essa decisão deve considerar fatores como o custo envolvido, a confiabilidade do software após a manutenção, a capacidade que possuirá de se adaptar a mudanças futuras, seu desempenho, limitações de suas atuais funções, e ainda as opções existentes no mercado que atendam o mesmo problema de maneira mais completa, rápida e barata.

Se o software atual funciona adequadamente, supõe-se que a organização que o utiliza terá uma preferência em aplicar a ele apenas os ajustes necessários em função de mudanças nos negócios ou outras necessidades de adaptações, do que substituí-lo por completo.

[5] Institute of Electrical and Electronics Engineers.

Abandonar sistemas existentes para iniciar projetos a partir do zero representa uma perda considerável (Silva & Santander, 2004).

Visando entender melhor as características de evolução de software, Lehman (1996), publicou suas oito "leis da evolução de software"[6]. De acordo com esse trabalho, seriam oito as principais características que regem o envelhecimento de um software, e foram determinadas após mais de vinte anos de estudos e observações. Inicialmente foram definidas apenas três leis (Lehman, 1974), que passaram a ser cinco (Lehman, 1980), alcançando seis leis (Lehman, 1991). Finalmente, após estudos mais longos com processos de software, as atuais oito leis foram publicadas. Essas leis consideram não apenas o software em si, mas as características da organização que o desenvolve e mantém.

A primeira lei, caracterizada pela evolução contínua, explica que um software desenvolvido para tratar um problema do mundo real, precisa ser constantemente adaptado, pois caso contrário, ele se tornará progressivamente menos satisfatório. Essa lei baseia-se na experiência, sendo que a evolução somente pode acontecer por meio de um processo de manutenção controlado e dirigido.

Na segunda lei, o autor explica que existe um aumento progressivo de complexidade do software à medida que este se desenvolve, o que pode ocasionar uma desestruturação crescente. Esse problema será inevitável, a menos que algum trabalho seja feito para reduzi-lo, e é ocasionado por alterações efetuadas em cima de outras alterações, na medida em que o software é adaptado ao seu ambiente.

A terceira lei prega uma auto-regulação do processo de evolução do software. Isso significa que a organização responsável pelo desenvolvimento desse software estabelecerá normas e verificações capazes de assegurar o entendimento das dificuldades e prover respostas que serão utilizadas para o crescimento e estabilização tanto do processo, como do produto de software. O gerenciamento de retornos positivos e negativos dos pontos de controle é um exemplo de mecanismos de estabilização. Existem vários outros, e juntos eles estabelecem uma dinâmica disciplinada cujos parâmetros são, pelo menos em parte, normalmente distribuídos.

A quarta lei trata da conservação da estabilidade organizacional, no sentido de que ao longo de todo ciclo de vida do software, a forma como a organização irá trabalhá-lo sempre produzirá resultados constantes. A lei sugere que mudanças de pessoal e recursos têm pouco efeito sobre a evolução do sistema.

Na quinta lei, é apresentada a característica de conservação de familiaridade, o que significa dizer que o conhecimento dos objetivos da organização ao lançar uma nova

[6] Do inglês, *laws of software evolution*.

atualização do software, é conhecido entre os membros da equipe que trabalha com esse software. Quanto mais mudanças forem necessárias ao software, maior será a dificuldade de manter toda a equipe ciente dos objetivos organizacionais. A taxa, qualidade de progresso e outros parâmetros são influenciados, até mesmo limitados, pela taxa de aquisição da informação necessária pelos participantes coletivamente e individualmente.

A sexta lei consagra a característica do contínuo crescimento do software, decorrente de novas funcionalidades, a fim de manter a satisfação do cliente. Essas mudanças podem ser correções de erros, adições de novas funcionalidades ou melhorias em funções pré-existentes. A maioria certamente não foi planejada pela equipe de desenvolvimento na época da primeira versão, ou foram causadas devido a alguma mudança no domínio operacional em que o software está inserido. Essas mudanças não planejadas inicialmente geram a necessidade de módulos extras para o software, causando assim um inevitável aumento do conteúdo funcional.

Na sétima lei, é apresentada a característica de qualidade decrescente do software, à medida que evolui. Isso ocorre uma vez que o software está inserido em um ambiente imprevisível, no qual existem possibilidades ilimitadas que entrarão em contraste com as características de evolução do software, limitadas pelo número de recursos e tempo disponível, expondo a suscetibilidade do software a eventos externos. Ainda que esse software funcionar satisfatoriamente por muitos anos, isto não será um indicativo de que continuará funcionando a contento nos anos vindouros. Essa característica somente poderá ser evitada com um esforço para detecção e correção contínuas.

Finalmente, na oitava lei, publicada somente na década de 1990, o autor apresenta a característica de sistema de retorno, o que significa que o sistema de evolução de um software será constantemente realimentado pelo retorno de seus usuários. A vida de um software é um ciclo bem estabelecido de retornos positivos e negativos dos usuários entre cada versão do software. Em longo prazo, a taxa de crescimento do sistema será estabelecida pela quantidade de retornos negativos e positivos, e controlado por fatores como quantidade de recursos (verba), número de usuários solicitando novas funcionalidades ou reportando algum erro, interesses administrativos e tempo entre uma versão e outra.

De uma maneira geral, as leis de Lehman abordam características aplicáveis a grandes corporações, que desenvolvem softwares complexos e com longo ciclo de vida. Não ficam claras quais as dificuldades para a manutenção de software. O que se tem é um panorama geral da evolução de software, apresentando em diversas leis a necessidade de intervenção no software para que problemas sejam evitados, mas nada diz a respeito das dificuldades que emergem dessa intervenção.

2.2.3 Natureza e Características

Atividades de manutenção de software são caracterizadas por intervenções no produto de software de forma a evitar sua deterioração. Um software não se desgasta como peças de um equipamento, mas se deteriora no sentido de os objetivos de suas funcionalidades cada vez menos se adequarem ao ambiente externo.

Do ponto de vista de desenvolvimento, Pfleeger (2001) explica que o foco das atenções está em produzir código que atenda aos requisitos e funcione corretamente. Isso inclui dizer que a cada estágio do desenvolvimento, haverá uma referência contínua a componentes produzidos em estágios anteriores. O desenvolvimento levará a uma integração dos componentes desenvolvidos, passando por etapas de revisão e testes para certificação de sua corretude, adequação aos requisitos a ao projeto. Resumindo, o desenvolvimento terá foco em considerar estágios anteriores do processo de maneira controlada.

A manutenção de software, por sua vez, incluirá não apenas considerar resultados de etapas anteriores, mas atividades do presente de forma a conseguir compreender o nível de satisfação dos usuários em relação ao software. Uma característica importante é também considerar o futuro durante a manutenção, buscando antever necessidades de mudanças nos negócios, o que causaria mudanças nesse software.

Outra característica fundamental está relacionada à imprevisibilidade da manutenção de software. Esse fato está relacionado à influência que o software sofre de fatores externos, naturalmente imprevisíveis (Bhatt *et al.*, 2004).

Na figura 2.2 é mostrada a curva de esforço despendido com a atividade de manutenção de software, destacando os picos de esforço ocasionados por fatores externos.

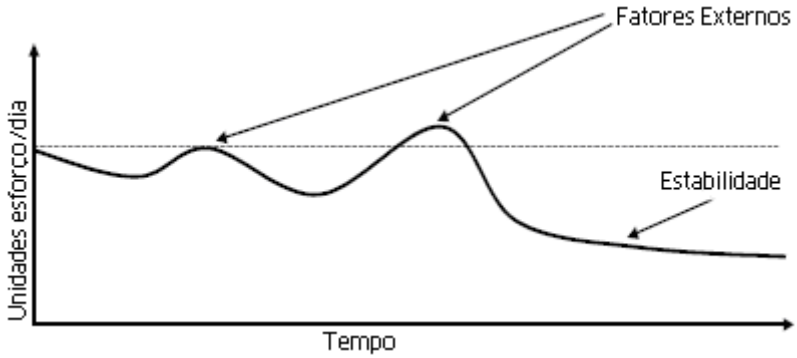

FIGURA 2.2: CURVA DE ESFORÇO PARA MANUTENÇÃO DE SOFTWARE (*ADAPTAÇÃO DE BHATT ET AL. 2004*)

Pela figura percebe-se que o software é sensível ao contexto no qual está inserido, estando sujeito a mudanças na medida em que seu ambiente muda, causando a necessidade de considerar e tentar prever esses fenômenos, tanto durante o desenvolvimento como durante a manutenção.

Um modelo de ciclo de vida de manutenção de software é proposto por Kung e Hsu (1998), no qual se destacam quatro fases de vida para uma aplicação de software: (i) introdução; (ii) crescimento (iii) maturidade; (iv) declínio. Para cada fase, os autores indicam qual o tipo de tarefa que será mais incidente, conforme mostrado na figura 2.3.

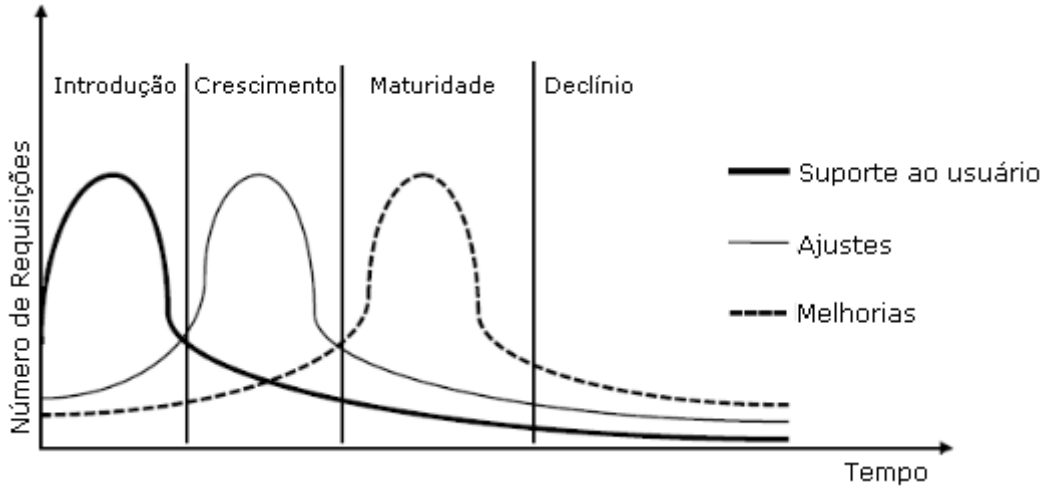

FIGURA 2.3: CICLO DE VIDA DE MANUTENÇÃO DE SOFTWARE (*ADAPTAÇÃO DE KUNG & HSU, 1998*)

Quanto tempo o software vai levar para entrar na fase de declínio é uma previsão quase impossível, visto que, uma vez na fase de maturidade, o software adquiriu certa estabilidade na organização, possivelmente já se enquadrando na classificação de sistema legado. Nesse sentido, a fase de declínio corresponderia àquela na qual uma decisão a respeito de manter ou substituir o software deve ser tomada, conforme exposto com mais detalhes no tópico 2.4, relacionado a sistemas legados.

2.2.4 Tipos de Manutenção

As ações ligadas à atividade de manutenção de software foram classificadas de acordo com sua natureza em três categorias (Lientz & Swanson, 1980): *corretivas*, *adaptativas* e *perfectivas*.

- Manutenções do tipo *corretivas* visam corrigir defeitos de funcionalidade, o que inclui acertos emergenciais de programa. Pfleeger (2001) expõe um exemplo desse tipo de manutenção, que consiste em um usuário apresentando um problema de impressão em um relatório. O número de linhas impresso por folha é muito grande, o que causa sobreposição de informações. O problema foi identificado como uma falha no *driver* da impressora, provocando a necessidade de se alterar o menu do relatório para aceitar um parâmetro adicional que determina o número máximo de linhas impressas por folha.

- Manutenções do tipo *adaptativas* referem-se a adequar o software ao seu ambiente externo. O exemplo apontado por Pfleeger (2001) ilustra bem essa categoria. Suponha um gerenciador de banco de dados, que faz parte um sistema maior de hardware e software. Em uma atualização do gerenciador, os programadores perceberam que as já existentes rotinas de acesso a disco precisavam agora de mais um parâmetro adicional. Essa manutenção corresponde a uma manutenção adaptativa, uma vez que teve por finalidade adequação do software ao seu ambiente e não a correção de um defeito.
- Manutenções do tipo *perfectivas* têm por objetivo acrescentar novos recursos de funcionalidade ao software, normalmente em razão de solicitações dos usuários. Significam ainda re-projetar partes de um software, de forma a tornar mais simples a compreensão e utilização do mesmo. Como exemplo, pode-se citar o pedido do usuário por um novo relatório com informações que até então não podiam ser obtidas do banco de dados.

A união das categorias adaptativa e perfectiva é sugerida por Pigoski (1996), que propõe uni-las em uma única denominada *aprimoramentos*[7]. Essa classificação estaria de acordo com organizações que geralmente utilizam o termo manutenção para se referir à execução de pequenas mudanças no software, enquanto o termo desenvolvimento é usado para os demais tipos de modificações.

Uma quarta categoria de manutenção é apresentada por alguns autores, como Pressman (2005). Essa categoria se refere a manutenções do tipo *preventivas* que buscam identificar previamente possíveis fontes de problemas no software e corrigi-las antecipadamente.

A IEEE (1998) modifica a definição apresentada inicialmente por Lientz e Swanson (1980), definindo uma categoria a mais chamada *emergencial*. Essa categoria é caracterizada pela execução de uma manutenção corretiva não planejada, com o intuito de manter o software operacional. Tal classificação insere a idéia de manutenção planejada e não-planejada, bem como manutenção reativa e pró-ativa, como é ilustrado no quadro 2.1.

QUADRO 2.1: DEFINIÇÕES IEEE PARA AS CATEGORIAS DE MANUTENÇÃO DE SOFTWARE

	Planejada	Não-Planejada
Reativa	Corretiva Adaptativa	Emergencial
Pró-ativa	Perfectiva	

[7] Do inglês, *enhancements*.

Para efeito de estudo, neste trabalho será considerada a classificação de tipos de manutenção definida por Pressman (2005): corretivas, adaptativas, perfectivas e preventivas.

Dentro dessa classificação adotada, um estudo com empresas de software (Souza *et al.*, 2004), constatou que entre as organizações pesquisadas, as manutenções do tipo corretivas prevaleceram com 50% dos resultados, seguido pelas perfectivas (30%) e adaptativas (20%). Na pesquisa de Souza não foi identificada nenhuma organização realizando manutenção do tipo preventiva.

Essa ausência de manutenção preventiva também pode ser verificada no estudo de caso que será apresentado no capítulo 3.

2.2.5 Custos e Desafios

A manutenção de software é uma operação importante pois consome a maior parte dos custos envolvidos no ciclo de vida de um software (SWEBOK, 2004), e a falta de habilidade em mudar um software rapidamente, e de maneira confiável, pode causar a perda de oportunidades de negócio (Bennett & Rajlich, 2000).

Embora não exista um consenso sobre o valor exato do custo atrelado à atividade de manutenção, as pesquisas na área apontam, na totalidade dos casos, sempre mais de 50% dos investimentos realizados no software. De fato, para Bhatt *et al.* (2004), esse percentual corresponde a algo entre 67% a 75% do investimento total, enquanto para Polo *et al.* (1999) corresponde a um valor entre 67% e 90%. Ainda de acordo com esse último autor, a razão do custo elevado deve-se, em parte, à própria natureza da atividade de manutenção, caracterizada principalmente pela imprevisibilidade. Além dos altos custos financeiros, essa é também a atividade que exige maior esforço dentre as atividades de engenharia de software, conforme apontado por Sneed (2003). Pressman (2005) ainda completa que o grande esforço necessário na manutenção se justifica pela abrangência do significado desse termo no contexto de software.

A importância financeira atrelada à manutenção de software é ainda agravada quando se leva em consideração o risco para as oportunidades de negócio, que podem ser causadas pela falta de gerenciamento e compreensão total da dinâmica da atividade. De acordo com a pesquisa realizada por Dart *et al.* (2001), esse gerenciamento deve considerar três fatores: (i) ferramentas, (ii) pessoas, (iii) processos, revelando-se, pois, uma atividade gerencial complexa.

Se por um lado a atividade de manutenção é dispendiosa, por outro ela é um desafio para as organizações que precisam considerá-la em seu dia-a-dia. Não é de se esperar que uma empresa de grande porte troque todos seus sistemas somente pelo fato de que a tecnologia

neles empregada está ultrapassada. Esses sistemas representam ativos importantes da organização e ela estará disposta a investir de maneira a manter seus valores (Sommerville, 2003).

Prever quando uma manutenção precisará ser conduzida é uma tarefa geralmente muito difícil de ser realizada. Essa dificuldade de previsão, e também controle sobre a manutenção, é explicada pelo fato de muitas vezes ficar a cargo de empresas terceirizadas a manutenção, revelando-se um problema já que normalmente essas organizações não possuem nenhum contato com o projeto inicial do software (Bhatt *et al.*, 2004). Ainda segundo esse autor, o aumento na complexidade dos softwares produzidos (tanto em termos de funcionalidades, como de técnicas) torna a previsão de esforços de manutenção muito vaga. Essa dificuldade em estimar esforços torna-se mais evidente quando se trata de sistemas legados, que serão discutidos com maiores detalhes no tópico 2.4.

2.3 Norma ISO/IEC 12207

Em 1987 a ISO e a IEC estabeleceram um Comitê Técnico Conjunto – JTC (*Joint Technical Committee*) sobre tecnologia da informação, com o objetivo de efetuar a normatização no campo de sistemas de tecnologia da informação (incluindo microprocessadores) e equipamentos (Singh, 1996).

Dentre os objetivos do comitê estava o estabelecimento de um padrão para processos de ciclo de vida de software, que culminou com a norma ISO/IEC 12207, que teve início em 1989. Esta norma deveria ser estabelecida como um *framework* adaptável, que pudesse ser usado para auxiliar na gerência e na construção do software.

Assim, a ISO/IEC 12207 foi publicada em 1 de agosto de 1995, todavia está em constante evolução, como pode ser comprovado pelas duas emendas que já recebeu (*amendment* 1, publicada em 2002, e a *amendment* 2, publicada em 2004).

Essa norma provê 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 um modelo de referência de processos. Sua organização se dá de acordo com o exposto na figura 2.4.

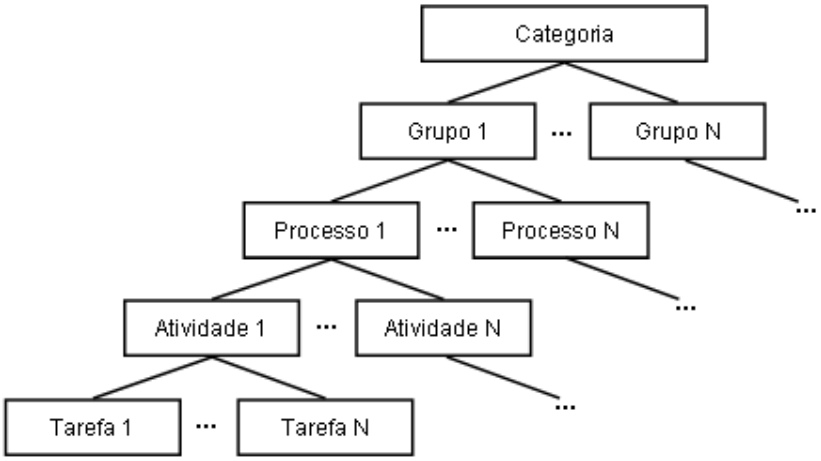

FIGURA 2.4: ESTRUTURAÇÃO DA NORMA ISO/IEC 12207

Ao todo são três categorias, totalizando dez grupos de processos com 47 processos no geral. Na figura 2.5, extraída da norma ISO/IEC 15504, é mostrada a organização da norma.

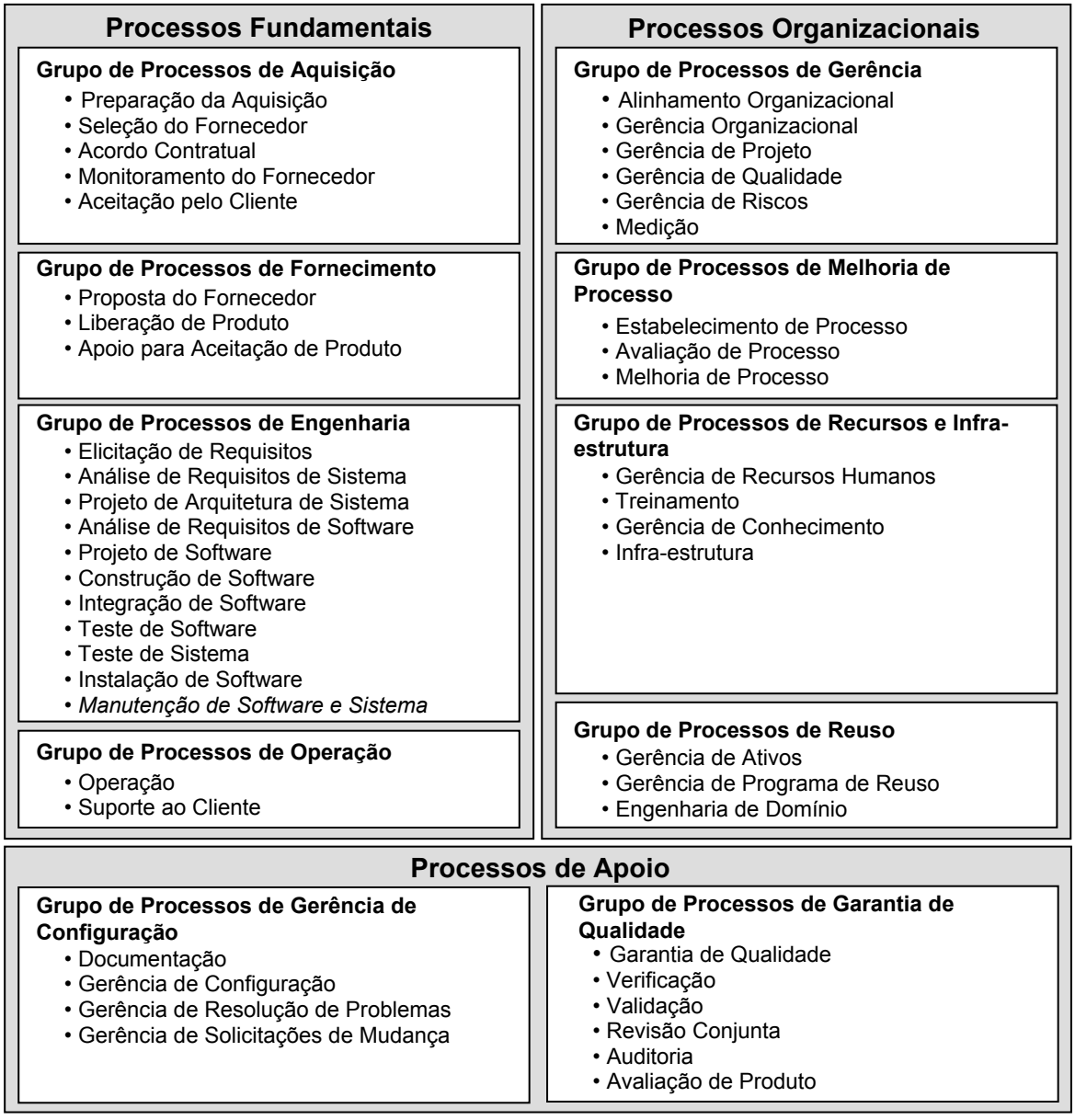

FIGURA 2.5: PROCESSOS DO CICLO DE VIDA DA NORMA ISO/IEC 12207

Para um melhor entendimento, é relevante descrever, ainda que em linhas gerais em um primeiro momento, quais os propósitos de cada um dos grupos de processos da norma. No capítulo 5 esses processos são retomados com o intuito de se estabelecer quais deles implicariam diretamente na redução de problemas de manutenção de software.

A – Categoria de Processos Fundamentais

Grupo de Processos de Aquisição: inclui processos a serem seguidos pelo cliente com a finalidade de aquisição de um produto ou serviço.

Grupo de Processos de Fornecimento: inclui processos a serem observados pelo fornecedor, com a finalidade de oferecimento de um produto ou serviço.

Grupo de Processos de Engenharia: inclui processos para elicitação e gerenciamento dos requisitos do cliente, bem como especificação, implementação e/ou manutenção do produto de software, considerando ainda sua relação com o sistema maior do qual fizer parte.

Grupo de Processos de Operação: inclui processos com a finalidade de oferecer suporte à transição do produto para o ambiente do cliente, provendo ainda a correta operação e uso desse produto ou serviço.

B – Categoria de Processos Organizacionais

Grupo de Processos de Gerência: inclui processos que englobam práticas que devem ser observadas por todos aqueles que gerenciam qualquer tipo de projeto ou processo relacionado ao ciclo de vida de software.

Grupo de Processos de Melhoria de Processo: inclui processos destinados à definição, criação e melhoria de processos realizados na organização.

Grupo de Processos de Recursos e Infra-estrutura: inclui processos com a finalidade de prover recursos humanos e infra-estrutura adequada para todos os processos da organização.

Grupo de Processos de Reuso: inclui processos com a finalidade de sistematicamente explorar oportunidades de reuso dentro da organização.

C – Categoria de Processos de Apoio

Grupo de Processos de Gerência de Configuração: inclui processos com a finalidade de controle e manutenção da integridade do produto desenvolvido pelos processos de engenharia.

Grupo de Processos de Garantia de Qualidade: inclui processos com o intuito de prover garantias de que os produtos de trabalho e processos estejam de acordo com as condições predefinidas e o planejado.

A manutenção de software aparece na norma como um dos processos dentro da categoria de processos fundamentais. Na figura 2.6 é apresentado quais são as atividades pertencentes a esse processo.

FIGURA 2.6: ATIVIDADES DO PROCESSO DE *MANUTENÇÃO DE SOFTWARE E SISTEMA* (ISO/IEC 12207)

- *Implantação do processo*: essa atividade inclui tarefas para o desenvolvimento de planos e procedimentos para manutenção de software, criando procedimentos para receber, gravar e monitorar pedidos de manutenção, e estabelecer uma interface organizacional com o processo de gerenciamento de configuração. A implementação do processo deve começar cedo no ciclo de vida do software, conforme reforça Pigoski (1996) ao dizer que os planos de manutenção devem ser preparados em paralelo com os planos de desenvolvimento. Essa atividade inclui definir o escopo de manutenção e a identificação e análise de alternativas, bem como organizar e contratar a equipe de manutenção, relacionando recursos e responsabilidades.
- *Análise do problema e da modificação*: em um primeiro momento, essa atividade tem por objetivo analisar a requisição de manutenção para classificá-la, podendo então determinar o escopo em termos de tamanho, custos e tempo necessário, destacando ainda sua prioridade. As demais tarefas dessa atividade focam o desenvolvimento e a documentação de alternativas para mudança de implementação e aprovação das opções adotadas.
- *Implantação da modificação*: essa atividade engloba a identificação dos itens que precisam ser modificados e os processos de desenvolvimento que precisarão ser

implementados. Outros requisitos da modificação incluem teste e validação de que as modificações estão corretamente implementadas e que os itens não modificados não foram afetados.

- *Revisão / aceitação da modificação*: as tarefas dessa atividade são dedicadas à confirmação da integridade do software modificado e à conclusão dos negócios com o cliente, quando este concorda e aprova satisfatoriamente a conclusão da requisição de manutenção. Muitos processos de apoio podem ser utilizados aqui, incluindo a garantia da qualidade de processo, o processo de verificação, o processo de validação e o processo de revisão conjunta.
- *Migração*: corresponde à atividade que ocorrerá quando o software for transferido de um ambiente de operação para outro. Será preciso o desenvolvimento de planos de migração e os usuários precisarão estar cientes dos requisitos, dos motivos do antigo ambiente não ser mais suportado e terem à disposição uma descrição do novo ambiente e sua data de disponibilidade. Outras tarefas dessa atividade concentram-se em operações paralelas do novo e antigo ambiente, incluindo a revisão de pós-operação para certificar-se do impacto da mudança de ambiente.
- *Descontinuação do software*: essa atividade consiste em descontinuar o software por meio da formalização, junto ao cliente, de um plano de descontinuação.

Um último comentário referente à norma refere-se ao fato de que ela não representa um modelo fixo ao qual uma organização que a adote se submete. Ao contrário disso, ela funciona como uma estrutura de apoio, devendo a organização que a adotar proceder com adaptações nas recomendações para a sua realidade.

2.4 Sistemas Legados

Um sistema legado normalmente representa um dos bens com maior valor econômico de uma organização (Visaggio, 2001).

Essa é a idéia central que envolve as decisões em torno de um software nessas condições, gerando muitas variáveis a considerar, entre elas: (i) é arriscado substituir um software que esteja estável; (ii) os usuários já estão acostumados com a forma de trabalhar com o software e uma mudança normalmente não seria bem vinda; (iii) a mudança vai exigir gastos com treinamento de todos os envolvidos com o uso desse software; (iv) a compra ou construção de um novo software pode extrapolar previsões de custos e prazos; (v) o novo software pode possuir menos funcionalidades, dificultando tarefas dos usuários etc.

Se por um lado o software não pode ficar como está, já que ou possui erros, ou não está mais atendendo às novas necessidades de negócios, por outro, substituí-lo pode trazer problemas ainda maiores.

Neste tópico serão consideradas as características desses softwares e algumas das soluções, e paliativos, já propostos, uma vez que se relacionam diretamente com a atividade de manutenção de software.

2.4.1 Conceitos

Um software desenvolvido no passado e ainda um utilização em uma determinada organização constitui um sistema legado.

Em função da dependência cada vez maior em relação a sistemas de software, as organizações em geral, especialmente as de grande porte, investiram muito dinheiro no desenvolvimento de softwares para atender às suas necessidades operacionais e de gerência. Considerando que a evolução da tecnologia, tanto de hardware como de software, foi muito rápida nos últimos anos, provocando o abandono de linguagens de programação mais antigas e a adoção de novas metodologias de desenvolvimento de software, é fortemente esperado que aqueles softwares, frutos de grandes investimentos das organizações, tenham sido construídos com alguma tecnologia ou método que hoje é obsoleto. Esses softwares, chamados de sistemas legados, constituem mais um desafio para a engenharia de software, especialmente para a atividade de manutenção de software, uma vez que dificilmente serão abandonados de imediato, justamente por causa do investimento que representam.

A vida útil de um software é muito variada. Alguns softwares de grande porte podem continuar em uso durante mais de dez anos, e em alguns casos organizações podem ainda contar com software desenvolvido há mais de vinte anos (Sommerville, 2003). Uma tentativa de quantificar a quantidade de software legado existente foi feita em 1990, e ainda naquela data, estimou-se que haveria no mundo cerca de 120 bilhões de linhas de código pertencentes a esse tipo de software (Ulrich, 1990).

2.4.2 Problemática dos Sistemas Legados

Várias abordagens foram propostas para gerenciar software legado. Soluções típicas são (Bennett *et al.*, 1999): (i) descartar o software e substituí-lo totalmente (muito inviável na grande maioria dos casos, por restrições financeiras e técnicas); (ii) tornar o software parte integrante de um novo software de maiores proporções; (iii) adiar a manutenção por mais algum tempo (nem sempre isso é possível), e finalmente, (iv) modificar o software para aumentar sua vida útil.

Quando a decisão de modificar o software é tomada, inicia-se um processo de entender esse software, bem como suas estruturas, uma vez que dificilmente o projeto inicial estará disponível, e se estiver, seguramente estará desatualizado.

As razões para explicar a dificuldade de manutenção em sistemas legados apresentada por Sommerville (2003) estão descritas a seguir:

- Diferentes partes do software foram desenvolvidas por equipes diferentes, o que implica em uma não uniformidade no estilo de programação ao longo dele todo;
- Partes do software, ou ele todo, podem ter sido desenvolvidas com o uso de alguma linguagem de programação obsoleta. Isso torna difícil encontrar profissionais com conhecimento e experiência na linguagem, o que pode forçar a terceirização da manutenção (normalmente a um custo elevado);
- A documentação existente é normalmente inadequada e desatualizada. Em muitos casos, a única documentação existente é o próprio código-fonte. Em outras situações mais graves, nem mesmo o código-fonte está disponível, somente a versão executável existe;
- Provavelmente os muitos anos de manutenções alteraram a estrutura do software, tornando-o progressivamente mais difícil de compreender. Novos programas podem ter sido adicionados e sua *interface* de comunicação com outras partes construída de maneira *ad-hoc*;
- O software pode ter sido *otimizado* para reduzir a ocupação de espaço ou melhorar sua velocidade de execução. Isso pode causar problemas de compreensão para profissionais que aprenderam somente as técnicas modernas de engenharia de software, não tendo tido contato com os *truques* usados no passado;
- Os dados processados pelo software podem estar armazenados em arquivos separados, com estruturas distintas e incompatíveis. Podem ainda existir duplicações de dados, erros e dados incompletos.

Vale aqui uma observação curiosa, referente ao crescimento do número de profissionais da área de software interessados em aprender linguagens e técnicas de programação *antigas*, justamente para trabalharem no aparente promissor campo da migração de sistemas legados. Organizações de porte médio e grande têm se dedicado à preparação de equipes para conduzir longos e complexos processos de migração de grandes sistemas legados, notadamente aqueles construídos com a linguagem de programação COBOL. Esse é provavelmente o fato que tem feito com que os livros de linguagens antigas estejam voltando

a circular, principalmente na indústria, entre profissionais da área.

2.5 Gerenciamento de Manutenção de Software

Com o aumento da necessidade de manutenção de software, principalmente de softwares mal desenvolvidos, as organizações têm buscado maneiras de lidar com o problema, já que não é possível trocar o software por um "modelo mais novo", como seria comum pensar em face de um equipamento com defeitos ou com funcionalidades limitadas.

Essa busca por alternativas viáveis economicamente e eficazes tecnicamente, acabam por produzir diferentes abordagens de resposta à necessidade de manutenção, respostas essas que não são sempre corretas e de aplicabilidade universal.

Está a cargo dos tomadores de decisão em relação a software decidir o que fazer e como proceder à necessidade de manutenção de software, o que envolverá sempre questões financeiras e a credibilidade da organização frente a seus clientes.

De Lucia *et al.* (2004) organizaram em quatro grupos as questões importantes a considerar por esses tomadores de decisão:

- *Programadores*: envolve as questões ligadas à equipe de engenheiros de software. Aqui estão incluídos assuntos importantes como a atitude dos profissionais diante de tarefas de manutenção e não de desenvolvimento, ressaltando a necessidade de avaliar questões não apenas técnicas, mas também de compreensão dos negócios do cliente. Nesse grupo estão incluídas ainda eventuais alegações de falta de *glamour* da atividade de manutenção, inclusive questões como a impossibilidade de desenvolvimento de carreira em manutenção de software, o que acaba por gerar grande rotatividade. Deve-se ainda considerar a habilidade que a equipe de engenheiros de software possui com a atividade de manutenção, avaliando-se a curva de aprendizado para a atividade.

- *Atitude Gerencial*: observa-se que a própria postura gerencial é a principal responsável por ditar os rumos e o sucesso ou fracasso do ambiente de manutenção. Esse grupo envolve as questões relacionadas ao estudo de programas de manutenção, que normalmente são escassas, e seu relacionamento com fatores como tempo e dinheiro. É justamente esse tipo de estudo que permite o sucesso em tomadas de decisão relacionadas a software. Um fator apontando nesse grupo refere-se à atitude gerencial de atribuir o desenvolvimento de projetos importantes a programadores como uma forma de presenteá-los por bom desempenho profissional. Essa atitude é agravada ainda mais quando é acompanhada de falta de treinamento e oportunidade para equipes de manutenção.

- *Código-Fonte*: definir critérios de qualidade para o código-fonte é outro grupo que envolve questões pertinentes. Arquitetura pobre de software e de estruturas de programa, documentação irregular, falta de padrões e *guidelines* para atividades de manutenção, incluindo ainda ausência de estudos de impactos de mudanças em outros trechos e módulos do software, representam considerações importantes a fazer.
- *Usuários:* os usuários devem ter participação ao longo de toda atividade de manutenção, para evitar que problemas de manutenibilidade no software possam gerar falta de credibilidade no trabalho do programador. Questões envolvendo confirmação e validação do usuário a respeito do software enquanto em processo de manutenção devem ser consideradas, para evitar erros de interpretação e de especificações. É fundamental reportar aos usuários os erros e dúvidas de interpretação.

Em razão dos muitos critérios a considerar, as decisões gerenciais podem ora fluir por um caminho, ora por outro, sendo importante o acompanhamento do resultado das decisões tomadas, efetuando ajustes o tempo todo, à medida que falhas forem diagnosticadas.

Ao tomador de decisão em organizações de software, é fundamental o conhecimento claro tanto dos objetivos de sua organização, como das características e dificuldades da atividade de manutenção de software, para então somente ser capaz de conduzir adequadamente sua equipe.

Dentro desse contexto, Basili (1990) propôs três paradigmas diferentes para tratar a manutenção de software: (i) Conserto Rápido[8]; (ii) Melhoria Interativa[9]; (iii) Reuso Total[10].

A idéia de conserto rápido compreende aplicar primeiramente as mudanças necessárias no código-fonte do software e então depois atualizar a documentação, nessa ordem. Essa abordagem corresponde à normalmente preferida pelo mantenedor, justificando a decisão pela urgência da manutenção (Visaggio, 2001). A decisão de alterar a documentação após a manutenção, no entanto, não é comumente uma boa prática, pois pressionado pelo tempo, o mantenedor executa mal, ou não executa, essa tarefa, dificultando cada vez mais manutenções posteriores. Na figura 2.7 é mostrado um paralelo entre o software atual e o derivado da manutenção, notando-se, que o ponto de partida é o código-fonte do software antes da manutenção.

[8] *Quick Fix*, nas palavras do autor.
[9] *Iterative Enhancement*, idem.
[10] *Full Reuse*, idem.

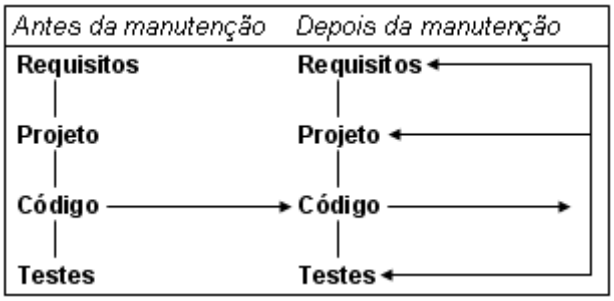

FIGURA 2.7: ESTRUTURA DA ABORDAGEM *CONSERTO RÁPIDO*

A proposta de melhoria interativa, por sua vez, propõe inicialmente adequar e atualizar a documentação de alto nível que será afetada, para então propagar as mudanças até o nível de código-fonte. Na figura 2.8 é representada essa idéia.

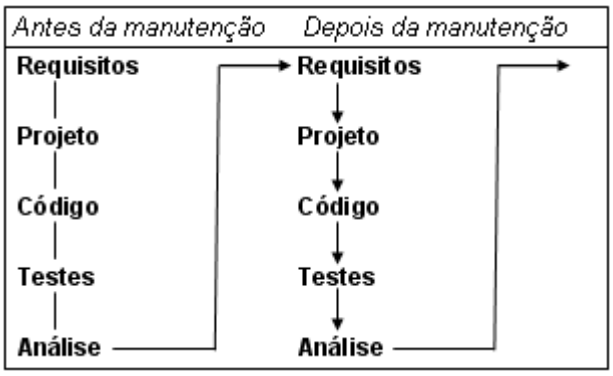

FIGURA 2.8: ESTRUTURA DA ABORDAGEM *MELHORIA INTERATIVA*

Finalmente, a última proposta, reuso total, consiste em construir um novo software, utilizando componentes do software antigo e outros disponíveis em um repositório. Na figura 2.9 essa idéia é mostrada.

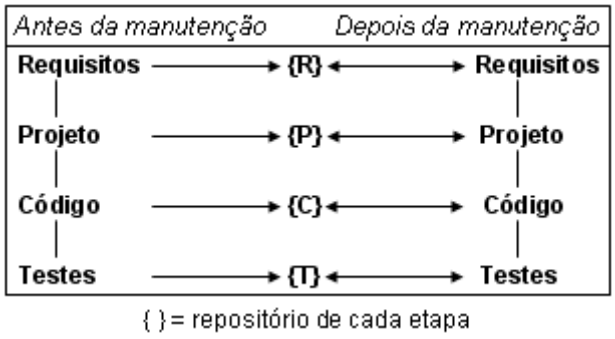

FIGURA 2.9: ESTRUTURA DA ABORDAGEM *REUSO TOTAL*

Essa última proposta não é freqüentemente usada no ambiente industrial, e uma das razões seria a deficiência ainda existente na tecnologia para gerenciar o repositório e o reuso de componentes de software.

A comparação entre os dois paradigmas mais comuns, o conserto rápido e a melhoria interativa, foi feito por Visaggio (2001) e o resultado apontou que a primeira abordagem

apenas é mais indicada do que a segunda, quando o impacto da modificação envolver poucos componentes, o que pode ser reforçado pelo menor tempo empregado para tornar a manutenção operacional. Porém, e principalmente, para alterações maiores, o emprego da melhoria contínua é a melhor indicação, pois permite antever defeitos e conseqüentemente reduzir re-trabalhos, contribuindo para a garantia de qualidade global do software. Esse último método funciona ainda como uma maneira automática de continuamente revisar a consistência do software.

Nesse ponto, o autor deste trabalho tem a acrescentar que verificou, por sua experiência na prática em algumas organizações, o emprego do método de corrigir o problema diretamente no código-fonte, exceção feita em algumas situações mais complexas que envolvem pequenas reuniões entre os profissionais responsáveis pela manutenção, produzindo em algumas delas um roteiro escrito de passos ou de fatores a observar para a manutenção. No entanto, na maior parte dos casos, esses registros não eram utilizados para a atualização da documentação de software existente. O procedimento mais adotado é o de *revisar* e completar a documentação de tempos em tempos, com base em visões gerais do software. É justo destacar, no entanto, que algumas organizações decidem por desenvolver internamente seu próprio sistema de controle de manutenções, no qual se registra requisições formais de manutenção, podendo ser mais ou menos completo dependendo da organização, contendo descrição da solução, tempo despendido, estrutura do software entre outros. Essa postura, no entanto, é normalmente adotada por organizações de médio e grande porte. Existem trabalhos dedicados a sistematizar e melhorar a construção desse tipo de ferramenta, como por exemplo, o trabalho de Koskinen *et al.* (2004), que propõe uma ferramenta baseada em hipertexto para auxiliar a recuperação de diversas informações sobre um software que sofrerá manutenção.

Outro desafio gerencial notável em manutenção de software relaciona-se à necessidade de estimar esforço para atividade, conforme explica De Lucia *et al.* (2004). Com uma maneira eficaz de estimar o esforço, fica mais fácil aos tomadores de decisão argumentar sobre as melhores opções para a organização, contribuindo para a decisão de terceirizar ou não a atividade. Esse auxílio na tomada de decisão envolve considerações como:

- Reavaliar a eficiência de processos de manutenção;
- Tratar a manutenção ou reengenharia;
- Fundamentar as decisões tomadas;
- Planejar o orçamento de equipe e avaliar a rotatividade.

Definir quem fará a manutenção é um ponto que pode levar ao fracasso ou ao sucesso dos esforços em manter um software em funcionamento e adequado às necessidades.

Portanto, decidir entre manter uma equipe interna de manutenção ou submeter o software à manutenção por outra organização exige estudo cuidadoso, e isso se justifica pela simples lembrança de que a organização pode ter dependência enorme em relação a esse software, necessitando de seu correto e adequado funcionamento o tempo todo.

Aplicar à manutenção de software as técnicas, ferramentas, modelos de processos, etc., próprios do desenvolvimento de software não constitui uma decisão correta, pois manutenção e desenvolvimento apresentam características distintas, como afirma Polo *et al.* (2003).

Algumas razões que justificam essas características distintas são apresentadas pelo mesmo autor:

- O esforço despendido em tarefas de desenvolvimento e manutenção não é igual, constituindo uma prática comum nas organizações concentrar a maior parte do esforço humano disponível em codificação e tarefas de teste durante o desenvolvimento.
- Outro fator de diferença está ligado ao fato de que a manutenção tem mais similaridade com um serviço do que com desenvolvimento, o que acaba implicando na necessidade de não aplicar as mesmas estratégias de desenvolvimento em manutenção.

Essa última razão pode ainda ser apontada como fator que justifica o aumento no número de organizações que oferecem serviços de manutenção de software, e esse aumento estaria ainda ligado à alta demanda já existente.

De fato, a prestação de serviços de manutenção de software, mais do que uma situação nova, na verdade, trata-se de um setor já competitivo, como afirma Bhatt *et al.* (2006). O autor explica que nos primeiros momentos o que se viu foram clientes contratando organizações de software e pagando com base no número de engenheiros de software envolvidos nas atividades de manutenção. Atualmente, observa uma mudança nesse cenário, com os clientes contratando as organizações por um preço fixo, estipulado por um período determinado e com base em estimativas a respeito da atividade de manutenção que será necessária.

2.6 Histórico dos Problemas de Manutenção de Software

Edward Yourdon (Yourdon, 1992), considerado um dos nomes mais importantes da análise estruturada clássica, estimou, ainda no início dos anos 1990, que a manutenção de software viria a ser um dos problemas relevantes no futuro. Essa previsão foi feita com base em constatações da época, como a política de entregar em tempo ao usuário um sistema que funcionasse, não se preocupando com questões de manutenibilidade. Mais do que isso,

Yourdon afirmou que a postura da alta direção das organizações que dependiam de software seria a de atacar o problema apenas quando ele emergisse de fato.

Antes, porém, do que afirmou Yourdon, ainda na década de 1980, estudos referentes a problemas com manutenção de software já ocorriam, e um pioneiro e relevante trabalho foi realizado por Lientz e Swanson (1980), publicado na forma de um livro. Esse trabalho descreve os resultados obtidos com estudos no intuito de averiguar a forma como as organizações tratavam a questão de manutenção de software na época.

A pesquisa contou com duas fases. Na primeira, realizada com 69 organizações, foram levantadas as características da atividade de manutenção de software por elas realizada. Essa fase inicial obteve as seguintes conclusões:

- A manutenção de software existente consome uma média de 48% das horas anuais do pessoal de sistemas e programação;
- Aproximadamente 60% dos esforços de manutenção são dedicados a manutenções do tipo perfectivas. Dentre esse percentual, dois terços se referem a esforços de melhoria de software;
- Problemas de natureza gerencial em manutenção foram vistos como mais importantes do que aqueles de natureza técnica.

Com base nessas verificações, uma segunda fase foi conduzida na pesquisa, nesse caso envolvendo 487 organizações, o que abrangia instituições governamentais, bancos, transportes, mineração, educação e indústrias de manufatura em geral, localizadas nos Estados Unidos e Canadá. O intuito dessa segunda fase foi o de validar os resultados obtidos na primeira, buscando uma compreensão mais profunda e abrangente.

Para se obter esses resultados, os autores buscaram saber das organizações as características dos esforços de manutenção empregados em sistemas considerados de grande importância, que constituíam resultados de investimentos significativos.

A coleta de dados se deu por meio de questionários enviados para 2000 profissionais que ocupavam cargos de diretoria, gerência ou supervisão de departamentos de processamento de dados. Os questionários foram cuidadosamente elaborados, e enviados por meio de cartas às pessoas, contendo já um envelope selado para resposta. Cada questionário consistia de dezenove páginas, com perguntas que envolviam totais, porcentagens e respostas sim ou não para diversas características da organização e da manutenção por ela praticada.

Uma característica importante do questionário foi considerar a confiabilidade das informações fornecidas, reconhecendo também que nem sempre era fácil ao profissional entrevistado obter os dados para responder algumas questões. Para isso, após cada questão,

era fornecido um quadro (quadro 2.2), que visava avaliar o conhecimento que o entrevistado tinha a respeito do que acabava de responder.

QUADRO 2.2: VERIFICAÇÃO DO CONHECIMENTO SOBRE CADA ITEM RESPONDIDO

Marque a sentença apropriada A resposta anterior é:
a. Razoavelmente precisa, baseada em dados confiáveis.
b. Uma estimativa grosseira, baseado em dados mínimos.
c. Uma suposição, sem base em dados.

Do total de questionários enviados, somente 487 respostas foram obtidas, o que representou uma taxa de resposta menor que 25%.

Nessa segunda fase, os principais resultados obtidos estão resumidos a seguir:

- Mais de um terço dos sistemas descritos tinham sua manutenção realizada por apenas uma pessoa;
- A maioria dos profissionais alocados para manutenção tinha outras tarefas ao mesmo tempo;
- Do total de manutenções, cerca de 20% representavam manutenções corretivas, 25% manutenções adaptativas e mais de 50% manutenções perfectivas (isso está de acordo com aos resultados obtidos na primeira fase da pesquisa);
- Quanto mais velho era o sistema, maior era o esforço que deveria ser empregado em sua manutenção, e mais sujeita a organização estava em perder o conhecimento em torno desse sistema, devido à rotatividade dos profissionais.

Finalmente, os autores chegaram aos principais problemas de manutenção enfrentados pelas organizações na época:

(i) Baixa qualidade da documentação dos sistemas;
(ii) Necessidade constante dos usuários por melhorias e novas funcionalidades;
(iii) Falta de uma equipe de manutenção;
(iv) Falta de comprometimentos com cronogramas;
(v) Treinamento inadequado do pessoal de manutenção;
(vi) Rotatividade dos profissionais.

Conforme explicam os autores, desses seis problemas, três estão relacionados com os usuários do software, dois estão relacionados a restrições gerenciais e um trata de problema com documentação, representando uma questão de natureza mais técnica.

No geral, foi verificada uma predominância de problemas não-técnicos sobre problemas técnicos, o que reforça a necessidade de uma maior atenção a questões gerenciais

em manutenção de software.

Em 1986, outro estudo (Chapin, 1986), envolveu 250 gerentes de manutenção de software e buscou saber quais eram os principais problemas de manutenção nas equipes por eles lideradas. Dos resultados obtidos, foi estabelecida uma classificação em oito categorias. Entre os problemas apontados, a grande maioria referia-se a características do próprio software, como documentação insuficiente e código excessivamente complexo ou mal estruturado.

Em outra pesquisa (Dekleva, 1990), realizada pelo *Software Maintenance Association* (SMA), o intuito foi a obtenção de respostas para a seguinte pergunta: "*Quais os três principais problemas de manutenção de software no seu departamento de sistemas de informação?*", sendo que os resultados permitiram apontar questões ligadas principalmente a dificuldades de gerenciamento, características de desenvolvimento de software, administração de equipes e mudanças no ambiente externo ao software.

Em 1992, valendo-se de uma técnica de pesquisa iterativa denominada *delphi*, que leva a um consenso entre os entrevistados, Dekleva (1992), obteve uma lista de problemas de manutenção muito parecida com os apresentados por Lientz e Swanson, incluindo apenas o problema da dificuldade da organização em medir o desempenho de sua equipe de manutenção. Novamente, foi constatado que problemas de ordem gerencial são absolutamente dominantes no contexto de problemas de manutenção de software. Na figura 2.10, extraída do trabalho de Dekleva (1992), é mostrada a relação de causa identificada entre os fatores envolvidos na atividade de manutenção. As más características ou mal gerenciamento de um fator, prejudicam o fator correlacionado.

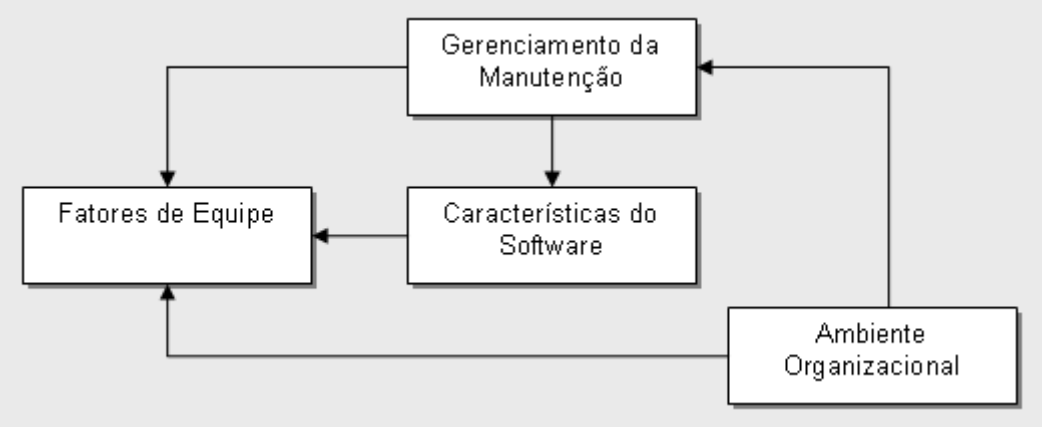

FIGURA 2.10: RELAÇÃO DE CAUSA ENTRE OS FATORES ENVOLVIDOS NA MANUTENÇÃO DE SOFTWARE

Existem ainda outros trabalhos com resultados semelhantes, como o de Dart *et al.* (2001), conduzido pelo *Software Engineering Institute* (SEI) junto a uma agência do governo dos Estados Unidos. Uma característica interessante apontada nesse estudo refere-se à

alegação dos profissionais encarregados de tarefas de manutenção de que acabam não tendo contato com ferramentas que representariam o *estado-da-arte* da tecnologia. Outro fator apontado refere-se ao fato de que as lições adquiridas com a experiência de manutenção em um determinado software não estavam sendo disseminadas para as outras equipes dentro da mesma organização.

2.7 Considerações Finais

A grande maioria dos estudos existentes na área de manutenção de software utiliza a metodologia de "pesquisar-e-transferir", no sentido de pesquisa fechada dentro do meio acadêmico, enquanto existe uma necessidade de utilizar a abordagem "indústria-como-laboratório", uma vez que é nas indústrias que a atividade de manutenção existe como atividade real e intensa no dia-a-dia, de acordo com a visão de Niessink (1999).

Nesse sentido, a manutenção de software deve ser considerada, assim como é a engenharia de software, com base em seu ambiente de aplicação, e essa abordagem representa a melhor fonte de evidências para sustentar e guiar as pesquisas de melhoria e sistematização da atividade.

CAPÍTULO

3

Estudo de Caso

3.1 Considerações Iniciais

Um estudo de caso foi conduzido com o objetivo de verificar os problemas de manutenção de software existentes em uma organização empenhada no desenvolvimento de software comercial. Essa organização mantém uma base de dados contendo registros históricos de manutenções efetuadas sobre um sistema de informação por ela desenvolvido e mantido. O resultado deste estudo de caso foi publicado em um artigo (Paduelli & Sanches, 2006), apresentado no *III Workshop de Manutenção de Software Moderna* (WMSWM'06), evento paralelo ao *V Simpósio Brasileiro de Qualidade de Software*. As subseções a seguir são baseadas no artigo.

3.2 Descrição da Organização

Criada em 1993, portanto com mais de 14 anos de existência, a organização estudada corresponde a uma *software-house*, que desenvolve e mantém alguns softwares voltados à área médica e de saúde. É composta por uma unidade na cidade de São Carlos - SP, responsável pelo desenvolvimento e manutenção, e outra unidade na cidade de São Paulo, dedicada à venda e consultoria.

O software estudado corresponde a um sistema de porte médio voltado à odontologia, mais precisamente ao gerenciamento de operadoras odontológicas, sendo, no entanto, um produto com características típicas à grande maioria dos softwares que se sujeitam a gerenciar

algum ramo de atividade comercial. São exemplos de recursos oferecidos pelo software: cadastro de clientes, cadastro de fornecedores, emissão de boletos bancários, controle financeiro, relatórios analíticos complexos, controle de estoques, suporte a multiusuário, controle de permissão de acesso a diferentes módulos, controle de fluxo de caixa, emissão de faturas, emissão de relatórios específicos para prestação de contas a órgãos de fiscalização do governo, integração com sistemas *web* etc.

Relativamente ao número de clientes, no momento da pesquisa, existiam 41 clientes (operadoras de odontologia), envolvendo desde organizações de pequeno porte, com até 5.000 associados cada, utilizando o software em cerca de 5 computadores, até empresas maiores, com mais de 50 máquinas em rede, utilizando o software de maneira simultânea e contando com até 150.000 associados. Normalmente essas empresas de maior porte são estruturadas por departamentos, e cada departamento utiliza um módulo específico do software.

3.3 Dinâmica de Manutenção

Neste tópico são abordadas as características da maneira de trabalhar da organização, expondo como registra, altera e entrega as suas manutenções.

Uma observação inicial se faz necessária, e se refere à possível confusão entre o que seria uma tarefa de desenvolvimento e o que seria uma atividade de manutenção, no contexto do software estudado. De uma forma objetiva, o software aqui referido encontra-se desenvolvido, uma vez que está em uso por muitos clientes, ou seja, é um *produto de software já entregue ao cliente*, conforme dita a definição da IEEE apontada no item 2.2.1. Partindo-se desse pressuposto, qualquer solicitação, seja por parte de clientes, seja por observação de algum profissional que atue sobre o software, será classificada como uma manutenção, ainda que exija criar um módulo novo ou refazer um já existente. Tais atividades se encaixam perfeitamente na classificação de tipos de manutenção apresentadas no item 2.2.4.

3.3.1 Registro de Manutenções

Os registros de manutenções são inseridos na base de dados de três formas distintas: (i) pelo próprio cliente, através do sistema de *help-desk* disponível no site da organização; (ii) pelos consultores da empresa, após visita ao cliente e levantamento de necessidades de manutenção; (iii) pelos próprios programadores, que podem identificar necessidades de manutenção à medida que "evoluem" o software. Uma interface específica baseada na *web* foi criada para a inserção desses registros na base.

Cada registro contém diversas informações, como por exemplo, qual o cliente que solicita, grau de prioridade, quem realizará a manutenção, tempo gasto previsto para a

atividade, tipo de manutenção, status da implementação, datas de inserção do chamado e data prevista para entrega, observações, descrição da solução adotada, entre outras.

Após o registro de um chamado, sua execução dependerá da análise de viabilidade, efetuada pelo responsável pelo produto. O procedimento adotado está resumido na figura 3.1.

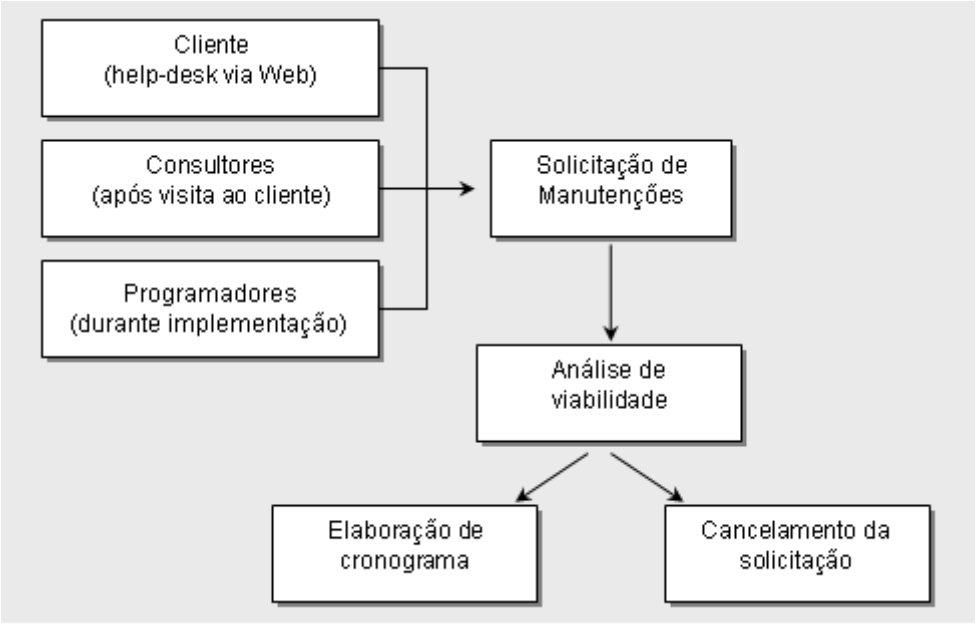

FIGURA 3.1: ETAPAS ENVOLVIDAS NA SOLICITAÇÃO DE MANUTENÇÕES

Eventualmente, solicitações de manutenção são consideradas inviáveis, sendo canceladas pelo responsável por essa avaliação. Uma vez cancelada, esse fato é informado ao respectivo cliente, podendo ser revisada a necessidade para que uma proposta de manutenção mais adequada seja registrada.

Uma característica do sistema de controle de chamados é a interação que oferece entre mantenedor/cliente. Essa interação ocorre da seguinte maneira: quando o cliente registra alguma necessidade de manutenção, automaticamente a equipe de suporte e o analista responsável pelos cronogramas e testes, recebem um *e-mail* informando que uma nova solicitação foi feita. Diversos *status* são atribuídos a uma solicitação, e a cada troca de status, o cliente pode receber um e-mail informando o andamento de sua solicitação. No quadro 3.1 estão representados os status possíveis para as solicitações, e em quais mudanças o cliente recebe um e-mail de notificação.

QUADRO 3.1: SITUAÇÕES POSSÍVEIS PARA OS CHAMADOS DE MANUTENÇÃO

Status do chamado	Cliente recebe *e-mail*
Pendente	Sim
Em execução	Sim
Cancelado	Sim
Em testes	Não
Em homologação	Sim
Concluído	Sim

Na situação "em testes", o cliente não recebe notificação, pois esse status é utilizado para controle interno. Uma solicitação "em testes" significa que está com a alteração no código-fonte efetuada e disponível para o analista responsável realizar os testes preliminares a fim de produzir a versão de homologação, que então é disponibilizada ao cliente. Somente quando a solicitação tiver sido testada e a versão de homologação estiver concluída e fornecida ao cliente, o analista responsável irá alterar o status para "em homologação", permitindo então o envio da notificação ao cliente. A situação "concluído" ocorre quando a versão de homologação finalmente for instalada no ambiente de produção do cliente.

Para o status de "cancelado", o cliente é informado apenas que sua solicitação foi negada, e que para maiores esclarecimentos ele deve entrar em contato com a equipe de suporte.

Por fim, destaca-se que o cliente pode, a qualquer momento, conferir os status de seus chamados, consultando sua área privativa no site da organização, que oferece acesso ao *help-desk* tanto para inclusão de novos chamados, como para acompanhamento por meio dos diferentes status.

3.3.2 Controle de Versões

Os clientes têm à sua disposição uma área privativa no site da organização, através do qual podem realizar o *download* de versões atualizadas do software, à medida que são disponibilizadas.

Essa disponibilização de versões ocorre mensalmente, podendo incluir rotinas de manutenção no banco de dados, quando necessárias, além dos componentes de software alterados. As atualizações de banco de dados são efetuadas automaticamente no momento em que o cliente abre o software pela primeira vez, após a atualização. Uma característica relevante é a de que o software sempre será o mesmo independente do cliente, ou seja, não existem versões distintas para clientes diferentes, todos utilizam uma versão única que pode ser configurada de acordo com necessidades específicas de cada cliente (módulo de *opções de sistema* do software).

Após a disponibilização de uma nova versão, o sistema de controle de chamados é consultado e as manutenções de software pendentes são consideradas, e então definidas quais entrarão na versão do mês seguinte. Essas manutenções são encaminhadas para os programadores (que consultam no sistema de chamados suas atividades pendentes), devendo seguir as datas programadas para entrega do módulo/arquivo alterado. As datas de entrega não são para o usuário final, mas sim para o analista responsável, que irá testá-las, e então disponibilizará uma versão de homologação do software. Essa versão é fornecida a um grupo

de clientes com os quais foi estabelecido um acordo para realização desses testes. Isso ocorre na última semana do mês que antecede aquele de disponibilização da nova versão.

A idéia de disponibilizar versões intermediárias a um ou mais clientes para testes reais em ambiente à parte do de produção, corresponde a uma das técnicas de teste apresentadas por Pressman (2005), intitulada *teste beta*. Segundo o autor, esse é o tipo de teste conduzido nas instalações do próprio cliente, sem acompanhamento do desenvolvedor, estando o cliente responsável por registrar problemas e informá-los em intervalos regulares.

Evidentemente entre a data de estabelecimento de cronograma para os chamados pendentes e a data de liberação da nova versão, surgem normalmente chamados novos e urgentes, que não podem esperar até próxima versão. No geral, procura-se evitar essa situação, mas quando ocorrem, versões intermediárias modificadas a partir da última versão oficial, somente com a alteração urgente ora solicitada, são disponibilizadas e apenas para o cliente que precisou da manutenção urgente.

Um padrão para nomenclatura das versões foi adotado, buscando atrelar a versão ao seu mês de lançamento, bem como informar o número da revisão atual da versão (caso a versão oficial tenha necessitado de alguma alteração emergencial). Na figura 3.2 é ilustrado um exemplo do padrão de numeração adotado.

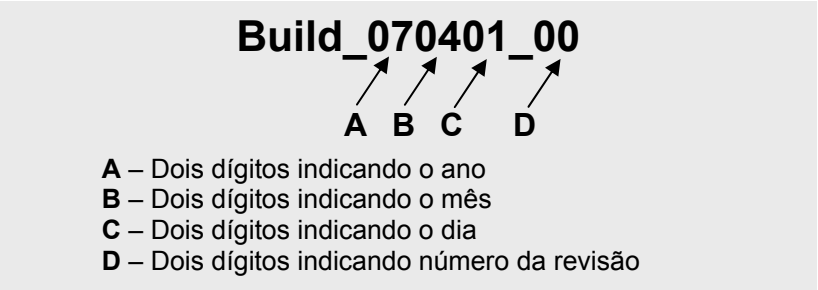

FIGURA 3.2: PADRÃO DE NUMERAÇÃO DE VERSÕES

Os dois últimos dígitos são incrementados em uma unidade, à medida que a versão for passando por manutenções emergenciais (aquelas que não podem ser acumuladas para a versão do mês seguinte). No exemplo da figura, a versão em questão é a oficial (revisão 00), do mês de abril de 2007.

A versão de homologação, uma vez testada pelo período de uma semana nos clientes selecionados, e com eventuais problemas detectados e corrigidos, é oficialmente convertida em uma versão final (“oficial”), e disponibilizada no site para todos os clientes, na primeira semana do mês seguinte ao de testes. Na figura 3.3 a seguir é ilustrado esse processo.

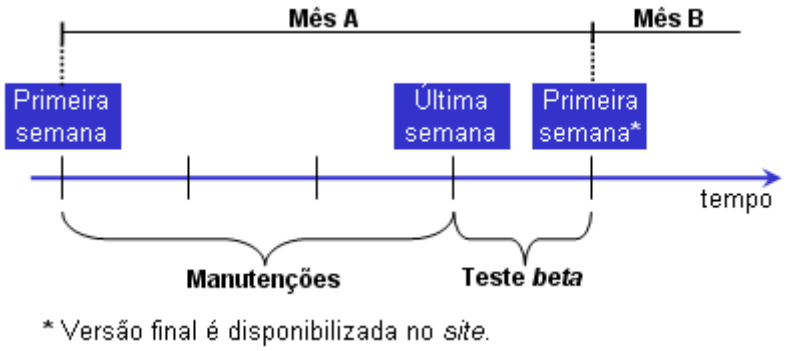

FIGURA 3.3: LINHA DO TEMPO PARA DISPONIBILIZAÇÃO DE VERSÕES

Durante a fase de manutenção no código-fonte, é utilizado um software de controle de versão (*Microsoft Source Safe*), que trabalha da seguinte forma: cada desenvolvedor que for alterar algum arquivo (que faz parte do projeto do software) realiza uma operação de *check-out*[11] desse arquivo do repositório (o qual armazena o projeto todo), ficando nesse momento "responsável" pelo arquivo, impedindo que qualquer outra pessoa possa realizar *check-out* do mesmo arquivo. Diz-se que o arquivo fica "travado" para o programador *x*. Somente quando a manutenção for concluída, o arquivo é submetido a um *check-in*[12], voltando para o repositório e ficando livre para uso por outro programador.

À medida que alterações vão sendo entregues e disponibilizadas no repositório, o analista responsável, realiza o *check-out* dos arquivos alterados e procede com testes preliminares. Uma vez que a data de liberação da nova versão para homologação se aproxima, esse analista realiza um *check-out* completo de todo o projeto do software, e realiza testes gerais, a fim de compor a versão de homologação para envio aos clientes pré-estabelecidos.

Durante a homologação, eventualmente esses clientes são acompanhados pessoalmente por algum consultor, ou remotamente via apoio por telefone, e-mail, *VNC* (software que permite conexão ao *desktop* remoto do cliente), entre outros.

Na medida em que problemas são verificados, se forem erros nas manutenções realizadas para a versão que será disponibilizada, esses erros são imediatamente corrigidos e a versão de homologação é atualizada. Se forem problemas novos (novas necessidades de alterações ainda não previstas), eles são registrados e então programados para disponibilização em versões posteriores do software.

3.4 Metodologia

A metodologia de pesquisa buscou, por um lado, estudar a base de dados na qual estão os registros de manutenção, e, por outro lado, entrevistar os profissionais responsáveis pela

[11] Ação de obter uma cópia do arquivo na máquina local (estando o repositório em um servidor).
[12] Ação inversa ao check-out. Devolver o arquivo alterado ao repositório, atualizando-o.

tarefa de manutenção. A forma como cada etapa foi conduzida está descrita a seguir.

3.4.1 Parte A – Base de Dados

Na primeira etapa, o foco dos esforços centrou-se em extrair da base de dados totais a respeito de diferentes características das manutenções efetuadas no software. Esses totais foram obtidos por meio de consultas *SQL* à base, sendo que as totalizações buscadas estão resumidas no quadro 3.2.

QUADRO 3.2: CARACTERÍSTICAS DE MANUTENÇÃO CONSIDERADAS E SEUS OBJETIVOS

Valor	Objetivo
Distribuição de solicitações mensalmente	Verificar algum padrão na distribuição
Tempo em dias para entrega de solicitações	Verificar o tempo de resposta ao cliente
Totais de solicitações por grupo de prioridade	Verificar a expectativa do usuário
Totais por tipo de manutenção *(vide item 2.2.4)*	Verificar distribuição entre os tipos
Tempo de desenvolvimento e manutenção	Verificar esforço necessário em cada fase

Finalmente, os números obtidos foram utilizados para construção de gráficos, mostrados no tópico 3.5.

3.4.2 Parte B – Questionário e Entrevista

A segunda etapa do levantamento de dados foi realizada com o auxílio de questionário e entrevistas.

O questionário elaborado (*apêndice A*) visou obter dos profissionais envolvidos com manutenção de software características de seu trabalho diário, coletando uma gama de informações para análise posterior junto com os dados da *Parte A*.

Após a elaboração do questionário foram realizadas entrevistas individuais com essas pessoas, procurando-se registrar todo comentário e informação relevante para o propósito da entrevista.

Detalhes da estratégia e da confecção do questionário podem ser vistas no *apêndice A*.

3.5 Análise dos Dados Coletados

Este tópico apresenta os resultados das duas formas utilizadas para obtenção de dados: a pesquisa na base de dados e as entrevistas. A base de dados, no momento da pesquisa, contava com mais de 3700 solicitações de manutenção, e as entrevistas foram realizadas com 8 pessoas ligadas diretamente à manutenção do software. Os subitens a seguir pormenorizam esses resultados.

3.5.1 Estatísticas sobre a Base de Dados

Partindo dos totais obtidos com a base de dados, algumas figuras puderam ser construídas.

Inicialmente, na figura 3.4, é apresentado o número de solicitações registradas mensalmente desde a implantação do sistema de registro de manutenções, em abril de 2004, até a data de realização deste trabalho.

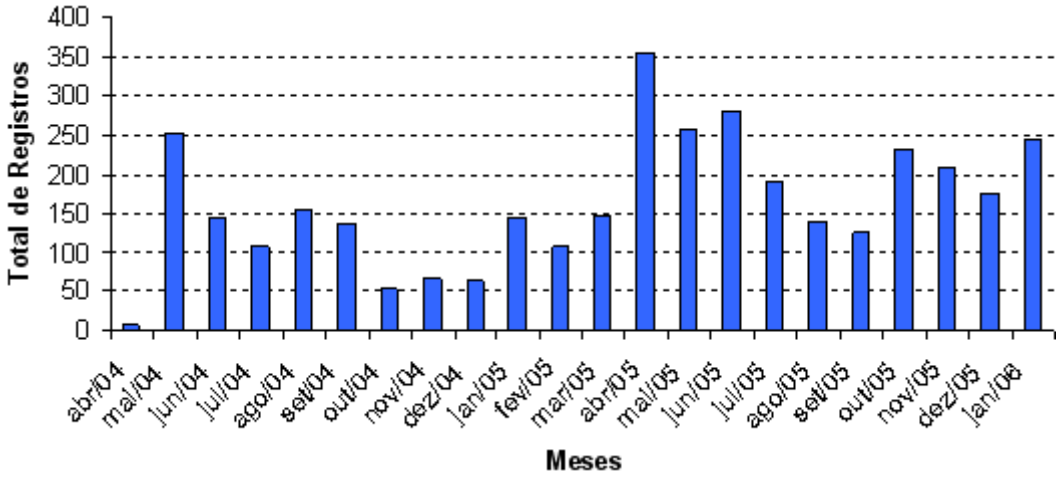

FIGURA 3.4: TOTAL MENSAL DE SOLICITAÇÕES DE MANUTENÇÃO

Nota-se que não existe uma constância no número de manutenções por mês, o que sugere uma imprevisibilidade nas necessidades de manutenção por parte dos clientes. Em abril de 2005 foi registrado o maior número de chamados de manutenção (355). Os dados representam uma média de 163,45 solicitações de manutenção por mês (desvio padrão de 83,7). Constatou-se ainda que a média de solicitações diária é de 8,59 novas requisições. Uma possível razão para a existência de épocas com maior incidência de pedidos de manutenção seria o fato de o número de solicitações entregues naquele mês ter sido maior, o que normalmente acarretaria mais *falhas colaterais*, ou seja, aquelas que acabam incidindo em outros módulos do software, apesar dos testes e da homologação em clientes.

Os registros de solicitações de manutenção incluem desde operações simples, que resultam em poucos dias para a implementação, até solicitações mais complexas, que podem levar semanas, e até meses, para serem concluídas. Solicitações desse porte normalmente exigem que o projeto de um ou mais módulos seja reestruturado, o que justifica o tempo gasto para entrega ao cliente.

A distribuição, em termos de dias gastos para a implementação de manutenções, é mostrada na figura 3.5.

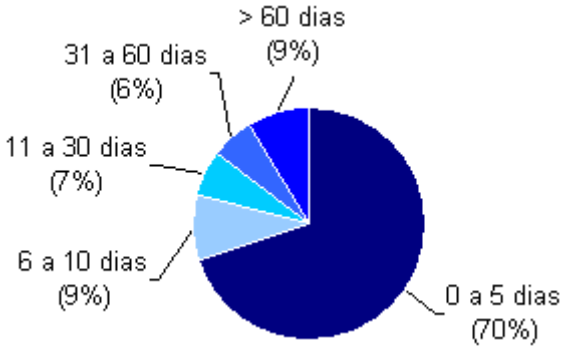

FIGURA 3.5: DIAS NECESSÁRIOS PARA A CONCLUSÃO DE MANUTENÇÕES

Um estudo do gráfico revela que as manutenções de menor grau de complexidade ocupam a maior parte do tempo dos mantenedores. No entanto, existem manutenções com complexidade suficiente para exigir mais de dois meses de trabalho.

Observou-se que a grande quantidade de manutenções de menor complexidade acaba por interferir no tempo de entrega das manutenções mais complexas, gerando atrasos. Isso ocorre em função da grande expectativa do cliente no sentido de obter respostas rápidas, o que força os mantenedores a liberar primeiro as manutenções mais simples. Na figura 3.6 é mostrado qual o grau de prioridade indicado pelo cliente ao fazer as suas solicitações de manutenção.

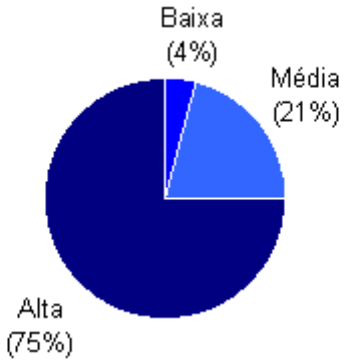

FIGURA 3.6: DISTRIBUIÇÃO DE PRIORIDADES DAS MANUTENÇÕES

Nota-se que o cliente busca respostas rápidas, considerando com prioridade elevada a maior parte dos chamados de manutenção efetuados.

Do ponto de vista dos quatro tipos de manutenção software (*adaptativa, perfectiva, corretiva* e *preventiva*), realizou-se a quantificação das manutenções efetuadas dentro desses tipos. Para esse levantamento, foi considerada a classificação das manutenções informada em cada registro feito na base de dados. O resultado obtido é apresentado na figura 3.7.

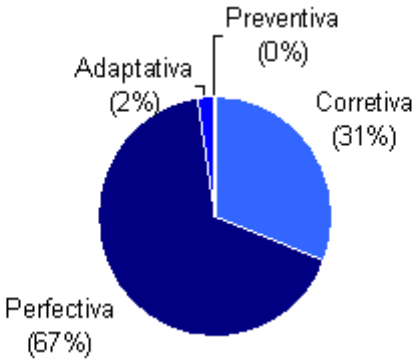

FIGURA 3.7: DISTRIBUIÇÃO DOS PROBLEMAS ENTRE OS TIPOS DE MANUTENÇÃO

Os valores apresentados apontam que o maior esforço da organização está em atender requisições de novas funcionalidades, mostrando enfoque em manutenção perfectiva. Uma constatação preocupante foi a de que não existe uma atenção em avaliar e prevenir problemas futuros no software (0% em manutenção preventiva), o que poderia ser feito por um processo de reengenharia. Fica evidente que o enfoque está em manter o software em funcionamento, adequando-o na medida em que as necessidades surgem, ou seja, seria uma postura de "aguardar acontecer" e não de "buscar prevenir".

É esperado, pela literatura, que o tempo empenhado em tarefas de manutenção exceda o tempo de desenvolvimento, e esse fato buscou ser comprovado pela comparação entre tempo gasto com o desenvolvimento da primeira versão entregue ao cliente e o tempo gasto com a manutenção até então. Para isso, elegeram-se três módulos representativos do software analisado (aqui referenciados por *módulo 1*, *módulo 2* e *módulo 3*), com o intuito de contabilizar o tempo gasto em desenvolvimento e o gasto em manutenção. O tempo de desenvolvimento inclui tempo gasto em projeto, codificação e testes.

Esse resultado é apresentado na figura 3.8, que considera os registros de manutenção efetuados até o momento de realização deste trabalho.

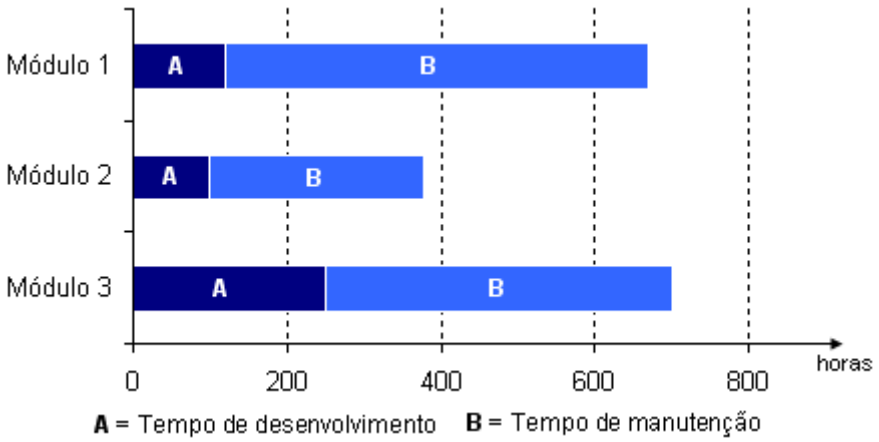

FIGURA 3.8: TEMPO CONSUMIDO EM DESENVOLVIMENTO *VERSUS* EM MANUTENÇÃO

De acordo com o mostrado na figura, percebe-se que a organização consumiu, nos três exemplos, mais de 60% do esforço empregado em atividades de manutenção, e esse percentual tende a subir com o passar do tempo.

3.5.2 Ambiente e Equipe de Manutenção

Durante a entrevista com os responsáveis pelas manutenções, constatou-se que não existe um procedimento padrão a ser seguido para a execução das manutenções. O que existe é uma preocupação em agendar datas de entrega para as solicitações cujos clientes responsáveis insistem em respostas mais rápidas, o que freqüentemente acaba inviabilizando a metodologia de disponibilização de atualizações descrita anteriormente. Casos de manutenções de urgência, quando ocorrem, são priorizados em detrimento das demais.

Observou-se também que o código-fonte modificado não é documentado de maneira adequada, sendo muitas vezes atribuídas somente pequenas notas, ou o número do chamado que resultou na modificação de um trecho de código. Esse número de chamado está relacionado com o registro de necessidade de manutenção efetuado pelo cliente, podendo um futuro mantenedor consultar do que se tratou a manutenção, buscando pelo número do chamado informado no código-fonte no sistema de registro de manutenções.

Os profissionais encarregados de tarefas de manutenção relataram ainda que nem sempre o cliente compreende totalmente o que está solicitando, muitas vezes gerando discussões entre empresa/cliente, o que está relacionado com os problemas de comunicação entre usuário e desenvolvedor. Muitas vezes o cliente acredita que uma determinada manutenção será suficiente para adequar o software à sua nova necessidade de negócio, quando na verdade não é suficiente. Essa dificuldade de compreensão do software pelo próprio cliente acaba por gerar situações de desgaste na relação entre empresa-cliente.

Prazo de entrega é sempre um problema quando não existe um processo para conduzir a manutenção. Esse foi outro fato verificado, uma vez que nem sempre a manutenção, com data agendada e informada ao cliente, pode ser entregue no prazo combinado, geralmente em função de re-trabalho em manutenções já realizadas, ou mudanças de prioridades de entrega.

Em parte esse atraso decorre do fato de o mantenedor ser também a pessoa que desenvolve, de forma que a tarefa de desenvolvimento precisa mesclar-se com a de manutenção, contribuindo para a diminuição do desempenho do mantenedor. Isso se agrava quando testes em uma nova funcionalidade estão sendo feitos em paralelo a alguma correção solicitada pelo cliente.

Outro ponto importante refere-se ao baixo interesse dos profissionais envolvidos em tarefas de manutenção, fato que pode ser comprovado pelo interesse desses profissionais em

deixar a atividade de manutenção para dedicar-se ao desenvolvimento. Esse interesse foi manifestado por todos os profissionais de manutenção entrevistados.

A política de testes utilizada, embora prevista como uma alternativa pela literatura, nesse caso específico não vinha apresentando os resultados esperados. Nem sempre o cliente tinha tempo e boa vontade para avaliar uma versão de testes em ambiente separado daquele de produção, culminando com problemas nas manutenções efetuadas surgindo quando a versão *oficial* já estava no site e em uso pela maior parte dos clientes em seus ambientes de produção.

3.6 Problemas Identificados

Após obter e analisar os dados, foi possível montar o quadro 3.3, que envolve os principais problemas de manutenção identificados na organização.

QUADRO 3.3: PROBLEMAS DE MANUTENÇÃO DE SOFTWARE – LISTA PARCIAL

Problemas Gerenciais	Ausência de um processo de manutenção de software
	Grande expectativa dos usuários
	Elevada rotatividade de membros e funções dentro da equipe
	Sobrecarga de tarefas
	Estimativa de prazo não condizente com a complexidade do software
	Baixa motivação entre profissionais de manutenção
	Ausência de manutenção preventiva
	Falhas de comunicação com o usuário
	Atrasos na entrega
Problemas Técnicos	Registro inexistente ou superficial de manutenções anteriores
	Ausência de um ambiente computacional específico para manutenção
	Validação insuficiente de manutenções efetuadas
	Documentação insuficiente ou superficial
	Falta de compreensão do software e suas estruturas

Os problemas, como pode ser visto no quadro, foram classificados de acordo com sua natureza em problemas gerenciais e técnicos. No item a seguir são tecidas considerações gerais sobre o resultado obtido.

3.7 Conclusões Parciais

A classificação dos problemas de manutenção identificados na organização quanto a sua natureza, permite observar que aqueles ligados a questões gerenciais estão mais presentes do

que os de caráter mais técnico. De fato, constatou-se que conciliar dificuldades técnicas de manutenção com os anseios dos usuários não é uma tarefa simples, e infelizmente não vem obtendo o sucesso desejado.

A ausência de um processo de manutenção de software formal, como dita normas de engenharia de software, não seria de todo condenável se o processo interno criado pela própria organização funcionasse de maneira satisfatória. O que se observou foi que a pressão gerada pelos clientes muitas vezes prejudica o cumprimento correto da seqüência de passos estipulada pela própria organização para tratar a manutenção de software.

Por fim, a inexistência de manutenção preventiva é um fato que precisa ser analisado em conjunto com os objetivos de negócio da organização e com seus projetos em relação à vida útil do software que mantém.

CAPÍTULO

4

Problemas de Manutenção de Software

4.1 Considerações Iniciais

Este capítulo apresenta uma comparação entre os problemas de manutenção de software atuais, identificados pelo estudo de caso, e aqueles registrados no passado, buscando inferir algum padrão de evolução nos mesmos. Além disso, estabelece um rol de problemas de manutenção de software, publicados por diferentes autores, na forma de uma lista de dificuldades inerentes à atividade de manutenção, para fins de análise nos próximos capítulos. Procurou-se estabelecer uma denominação direta e simples para os problemas, facilitando a leitura, o entendimento e posteriores referências.

4.2 Problemas do Passado *versus* Problemas Atuais

Os problemas identificados no estudo de caso permitem o surgimento da seguinte questão: *Os problemas de manutenção de software de hoje guardam relação com aqueles verificados no passado?* A resposta a essa questão fornecerá indícios da maneira como a mudança da tecnologia pode estar ou não influenciando tanto o agravamento como o surgimento de problemas.

Partindo desse princípio, constatou-se que existe um alinhamento mais ou menos regular entre os problemas do passado com os atuais, o que permitiu a construção do quadro 4.1, no qual é mostrado esse fato por meio de um paralelo entre os problemas apontados no

tópico 2.6, obtidos por Lientz e Swanson (1980), e os identificados no estudo de caso. O quadro associa os problemas relacionando-os de acordo com a sua natureza.

QUADRO 4.1: COMPARAÇÃO ENTRE PROBLEMAS DE MANUTENÇÃO DE SOFTWARE

Problemas apontados por Lientz e Swanson (1980)	Problemas atuais
Baixa qualidade da documentação dos sistemas	*- Registro inexistente ou superficial de manutenções anteriores;* *- Documentação insuficiente ou superficial.*
Necessidade constante dos usuários por melhorias e novas funcionalidades	*- Grande expectativa dos usuários;* *- Falhas de comunicação com o usuário.*
Falta de uma equipe de manutenção	*- Ausência de um processo de manutenção de software;* *- Sobrecarga de tarefas;* *- Ausência de manutenção preventiva;* *- Ausência de um ambiente computacional específico para manutenção.*
Falta de comprometimentos com cronogramas	*- Estimativa de prazo não condizente com a complexidade do software;* *- Atrasos na entrega.*
Treinamento inadequado do pessoal de manutenção	*- Baixa motivação entre profissionais de manutenção;* *- Validação insuficiente de manutenções efetuadas;* *- Falta de compreensão do software e suas estruturas.*
Rotatividade dos profissionais	*- Elevada rotatividade de membros e funções dentro da equipe.*

Embora não se estabeleça uma equivalência exata, esse quadro mostra uma semelhança muito grande entre a natureza dos problemas obtidos em cada levantamento. Esse fato é favorável aos objetivos deste trabalho, pois mostra que existe uma relativa estabilidade entre os problemas.

Essa constatação permite que uma sistematização do estudo possa ser estabelecida. Se os problemas atuais fossem muito diferentes dos verificados no passado, então seria difícil decidir sobre quais deles buscar técnicas e procedimentos para a redução dos esforços empregados, porém esse fato não se verificou. Isso permite concluir que é possível estabelecer estudos e respostas sobre problemas de manutenção de software que não se tornarão totalmente obsoletos quando uma nova tecnologia surgir.

4.3 Problemas de Manutenção Publicados

A relação de problemas de manutenção de software identificada pode ser visualizada no quadro 4.2, construído de forma a mostrar os problemas e suas respectivas fontes. Em virtude da lista relativamente extensa, optou-se por dividi-la em categorias para promover melhor organização e facilidade de leitura. A numeração seqüencial atribuída a cada problema tem o intuito único de facilitar as referências posteriores, não guardando qualquer outro significado. Faz-se necessário dizer que a separação dos problemas a seguir não é rigorosamente exata, uma vez que muitos deles poderiam ser classificados em mais de uma categoria.

QUADRO 4.2: PROBLEMAS DE MANUTENÇÃO DE SOFTWARE – LISTA GERAL

Fatores de Gerência	
Problema	**Referências**
01. Falta de comprometimento com cronogramas	Lientz e Swanson (1980); Estudo de Caso
02. Treinamento inadequado da equipe de manutenção	Lientz e Swanson (1980); Dekleva (1992); Dart *et al.* (2001); Estudo de Caso
03. Ausência de um profissional responsável exclusivamente ao controle de configuração de software	Dart *et al.* (2001)
04. Visão organizacional diferenciada para a equipe de desenvolvimento e para a equipe de manutenção	Dart *et al.* (2001)
05. Dificuldades de comunicação entre a equipe de manutenção e a própria organização	Dart *et al.* (2001)
06. Software desenvolvido é visto pela gerência como não construído para a manutenção	Dart *et al.* (2001)
07. Mantenedores desconhecem planos da organização com relação à equipe de manutenção	Dart *et al.* (2001)
08. Contratação de temporários para auxílio na manutenção leva ao menor controle sobre a atividade	Dart *et al.* (2001)
09. Experiências com manutenções anteriores não são disseminadas dentro da própria organização e entre novos membros da equipe	Dart *et al.* (2001); Estudo de Caso
10. Dificuldade na medição do desempenho da equipe de manutenção	Dekleva (1992)
11. Ausência de adoção de padrões, metodologias e procedimentos de manutenção	Dekleva (1992); Estudo de Caso
12. Falta de suporte da gerência	Dekleva (1992)
13. Sobrecarga de tarefas	Estudo de Caso
14. Ausência de manutenção preventiva	Estudo de Caso
15. Estimativa de prazo não condizente com a complexidade do software	Estudo de Caso
Fatores de Infra-estrutura	
Problema	**Referências**
16. Necessidade de suporte automatizado à gerência de configuração de software	Dart *et al.* (2001)
17. Ferramentas para manutenção precisam ser diferentes daquelas de desenvolvimento	Dart *et al.* (2001)
18. Falta de recursos tecnológicos adequados	Dart *et al.* (2001); Estudo de Caso
19. Estagnação tecnológica no ambiente de trabalho	Dart *et al.* (2001)
Fatores humanos – equipe	
Problema	**Referências**
20. Falta de uma equipe de manutenção	Lientz e Swanson (1980)
21. Elevada rotatividade dos profissionais	Lientz e Swanson (1980); Dekleva (1992); Estudo de Caso

22. Mantenedores lamentam por não possuírem contato com estado-da-arte da tecnologia	Dart *et al.* (2001)
23. Preferência dos profissionais por trabalhos de desenvolvimento	Dart *et al.* (2001)
24. Manutenção de software é vista como uma tarefa não prestigiosa	Dekleva (1992); Dart *et al.* (2001); Estudo de Caso
25. Falhas de comunicação com o usuário	Estudo de Caso
26. Métodos inadequados de testes	Dekleva (1992); Estudo de Caso
27. Documentação insuficiente ou superficial (software alterado)	Lientz e Swanson (1980); Dekleva (1992) ; Dart *et al.* (2001); Estudo de Caso
Fatores humanos – cliente	
Problema	**Referências**
28. Necessidade constante dos usuários por melhorias e novas funcionalidades	Lientz e Swanson (1980); Estudo de Caso
29. Falta de compreensão dos usuários a respeito de suas reais necessidades de software	Estudo de Caso
30. Mudanças freqüentes de prioridades (clientes)	Dekleva (1992)
Fatores de software	
Problema	**Referências**
31. Baixa qualidade da documentação dos sistemas (software original)	Lientz e Swanson (1980); Dekleva (1992); Dart *et al.* (2001); Estudo de Caso
32. Má qualidade do código-fonte original	Dekleva (1992); Dart *et al*. (2001)
33. Necessidades de integração com softwares incompatíveis	Dart *et al.* (2001)
34. Plataformas heterogêneas dificultam a definição de ferramentas adequadas	Dart *et al.* (2001)

A relação anterior, embora seguramente não exaustiva, apresenta de forma geral os principais problemas que recaem sobre a atividade de manutenção de software. Outros problemas que eventualmente existam e não foram citados, são demasiados específicos de certo domínio, ou fortemente relacionados a algum dos problemas já citados. Dessa forma, acredita-se que a listagem anterior seja representativa das dificuldades mais importantes em manutenção de software, não se prendendo a nenhum domínio específico de software.

4.4 Relacionamento entre os Problemas de Manutenção e os Grupos de Processos da Norma ISO/IEC 12207

Conforme já apresentado neste trabalho, a norma ISO/IEC 12207 representa um padrão internacionalmente adotado que engloba processos para o ciclo de vida de software. Neste tópico, a norma é utilizada para criar uma relação de causa e conseqüência com os problemas de manutenção de software apresentados no rol anterior.

Essa relação se estabelece da seguinte maneira: a norma apresenta grupos de processos (que por sua vez englobam diversos outros processos), que visam atingir alguma fase do ciclo de vida de software, guiando-o com as melhores práticas. Assim, a não verificação de um grupo de processos, acarretará problemas para a respectiva fase (e conseqüências para as futuras).

Dessa maneira, foi possível a construção do quadro 4.3, no qual são relacionados os grupos de processos da norma, com os problemas de manutenção, procurando estabelecer uma relação de causa (a não verificação do respectivo grupo) e conseqüência (problema derivado). Para essa distribuição, verificaram-se, para cada grupo, seus processos e as determinações de suas respectivas tarefas. O não atendimento a uma das tarefas do processo, automaticamente relaciona o problema ao processo, e em um nível mais alto, ao grupo ao qual esse processo pertence.

QUADRO 4.3: GRUPOS DE PROCESSOS NÃO OBSERVADOS E AS RESPECTIVAS CONSEQÜÊNCIAS

	Problemas
Categoria Processos Fundamentais	
Grupo de Processos de Aquisição	
Grupo de Processos de Fornecimento	
Grupo de Processos de Engenharia	(11), (26), (29), (33), (34)
Grupo de Processos de Operação	(25)
Categoria Processos Organizacionais	
Grupo de Processos de Gerência	(01), (04), (05), (06), (07), (08), (12), (13), (14), (15), (20), (21), (24), (30), (31), (32)
Grupo de Processos de Melhoria de Processo	
Grupo de Processos de Recursos e Infra-estrutura	(02), (09), (10), (17), (18), (19), (22), (23)
Grupo de Processos de Reuso	
Categoria Processos de Apoio	
Grupo de Processos de Gerência de Configuração	(03), (16), (27), (28)
Grupo de Processos de Garantia de Qualidade	

A distribuição gráfica das informações anteriores está representada na figura 4.1.

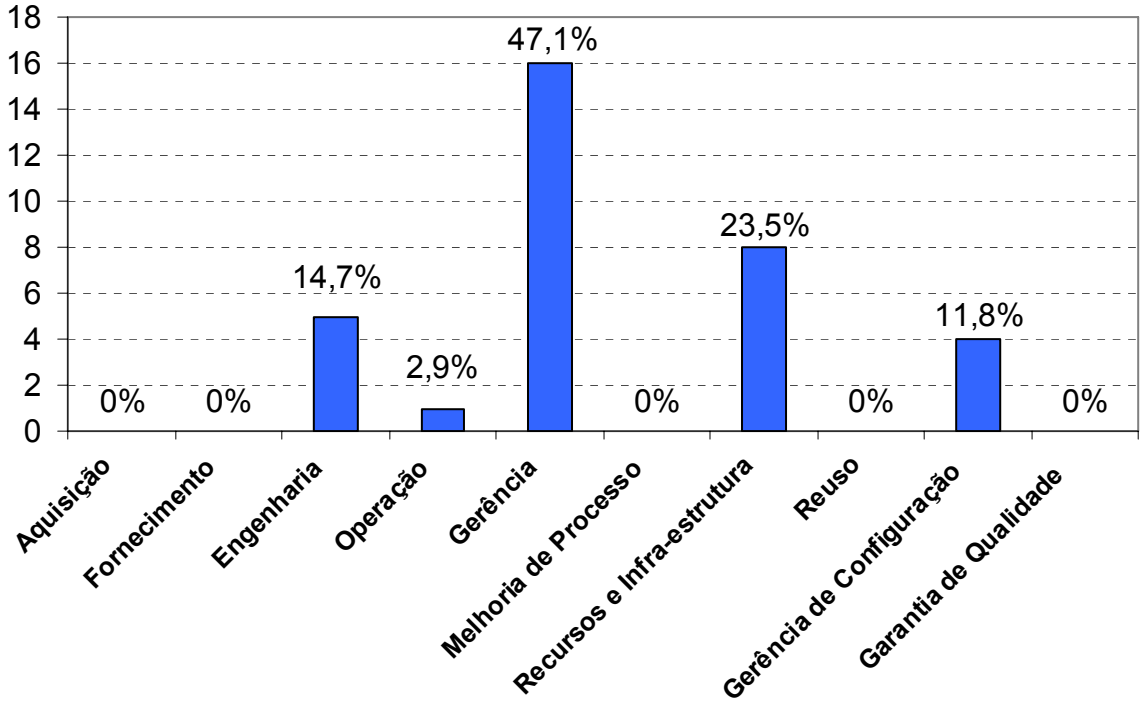

FIGURA 4.1: DISTRIBUIÇÃO DOS PROBLEMAS NOS GRUPOS DE PROCESSOS

Percebe-se, pela figura anterior, que existe uma grande concentração de problemas relacionados principalmente com fatores de ordem gerencial, ligados à interação com pessoas e processos, respondendo por 47,1% dos problemas.

Dentre os dez grupos de processos, cinco ficaram sem nenhum problema relacionado. Os grupos de *aquisição* e de *fornecimento* apresentam uma razão clara: estão relacionados à criação de um produto de software novo, portanto fortemente ligadas ao desenvolvimento, o que não foi considerado. O grupo de *melhoria de processo* pode ser visto sob duas perspectivas, que consideram a existência ou não de processos implantados na organização. Na primeira das visões, assumindo que a organização possua processos implantados, os problemas de manutenção poderiam ser associados a falhas nesses processos, alegando-se que seriam essas falhas as responsáveis diretamente pela maioria dos problemas de manutenção. Isso, no entanto, não corresponde à realidade, já que normalmente os processos não existem nas organizações.

Em outra visão do grupo de melhoria de processo, agora considerando que a organização não possua processos implantados, seria inadequado valer-se desse fato para se associar a esse grupo praticamente todos os problemas de manutenção, sob a alegação de que existem como resultado da ausência de processos adequados para tratar a manutenção de software. A inadequação desse posicionamento se justifica pelo fato de que é possível estabelecer uma melhor relação dos problemas com outros processos da norma, muitas vezes de forma direta, valendo-se para isso das tarefas estipuladas para cada processo. O não

cumprimento de uma tarefa específica, automaticamente relaciona o processo correspondente com algum dos problemas já apresentados. Assim, pode-se concluir que a norma é capaz de fornecer indícios de quais processos específicos inexistem ou são falhos para a ocorrência de cada problema, permitindo então distribuí-los de forma mais precisa e não associá-los todos ao grupo de melhoria de processo sob a alegação genérica de sua ausência na organização.

O próximo grupo a considerar é o de *reuso*, que também parece ter uma razão clara para ter permanecido sem problemas associados. Não é simples tratar o reuso de artefatos de software nem mesmo durante o desenvolvimento. O que dizer então da atividade de manutenção, normalmente com necessidades imprevisíveis e muitas vezes inéditas para o contexto do software. Dificilmente será possível valer-se de soluções prontas, o que torna o grupo de processos de reuso com pouca ou nenhuma importância para a redução de problemas de manutenção de software.

Finalmente, o grupo de processos de *garantia de qualidade* também não teve problemas relacionados em razão de existirem outros processos da norma com tarefas específicas não atendidas que melhor se encaixam como fonte do problema. Por exemplo, considere o processo de *garantia de qualidade* (os processos do grupo são: *garantia de qualidade, verificação, validação, revisão conjunta, auditoria* e *avaliação de produto*). O objetivo principal desse processo é prover garantias de que os produtos de trabalho e processos obedeçam aos planos predefinidos para o software. Note, no entanto, que os problemas de manutenção de software normalmente são frutos da ausência de planejamento ou do planejamento falho da manutenção, sendo que o correto planejamento e execução são tarefas reservadas a outro processo (processo de *Manutenção de Software e Sistema*). Assim, a existência de processos com tarefas mais específicas acabou por evitar que problemas de manutenção fossem relacionados a processos do grupo de garantia de qualidade. No entanto, é fácil perceber que a palavra *qualidade* é extremamente conhecida em diversos contextos, mesmo aqueles não relacionados a software, chegando a ser quase intuitivo pensar que algo feito com qualidade resultará em menos problemas ou maior satisfação do usuário. Esse raciocínio leva à conclusão de que, embora sem problemas associados diretamente, os processos do grupo de garantia de qualidade devem ser observados como recurso de apoio à execução com qualidade das tarefas de manutenção como um todo.

4.5 Considerações Finais

Embora tenha se produzido neste capítulo uma relação de problemas com base em apontamentos feitos por outros autores, e também pelo estudo de caso realizado, entende-se que não se trata de uma relação fixa e exaustiva. Em diferentes abordagens futuras, nomes

diferentes podem ser utilizados para se referirem às mesmas dificuldades. É razoável pensar ainda que novos problemas surjam, constituindo novas categorias de problemas, distintas das aqui definidas.

Esse fato, porém, não invalida ou torna de todo desatualizado o que foi exposto, já que a literatura, e também o estudo de caso apresentado, apontam uma relativa estabilidade entre os problemas mais relevantes ao longo do tempo.

É admissível, também, que o surgimento de novas técnicas de programação, novas tecnologias e metodologias, influenciem a forma como as organizações trabalham, e nesse caso poderiam reduzir os problemas aqui apresentados.

Outra observação necessária se faz com relação aos problemas e os tamanhos de software sobre os quais incidem. O que ocorre, de fato, é que os problemas apresentados incidem de maneira geral sobre qualquer tamanho de software, variando apenas a ênfase que o problema representa para cada tamanho de software.

Por fim, a relação estabelecida entre os problemas e os grupos da norma, permite uma visão inicial das características do processo de ciclo de vida de software que devem ser tomadas com maior atenção para o contexto de minimização de problemas de manutenção de software. No capítulo seguinte, esse relacionamento é retomado e considerado com maiores detalhes.

CAPÍTULO

5

Alternativas de Redução dos Problemas de Manutenção de Software

5.1 Considerações Iniciais

As alternativas de respostas para os problemas de manutenção de software apresentadas neste capítulo estão embasadas em três fontes distintas: na norma ISO/IEC 12207, nas soluções encontradas na organização analisada no estudo de caso, e nas propostas de outros autores que se dedicaram ao assunto.

Acredita-se que a norma seja um excelente veículo para conduzir uma organização de software pelas melhores práticas, já que seu conteúdo é resultado de um esforço fundamentado no que poderia se chamar estado-da-arte da engenharia de software. Outra consideração importante diz respeito à maneira como a norma se apresenta. Ela não é um modelo fechado que diz *como* fazer e manter software da melhor forma, mas sim, mostra *o que* deve ser considerado para que isso seja alcançado. Dessa forma, a norma funciona como um guia para ser seguido por organizações interessadas em tratar sistematicamente seu processo de software.

O estudo de caso apresentado anteriormente é estendido neste capítulo, expondo agora práticas adotadas pela organização que trouxeram resultados satisfatórios para a prevenção ou contenção de problemas de manutenção de software em seu ambiente de trabalho.

Finalmente, os mesmos autores que identificaram problemas de manutenção de software em seus trabalhos também contribuem para fornecer propostas de redução desses problemas.

5.2 Extensão do Estudo de Caso

Neste tópico é apresentada a extensão do estudo de caso mostrado no capítulo 3.

Essa extensão foi possível graças à constatação de que a organização estava em busca freqüente por melhorias em seu ambiente de trabalho, e em sua capacidade de resposta satisfatória aos clientes, que vinham crescendo em número, sem, no entanto, valer-se da adoção de um processo extenso e sistemático como o sugerido pela norma ISO/IEC 12207.

O objetivo foi verificar quais mudanças a organização vinha adotando no seu dia-a-dia que estivessem contribuindo positivamente para a diminuição de seus problemas de manutenção de software, ou ainda, o que os profissionais empenhados nessa atividade achavam que deveria ser feito para que essa redução ocorresse.

5.2.1 Metodologia

A metodologia aplicada na extensão do estudo de caso também foi baseada em questionários e entrevistas (*apêndice A*).

O intuito principal desses questionários foi o de servirem para a abertura de uma discussão na qual exemplos de soluções eficientes fossem citados pelos entrevistados. Como o objetivo era que a organização mostrasse quais atitudes vinha adotando para tratar de maneira satisfatória sua atividade de manutenção, não havia como estipular questões prévias, por isso o questionário reduzido funcionou somente como recurso para início de entrevista. De fato, as soluções citadas, bem como observações e todas as informações relevantes para o contexto da entrevista, foram registradas pelo entrevistador em documento a parte e posteriormente compiladas.

5.2.2 Soluções Identificadas

Os resultados obtidos foram organizados e utilizados para elaborar os itens a seguir, que apresentam soluções apontadas pelos entrevistados como favoráveis para a atividade de manutenção de software, contribuindo para reduzir um ou mais dos problemas conhecidos.

- *Melhoria em treinamento:* uma prática relevante adotada pela organização diz respeito a treinamento. A solução considerada foi a de verificar entre os profissionais interessados, quais eram as ferramentas de software que gostariam de um aprofundamento em conhecimentos, por meio de certificações. A partir desse levantamento, a organização se comprometeu em comprar os livros necessários à preparação para alguns exames de certificação oficial. Como exemplo, tem-se uma das modalidades de certificação de *DBA SQL Server* da Microsoft. O gerenciador de banco de dados *SQL Server* corresponde ao utilizado pela organização, e sua

configuração correta, seja em termos de desempenho ou de funcionalidades, é de fundamental importância. Assim, a organização comprou os livros necessários à preparação, e que normalmente têm um custo relativamente alto, se comprometendo ainda a reembolsar o valor pago para realização do exame, caso o profissional fosse aprovado. No momento das entrevistas, algumas pessoas já haviam conseguido aprovação no módulo inicial da certificação. Esse conhecimento adicional em banco de dados é de relevância para manutenção, já que o software da empresa tem muitas partes implementadas na forma de *stored procedures*, e muitas das manutenções são efetuadas nesses componentes.

- *Gerenciamento do conhecimento:* ligado diretamente ao problema da rotatividade elevada, a organização vinha se empenhando em sistematizar os problemas de manutenção, principalmente os problemas de atendimento ao cliente via suporte e *help-desk*, montando uma base de conhecimentos onde fosse possível consultar as soluções possíveis para um problema freqüente. Para isso, uma ferramenta *web* de colaboração on-line foi instituída, e nela registrado, na forma de perguntas e respostas, as maneiras de abordar inúmeros problemas. A atitude gerencial inicial para a composição das primeiras questões dessa base foi a obtenção, junto à equipe de suporte, de uma lista de problemas mais comuns. O passo seguinte foi reunir consultores da empresa que trabalham na cidade de São Paulo, para em reuniões conjuntas entre equipe de suporte e consultores, elaborar as primeiras respostas à lista de problemas de suporte. Tais questões foram posteriormente inseridas na ferramenta web, de forma que qualquer pessoa dentro da organização pudesse ter acesso e editar seu conteúdo, como forma de complementação.

- *Melhoria na documentação oferecida ao cliente*: da mesma forma que foi criado um repositório de problemas e soluções para a própria organização, também se criou algo semelhante em relação aos clientes. Com o uso da mesma ferramenta web, uma interface de consulta foi disponibilizada aos clientes, permitindo a obtenção de respostas a problemas comuns que o suporte vinha reiteradas vezes atendendo, com uma linguagem padronizada e uniforme. Questões de caráter técnico, que envolvesse informações de código-fonte ou configuração de banco de dados, não ficavam disponíveis aos clientes (essa classificação de o que o cliente podia consultar e o que não podia, era feita principalmente pela equipe de suporte, mas de uma forma geral todos podiam interferir).

- *Padronização no fluxo de informações:* buscando corrigir problemas de comunicação com os usuários, muitas vezes por motivos de o usuário informar partes de seu problema para pessoas diferentes do suporte, o que tornava a compreensão do todo dificultada e muitas vezes gerava insatisfação desse cliente, foi uniformizada a maneira como esse tipo de informação fluiria na organização. O fluxo estabelecido foi o seguinte: todo problema de software deveria ser primeiramente cadastrado no sistema de *help-desk*, para então algum contato via telefone ou e-mail poder ser estabelecido. O sistema de help-desk possuía recurso de interação entre usuário e suporte, registrando-se ali (e não por *e-mail*, como antes) todas as interações e detalhes do problema em questão. Com isso evitou-se a perda de informações passadas pelo cliente, o que impactava na manutenção, pois chamados de alteração eram estabelecidos sem o devido detalhamento, o que causava má especificação de requisitos de alteração, gerando conflitos do próprio suporte com a equipe de manutenção.

- *Melhoria em cronogramas*: um dos grandes problemas em manutenção é a questão de prazos e o cumprimento de cronogramas estabelecidos junto aos clientes. A organização, visando minimizar esse tipo de problema, adotou a postura de assumir de fato o cronograma junto ao cliente, garantindo isso da seguinte forma: após serem agendadas as datas de entrega de manutenções, cada profissional responsável pela implementação deveria informar, diariamente, ao coordenador o andamento de suas atividades. Ao menor sinal de contratempos que fossem levar a um não-cumprimento da data de entrega estipulada, o coordenador indicava alguém ou ele próprio assumia a responsabilidade de auxiliar na manutenção para que o prazo fosse cumprido. Essa postura forçou a contratação de novos programadores.

- *Divisão de tarefas*: o sistema de registro de chamados de manutenções passou a abrigar mais um campo de informação: o nível de complexidade da manutenção, estipulado pelo coordenador. Isso permitiu um melhor controle do problema da sobrecarga de tarefas, o que seguramente foi auxiliado pela contratação de novos programadores.

- *Melhoria em testes:* essa foi uma das mudanças significativas para a redução do número de problemas de software que acabavam no ambiente de produção dos clientes. A medida adotada foi a de insistir no uso de *teste beta* (testes pelo cliente, em seu ambiente), porém a maneira de negociação foi alterada. A organização se comprometeu a preparar todo ambiente paralelo ao de produção, necessário para

efetuar os testes, sempre que uma versão de testes fosse disponibilizada, ficando a cargo do cliente apenas proceder com os testes, sem necessidade de qualquer configuração de software. O cliente deveria oferecer apenas os recursos de hardware (computador, espaço em disco rígido para replicação do banco de dados de produção etc.). Além disso, a liberação da versão inicial de testes passou a ocorrer somente após revisões mais rígidas realizadas internamente na organização por uma pessoa dedicada exclusivamente a essa tarefa.

Por fim, verificou-se que a questão da documentação interna das manutenções era uma das preocupações da organização, porém ainda sem solução satisfatória implementada, portanto não citada aqui.

Embora não se tenha adotado um processo de manutenção complexo e rígido, a lista de modificações internas citada contribuiu para a diminuição daqueles problemas mais simples e que causavam grande impacto negativo junto aos clientes.

No entanto, uma nova dificuldade parece ter surgido, e também já estava sendo combatido pela gerência através de maiores exigências, que era justamente o fato de os procedimentos novos adotados nem sempre estarem sendo seguidos, como por exemplo, a necessidade de sempre registrar na base de conhecimento as soluções adotadas para problemas que ainda não constavam dessa base.

5.2.3 Considerações Gerais

As soluções encontradas pela organização pesquisada não representam o resultado do emprego de alguma norma ou processo bem definido, como dito anteriormente.

Foi justamente esse fato que permitiu utilizar essas soluções como alternativas para abordar os problemas de manutenção de software. Isso porque é importante que seja mostrado tanto o aspecto rigorosamente formal proposto pela norma ISO/IEC 12207, como também alternativas de soluções. É valioso destacar que alternativas como as aqui apresentadas podem até mesmo serem mais viáveis do que a adoção da própria norma, dependendo do porte da organização e de sua estruturação.

O fundamental é que as organizações de software se empenhem em compreender as características e dificuldades de si próprias, buscando continuamente por falhas e possíveis formas de tratá-las, o que muitas vezes pode ocorrer pela simples mudança de atitudes e controle mais rígido de tarefas. O uso de ferramentas de software também é um ponto importante, e nesse exemplo houve o uso de ferramentas para auxílio no tratamento de informações.

5.3 Soluções com base na Norma ISO/IEC 12207

O relacionamento entre os problemas de manutenção de software mostrado no capítulo anterior pode ser estendido para especificar também a qual processo do grupo o problema de manutenção de software se relaciona (lembre-se que essa relação é devida ao *não cumprimento* de tarefas do processo correspondente).

Para criar esse relacionamento, o quadro 5.1 foi construído. Nele são apresentados todos os grupos e processos da norma, mostrando em quais desses processos os problemas de manutenção de software se encaixam, de forma individual.

QUADRO 5.1: PROCESSOS DA NORMA ISO/IEC 12207 E PROBLEMAS DE MANUTENÇÃO DE SOFTWARE

	Problema
Categoria Processos Fundamentais	
Grupo de Processos de Aquisição	
Preparação da Aquisição	
Seleção do Fornecedor	
Acordo Contratual	
Monitoramento do Fornecedor	
Aceitação pelo Cliente	
Grupo de Processos de Fornecimento	
Proposta do Fornecedor	
Liberação de Produto	
Apoio para Aceitação de Produto	
Grupo de Processos de Engenharia	
Elicitação de Requisitos	**29.** Falta de compreensão dos usuários a respeito de suas reais necessidades de software
Análise de Requisitos de Sistema	
Projeto de Arquitetura de Sistema	
Análise de Requisitos de Software	**33.** Necessidades de integração com softwares incompatíveis **34.** Plataformas heterogêneas dificultam a definição de ferramentas adequadas
Projeto de Software	
Construção de Software	
Integração de Software	
Teste de Software	**26.** Métodos inadequados de testes
Integração de Sistema	
Teste de Sistema	
Instalação de Software	

Manutenção de Software e Sistema	**11.** Ausência de adoção de padrões, metodologias e procedimentos de manutenção
Grupo de Processos de Operação	
Operação	
Suporte ao Cliente	**25.** Falhas de comunicação com o usuário
Categoria Processos Organizacionais	
Grupo de Processos de Gerência	
Alinhamento Organizacional	**07.** Mantenedores desconhecem planos da organização com relação à equipe de manutenção
Gerência Organizacional	**04.** Visão organizacional diferenciada para a equipe de desenvolvimento e para a equipe de manutenção **05.** Dificuldades de comunicação entre a equipe de manutenção e a própria organização **20.** Falta de uma equipe de manutenção **24.** Manutenção de software é vista como uma tarefa não prestigiosa
Gerência de Projeto	**01.** Falta de comprometimento com cronogramas **06.** Software desenvolvido é visto pela gerência como não construído para a manutenção **08.** Contratação de temporários para auxílio na manutenção leva ao menor controle sobre a atividade **12.** Falta de suporte da gerência **13.** Sobrecarga de tarefas **15.** Estimativa de prazo não condizente com a complexidade do software
Gerência de Qualidade	**14.** Ausência de manutenção preventiva
Gerência de Riscos	**21.** Elevada rotatividade dos profissionais **30.** Mudanças freqüentes de prioridades (clientes) **31.** Baixa qualidade da documentação dos sistemas (software original) **32.** Má qualidade do código-fonte original
Medição	
Grupo de Processos de Melhoria de Processo	
Estabelecimento de Processo	
Avaliação de Processo	
Melhoria de Processo	
Grupo de Processos de Recursos e Infra-estrutura	
Gerência de Recursos Humanos	**10.** Dificuldade na medição do desempenho da equipe de manutenção **23.** Preferência dos profissionais por trabalhos de desenvolvimento
Treinamento	**02.** Treinamento inadequado da equipe de manutenção **22.** Mantenedores lamentam por não possuírem contato com estado-da-arte da tecnologia
Gerência de Conhecimento	**09.** Experiências com manutenções anteriores não são disseminadas dentro da própria organização e entre novos membros da equipe

Infra-estrutura [13]	**17.** Ferramentas para manutenção precisam ser diferentes daquelas de desenvolvimento **18.** Falta de recursos tecnológicos adequados **19.** Estagnação tecnológica no ambiente de trabalho
Grupo de Processos de Reuso	
Gerência de Ativos	
Gerência de Programa de Reuso	
Engenharia de Domínio	
Categoria Processos de Apoio	
Grupo de Processos de Gerência de Configuração	
Documentação	**27.** Documentação insuficiente ou superficial (software alterado)
Gerência de Configuração	**03.** Ausência de um profissional responsável exclusivamente ao controle de configuração de software **16.** Necessidade de suporte automatizado à gerência de configuração de software
Gerência de Resolução de Problemas	
Gerência de Solicitações de Mudança	**28.** Necessidade constante dos usuários por melhorias e novas funcionalidades
Grupo de Processos de Garantia de Qualidade	
Garantia de Qualidade	
Verificação	
Validação	
Revisão Conjunta	
Auditoria	
Avaliação de Produto	

Percebe-se que não são todos os processos da norma que quando não observados contribuem para os problemas de manutenção de software. Porém, aqueles que estão diretamente ligados ao surgimento dos problemas, precisam de um entendimento mais detalhado, explorando-se também as tarefas previstas para cada um.

No tópico seguinte, cada processo que apresentou associação com problemas de manutenção é descrito em um quadro, que apresenta os resultados esperados de uma implementação com sucesso do processo e suas respectivas tarefas, de acordo com o que expõe a norma. As tarefas podem ser consideradas como as soluções possíveis para os problemas atrelados a cada processo. Imediatamente após cada quadro, são tecidas observações ou comentários a respeito do processo no contexto de manutenção de software.

[13] De acordo com a norma, infra-estrutura pode incluir hardware, software, métodos, ferramentas, técnicas, padrões e outras facilidades para desenvolvimento, operação e manutenção.

5.3.1 Grupo de Processos de Engenharia

Um total de 14,7% dos problemas de manutenção de software foi classificado em processos pertencentes ao grupo de processos de engenharia.

No quadro 5.2 é apresentado o primeiro processo considerado desse grupo, o processo de *elicitação de requisitos.*

QUADRO 5.2: ENGENHARIA: PROCESSO DE *ELICITAÇÃO DE REQUISITOS*

Grupo de Processos de Engenharia
Processo: Elicitação de Requisitos
Objetivos: Reunião, processamento e monitoramento da evolução das necessidades do cliente e de seus requisitos através da vida do produto e/ou serviço.
Resultados esperados de uma implementação com sucesso: (i) Contínua comunicação com o cliente; (ii) Ajuste dos requisitos definidos com o cliente; (iii) Estabelecimento de um mecanismo para avaliar e recepcionar mudanças de requisitos do cliente, resultado de mudanças de necessidades do mesmo; (iv) Elaboração de um mecanismo para continuamente monitorar as necessidades do cliente; (v) Criação de um meio que permita ao cliente consultar o *status* de suas requisições; (vi) Gerenciamento de melhorias derivadas de troca de tecnologias ou mudanças de necessidades do cliente.
Tarefas necessárias
Obtenção dos requisitos do cliente: consiste em obter e definir quais são os requisitos do cliente, por meio de solicitações diretas e pela especificação de entradas para o seu sistema. Isso pode ser obtido por revisão dos propósitos de negócio do cliente, verificação do ambiente operacional e de hardware e ainda outros documentos.
Entendimento das expectativas do cliente: tem por objetivo assegurar que tanto fornecedor quanto o cliente entenderam cada requisito da mesma maneira. Revisões com os clientes para um entendimento melhor das suas necessidades e expectativas devem ser conduzidas.
Concordância com os requisitos: busca obter formalmente junto ao cliente e seus interessados a concordância total com relação aos requisitos estipulados.
Estabelecimento final dos requisitos: essa tarefa produz um documento formal considerado o estabelecimento final dos requisitos para uso na fase de projeto e para confronto com futuras mudanças de necessidades do cliente.
Gerenciamento de mudanças de requisitos do cliente: consiste em gerenciar todas as mudanças do cliente em relação ao documento original de requisitos, garantindo que melhorias resultantes de mudança de tecnologias ou de necessidades do cliente possam ser identificadas e aquelas que sejam afetadas por mudanças possam ser analisadas em relação a impacto e riscos, permitindo a tomada de ações.
Estabelecimento de mecanismo de comunicação com cliente: consiste em prover meios para que o cliente possa ser alertado do status e disposição das suas mudanças de requisitos.

Comentário sobre o processo:

A clara definição e entendimento comum dos requisitos é uma das características mais relevantes para manutenção de software, porém, as sugestões da norma podem ser demasiadas formais para a grande massa de pequenas manutenções normalmente existentes. Exigir que o cliente documentasse todos os pedidos da maneira como a norma expõe pode não ser adequado para a grande maioria dos pequenos ajustes, o que poderia levar a um descontentamento do cliente, o qual consideraria a organização que contratou extremamente burocrática.

A solução mais evidente, a exemplo do que fazem muitas organizações, é o uso de sistemas *on-line* de *help-desk*, reservando apenas às manutenções mais complexas um procedimento formal de documento de requisitos.

Por meio de um help-desk, é possível documentar tanto as solicitações do cliente, como as interações da equipe de suporte/manutenção, atendendo também ao requisito da norma de criação de um mecanismo para consulta ao status da uma determinada solicitação pelo cliente. Essa é uma forma de evitar outro problema importante que surge com a obtenção dos requisitos: a perda de informações. Se os requisitos forem passados via telefone, e-mail, fax, etc., a experiência mostra que a perda dessas informações é comum e leva a um descontentamento geral, com conseqüente perda de credibilidade da organização frente ao cliente.

Na seqüência, o próximo processo a ser considerado é o de *análise de requisitos de software*. No quadro 5.3 é exposto esse processo.

QUADRO 5.3: ENGENHARIA: PROCESSO DE *ANÁLISE DE REQUISITOS DE SOFTWARE*

Grupo de Processos de Engenharia
Processo: Análise de Requisitos de Software
Objetivos: Estabelecimento dos requisitos de elementos de software para o sistema.
Resultados esperados de uma implementação com sucesso: (i) Definição dos requisitos necessários aos elementos de software do sistema, bem como suas interfaces; (ii) Requisitos de software são analisados quanto à corretude e testabilidade; (iii) O impacto dos requisitos de software no ambiente operacional é compreendido; (iv) Consistência e rastreabilidade são estabelecidas entre os requisitos de software e os requisitos de sistema; (v) Definição da priorização de implementação para os requisitos de software;

(vi)	Aprovação e atualização dos requisitos de software sempre que necessário;
(vii)	Avaliação das mudanças de requisitos de software com relação a custos, cronograma e impacto técnico;
(viii)	Composição final dos requisitos de software e comunicação a todas as partes afetadas.
Tarefas necessárias	
Especificação de requisitos de software: definir, analisar e priorizar requisitos de software operacionais e não-operacionais do sistema, registrando no documento de requisitos de software.	
Determinação do impacto no ambiente operacional: determinar as interfaces entre os requisitos de software e outros elementos do ambiente operacional e o impacto que esses requisitos terão.	
Desenvolvimento de um critério de teste de software: usar os requisitos de software para definir critérios de aceitação para os testes do produto de software, que devem mostrar conformidade com esses requisitos de software.	
Garantia de consistência: garantir a consistência entre a análise dos requisitos de sistema e a análise dos requisitos de software. Essa consistência deve ser mantida por meio do estabelecimento e manutenção da rastreabilidade entre os requisitos de sistema e os requisitos de software.	
Avaliar e atualizar os requisitos de software: avaliar os requisitos junto ao cliente, aprovando as mudanças e atualizando os requisitos de especificação de software.	
Comunicar os requisitos de software: estabelecer mecanismos de comunicação para disseminação dos requisitos de software, atualizando os mesmos para todas as partes que forem precisar.	

Comentário sobre o processo:

No contexto de manutenção de software, os requisitos são freqüentemente de inclusão de novas funcionalidades ou de adaptações nas já existentes. Assim, supondo que o atual software esteja com relativa estabilidade, modificações precisam ser cuidadosamente consideradas, já que não é verdade que uma alteração no software sempre garantirá que ele estará melhor adequado à realidade da organização. Problemas como efeito cascata de alterações, que geram problemas em outros módulos do sistema, precisam ser considerados. A análise de requisitos em muitas situações pode encontrar outros meios de atender à necessidade do cliente sem exigir mudança no software (o software normalmente tem recursos que são desconhecidos ao cliente). Nesse ponto é necessário um serviço de suporte eficiente da organização que desenvolve e mantém o produto, para que possam ser percebidas as situações nas quais recursos pouco conhecidos do software podem ser utilizados, evitando-se assim manutenções desnecessárias.

O próximo processo considerado é o de *teste de software*, conforme é mostrado no quadro 5.4.

QUADRO 5.4: ENGENHARIA: PROCESSO DE *TESTE DE SOFTWARE*

Grupo de Processos de Engenharia
Processo: Teste de Software
Objetivos: Confirmação de que o produto de software integrado está de acordo com os requisitos previamente definidos.
Resultados esperados de uma implementação com sucesso: (i) Critério de integração de software é definido para demonstrar a conformidade com os requisitos de software; (ii) O software integrado é verificado usando-se o critério criado; (iii) Os resultados dos testes são registrados; (iv) Uma estratégia de regressão é definida para ser usada para re-testar o software quando uma alteração em seus itens for realizada.
Tarefas necessárias
Desenvolvimento de testes para o produto de software integrado: descrever os testes a serem efetuados no produto de software integrado, indicando os requisitos sendo checados, entrada de dados e critério de verificação. O conjunto de testes deve mostrar conformidade com os requisitos de software.
Teste do produto de software integrado: teste do produto de software utilizando-se o critério estabelecido, registrando os resultados. Atualização da documentação quando necessário.
Teste de regressão do produto de software integrado: desenvolver uma estratégia de testes de regressão para re-testar o software integrado. Se mudanças forem feitas nos itens de software, proceder com os testes de regressão conforme critério estabelecido.

Comentário sobre o processo:

Testes de manutenções efetuadas, principalmente em sistemas legados (normalmente softwares de grande porte e com estrutura e documentação comprometidas), representam uma dificuldade grande para a disponibilização de um produto estável. Esse fato torna o planejamento de testes fundamental para evitar que ocorram muitas idas e voltas do software ao ambiente de produção do cliente, em razão de erros.

Quando a manutenção ocorre em um software que ainda não constitui um sistema legado, como aqueles em pleno processo de crescimento e agregação de novas funcionalidades, a importância dos testes é acentuada principalmente no quesito de integração. Testar um pequeno módulo, ou uma pequena funcionalidade alterada, pode ser simples e realizado pelo próprio programador, porém é no momento da integração que o efeito cascata de uma alteração com problemas pode gerar situações futuras indesejadas. Garantir que o produto de software integrado funcione por completo é o mínimo esperado pelos clientes. Se a organização falha continuamente nesse ponto, a desconfiança do cliente no software é logo verificada.

Finalmente, o último processo considerado no grupo de processos de engenharia é o de *manutenção de software e sistema*. No quadro 5.5 esse processo é descrito.

QUADRO 5.5: ENGENHARIA: PROCESSO DE *MANUTENÇÃO DE SOFTWARE E SISTEMA*

Grupo de Processos de Engenharia
Processo: Manutenção de Software e Sistema
Objetivos: Modificação de um sistema/produto de software após entrega ao cliente, para corrigir falhas, melhorar o desempenho ou outros atributos, ou ainda adaptá-lo para um ambiente modificado.
Resultados esperados de uma implementação com sucesso: (i) Uma estratégia de manutenção é desenvolvida para gerenciar modificações, migrações e retirada de produtos de acordo com a estratégia adotada; (ii) Identificação do impacto das mudanças no sistema/software existente para a organização, operações ou interfaces; (iii) Documentação de sistema/software afetada é atualizada conforme necessário; (iv) Produtos modificados são desenvolvidos com testes associados que demonstram que outros requisitos não estão comprometidos; (v) Produtos atualizados são migrados para o ambiente do cliente; (vi) Via requisição, produtos são retirados de uso de uma maneira controlada que minimize os distúrbios para o cliente; (vii) O sistema/software modificado é comunicado a todas as partes afetadas.
Tarefas necessárias
Desenvolvimento de uma estratégia de manutenção: desenvolver a estratégia para manutenção, migração e retirada de produtos de maneira consistente com os requisitos de manutenção, comunicando a estratégia e possíveis políticas de garantias.
Análise de problemas do usuário e mudanças: analisar os problemas do usuário, suas requisições e mudanças necessárias, avaliando o impacto de diferentes opções para modificar o sistema ou software existente, interfaces e requisitos. Documentar a opção escolhida.
Implementar e testar modificações: determinar quais produtos precisam ser mudados. Implementar, testar e documentar as modificações selecionadas, demonstrando que o sistema e requisitos de software, bem como a integridade, não serão comprometidos com a alteração.
Atualizar o sistema do cliente: migrar o sistema e software atualizados para o ambiente do cliente. Prover conforme apropriado: notificação dos planos de migração e atividades, operação paralela do antigo e do novo sistema e necessidades de treinamento. Proceder com uma atividade de pós-operação para revisar e assegurar o impacto da modificação.
Retirada do produto de software: seguindo aprovação, retirar o sistema obsoleto do ambiente do usuário, provendo conforme apropriado: notificação dos planos de retirada e atividades, operação paralela com a troca de sistemas, conversão de dados para o sistema novo, arquivamento do sistema e dados, treinamento de usuários e suporte.
Comunicar modificações: estabelecer mecanismos de comunicação para prover a disseminação das modificações de sistema/software a todas as partes afetadas.

Comentário sobre o processo:

A previsão, pela norma, de um processo específico para manutenção de software, vem diretamente ao encontro dos propósitos deste trabalho. Uma política de manutenção focada no entendimento da dinâmica da atividade significa economia de tempo e redução de custos.

As tarefas da norma buscam conscientizar a organização de que a manutenção não pode ser uma tarefa subsidiária dentro da empresa, sendo acionada ocasionalmente quando necessária, sem um planejamento e acompanhamento maior. A razão disso é a sua característica de atividade inevitável. É praticamente impossível imaginar um software que funcionará em uma empresa dinâmica sem necessidade de modificações e de inclusão de novas funcionalidades.

Uma observação importante sobre esse processo refere-se novamente à necessidade de avaliar a real necessidade do cliente, mantendo para isso um meio eficiente de comunicação e um conhecimento sobre o negócio dele (ainda que pontual).

5.3.2 Grupo de Processos de Operação

Um total de 2,9% dos problemas de manutenção de software foi classificado em processos pertencentes ao grupo de processos de operação, sendo somente um processo desse grupo referenciado. O processo referenciado foi o de *suporte ao cliente*, conforme apresentado no quadro 5.6.

QUADRO 5.6: OPERAÇÃO: PROCESSO DE *SUPORTE AO CLIENTE*

Grupo de Processos de Operação
Processo: Suporte ao Cliente
Objetivos: Estabelecimento e manutenção de um aceitável nível de serviços através de assistência e consultoria ao cliente para dar apoio ao uso efetivo do produto.
Resultados esperados de uma implementação com sucesso: (i) Identificação e monitoração dos serviços necessários para o suporte ao cliente; (ii) Avaliação da satisfação do cliente tanto com relação ao suporte oferecido como com relação ao produto; (iii) Suporte operacional é oferecido através do tratamento das questões do cliente, incluindo suas novas requisições; (iv) As necessidades de suporte do cliente são conhecidas por meio do oferecimento de serviços apropriados.
Tarefas necessárias
Estabelecer suporte do produto: estabelecer um serviço por meio do qual o cliente possa apresentar problemas e questões encontradas no uso do produto e receber ajuda para solucioná-los.

Prover treinamento dos usuários: oferecer treinamento e documentação ao usuário, de forma que o produto possa ser eficientemente utilizado.
Monitorar o desempenho: monitorar o desempenho operacional do produto, de forma a estar ciente dos problemas que podem impactar no nível de serviço.
Determinar o nível se satisfação do cliente: determinar o nível de satisfação do cliente com os produtos e serviços oferecidos.
Comparar a satisfação do cliente: comparar e monitorar o nível obtido de satisfação do cliente com as indústrias do setor e, quando possível, com os concorrentes diretos.

Comentário sobre o processo:

Prover suporte efetivo ao produto de software tornou-se um fator básico de competitividade. Diante dessa realidade, e considerando que muitos dos problemas com operação de software estão relacionados ao desconhecimento do usuário em relação à maneira correta de utilizar o software, planejar e administrar a forma como o serviço será oferecido representa característica básica de uma organização que comercializa e mantém produto de software.

No contexto de manutenção de software, o papel do suporte ao produto tem grande importância no sentido de evitar necessidades de mudanças. Pessoal de suporte bem treinado pode ser extremamente útil para oferecer alternativas ao usuário, evitando manutenções freqüentemente desnecessárias.

5.3.3 Grupo de Processos de Gerência

Um total de 47,1% dos problemas de manutenção de software foi relacionado aos processos do grupo de gerência. Esse foi o grupo que mais apresentou problemas associados, e isso vem ao encontro dos resultados encontrados no estudo de caso apresentado, que aproxima os problemas de questões mais de ordem gerencial do que de ordem técnica.

O primeiro processo desse grupo é o de *alinhamento organizacional*, conforme descrito no quadro 5.7.

QUADRO 5.7: GERÊNCIA: PROCESSO DE *ALINHAMENTO ORGANIZACIONAL*

Grupo de Processos de Gerência
Processo: Alinhamento Organizacional
Objetivos: Capacitação dos processos de software necessários à organização para prover produtos de software e serviços, de forma que fiquem consistentes com seus objetivos de negócios.
Resultados esperados de uma implementação com sucesso: (i) Definição dos objetivos de negócios da organização;

(ii)	Definição de seu *framework* de processos, que inclui uma série de processos de software necessários para alcançar os objetivos de negócios da organização;
(iii)	Elaboração de uma estratégia para definição, implementação e melhoria de processos;
(iv)	A missão da organização, seus valores principais, visões e objetivos são apresentados a todos os funcionários;
(v)	Comum visão da organização pelos seus funcionários, compartilhando o entendimento dos objetivos de negócios para que possam realizar suas funções eficientemente;
(vi)	Cada indivíduo na organização deve entender seu papel para alcançar os objetivos de negócios e avaliar se está apto a cumprir esse papel.
Tarefas necessárias	
Desenvolver uma visão estratégica: desenvolver uma visão estratégica para a organização, que seja capaz de identificar os objetivos de negócios e sua relação com funções de engenharia de software e de sistemas.	
Definir o framework de processos: identificar os processos que precisam ser executados para que seja possível alcançar os objetivos de negócios.	
Prover comprometimento gerencial: prover suporte gerencial para a aplicação e melhoria estratégica de processos.	
Comunicar a visão e objetivos: explicar a visão e objetivos estratégicos da organização para todos os indivíduos que trabalhem para ela, utilizando mecanismos adequados de comunicação e gerenciamento.	
Garantir o compartilhamento de uma visão comum: garantir que cada indivíduo da organização compreenda a visão comum e se comprometa a cumprir sua função individual com eficiência.	
Permitir participação ativa: permitir que cada indivíduo contribua para se alcançar os objetivos de negócios e apresentar iniciativas de melhoria de processos.	

Comentário sobre o processo:

A postura gerencial, como em todos os demais processos envolvidos no ciclo de vida de software, tem papel fundamental na atividade de manutenção. Um único funcionário de desenvolvimento preocupado em produzir software com alta manutenibilidade dificilmente conseguirá apoio dos demais programadores e analistas, sem o devido suporte gerencial. Desenvolver software com alta manutenibilidade é algo de natureza estratégica e, portanto, deve partir da consciência dos altos administradores, como uma postura global da organização.

Para complementar o papel da gerência no contexto de manutenção de software, as decisões afetas ao planejamento e aos objetivos de longo prazo sobre o software não devem ser mantidas de forma fechada entre os membros da cúpula da organização, mas, pelo contrário, devem ser compartilhadas com as suas equipes, deixando claro ao desenvolvedor/mantenedor a sua importância e contribuição para o propósito maior do

software em que trabalha. Visão diferenciada para equipes de desenvolvimento e de manutenção é o principal ponto a se evitar.

O próximo processo do grupo de gerência é o de *gerência organizacional*, apresentado no quadro 5.8.

QUADRO 5.8: GERÊNCIA: PROCESSO DE *GERÊNCIA ORGANIZACIONAL*

Grupo de Processos de Gerência
Processo: Gerência Organizacional
Objetivos: Estabelecimento e controle de práticas de gerenciamento de software durante a execução dos processos necessários ao provimento de produtos/serviços de software.
Resultados esperados de uma implementação com sucesso: (i) Investimento em infra-estrutura de gerenciamento adequada; (ii) Identificação das melhores práticas para apoiar a implementação do gerenciamento efetivo da organização e de seus projetos; (iii) Constituição de uma base para avaliação da conquista ou não dos objetivos de negócios da organização.
Tarefas necessárias
Identificar a estrutura de gerenciamento: identificar a infra-estrutura de gerenciamento apropriada para a execução de práticas de gerenciamento de software que forem consistentes com os objetivos de negócios da organização.
Investir em infra-estrutura de gerenciamento de software: investir na infra-estrutura apropriada de gerenciamento de software, segundo o escopo amplo da organização.
Identificar as práticas de gerenciamento de software: identificar efetivas práticas de gerenciamento de software para apoiar a organização e a implementação de software.
Avaliar a efetividade: avaliar a efetividade das práticas de gerenciamento de software implementadas para alcançar os objetivos de negócios da organização.
Prover suporte para a adoção das melhores práticas: utilizar abordagens de incentivo e infra-estrutura de gerenciamento de software para apoiar a implementação de práticas de gerenciamento, registrando-se as melhores para disseminação na organização, como parte de seus bens.

Comentário sobre o processo:

Citar a adoção de práticas modernas de gerência organizacional, bem como o uso de ferramentas e meios de monitoração do desempenho global tanto das pessoas como dos demais recursos de infra-estrutura, parece ser desnecessário, uma vez que isso já é a realidade de organizações competitivas. Porém, não é desnecessário afirmar que mesmo em um ambiente controlado e adaptado, problemas surgem, e no contexto de manutenção de software

eles podem representar um entrave para a efetivação de outros processos, como o alinhamento organizacional citado no processo anterior.

Os problemas de manutenção de software relacionados com o processo de gerência organizacional são totalmente de natureza humana e relacionados com a tradicional postura de valorização, muitas vezes absoluta, do desenvolvimento frente a manutenção de software. Todavia, não seria justo acusar a organização que adota tal postura como incapaz de visualizar e entender de maneira ampla seu negócio de software. Essa postura clássica, e resultante da própria história de evolução da engenharia de software, vem agora gradativamente sendo alterada, sendo que a velocidade dessa alteração depende muito de uma análise cuidadosa dos objetivos de médio e longo prazo de cada organização.

Na seqüência, o próximo processo relevante dentro do grupo de gerência é o de *gerência de projeto*, conforme mostrado no quadro 5.9.

QUADRO 5.9: GERÊNCIA: PROCESSO DE *GERÊNCIA DE PROJETO*

Grupo de Processos de Gerência
Processo: Gerência de Projeto
Objetivos: Identificação, estabelecimento, coordenação e monitoração de atividades, tarefas e recursos necessários a um projeto para produzir um produto ou serviço no contexto de requisitos e limitações.
Resultados esperados de uma implementação com sucesso: (i) Definição do escopo de trabalho para o projeto; (ii) Verificação das possibilidades de execução do projeto com os recursos disponíveis e limitações impostas; (iii) Estimativa das tarefas e dimensionamento dos recursos necessários; (iv) Identificação e monitoramento de interfaces necessárias entre elementos do projeto e entre outros projetos e unidades organizacionais; (v) Desenvolvimento e implementação de planos para a execução do projeto; (vi) Monitoração do progresso do projeto e elaboração de relatórios; (vii) Tomada de ações de correção de desvios quando objetivos não forem alcançados.
Tarefas necessárias
Definição do escopo do trabalho: definir o trabalho necessário ao cumprimento do projeto.
Definição do ciclo de vida do projeto: definir o ciclo de vida e estratégia de desenvolvimento para o projeto, que deve ser apropriado ao seu contexto, escopo e complexidade.
Avaliação da possibilidade de execução do projeto: avaliar as possibilidades de se alcançar os objetivos do projeto, diante dos recursos e limitações existentes.
Determinar e manter estimativas para os atributos do projeto: definir e manter conjuntos de atributos validados por etapas de projeto (atributos podem incluir: negócios e objetivos de

qualidade do projeto, estimativas de tamanho e complexidade, esforço necessário e orçamento).
Definição de tarefas e atividades do projeto: identificar atividades e tarefas de projeto de acordo com o ciclo de vida, definindo ainda as dependências entre elas.
Definição de necessidades de experiência, conhecimento e habilidades: identificar a experiência e habilidades necessárias ao projeto, e aplicá-las para selecionar indivíduos e equipes.
Definição de cronograma: alocar recursos para as atividades e determinar a seqüência e cronograma de execução das atividades ligadas ao projeto.
Identificar e monitorar as interfaces de projeto: identificar e concordar com interfaces do projeto com outros projetos ou com outras partes afetadas, monitorando os compromissos acordados.
Alocar responsabilidades: identificar as contribuições específicas de cada indivíduo e de cada grupo necessárias ao projeto, alocando suas responsabilidades específicas, e garantindo que os comprometimentos foram entendidos e aceitos por todos.
Estabelecer o plano de projeto: definir e manter um plano mestre de projeto, bem como outros planos relevantes, para cobrir o escopo e objetivos do projeto, seus recursos, infra-estrutura, interfaces e mecanismos de comunicação necessários.
Implementar o plano de projeto: implementar as atividades planejadas para o projeto, registrando o status do progresso e reportando-o às partes afetadas.
Monitorar os atributos do projeto: monitorar o escopo do projeto, seu orçamento, custos, recursos e outros atributos necessários, documentando desvios significativos que possam ocorrer.
Revisar o progresso do projeto: relatar regularmente e revisar o status do projeto e seu desempenho frente ao plano de projeto.
Agir para corrigir desvios: tomar ações quando os objetivos de projeto não forem alcançados, para corrigir desvios com relação ao planejamento, prevenindo a recorrência de problemas já identificados.

Comentário sobre o processo:

Esse foi o processo com maior quantidade de problemas de manutenção de software associados, apesar de ser um tema tradicional em administração. Aparentemente, os problemas resultantes de uma gerência falha de projetos são bem conhecidos em todos os contextos, não sendo exclusivos de manutenção de software. Como exemplos têm-se a falta de comprometimento com cronogramas, falta de suporte da gerência, sobrecarga de tarefas e estimativa de prazos não condizentes com a complexidade do trabalho. Isso mostra que apesar da teoria estar evoluída, é preciso um controle muito próximo no dia-a-dia para se evitarem tais problemas comuns.

No contexto de manutenção de software, no entanto, alguns outros problemas se sobressaem e estão intimamente ligados ao trabalho com software. Entre eles está a conhecida idéia falsa de que contratar novos integrantes para uma equipe que trabalha com desenvolvimento ou manutenção de software trará mais agilidade ao processo. Isso pode ser verificado no problema da contratação de temporários que leva à diminuição do controle

sobre a atividade de manutenção. Esse problema se une ao do software não ser visto pela gerência como um produto que precisa, ainda durante o desenvolvimento, ter a manutenção trabalhada. A união desses dois problemas revela um outro: a falta de conhecimento sobre o processo de software como um todo. E aqui vale uma observação prática importante: não se podem atribuir a gerentes de projetos que não são de software, tarefas de gerência de projetos de software, sem assegurar que estes conheçam as peculiaridades inerentes ao trabalho com software e seu ciclo de vida.

O processo seguinte, *gerência de qualidade*, é apresentado no quadro 5.10.

QUADRO 5.10: GERÊNCIA: PROCESSO DE *GERÊNCIA DE QUALIDADE*

Grupo de Processos de Gerência
Processo: Gerência de Qualidade
Objetivos: Alcançar a satisfação do cliente, por meio da monitoração da qualidade dos produtos ou serviços, a nível organizacional e de projeto, para assegurar que estão de acordo com os requisitos.
Resultados esperados de uma implementação com sucesso: (i) Estabelecimento dos objetivos de qualidade com base nos requisitos explícitos e implícitos do cliente; (ii) Definição de uma estratégia completa de desenvolvimento para alcançar os objetivos definidos; (iii) Implantação de um sistema de gerenciamento de qualidade; (iv) Execução de atividades de controle e garantia de qualidade, avaliando-se seu desempenho; (v) Monitoração do desempenho com relação aos objetivos de qualidade; (vi) Tomada de providências quando objetivos de qualidade não forem alcançados.
Tarefas necessárias
Estabelecer objetivos de qualidade: estabelecer objetivos de qualidade para o produto e processo, preferencialmente de forma quantitativa, baseando-se nos requisitos de qualidade do cliente, e considerando-se também aqueles implícitos relevantes ao ambiente desse cliente.
Definir estratégia global: definir uma estratégia global, incluindo os recursos e responsabilidades, bem como os objetivos a serem alcançados e a contínua melhoria da qualidade.
Definir critérios de qualidade: definir padrões, referências e métricas que irão assegurar e verificar o alcance dos objetivos de qualidade, bem como definir o critério de aceitação que irá ajudar a decidir quando objetivos de qualidade foram ou não alcançados.
Estabelecer um sistema de gerenciamento de qualidade: estabelecer e manter um sistema de gerenciamento de qualidade para planejar, implementar, monitorar e controlar ações preventivas e corretivas.
Avaliar o cumprimento de objetivos de qualidade: revisar regularmente o atendimento de objetivos, no nível mais alto de gerenciamento, usando os critérios definidos.

Tomar ações corretivas ou preventivas: tomar ações corretivas e preventivas, em nível de organização e projeto, quando os objetivos de qualidade não forem alcançados.
Coletar resultados: coletar o retorno do cliente, do projeto, dos processos e das pessoas, para verificar a contínua melhora da qualidade tanto na organização como no projeto.

Comentário sobre o processo:

De uma maneira geral, *quase todos* os problemas de manutenção de software poderiam ser reduzidos ou evitados com o controle da qualidade. Especificamente na classificação adotada, o problema da *ausência de manutenção preventiva* se relacionou com a gerência da qualidade, pelo fato da norma prever para esse processo a tomada de ações corretivas e preventivas, que vem ao encontro da idéia de buscar antever problemas em um produto de software, procedendo-se com sua correção ou ajuste.

É fato conhecido que na prática muitas vezes a garantia da qualidade é prejudicada por questões como tempo curto e restrições financeiras. No entanto, isso é mais uma questão cultural do que de limitação real. Por exemplo, se existe limitação financeira para o projeto, é razoável pensar que se deva produzir exatamente o que o cliente deseja (idéia de *verificação*) e de forma correta (conceito de *validação*), evitando problemas futuros com correções e modificações perfeitamente evitáveis.

Dentro de manutenção de software, a garantia da qualidade deve estar enquadrada como objetivo estratégico da organização e disseminada da gerência para os funcionários de nível operacional. É a postura gerencial e a consciência de equipe que leva a produção de trabalhos com qualidade. Parece razoável pensar que se o foco for a qualidade da manutenção, problemas como a *ausência de adoção de padrões* ou *treinamento inadequado da equipe de manutenção* poderiam deixar de figurar na lista de problemas de manutenção de software.

Por fim, o último processo considerado no grupo de gerência é o de *gerência de riscos*, conforme mostrado no quadro 5.11.

QUADRO 5.11: GERÊNCIA: PROCESSO DE *GERÊNCIA DE RISCOS*

Grupo de Processos de Gerência
Processo: Gerência de Riscos
Objetivos: Identificação, análise, tratamento e monitoração de riscos de projeto continuamente.
Resultados esperados de uma implementação com sucesso: (i) Determinação do escopo do gerenciamento de riscos; (ii) Definição e implementação da apropriada estratégia de gerenciamento de riscos; (iii) Identificação dos riscos conforme se desenvolvem ao longo da condução do projeto;

(iv)	Determinação dos recursos de tratamento dos riscos com base na análise e priorização dos mesmos;
(v)	Implementação do correto tratamento para evitar o impacto do risco, baseando-se na prioridade e probabilidade de sua ocorrência.
Tarefas necessárias	
Estabelecer o escopo do gerenciamento de riscos: determinar o escopo do gerenciamento de riscos a ser implementado, observando-se as políticas de gerenciamento de riscos da organização.	
Definir as estratégias de gerenciamento de riscos: definição das apropriadas estratégias, análises, tratamentos e monitoramento de riscos, sempre em relação à organização e ao projeto.	
Identificar os riscos: identificar os riscos do projeto, inicialmente com relação à estratégia de projeto, e posteriormente, com os que surgirem durante a condução do projeto.	
Analisar os riscos: analisar os riscos para determinar prioridades e quais recursos de monitoramento aplicar.	
Definir e executar ações de tratamento de riscos: para cada risco ou conjunto de riscos semelhantes, definir e executar ações apropriadas para reduzi-los a um nível aceitável.	
Monitorar os riscos: monitorar o estado corrente de cada risco e assegurar a efetividade das ações de tratamento.	
Tomar ações corretivas: tomar ações corretivas apropriadas para reduzir ou evitar o impacto de cada risco, quando verificado que o tratamento de risco não está atingindo o progresso previsto.	

Comentário sobre o processo:

É fato conhecido que riscos são circunstâncias inerentes a qualquer projeto. A diferença entre um desastre ou a definição de uma alternativa quando um risco ocorre está no planejamento e nas formas controle que se estabeleceu sobre a possibilidade dos mesmos ocorrerem.

Dada a característica de ser normalmente impossível prever se um risco ocorrerá e quando ocorrerá, somente resta valer-se da verificação de dados históricos, ou mesmo do uso de heurísticas, para determinar quais os riscos potenciais conhecidos e então estabelecer formas de reação a eles caso ocorram.

Dentro do contexto de manutenção de software, foi possível listar pelo menos quatro riscos muito comuns, perante os quais é possível adotar medidas de contenção. Percebe-se que o problema da *elevada rotatividade dos profissionais* ligados à manutenção pode ser reduzido com programas de motivação e de justo trato de recursos humanos. Já o problema das *mudanças freqüentes de prioridades* por parte dos clientes pode ser combatido com a *definição e execução de ações de tratamento de riscos*, conforme sugere uma das tarefas da norma.

Por fim, os últimos dois riscos que se transformam em problemas de manutenção de software são ligados a questões técnicas, como a *má qualidade do código-fonte original* e a *baixa qualidade da documentação*. A exemplo dos demais riscos, o estudo prévio de como reagir a eles é a única preparação com a qual a organização pode contar, sendo que nesses dois últimos casos, uma possível preparação seria o acordo antecipado com o cliente sobre a possibilidade de se rever os prazos estabelecidos.

5.3.4 Grupo de Processos de Recursos e Infra-estrutura

Um total de 23,5% dos problemas de manutenção de software foi relacionado a problemas com a ausência de observância dos processos desse grupo, colocando-o em uma posição bastante relevante para o contexto de problemas de manutenção de software.

O primeiro processo que mereceu destaque nesse grupo foi o de *gerência de recursos humanos*, conforme mostrado no quadro 5.12.

QUADRO 5.12: RECURSOS E INFRA-ESTRUTURA: PROCESSO DE *GERÊNCIA DE RECURSOS HUMANOS*

Grupo de Processos de Recursos e Infra-estrutura
Processo: Gerência de Recursos Humanos
Objetivos: Prover à organização, e aos seus projetos, os indivíduos que possuam habilidades e conhecimentos para cumprir eficientemente seus papéis, trabalhando em grupos coesivos.
Resultados esperados de uma implementação com sucesso: (i) Identificação e contratação dos indivíduos com as habilidades e competências requeridas; (ii) Suporte à efetiva interação entre indivíduos e grupos; (iii) Força de trabalho capaz de utilizar suas habilidades para compartilhar informações e coordenar suas atividades eficientemente; (iv) Definição de critérios objetivos para monitorar o desempenho de cada grupo e indivíduo, para o provimento de retorno e melhoria de desempenho.
Tarefas necessárias
Identificar as habilidades e competências necessárias: identificar e avaliar habilidades e competências necessárias para atingir os objetivos da organização.
Determinar o critério de avaliação: definir critério objetivo que possa ser usado para avaliar candidatos e assegurar desempenho da equipe.
Recrutar pessoal qualificado: estabelecer um programa sistemático para recrutamento de pessoal qualificado que se encaixe nas necessidades da organização.
Desenvolver habilidades e competências: definir e prover oportunidades de desenvolvimento de habilidades e competências.
Definir a organização de equipes de projeto: definir a estrutura e regras de operação sobre as quais as equipes trabalharão.

Motivar equipes de projeto: motivar equipes para a execução de seus trabalhos, assegurando que todos têm: (i) entendimento do seu trabalho, (ii) uma visão de comum interesse, (iii) mecanismos apropriados de comunicação e trabalho, (iv) suporte da gerência no que o indivíduo estiver executando.
Manter interações entre equipes de projeto: obter e manter um acordo no gerenciamento de interações entre equipes.
Avaliar desempenho: avaliar o desempenho de pessoal, tanto individualmente como por equipes, no que se refere às suas contribuições aos objetivos da organização como um todo. Assegurar que os fatos levantados serão discutidos com o pessoal.
Prover retorno sobre o desempenho: prover retorno às pessoas, referente aos resultados de avaliações efetuadas.
Manter registros de pessoal: manter registros adequados de pessoal, incluindo não apenas detalhes pessoais, mas também informações a respeito de habilidades, treinamentos completados e avaliações de desempenho.

Comentário sobre o processo:

Os problemas de manutenção relacionados a esse processo podem ser vistos como um reflexo da postura clássica das organizações: a atenção voltada quase exclusivamente para o desenvolvimento de software.

Uma característica importante da postura organizacional que se relaciona tanto com esse processo, como com o de *treinamento*, conforme é apresentado no próximo quadro, refere-se à estratégia de manter os profissionais mais experientes, e com maior conhecimento sobre o software, em tarefas de desenvolvimento, reservando aos novatos as tarefas de manutenção. Essa postura evidencia o desprestígio da tarefa de manutenção pela gerência o que justifica diretamente um dos problemas associados a esse processo, o problema da *preferência dos profissionais por trabalhos de desenvolvimento*. Percebe-se aqui, novamente, um problema cultural funcionando como fonte para problemas de manutenção.

Nesse sentido, a gerência de recursos humanos deve ser considerada no contexto de manutenção de software, e não de forma genérica, isso devido às características diferenciadas que essa atividade apresenta em face de outros tipos de projetos não de software. Um dos pontos a considerar com cuidado é o de avaliar a possibilidade de maiores benefícios em se colocar profissionais mais experientes em tarefas de manutenção e não novatos recém contratados.

O próximo processo a ser estudado refere-se ao processo de *treinamento*, conforme mostrado no quadro 5.13.

QUADRO 5.13: RECURSOS E INFRA-ESTRUTURA: PROCESSO DE *TREINAMENTO*

Grupo de Processos de Recursos e Infra-estrutura
Processo: Treinamento
Objetivos: Prover a organização, e seus projetos, de indivíduos que possuam as habilidades e conhecimentos necessários à execução de seus papéis eficientemente.
Resultados esperados de uma implementação com sucesso: (i) Treinamentos são desenvolvidos ou adquiridos para atender às necessidades de treinamento tanto da organização como de projeto; (ii) Treinamentos são conduzidos para assegurar que todos os indivíduos possuam as habilidades necessárias para executar suas tarefas, usando mecanismos como estratégias de treinamento e materiais.
Tarefas necessárias
Desenvolver a estratégia para treinamento: desenvolver a estratégia de treinamento, incluindo como as necessidades de treinamento serão identificadas, como os treinamentos necessários serão desenvolvidos ou adquiridos, e como os treinamentos serão realizados.
Identificar necessidades de treinamento: identificar e avaliar habilidades e competências para prover e melhorá-las através de treinamentos.
Desenvolver ou adquirir treinamentos: desenvolver ou adquirir treinamentos que representem necessidades comuns de treinamento.
Preparar para a execução do treinamento: identificar e preparar a execução das sessões de treinamento, incluindo disponibilidade de materiais e de pessoal a ser treinado.
Treinar o pessoal: treinar o pessoal para possuírem conhecimento e habilidades necessárias à execução de seus papéis.
Manter registros de treinamentos: manter adequado registro de treinamentos completados.
Avaliar efetividade do treinamento: identificar e avaliar o valor adicionado oferecido por cada sessão de treinamento, incluindo a avaliação do material de treinamento utilizado.

Comentário sobre o processo:

Os problemas de manutenção associados a esse processo são o *treinamento inadequado da equipe de manutenção* e o fato de os *mantenedores lamentarem por não possuírem contato com o estado-da-arte da tecnologia*. Percebe-se aqui que a valorização da atividade de manutenção poderia diminuir ambos os problemas: por um lado entendendo-se o papel importante da manutenção de software, o problema da falta ou inadequação do treinamento seria atacado diretamente, com o surgimento de programas de treinamento específicos e focados nas peculiaridades da atividade. Por outro lado, a questão do não contato com o estado-da-arte também seria reduzido.

O problema do não contato com o estado-da-arte é algo relativo. Hoje um mantenedor afirma isso, por exemplo, porque não está em contato com a linguagem de programação da

moda ou a última metodologia para desenvolvimento ágil. Tal fato se deve à valorização do desenvolvimento. Porém, se a manutenção passar a ser outro ponto forte de atenção, a pessoa encarregada de tarefas de manutenção terá a possibilidade de estar em contato com técnicas e ferramentas que são estado-da-arte, mas da manutenção. Por exemplo, têm-se as técnicas modernas de decompilação, as ferramentas de recuperação de documentação etc. Dessa forma o não contato com o estado-da-arte representa uma alegação referente aos artefatos utilizados em desenvolvimento, uma vez que infelizmente ainda são pouco conhecidos aqueles possíveis de serem utilizados em manutenção.

O próximo processo considerado no grupo de recursos e infra-estrutura é o de *gerenciamento do conhecimento*, conforme mostrado no quadro 5.14.

QUADRO 5.14: RECURSOS E INFRA-ESTRUTURA: PROCESSO DE *GERENCIAMENTO DO CONHECIMENTO*

Grupo de Processos de Recursos e Infra-estrutura
Processo: Gerenciamento do Conhecimento
Objetivos: Garantia de que o conhecimento individual, informações e habilidades sejam coletados, compartilhados, reusados e melhorados através da organização.
Resultados esperados de uma implementação com sucesso: (i) Infra-estrutura é estabelecida e mantida para o compartilhamento de informação através da organização; (ii) Conhecimento é disponibilizado e compartilhado através da organização; (iii) A organização selecionará a apropriada estratégia de gerenciamento do conhecimento.
Tarefas necessárias
Estabelecer um sistema de gerenciamento do conhecimento: estabelecer e manter uma infra-estrutura de gerenciamento do conhecimento, bem como mecanismos para suportar as atividades de identificar, classificar, exportar e usar o conhecimento.
Criar uma rede de contribuidores de conhecimento: estabelecer uma rede de especialistas provendo sua natural interação.
Desenvolver uma estratégia de gerenciamento do conhecimento: definir uma estratégia apropriada de gerenciamento do conhecimento baseada em necessidades organizacionais, individuais, de domínio e de projetos.
Capturar o conhecimento: identificar e registrar cada item de conhecimento de acordo com a classificação estabelecida e critério de disseminação.
Disseminar a propriedade do conhecimento: compartilhar o conhecimento com especialistas, usuários e projetistas.
Melhorar o conhecimento disponível: validar e enriquecer o conhecimento armazenado, assegurando sua adequação aos valores da organização.

Comentário sobre o processo:

Manutenção de software é uma tarefa imprevisível. Isso resulta no fato de que nem sempre é possível aplicar a ela técnicas pré-moldadas típicas de processos de desenvolvimento. Nesse sentido, cada manutenção pode se tornar uma fonte de novos conhecimentos e novas descobertas heurísticas.

É justamente esse conhecimento derivado de manutenções passadas que deve ser preservado e, principalmente, disseminado entre pessoas e equipes diferentes. Essa ação pode gerar economia de tempo e de recursos, acelerando a liberação das alterações para o cliente. Infelizmente, notou-se que um dos problemas de manutenção apontados refere-se justamente à perda ou não disseminação de conhecimentos adquiridos com os demais membros interessados da organização. Isso pode ser evitado com a construção de sistemas internos paralelos de apoio, dedicados a armazenar e disponibilizar esse tipo de informação.

Por fim, o último processo desse grupo que recebe destaque é o processo de *infra-estrutura*, mostrado no quadro 5.15.

QUADRO 5.15: RECURSOS E INFRA-ESTRUTURA: PROCESSO DE *INFRA-ESTRUTURA*

Grupo de Processos de Recursos e Infra-estrutura
Processo: Infra-estrutura
Objetivos: Manter a infra-estrutura confiável e estável, uma vez que é necessária para o apoio do desempenho de qualquer outro processo.
Resultados esperados de uma implementação com sucesso: (i) Definição dos requisitos de infra-estrutura necessários; (ii) Identificação e especificação dos elementos de infra-estrutura; (iii) Aquisição dos elementos de intra-estrutura; (iv) Implementação dos elementos de infra-estrutura.
Tarefas necessárias
Identificar o escopo do processo de infra-estrutura: identificar os procedimentos, padrões, ferramentas e técnicas que o processo de infra-estrutura deve apoiar.
Definir os requisitos do processo de infra-estrutura: definir os requisitos de infra-estrutura para apoiar o desempenho dos processos relacionados. Pode incluir: (i) segurança, (ii) facilidades de acesso remoto, (iii) backup e recuperação de dados, (iv) requisitos de manutenção, etc.
Adquirir e prover processo de infra-estrutura: adquirir e prover um processo de infra-estrutura, que satisfaça aos requisitos.
Estabelecer o processo de infra-estrutura: reunir e integrar os elementos do processo de infra-estrutura, provendo um ambiente efetivo que apóie a implementação dos processos da organização.

Prover apoio para o processo de infra-estrutura: prover suporte para aqueles que utilizam o processo de infra-estrutura.
Manter o processo de infra-estrutura: realizar manutenção no processo de infra-estrutura com o propósito de corrigir defeitos e melhorar o desempenho.

Comentário sobre o processo:

O processo de infra-estrutura no contexto de contenção e prevenção de problemas de manutenção de software se destaca no sentido de fornecer ferramentas e ambiente diferenciado para manutenção.

Normalmente, o que ocorre nas empresas é a união, em um mesmo ambiente computacional, de ferramentas e atividades de desenvolvimento e manutenção, o que causa freqüentemente problemas com sobreposição de arquivos, perda de modificações e propagação de falhas. A solução para esses problemas não se resume em adotar um software de controle de versão. A solução precisa incluir uma separação entre desenvolvimento e manutenção e principalmente processos para se conduzir ambas as atividades.

É conhecida a relativa dificuldade de se unir um projeto modificado com o projeto paralelo em desenvolvimento. Isso porque no mesmo instante em que um software esteja sendo modificado em um determinado ponto, ele pode estar em desenvolvimento por outra equipe, agregando-se funcionalidades novas, não necessariamente solicitadas pelos clientes, mas como forma de sofisticação e complementação do software original. Nesse momento surge a dificuldade de unir eventualmente mesmos arquivos modificados em locais diferentes. Situações como essas devem ser analisadas e amparadas por ferramentas através de um processo de infra-estrutura.

5.3.5 Grupo de Processos de Gerência de Configuração

Um total de 11,8% dos problemas de manutenção de software foi relacionado à má implementação de processos do grupo de processos de gerência de configuração. São três os processos desse grupo que merecem destaque nesse contexto.

O primeiro processo, *documentação*, é apresentado no quadro 5.16.

QUADRO 5.16: GERÊNCIA DE CONFIGURAÇÃO: PROCESSO DE *DOCUMENTAÇÃO*

Grupo de Processos de Gerência de Configuração
Processo: Documentação
Objetivos: Desenvolvimento e manutenção do registro de informações produzidas por um processo.
Resultados esperados de uma implementação com sucesso:

(i)	Desenvolvimento da estratégia para identificação da documentação que será produzida durante o ciclo de vida de um produto ou serviço;
(ii)	Identificação dos padrões a serem aplicados para o desenvolvimento da documentação;
(iii)	Identificação da documentação a ser produzida pelo processo ou projeto;
(iv)	O conteúdo e propósito de toda documentação é especificado, revisado e aprovado;
(v)	Documentação é desenvolvida de acordo com os padrões definidos;
(vi)	Documentação é mantida de acordo com os critérios definidos.
Tarefas necessárias	
Desenvolver a estratégia de gerenciamento da documentação: determinar a estratégia de gerenciamento da documentação, que deve indicar o que deve ser documentado a cada estágio do ciclo de vida do produto/serviço.	
Estabelecer padrões para documentos: estabelecer padrões para desenvolver, modificar e manter documentos.	
Especificar requisitos de documentos: especificar os requisitos para os documentos, como título, data, identificação, versão, autores, revisores, propósito, lista de distribuição etc.	
Identificar os documentos a serem produzidos: para qualquer ciclo de vida, identificar os documentos a serem produzidos.	
Desenvolver os documentos: desenvolver os documentos requeridos ao ponto corrente do processo, de acordo com os padrões e política estabelecidos.	
Checar os documentos: revisar os documentos antes da distribuição e autorizá-los antes de cada distribuição/atualização.	
Distribuir os documentos: distribuir os documentos de acordo com as formas de distribuição determinadas, confirmando o recebimento quando necessário.	
Manter os documentos: manter os documentos atualizados de acordo com a estratégia de documentação empregada.	

Comentário sobre o processo:

A documentação de software é uma das tarefas mais trabalhosas em manutenção de software. Isso porque cada manutenção individual deve ser documentada, e o projeto do software como um todo atualizado. O desenvolvimento de documentos padronizados e processos formais de atualização de documentação podem não ser viáveis para manutenções corriqueiras e de pequeno porte. Enfrentar um processo burocrático a cada pequena manutenção acaba desestimulando a execução dessa atividade e conseqüentemente levando à progressiva desatualização da documentação do projeto.

Cada organização deve avaliar o produto de software que desenvolve do ponto de vista da documentação, estabelecendo-se uma maneira prática de manter essa documentação atualizada. O interessante é facilitar no caso de pequenas manutenções, reservando somente às

de grande porte processos mais formais de construção e atualização de documentos. Novamente o uso de sistemas internos de apoio é uma alternativa favorável.

O próximo processo a receber atenção nesse grupo é o de *gerência de configuração*, conforme mostrado no quadro 5.17.

QUADRO 5.17: GERÊNCIA DE CONFIGURAÇÃO: PROCESSO DE *GERÊNCIA DE CONFIGURAÇÃO*

Grupo de Processos de Gerência de Configuração
Processo: Gerência de Configuração
Objetivos: Estabelecimento e manutenção da integridade dos itens de um processo (como exemplo, um módulo do produto de software em desenvolvimento), tornando-os disponíveis às partes interessadas.
Resultados esperados de uma implementação com sucesso: (i) Desenvolvimento de uma estratégia de gerenciamento de configuração; (ii) Itens/produtos gerados por um processo ou projeto são identificados, definidos e confirmados; (iii) Modificações e atualizações nos itens/produtos são controladas; (iv) Modificações e atualizações são disponibilizadas a todas as partes afetadas; (v) O status e modificações dos itens/produtos são registrados e reportados; (vi) A completude e consistência dos itens/produtos são asseguradas; (vii) O armazenamento, manipulação e entrega dos itens/produtos é controlado.
Tarefas necessárias
Desenvolver a estratégia de gerenciamento de configuração: determinar a estratégia de gerenciamento de configuração, incluindo as atividades e cronograma.
Identificar os itens de configuração: identificar os itens de configuração que precisam ser individualmente armazenados, testados, revisados, usados, alterados, entregues e mantidos.
Estabelecer a estrutura e hierarquia de arquivo e diretórios: prover meios eficientes de acesso e armazenamento de entidades necessárias.
Estabelecer baselines: estabelecer as baselines internas e de entrega, que devem armazenar todos os itens configurados necessários ao estágio respectivo do projeto (obs.: uma baseline representa um "pacote" de itens validados e considerados de acordo com os propósitos do projeto).
Manter a descrição dos itens de configuração: manter uma descrição atualizada de cada item de configuração.
Controlar modificações e atualizações: estabelecer um mecanismo para registrar e atualizar os itens.
Manter histórico de item de configuração: manter um histórico de cada item de configuração em detalhamento suficiente para permitir recuperar toda versão anterior, quando necessário.
Relatar status da configuração: relatar o status de cada item de configuração e seu relacionamento com a atual integração do sistema.

Verificar informações sobre os itens configurados: verificar se a informação sobre os itens configurados e suas estruturas, fornecidos pelo relatório de status, está completa e assegurar ainda a consistência desses itens.
Gerenciar o backup, armazenamento, manipulação e entrega de itens de configuração: assegurar a integridade dos itens configurados, bem como dos recursos de backup e armazenamento. Controlar também a manipulação e entrega dos itens configurados.

Comentário sobre o processo:

A quantidade de arquivos modificados em uma manutenção, ainda que de pequeno porte, pode ser grande. A esse fato inclui-se o problema de os mesmos arquivos poderem estar sofrendo outras modificações por parte da equipe de desenvolvimento. Esse é apenas um dos problemas que precisam ser tratados nesse processo, e está relacionado com a questão de conciliar manutenção com desenvolvimento.

Outro fato ligado ao processo de gerência de configuração é o estabelecimento de componentes e versões *baselined*. Se o produto de software está, de um lado, em evolução pela equipe de desenvolvimento e, por outro lado, em fase de adaptação e modificação por equipe de manutenção, torna-se difícil e necessário garantir uma forma de estabelecer e armazenar versões testadas e confiáveis do software como um todo. A ausência de um processo adequado pode levar a perda de código e alterações, bem como a entrega de versões do software com novos problemas para o cliente.

Finalmente, o último processo a ser destacado é o de *gerência de solicitações de mudança*. Esse processo é mostrado no quadro 5.18.

QUADRO 5.18: GERÊNCIA DE CONFIGURAÇÃO: PROCESSO DE *GERÊNCIA DE SOLICITAÇÕES DE MUDANÇA*

Grupo de Processos de Gerência de Configuração
Processo: Gerência de Solicitações de Mudança
Objetivos: Assegurar que as solicitações de mudança sejam gerenciadas, monitoradas e controladas.
Resultados esperados de uma implementação com sucesso: (i) Desenvolvimento de uma estratégia de gerenciamento; (ii) Requisições de mudança são identificadas e registradas; (iii) Dependências e relações com outros pedidos de mudança são identificadas; (iv) Definição de critérios para confirmação da implementação de mudanças; (v) Priorização de requisições de mudança e estimativas de recursos necessários; (vi) Aprovação de mudanças com base na prioridade e disponibilidade de recursos; (vii) Mudanças aprovadas são implementadas; (viii) Possibilidade de conhecimento do status de todos os pedidos de mudança.

Tarefas necessárias
Desenvolver uma estratégia de gerenciamento de mudanças: estabelecimento de uma estratégia de gerenciamento de mudanças para assegurar que as mudanças possam ser descritas, registradas, analisadas e implementadas.
Registrar os pedidos de mudanças: cada requisição de mudança deve ser unicamente identificada e registrada.
Registrar o status dos pedidos de mudança: requisições de mudança devem possuir um status associado para permitir facilidade no monitoramento.
Estabelecer as dependências e relações com outros pedidos: identificar o relacionamento do pedido de mudança com outros pedidos para estabelecer suas dependências (por exemplo, mudanças no mesmo componente de software ou mudanças requisitadas para uma futura versão já planejada do software).
Avaliar o impacto da mudança: avaliar o impacto da mudança, os recursos e os potenciais benefícios resultantes.
Identificar as atividades de verificação e validação necessárias: antes de implementar a mudança, o escopo das atividades de verificação e validação correspondentes deve ser identificado.
Aprovar as mudanças antes de implementação: todas as mudanças devem ser aprovadas antes da implementação.
Incluir mudança em cronograma: as mudanças aprovadas devem ser agendadas para implementação.
Revisar a mudança implementada: todas as mudanças devem ser revisadas após implementação e antes de serem concluídas, de modo a garantir que tiveram os efeitos desejados e atingiram os objetivos.

Comentário sobre o processo:

Mudanças talvez seja a palavra mais presente no contexto de manutenção de software e mudanças podem surgir em um volume muito grande, o que dependerá do estágio de evolução do software e do domínio ao qual atende. A prática mostra que quando o volume de solicitações de mudança é grande, inúmeros outros problemas surgem derivados desse fato.

Um processo adequado de gerência de solicitações de mudança deve ser capaz de filtrar o que de fato resultará em uma alteração no produto de software e o que pode ser resolvido por outros meios ou valendo-se de recursos já existentes e pouco conhecidos do software. Além de saber separar o que é mesmo necessário do que é mero preciosismo do usuário, esse processo deve atentar-se especialmente para a questão da priorização dos chamados. Isso porque o usuário normalmente irá solicitar alterações em cima de alterações e a probabilidade de se perder o controle do que está sendo feito é alta. Uma dica aqui é estabelecer um mecanismo de comunicação eficiente com o usuário, preferencialmente através de um sistema paralelo ao qual o usuário possa consultar e que possa também ser usado pela organização para expor dificuldades e possíveis sobreposições de chamados

considerados urgentes, deixando claro ao cliente os fatos que justificam a negação ou adiamento de uma determinada manutenção.

5.4 Propostas na Literatura

Alguns autores que se dedicaram a pesquisar problemas de manutenção de software também mostraram propostas de soluções ou de redução desses problemas.

As soluções apresentadas são semelhantes e pode-se dizer que estão muito bem resumidas e apresentadas em Dart *et al.* (2001), que organizou suas propostas de intervenção nos problemas de manutenção em três grandes grupos: recomendações quanto a processos, quanto a pessoas e quanto a ferramentas.

Propostas quanto aos processos:

- Reexaminar políticas corporativas à luz de ferramentas CASE;
- Enfatizar a importância de garantir que todo código esteja completamente e cuidadosamente documentado;
- Discernir as diferenças de necessidades entre projetos de manutenção e projetos de novos softwares;
- Implantar filosofia de *projeto-para-manutenção*;
- Planejar apropriadamente o tempo em cronogramas que incluam documentação e testes;
- Promover os sucessos anteriores com relatórios de lições aprendidas;
- Instituir e reforçar práticas de garantia de qualidade;
- Investigar porque a comunicação interna na organização ocasionalmente torna-se ineficiente;
- Melhorar as interações dos clientes com os desenvolvedores e mantenedores;
- Melhorar a seleção de fornecedores e monitorar processos, planos de estratégia e tomadas de decisões.

Propostas quanto às pessoas:

- Disseminar o conhecimento relativo ao estado-da-arte de ferramentas e conhecimentos;
- Associar pessoas a papéis particulares (exemplo: testes) para projetos e oferecer suporte corporativo a esses papéis;

- Encorajar as pessoas mais experientes para se tornarem mantenedores;
- Melhorar o prestígio das tarefas de manutenção;
- Melhorar a comunicação entre grupos por meio de recursos diversos (quadros, ferramentas eletrônicas etc.), registrando-se a experiência individual e inter-projetos, elegendo-se alguém para organizar e promover essa comunicação;
- Tornar mais efetivo o treinamento, especialmente com relação ao uso de ferramentas, padrões e documentação;
- Estabelecer um recurso de consulta a informações técnicas para suporte da gerência.

Propostas quanto às ferramentas:

- Investir em ferramentas mais efetivas, principalmente aquelas que suportarem engenharia reversa, reengenharia, testes, gerência de configuração e documentação;
- Encorajar fornecedores a construírem ou adaptarem suas ferramentas, especialmente para suportarem as necessidades da organização;
- Melhorar a qualidade das ferramentas desenvolvidas internamente;
- Avaliar novas ferramentas e determinar a sua aplicabilidade em projetos específicos;
- Melhorar as atividades e uso de ferramentas que centralizem comunicação entre projetos;
- Encontrar soluções de ferramentas que suportem plataformas heterogêneas da organização (quando houver);
- Encorajar o pessoal técnico para freqüentemente comunicarem suas necessidades de ferramentas à gerência superior.

5.5 Considerações Finais

Os exemplos de soluções possíveis para os problemas de manutenção de software apresentados neste capítulo, principalmente os derivados da interpretação das recomendações da norma, mostram como pode ser extenso e detalhadamente tratado o assunto.

Nesse sentido, este capítulo representa um documento de que existem maneiras de se tratar os problemas de manutenção de software, residindo na cultura organizacional o principal entrave para se atacar de forma ampla as dificuldades. Certamente a formação acadêmica de seus funcionários, voltada para a conscientização da importância da manutenção de software e das alternativas de resposta aos seus problemas, levaria a uma predisposição maior para a mudança organizacional.

CAPÍTULO

6

Características do Ensino de Manutenção de Software

6.1 Considerações Iniciais

Dentre as atividades previstas no ciclo de vida de software, a manutenção é a que possui a menor atenção e conseqüente incipiência em seu ensino (Dias, 2004), apesar de não ser uma atividade nova, já que o seu surgimento está relacionado ao surgimento dos primeiros softwares (Zvegintzov & Parikh, 2005).

Para ensinar manutenção de software, além de conhecer profundamente essa atividade é fundamental saber o que vem sendo feito pela comunidade científica no intuito de transmitir o conhecimento a respeito. Essa compreensão pode ser útil para a extração de recomendações do que deve ser aperfeiçoado e também de erros que já foram identificados, de forma a poder propor soluções não absolutamente novas, mas baseadas em correções de erros e melhoria dos acertos de experiências anteriores.

6.2 Panorama do Ensino de Engenharia de Software

Contando com mais de trinta anos de existência, a engenharia de software ainda é uma área de estudo recente, tendo alguns de seus tópicos sem consenso entre a comunidade de pesquisadores, não possuindo um corpo de conhecimento estabilizado (Castro *et al.*, 2000).

Ensinar engenharia de software vem sendo um desafio através dos anos e dados mostram que ainda existe uma grande carência de qualidade na forma como os softwares são desenvolvidos. Isso pode ser comprovado por uma estatística relativamente recente que indica

que 40 a 50% dos softwares de usuário apresentam defeitos não-triviais (Boehm & Basili, 2001).

Um dos problemas de ensinar engenharia de software está ligado à vasta quantidade de termos e conceitos, o que pode resumir uma disciplina de engenharia de software em uma disciplina de termos e conceitos de engenharia de software (Stiller & LeBlanc, 2002). Ainda segundo a visão desse autor, é necessário que exista um projeto prático para prover um contexto para os termos e conceitos, pois caso contrário os alunos não reterão o conteúdo de maneira apropriada.

De acordo com Soares (2005), fatores relacionados com a questão técnica (metodologias, linguagens etc.) em engenharia de software estão mais amadurecidos do que aqueles ligados a características sociais (como trabalho em equipe), sendo necessária a união desses fatores para um ensino adequado da disciplina.

Diversas são as propostas para ensino de engenharia de software, algumas preocupadas essencialmente com a forma de ensinar, enquanto outras com o conteúdo a ensinar, como exemplificado a seguir:

- Em Tomayko (1987), distinguem-se quatro modos em que esta disciplina pode ser ministrada: aulas expositivas; aulas expositivas e seminários; aulas expositivas e estudos de casos com pequenas equipes; aulas expositivas e estudos de casos com grandes equipes, de quinze a trinta alunos, sendo o professor o gerente da equipe.
- Em Silva *et al.* (2003), professores apresentam uma experiência de ensino da disciplina usando a metodologia ágil XP (Beck, 1999) e buscando enfocar a implementação baseada em *web*, com ferramentas como PHP, HTML e XML.
- Em Hazzan e Dubinsky (2003), demonstra-se a preocupação em se tratar as questões humanas do desenvolvimento de software.

A questão de ensino de engenharia de software como disciplina não isolada é exposta por Ghezzi e Mandrioli (2005) ao dizer que a engenharia de software precisa ser ensinada de forma atrelada ao seu contexto. Isso significa que o engenheiro de software deve não apenas conhecer técnicas de engenharia de software, mas também linguagens de programação, algoritmos e estrutura de dados, banco de dados, sistemas operacionais, etc., que estarão ainda ligados ao ambiente para o qual o software for ser desenvolvido (organização comercial, indústria, sistema de missão crítica etc.). Em suma, o ensino de engenharia de software deve abordar, na medida do possível, assuntos muito diversos.

Os conhecimentos necessários a um engenheiro de software, de acordo com Ghezzi e Mandrioli (2005), estão relacionados ao domínio amplo de:

- Fundamentos teóricos da disciplina;
- Métodos de projeto;
- Tecnologia e ferramentas.

Além disso, o profissional precisa ter habilidades para:

- Manter seu conhecimento a respeito de novas abordagens e tecnologias;
- Interagir com outras pessoas (freqüentemente de culturas diferentes);
- Entender, modelar, formalizar e analisar um *novo* problema;
- Reconhecer problemas recorrentes e reusar ou adaptar soluções conhecidas;
- Gerenciar processos e coordenar o trabalho de pessoas diferentes.

Uma observação importante se refere ao fato de que algumas dessas habilidades somente podem ser totalmente entendidas fora do ambiente acadêmico, ou seja, com a prática no mundo real. Aqui é possível um paralelo com o ensino de manutenção de software, que guarda também essa característica de total aprendizado somente com a prática, o que vem a reforçar os objetivos deste trabalho, quando tenta aproximar problemas da prática com a teoria acadêmica.

6.3 Questões sobre o Ensino de Manutenção de Software

O ensino de manutenção de software na indústria e dentro do meio acadêmico, por meio de disciplina específica, está recebendo atenção crescente. No geral, na quase totalidade dos casos, o enfoque das universidades está em ensinar desenvolvimento de software, incluindo a manutenção como um item dentro do desenvolvimento, muitas vezes recebendo pouca ênfase.

Dias (2004) explica que o enfoque em desenvolvimento dentro do meio acadêmico é uma postura tendenciosa, e que considera a manutenção como parte do processo de desenvolvimento, ou em outros casos quando o processo de desenvolvimento proposto é iterativo, a cada iteração a atividade de manutenção aparece.

Criar uma forma explícita de abordar a manutenção de software dentro de universidades é uma decisão que vem sendo tratada com cautela. Cardow (1992) já havia apresentado três razões pelas quais se justificava o adiamento do ensino de manutenção: (i) essa área é um subconjunto de um grande corpo de conhecimento, que adequadamente cobre o tópico; (ii) não há interesse suficiente para se preocupar com o estudo desse curso; (iii) não existe uma quantidade suficiente de material para apresentar. Ele complementa que a visão até então era de que a manutenção representava uma continuação do processo de

desenvolvimento de software e que não necessitaria de uma atenção maior.

No entanto, nesse mesmo trabalho, Cardow mostra que essa visão é errônea, já que o desenvolvimento sempre produz software com necessidade de manutenção, em função de mudanças de requisitos, escolhas erradas de projeto ou má interpretação do mesmo e ainda por ser a atividade de testes muitas vezes reduzida por limitações de tempo e custos.

O interesse didático com manutenção de software, por sua vez, também existe, como Cardow explica por estudos realizados naquela época junto ao exército americano e na indústria em geral, bem como atualmente pela existência de congressos destinados exclusivamente a tratar essa questão.

Por fim, a afirmação de que a quantidade de material existente não é suficiente para elaborar um curso também não deve ser considerada válida, pois existe uma quantidade crescente de bons artigos apresentados em congressos e eventos específicos de manutenção de software ou não, ao redor do mundo, bem como diversas pesquisas.

A dificuldade, portanto, está centrada no conteúdo a ser ensinado. Esse conteúdo deve ser suficientemente fundamentado em resultados de situações reais e não puramente acadêmicas. Assim, além dos conceitos teóricos básicos, o ensino deve acatar dificuldades persistentes verificadas na indústria.

6.4 Padrões de Currículo para Cursos de Computação

No Brasil, a Sociedade Brasileira de Computação[14] (SBC), é uma organização que fomenta e desenvolve pesquisa científica na área da computação. Ela estabelece alguns planos pedagógicos recomendados para cursos de graduação na área de computação. Mais especificamente, são propostos planos de ensino para os cursos de Licenciatura em Computação, Bacharelado em Engenharia de Computação, Bacharelado em Sistemas de Informação e Bacharelado em Ciência da Computação, que são os nomes padronizados pela instituição para os cursos de graduação na área pelo país.

Para o curso de Licenciatura em Computação, Dahmer *et al.* (2002) explica que o objetivo é formar professores, especialistas da área de computação, capazes de tratar os conteúdos da ciência da computação, nos anos finais do ensino fundamental (5ª a 8ª séries) e médio, podendo abordar esses conteúdos de forma integrada ou não a outras disciplinas, ou tratados como projetos, temas transversais ou atividades colaborativas.

O plano da disciplina é organizado por matérias que possuem disciplinas que cobrem total ou parcialmente uma matéria.

[14] http://www.sbc.org.br

Analisando a grade curricular proposta para esse curso, verifica-se que o total de horas previsto para a matéria engenharia de software é de 210 horas, sendo representado por apenas uma disciplina. O termo manutenção de software não aparece, de forma que se algo nesse sentido é ensinado, isso deverá ser feito dentro da única disciplina oferecida, e será muito ligado à visão do professor que ministrará essa disciplina, de forma que o enfoque em manutenção poderá ser variado, ou até não existir.

Para o curso de Bacharelado em Engenharia de Computação, Teixeira *et al.* (2002) explica que o objetivo é formar recursos humanos para o desenvolvimento e aplicação de tecnologias da computação no controle de processos e automação industrial. O currículo proposto para esse curso reserva um total de 680 horas para a matéria de engenharia de software, distribuídas por oito disciplinas. Fica claro um enfoque bem maior em engenharia de software quando comparado com o curso de licenciatura. O plano de engenharia de computação considera que o egresso poderá exercer atividades de manutenção de software e dentro da disciplina *Métodos e Ferramentas de Engenharia de Software* apresenta, como um dos objetivos, capacitar o aluno no uso de métodos e ferramentas para manutenção de software.

No curso de Bacharelado em Sistemas de Informação, Costa *et al.* (2000) explica que o objetivo do curso é a formação de profissionais para atuação em planejamento, análise, utilização e avaliação de modernas tecnologias de informação aplicadas às áreas administrativas e industriais, em organizações públicas e privadas.

Por se tratar de um curso que oferece uma visão de administração de empresas e sistemas de informação, esperava-se que esse pudesse ser o curso mais propenso em ensinar manutenção de software por meio de uma disciplina específica, isso porque o egresso poderá deparar-se com situações onde precisará decidir sobre o futuro de sistemas legados, por exemplo, determinando sua descontinuidade ou manutenção. Porém, esse fato não se verificou. O curso reserva um total de 432 horas para a matéria engenharia de software, distribuídas em quatro disciplinas. Em nenhuma delas está previsto explicitamente o ensino de manutenção de software. A disciplina de *Engenharia de Software Aplicada* prevê exercitar com o aluno o ciclo de vida de software, subentendendo-se que a atividade de manutenção possa ser mesclada nessa disciplina, novamente estando dependente do perfil do professor que a ministrar.

Finalmente, no curso de Bacharelado em Ciência da Computação, fechando os quatro cursos de computação definidos pela SBC, Ferreira *et al.* (2001) explica que o objetivo é formar recursos humanos para o desenvolvimento tecnológico da computação. Nesse curso, 144 horas são reservadas para a matéria engenharia de software, distribuídas em duas

disciplinas. Apesar de o curso prever a capacitação do egresso para resolver problemas de manutenção de software, em nenhuma das disciplinas o ensino é explicitamente previsto. Novamente, vale o que foi dito para os cursos anteriores, o ensino de manutenção de software poderá ser mesclado nas disciplinas previstas para engenharia de software, mas isso dependerá do perfil do professor que ministrar as aulas, o que certamente implicará em abordagens diferentes cada vez que as disciplinas forem ensinadas por professores com visões diferentes.

No quadro 6.1 é resumido o plano pedagógico para os quatro cursos descritos. São relacionados os nomes dos cursos, as disciplinas para o tema engenharia de software e o total de horas previsto para cada disciplina.

QUADRO 6.1: CURSOS DE COMPUTAÇÃO E DISCIPLINAS DE ACORDO COM A SBC

Curso	Disciplinas de Engenharia de Software	Total de Horas	Total Geral
Licenciatura em Computação	Engenharia de Software	210	**210 h**
Engenharia de Computação	Engenharia de Software I Engenharia de Software II Engenharia de Software III Gerência de Configuração Processo de Engenharia de Software Métodos e Ferramentas para Eng. de Soft. Qualidade de Engenharia de Software Tópicos Especiais em Eng. de Software	120 80 80 80 80 80 80 80	**680 h**
Sistemas de Informação	Engenharia de Software A Engenharia de Software B Engenharia de Software Aplicada Tópicos Especiais em Eng. de Software	72 72 72 72	**288 h**
Ciência da Computação	Engenharia de Software Laboratório de Engenharia de Software	72 72	**144 h**

Uma outra organização, de importância internacional, que também deve ser considerada do ponto de vista de recomendações curriculares, é a *Association for Computing Machinery*[15] (ACM).

A ACM corresponde a uma organização educacional e científica, dedicada ao avanço das artes, ciências e aplicações de tecnologia de informação (Blasi, 1999).

Um dos objetivos da organização é desenvolver recomendações curriculares, que pela relevância das publicações acabam por influenciar o ensino acadêmico na área de computação em muitas instituições ao redor do mundo (Rodriguez, 2004).

Em uma publicação (Shackelford *et al.*, 2004) são discutidas as características de planos de cursos de computação nos Estados Unidos, explicando o foco desses cursos antes

[15] http://www.acm.org

de 1990 e após essa data, na qual significativos avanços proporcionaram uma adaptação dos currículos para os cursos de Engenharia de Computação, Ciência da Computação e Sistemas de Informação. Tais avanços incluem a importância atribuída ao tema de engenharia de software dentro do contexto de ciência da computação.

A importância que o tema engenharia de software assumiu a partir de 1990 nos Estados Unidos está atrelada ao fato de que quanto mais a computação é usada para abordar quantidades crescentes de problemas complexos, criar software de confiança se torna mais difícil. Na medida em que os problemas se tornam cada vez mais complexos, não é possível que apenas uma pessoa entenda o software por completo, considerando ainda que diferentes módulos de um software possam precisar interagir de forma imprevisível. O uso da computação para tarefas críticas, que envolvam segurança de pessoas, também pode ser considerado um dos fatores que influenciaram o desenvolvimento da engenharia de software.

De uma forma geral, o que foi possível compreender era que produzir software consistia em uma tarefa difícil, muito cara e muito necessária.

Segundo ainda o mesmo autor, a percepção de todos esses fatos fez com que a engenharia de software se tornasse uma disciplina fundamental em cursos de computação, sendo aplicada inicialmente em cursos de computação na Inglaterra e Austrália durante a década de 1980 e adotada mais tarde, durante a década de 1990, nos Estados Unidos.

Percebe-se que essa evolução da engenharia de software é contínua, e os currículos propostos pela ACM já consideram a atividade de manutenção de software como uma disciplina específica.

A importância da disciplina difere de acordo com o curso de computação em questão. Para mostrar essa importância relativa, Shackelford propõe em seu trabalho uma escala de 0 a 5, que representa um índice de ênfase atribuída à disciplina. O valor 5 (cinco) representaria a máxima ênfase possível, enquanto 0 (zero) indicaria nenhuma relevância dentro do curso considerado. Para cada disciplina dentro de um curso, a escala diria qual a mínima e qual a máxima ênfase que a disciplina pode ter dentro do currículo do curso. Essa estipulação de máximo e mínimo visa oferecer flexibilidade para a grade curricular como um todo, que poderia variar as horas-aula dentro da faixa de máxima e mínima ênfase de cada disciplina.

As ênfases propostas para a disciplina de manutenção de software foram consideradas para elaborar o quadro 6.2, no qual é mostrada a distribuição dessas ênfases para cada um dos três cursos de computação em questão.

QUADRO 6.2: ÊNFASES SUGERIDAS PARA A DISCIPLINA DE MANUTENÇÃO DE SOFTWARE

Disciplina	EC*		CC**		SI***	
	mín.	*máx.*	*mín.*	*máx.*	*mín.*	*máx.*
Evolução e Manutenção de Software	**2**	**4**	**1**	**1**	**1**	**2**

* EC = Engenharia de Computação, ** CC = Ciência da Computação, *** SI = Sistemas de Informação

Percebe-se pelos números uma importância maior dada ao tema no curso de Engenharia de Computação (prevê ênfase de até 4 pontos).

Embora esses números ainda possam representar um enfoque baixo, a proposta da ACM já considera a manutenção de software como uma disciplina em cursos de graduação em computação, e é esperado que no Brasil também surjam recomendações nesse sentido, considerando a representatividade e influência internacional da ACM.

6.5 Experiências com o Ensino de Manutenção de Software

As tentativas de ensinar manutenção de software no meio acadêmico têm aumentado e alguns exemplos são discutidos nesta seção.

Em seu trabalho, Andrews (Andrews & Lutfiyya, 2000) apresenta a descrição e os resultados obtidos com a criação de um curso de manutenção de software de duas horas semanais (com uma duração total de 26 semanas), em uma universidade canadense[16] para alunos de graduação.

Para esse curso, a metodologia escolhida consistia em trabalhar teoria e prática com os alunos, por meio de projetos práticos desenvolvidos durante a disciplina. Essa abordagem envolvia dividir os alunos em grupos e cada grupo conduziria um projeto, que era motivado pelos diferentes tipos de tarefas em manutenção de software.

Andrews explica que dentre os tipos de manutenção de software, o tipo manutenção corretiva não era abordado pela disciplina, uma vez que os alunos já haviam tratado de correções de erros em software em disciplinas anteriores. O foco estava em projetos que priorizassem manutenções adaptativas e perfectivas (nota-se que manutenção do tipo preventiva não era considerada).

Em um primeiro momento, os alunos organizaram-se em grupos de no máximo seis alunos e a cada grupo foi atribuído um projeto. Tais projetos objetivavam, por exemplo, adaptar software para uso em diferentes máquinas, escrever novos módulos de modo que esses módulos fizessem comunicação com sistemas legados e ainda alguns projetos envolviam entender e efetuar manutenções diversas em software-livres. No geral as atividades

[16] University of Western Ontario (UWO).

de cada grupo consistiam em entender o código-fonte, realizar instalação, implementar pequenas características novas e a produção de relatórios para apresentação em sala de aula.

O autor descreve os problemas iniciais dessa abordagem, que envolveu o cancelamento dos projetos de alguns grupos e a atribuição de novos. O cancelamento ocorreu em função dos softwares proprietários em uso não poderem ser submetidos a algumas atividades previstas na disciplina, como instalação. Outros problemas como má organização das equipes e falta de tempo para execução completa dos projetos foram citados.

Finalmente, como resultados finais da experiência, os professores puderam constatar que existia uma tendência nos resultados apresentados no sentido de trabalhar atividade de desenvolvimento e não de manutenção. Observaram que os temas de arquitetura de software e padrões de projeto deveriam ter sido apresentados aos alunos antes de iniciarem os projetos, já que foi uma deficiência identificada nos trabalhos entregues.

O relato anterior foi publicado no ano 2000, o que mostra que a preocupação em ensinar manutenção de software não é absolutamente recente.

Em outro trabalho, dessa vez no Brasil, é descrito a experiência obtida com a implantação de uma disciplina de manutenção de software em um curso de pós-graduação lato sensu[17] (Dias, 2004).

Nesse curso, a disciplina de manutenção é descrita com o objetivo de formar profissionais para "(i) exercer a gestão das manutenções de software, compreendendo e avaliando as possibilidades de uso de tecnologias de última geração no ambiente das organizações; (ii) realizar pesquisas tecnológicas, na área da gestão de manutenções de software; e (iii) projetar, desenvolver e implantar projetos (diretamente relacionados com manutenção de software), buscando neles seu 'estado da arte' para propiciar incremento no desempenho profissional em organizações públicas e privadas".

O escopo definido para a disciplina é bem amplo e o autor descreve brevemente a ementa estabelecida para buscar os objetivos propostos. Percebe-se que os assuntos da ementa foram definidos com base nas experiências dos professores, em observações gerais do ambiente organizacional e na teoria clássica de engenharia de software.

A disciplina proposta tem duração de 20 horas, distribuídas em quatro aulas expositivas. Nesse período é apresentado, por exemplo, a classificação dos tipos de manutenção de software (corretiva, adaptativa, perfectiva, preventiva), e a cada aula, os alunos recebem um artigo em cima do qual devem preparar um relatório para a aula seguinte, destacando: resumo da essência do artigo, principais problemas apontados pelo autor, principais soluções consideradas e quais os questionamentos mais relevantes que surgem da

[17] Gestão de Tecnologias da Informação.

discussão do tema. A disciplina introduziu ainda o uso de uma lista de discussão via internet, para discussões de assuntos relevantes ao tema.

Entre os resultados, obtidos por meio de uma pesquisa com os egressos da disciplina, verificou-se que 50% dos alunos não tinham conhecimento da importância do tema, 100% disseram que passaram a se interessar pelo tema, 62,5% iniciaram a definição de um processo de manutenção em seu ambiente de trabalho; ocorreu um acréscimo de 18,18% no uso de ferramentas CASE para apoiar essa atividade; houve um crescimento de 14,29% em termos de controle gerencial dessa atividade; 33,33% dos alunos presenciaram a criação de um setor ou grupo específico para administrar as manutenções em suas empresas; 100% concordaram com a relevância do tema; e 100% julgaram a disciplina realmente necessária para o curso.

Um resultado interessante do relato anterior é a constatação da relevância que os egressos deram ao tema e as iniciativas de aplicação de conceitos teóricos de manutenção de software na prática.

6.6 Considerações Finais

Definir o conteúdo a ser abordado em uma disciplina de engenharia de software ainda não é um consenso entre os professores. Por ser um tema extremamente extenso, surgem focos ora em um ponto, ora em outro. No entanto, de uma forma geral, sabe-se ensinar engenharia de software nas universidades, em função também da própria experiência e cultura que já se adquiriu em torno do tema.

De maneira semelhante, o ensino de manutenção de software corresponde a outro ponto de divergências em relação ao ensino, que diferentemente da engenharia de software como um todo, ainda não possui um histórico de experiências que sustentem uma ou outra forma de ensinar.

Tendências internacionais apontam que o ensino de manutenção de software de forma específica, por meio de uma disciplina, não deve tardar a fazer parte de currículos de cursos de computação. Nesse sentido, buscar o melhor conteúdo a ensinar é um desafio ao qual este trabalho se propõe a ajudar, em uma tentativa de evitar que um conteúdo adequado demore a surgir dentro do meio acadêmico. O objetivo é trazer idéias que facilitem a formação adequada de profissionais aptos a lidar com manutenção de software mais facilmente.

CAPÍTULO

7

Perspectivas para uma Disciplina de Manutenção de Software

7.1 Considerações Iniciais

Neste capítulo são resumidas as propostas de resposta aos problemas de manutenção de software em diretrizes para a elaboração de uma disciplina específica sobre o assunto no meio acadêmico. O objetivo é apresentar caminhos que orientem quais pontos devem ser considerados com cuidado em uma futura implementação prática da disciplina.

A experiência do autor com manutenção de software representa uma característica favorável no momento de tecer indicações do conteúdo a ser abordado pela disciplina, pois permite confrontar o aprendizado que adquiriu nessa área durante seu curso de graduação com a experiência prática desde então obtida com manutenção de software.

Relativamente ao aprendizado durante a graduação, é notável a visão limitada que foi formada, pois apenas em alguns momentos dentro de disciplinas de desenvolvimento de software esse conteúdo foi abordado. De fato, ao experimentar os problemas reais no dia-a-dia de organizações que produzem e mantêm produtos de software em constante evolução, foi sensível a deficiência em entender alguns pontos. Primeiramente, existiu a deficiência em perceber a importância que a manutenção de software pode representar para a organização, obrigando-a a tratar o assunto de maneira independente e sistêmica. Em segundo lugar, não foi imediata a percepção do perigo de se criar uma falta de credibilidade da organização frente aos seus clientes, uma vez que reiteradamente problemas relacionados a alterações de software não fossem resolvidos ou fossem de maneira insatisfatória.

Já durante seu programa de pós-graduação, o autor teve contato mais aprofundado com o tema de manutenção de software, quando cursou disciplinas que focaram qualidade de software e processo de software, incluindo de maneira mais pausada e detalhada, a dinâmica e problemática da atividade, principalmente podendo confrontar a estrutura de normas e processos para desenvolvimento com aquelas para manutenção.

A união de todos esses fatores com a compreensão dos problemas exposta até aqui possibilitou a elaboração deste capítulo.

7.2 Visão Pedagógica

As idéias que envolvem as diretrizes para o ensino de manutenção de software estão cunhadas em um conceito pedagógico apresentado por Nunes (2005), focado na resolução de problemas. Essa abordagem pode ser utilizada para a elaboração de planos pedagógicos e fundamenta-se na verificação de quais problemas os egressos de um determinado curso se deparam quando ingressam no mercado de trabalho.

As formas tradicionais utilizadas para o projeto de planos pedagógicos são chamadas de abordagem *orientada a competências e habilidades* e abordagem *orientada a conteúdos*.

Zarifian (2001) explica que na abordagem orientada a competências e habilidades, o projetista do plano pedagógico procura visualizar as funções que o egresso precisará exercer, preocupando-se, portanto, com o desenvolvimento das habilidades que serão necessárias para isso. Essas funções podem ser tanto aquelas tradicionais, como outras criadas mais recentemente, em decorrência de progresso tecnológico. Nunes revela que muitas das funções tradicionais estão apresentadas na Classificação Brasileira de Ocupações (CBO)[18], como por exemplo: (i) administração de redes de computadores; (ii) administração de bases de dados; (iii) análise de sistemas etc. Cada uma dessas funções engloba capacidades que os egressos devem possuir, como o conhecimento de certas ferramentas, devendo o plano pedagógico oferecer o conhecimento necessário. Ainda segundo esse autor, trata-se de uma abordagem focada no mercado de trabalho, estando detalhada no texto de Zarifian e sugerida no Conselho Nacional de Educação (CNE)[19].

Especificamente sobre manutenção de software, existe uma breve referência nas descrições da CBO referente às atividades do *analista de sistemas*, porém na estrutura chamada de *tabela de atividades* (nome dado pela CBO), nada aparece. A tabela de atividades da profissão de analista de sistemas e afins pode ser visualizada no *apêndice B*.

[18] http://www.mtecbo.gov.br.
[19] Parecer nº.: CNE/CES 583/2001 - www.mec.gov.br.

Os cursos no Brasil que seguem a abordagem de orientação a competências e habilidades são aqueles chamados de profissionalizantes, independentemente da modalidade: bacharelado, seqüenciais de formação específica, tecnológico e técnico (Nunes, 2005).

De outro lado, a abordagem orientada a conteúdos ignora o mercado de trabalho, centrando a formação dos alunos nas disciplinas básicas de computação, principalmente matemática, computabilidade e algoritmos. Segundo essa visão, os egressos assim preparados estarão aptos a atuar com criatividade, resolvendo qualquer problema de computação no mercado de trabalho, sendo capazes ainda de produzir novas tecnologias. A idéia é que, produzindo novas tecnologias (ferramentas), esses egressos vão demandar outros com formação orientada por competências e habilidades, formando um ciclo.

Existem defensores das duas abordagens, sendo essa última adotada pela ACM e aplicada em seus c*omputing curricula*, orientação curricular para cursos de computação, editados anualmente. No Brasil, o Ministério da Educação e Cultura (MEC) também recomenda em suas *Diretrizes Curriculares da Área de Computação* a mesma abordagem. Finalmente, os currículos de referência para cursos de computação da SBC, citados no capítulo anterior (item 6.4) também são construídos de acordo com a idéia de orientação a conteúdos.

Os cursos de computação que adotam a orientação a conteúdos no Brasil são categorizados como *acadêmicos* e normalmente são da modalidade bacharelado (Nunes, 2005).

Por fim, a terceira abordagem para planos pedagógicos vem diretamente ao encontro das idéias deste trabalho: abordagem *orientada a problemas*, ou seja, são os problemas práticos que servirão de indicadores para a elaboração de um plano pedagógico.

A idéia nessa abordagem é especificar uma classe de problemas encontrada por egressos, e a partir dela fixar os conhecimentos necessários para o entendimento dos problemas e conseqüente resolução dos mesmos. Conforme explica Nunes, a solução dos problemas pode ser complementada pela construção de ferramentas (de propósito geral ou específico, dependendo da classe de problemas).

No caso deste trabalho, a mesma idéia é utilizada, ou seja, a verificação de quais são os problemas de manutenção de software que incidem no trabalho de profissionais dedicados a essa tarefa, produzindo então a *classe de problemas* sugerida pela metodologia. Tendo-se a classe, o passo seguinte é a fixação dos conhecimentos necessários e a construção de ferramentas. Neste trabalho, a fixação de conhecimentos e a construção de ferramentas se fundiram na busca de soluções e propostas para resolução dos problemas de manutenção, culminando com diretrizes para a elaboração de uma disciplina de manutenção de software.

Na abordagem de orientação a problemas, Nunes alerta que a idéia pode apresentar distorções caso os problemas sejam muito específicos de certo domínio, o que fatalmente levaria a soluções aplicáveis em um único contexto.

Comparando-se novamente com o propósito deste trabalho, o alerta do autor parece não ser preocupante, uma vez que os problemas de manutenção de software são normalmente os mesmos para diferentes épocas e domínios, abrangendo, portanto, dificuldades comuns e não específicas de uma aplicação.

7.3 Procedimentos para Disciplina de Manutenção de Software

Neste tópico é apresentada a estrutura utilizada para expor os assuntos que se mostraram indispensáveis a uma disciplina de manutenção de software focada na resposta a problemas de manutenção. Os subitens a seguir detalham essa estrutura.

7.3.1 Modelo Geral

Em sua última edição (março de 2006), o *computing curricula* apresenta um corpo de conhecimento em torno de currículos de cursos de computação, e o organiza hierarquicamente em disciplinas, módulos (essenciais e optativos) e tópicos.

De maneira semelhante, a proposta de conteúdo para uma disciplina de manutenção de software obedece a uma hierarquia de módulos e tópicos, como a sugerida pela ACM para engenharia de software. Na figura 7.1 é ilustrada essa hierarquização.

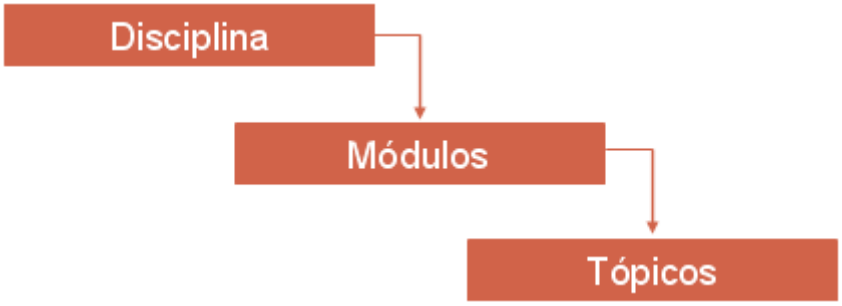

FIGURA 7.1: ESTRUTURA HIERÁRQUICA ADOTADA PARA OS ASSUNTOS DA DISCIPLINA

Em um nível inicial, tem-se a disciplina como um todo. No nível imediatamente seguinte surgem quatro módulos considerados essenciais, e para cada módulo existem conceitos a serem tratados, apresentados na forma de tópicos que se baseiam em: a) nas determinações da norma ISO/IEC 12207 (levando-se em consideração o exposto no capítulo 5), b) nos resultados do estudo de caso e c) nas recomendações apresentadas no item 5.4. Os tópicos representam os assuntos que são de fato o conteúdo que será tratado junto aos alunos.

Os tópicos de cada módulo se baseiam na abordagem pedagógica de *orientação a problemas*, apresentado no item 7.2 deste capítulo. Acredita-se que em razão da grande

importância prática da atividade, essa seja a abordagem mais correta, por basear-se no estudo e entendimento dos problemas reais.

7.3.2 Estruturação de Módulos

A especificação dos assuntos começa com a definição dos módulos. Para isso, optou-se pela definição de quatro grandes módulos: *Conceitos, Ambiente de Manutenção, Manutenibilidade Durante o Desenvolvimento* e *Manutenção no Produto Final de Software.* Cada um desses módulos é descrito no quadro 7.1, expondo-se o nome de cada módulo e seus objetivos gerais.

QUADRO 7.1: MÓDULOS PARA UMA DISCIPLINA DE MANUTENÇÃO DE SOFTWARE

Módulo A: *Conceitos*

Objetivos

No módulo inicial, *conceitos*, ocorre a aproximação inicial do aluno com o conteúdo que será abordado. Os tópicos desse módulo visam oferecer uma visão geral e abrangente da importância e significado da matéria. Isso inclui posicionar a manutenção de software dentro do ciclo de vida de software, apresentar suas características, importância prática e principalmente expor os problemas de manutenção. É importante que desde o início o aluno tenha contato com a natureza de tais problemas, caracterizada pela incidência maior de fatores de ordem gerencial do que de ordem técnica.

Módulo B: *Ambiente de Manutenção*

Objetivos

No módulo relacionado ao *ambiente de manutenção* são apresentados os recursos necessários ao cumprimento com sucesso da atividade de manutenção, tanto no que se refere aos fatores de infra-estrutura (ferramentas, recursos computacionais etc.), como aos fatores de ordem gerencial, por exemplo, processos e administração de pessoas. Um dos grandes focos é o de expor maneiras de tornar eficiente a comunicação, tanto interna como em relação aos clientes. Aliado a isso, esse módulo preocupa-se em mostrar que o registro e compartilhamento de experiências de manutenções passadas com novos membros de equipe, ou mesmo com outras equipes, é indispensável.

Módulo C: *Manutenibilidade Durante o Desenvolvimento*

Objetivos

No módulo destinado à *manutenibilidade durante o desenvolvimento*, o intuito maior é o de apresentar quais características precisam ser consideradas durante o desenvolvimento de software, para que se produzam facilidades posteriores durante atividades de manutenção. Aqui o apoio gerencial é fundamental, sendo necessário que a alta administração da organização possua um entendimento abrangente das características do ciclo de vida de um software. Termos chaves nesse módulo são: documentação, verificação e validação e qualidade de software.

Módulo D: *Manutenção no Produto Final de Software*

Objetivos

No último módulo, focado na *manutenção no produto final de software*, algumas questões fundamentais que apareceram no módulo anterior são retomadas. Documentação, verificação e validação e qualidade de software voltam em pauta, agora com uma visão diferente, focada na alteração de um produto de software já pronto. São incluídas aqui novas questões gerenciais e de equipe, como o problema da delegação de tarefas, administração de riscos, controle de mudanças, manutenção preventiva e principalmente análise e estudos sobre a viabilidade das mudanças e o impacto delas no sistema como um todo.

Uma vez que os módulos e seus principais objetivos foram determinados, os tópicos que cada um deverá conter podem ser estabelecidos.

A especificação dos tópicos para cada módulo é feita de forma a apresentar o tópico e a respectiva constatação que justificou sua existência. Essa constatação está baseada nas seguintes fontes:

- Teoria fundamental e conceituação básica necessária a qualquer disciplina;
- Resultados do estudo de caso apresentado no capítulo 3;
- Resultados do estudo da relação de problemas de manutenção de software com as recomendações dos processos da norma ISO/IEC 12207, apresentado no capítulo 5 (especificamente com base nas determinações das tarefas de cada processo);
- Sugestões de solução de outros autores, conforme apresentado também no capítulo 5;
- Experiência pessoal do autor na área de manutenção de software.

Os quatro itens a seguir detalham a estrutura de tópicos prevista para cada módulo, sendo que nas *considerações finais* deste capítulo é feita uma análise crítica da estrutura apresentada.

7.3.3 Módulo A – Conceitos

O módulo de conceitos, diferentemente dos demais, possui a maioria de seus tópicos fortemente ligados apenas à teoria fundamental e a conceituação básica, em razão da característica que possui de introdução ao tema.

No quadro 7.2 são apresentados os tópicos e suas respectivas fundamentações.

QUADRO 7.2: TÓPICOS DO MÓDULO A – CONCEITOS

Tópico	Fundamentação
Ciclo de vida de software	(TF)
Os diferentes significados de manutenção de software	(TF)
Tipos de manutenção de software	(TF)
Importância prática – sistemas legados	(TF)
Causas da evolução do software	(TF)
Apresentação e caracterização dos problemas	(TF), (EC), (OA)
Natureza dos problemas	(TF), (EC)

Legenda:

TF = Teoria Fundamental, **EC** = Estudo de Caso, **OA** = Outros Autores (vide item 5.4).

Cada tópico é descrito, em termos de conteúdo a abordar, nos itens a seguir.

- *Ciclo de vida de software*: os objetivos neste tópico incluem a apresentação do ciclo de vida de software e de suas características gerais, destacando a fase de manutenção e descontinuação de software. O intuito maior é que o aluno perceba em que ponto se encaixa o assunto que será estudado.
- *Os diferentes significados de manutenção de software*: o intuito principal neste tópico é diferenciar o significado do termo manutenção dentro do contexto de software e fora dele, expondo as peculiaridades e abrangência do termo quando aplicado a software.
- *Tipos de manutenção de software*: neste tópico o objetivo é mostrar as tradicionais classificações de manutenções (corretivas, adaptativas, perfectivas e preventivas), bem como os conceitos de manutenção reativa e pró-ativa.
- *Importância prática – sistemas legados*: os objetivos neste tópico englobam a apresentação do conceito de sistemas legados e das razões de sua manutenção e necessidade de continuidade de operação, apresentando-se características desses sistemas que dificultam a atividade de manutenção de software.
- *Causas da evolução do software*: exposição de conceitos de evolução de software. Por que o software evolui e quais as razões que obrigam a constante adaptação dos mesmos. Exemplificação de softwares que não precisam de manutenção (como aqueles que codificam alguma regra matemática imutável), e caracterização de softwares comerciais.
- *Apresentação e caracterização dos problemas*: apresentação dos problemas de manutenção de software descritos neste trabalho. A apresentação dos problemas deve incluir uma explicação sobre o significado de cada um e também a característica de relativa estabilidade dos tipos de problemas através dos anos.
- *Natureza dos problemas*: finalmente, após a exposição dos problemas de manutenção, neste último tópico são apresentadas as características desses problemas, expondo o fato de serem predominantemente de origem gerencial e não técnica, como pode indicar a intuição.

7.3.4 Módulo B – Ambiente de Manutenção

O módulo destinado ao ambiente de manutenção visa apresentar os recursos necessários para a execução com sucesso da atividade de manutenção. Esses recursos não se restringem a

ferramentas e máquinas, mas estão ligados principalmente ao apoio gerencial e à definição de processos adequados a diferentes etapas da manutenção.

No quadro 7.3 são mostrados os tópicos considerados para esse módulo.

QUADRO 7.3: TÓPICOS DO MÓDULO B – AMBIENTE DE MANUTENÇÃO

Tópico	Fundamentação
Gestão para manutenção	(TF), (OA), (6)
Equipe de manutenção	(TF), (EC), (OA), (11)
Gerenciamento de comunicação e do fluxo de conhecimento	(TF), (EC), (OA), (1), (2), (4), (6), (7), (9), (11), (13), (15), (16)
Definição, avaliação e provimento do ambiente de manutenção	(TF), (OA), (7), (11), (14)
Organização, treinamento e avaliação de equipes	(TF), (EC), (OA), (9), (11), (12)
Controle e gerência de configuração de software	(TF), (16)
Processos e padrões para manutenção	(TF), (4), (6), (7), (9)
Definição de recursos de suporte ao produto de software	(TF), (EC), (OA), (5)

Legenda (comum aos módulos *B*, *C* e *D*):

TF = Teoria Fundamental
EC = Estudo de Caso
OA = Outros Autores (vide item 5.4)
(1) = Processo de Elicitação de Requisitos (grupo de Engenharia)
(2) = Processo de Análise de Requisitos de Software (grupo de Engenharia)
(3) = Processo de Teste de Software (grupo de Engenharia)
(4) = Processo de Manutenção de Software e Sistema (grupo de Engenharia)
(5) = Processo de Suporte ao Cliente (grupo de Operação)
(6) = Processo de Alinhamento Organizacional (grupo de Gerência)
(7) = Processo de Gerência Organizacional (grupo de Gerência)
(8) = Processo de Gerência de Projeto (grupo de Gerência)
(9) = Processo de Gerência de Qualidade (grupo de Gerência)
(10) = Processo de Gerência de Riscos (grupo de Gerência)
(11) = Processo de Gerência de Recursos Humanos (grupo de Recursos e Infra-estrutura)
(12) = Processo de Treinamento (grupo de Recursos e Infra-estrutura)
(13) = Processo de Gerência do Conhecimento (grupo de Recursos e Infra-estrutura)
(14) = Processo de Infra-estrutura (grupo de Recursos e Infra-estrutura)
(15) = Processo de Documentação (grupo de Gerência de Configuração)
(16) = Processo de Gerência de Configuração (grupo de Gerência de Configuração)
(17) = Processo de Gerência de Solicitações de Mudança (grupo de Gerência de Configuração)

De maneira semelhante ao módulo anterior, a seguir é descrito o conteúdo geral que cada tópico deve abordar.

- *Gestão para manutenção*: neste tópico deve ser apresentado o problema existente em aplicar critérios de gestão não adequados ao contexto de software. É importante destacar que além de ser necessária uma gerência capaz de compreender as

peculiaridades do ciclo de vida de software, é também preciso considerar a manutenção de software desde as primeiras etapas do desenvolvimento. Essa postura visa contribuir tanto para a existência de um ambiente próprio para a manutenção, como também para o surgimento de uma consciência comum a respeito das dificuldades que poderão advir da necessidade de manter o software em funcionamento através dos anos.

- *Equipe de manutenção*: os objetivos neste tópico incluem apresentar a necessidade de existência de uma equipe específica para manutenção de software dentro da organização, bem como as características que essa equipe deve possuir. Entre essas características estão, por exemplo, qual o papel de cada membro e a necessidade de adoção de ferramentas destinadas a tratar questões específicas de manutenção de software.

- *Gerenciamento de comunicação e do fluxo de conhecimento:* apresentação da importância da comunicação eficiente entre membros de uma mesma equipe e entre equipes diferentes, principalmente no que se refere à troca de experiências e ao uso de soluções eficientes adotadas em situações passadas. Explorar os problemas das falhas de comunicação pode ser uma maneira eficaz de ensinar a importância da comunicação eficiente. A proposta de ferramentas para auxílio dessa necessidade também deve ser considerada.

- *Definição, avaliação e provimento do ambiente de manutenção*: o assunto neste tópico está ligado ao da gestão para manutenção e acaba sendo uma conseqüência do implemento daquele. Inclui o delineamento e monitorado do ambiente de manutenção, o que envolve verificar a necessidade de ferramentas, pessoal e ainda o desempenho dos mesmos do ponto de vista geral, como necessidades comuns do ambiente e não necessidades de treinamento de um único indivíduo. O ambiente de manutenção como um todo deve ser considerado.

- *Organização, treinamento e avaliação de equipes*: neste tópico, diferentemente do anterior, deve ser tratada a necessidade de avaliação individual e a proposição de programas de treinamento focado em pequenos grupos. O papel individual de cada indivíduo dentro da equipe deve ser considerado, e deve ser mostrada a necessidade de entender as habilidades individuais, para que se possa atrelar a cada um o papel que melhor se adeque às suas habilidades pessoais.

- *Controle e gerência de configuração de software*: expor conceitos básicos de configuração de software e como esses conceitos se aplicam a manutenção de

software. A idéia principal é mostrar as conseqüências de falhas na gerência de configuração e seu impacto no produto de software alterado.

- *Processos e padrões para manutenção*: o objetivo neste tópico é esclarecer, em um primeiro momento, a idéia de processo, apresentando, na medida do possível, conceitos e normas para esse fim. Em um passo seguinte, devem ser expostas alternativas menos rígidas e formais, mas também eficazes, para diminuir os problemas de manutenção de software, bem como para permitir maior controle sobre a atividade como um todo.
- *Definição de recursos de suporte ao produto de software*: neste tópico, o aluno deve ser conscientizado de que manter um software é uma tarefa muito árdua e freqüentemente responsável por grande desgaste na relação empresa-cliente. Tão importante quando desenvolver o software para o cliente é acompanhar o uso do mesmo e suas necessidades de ajustes. Do ponto de vista prático, a questão do suporte parece ser um dos pontos menos conhecidos por recém-formados. Muitos acreditam que é possível o uso de técnicas *mirabolantes* de desenvolvimento para produzir um software que nunca precisará de manutenção. A percepção da total impossibilidade disso muitas vezes só é conseguida com a prática, sendo de grande valor tentar antecipar esse fato aos alunos.

7.3.5 Módulo C – Manutenibilidade Durante o Desenvolvimento

No módulo de manutenibilidade durante o desenvolvimento, o foco está na idéia do *desenvolvimento para manutenção*. Isso significa que a preocupação agora deve ser diferente daquela do passado, ligada normalmente apenas a fatores próprios do desenvolvimento. Características que tornem o software mais fácil de ser modificado e ampliado, provavelmente em um futuro próximo à entrega ao cliente, devem ser consideradas e aplicadas ao software ainda em fase de desenvolvimento. Os requisitos do cliente deverão ser interpretados já com a abordagem de que em breve mudanças precisarão ser feitas.

Alguns tópicos são apresentados na forma de *parte 1* e *parte 2*. Essa divisão foi adotada por julgarem-se necessárias visões diferenciadas para um mesmo assunto em fases distintas do ciclo de vida de software.

Note-se, no entanto, que os tópicos apresentados a seguir são apenas aqueles que se mostraram importantes do ponto de vista de prevenção de problemas de manutenção de software, já que essa atividade somente aparecerá de fato quando o produto final for entregue. Fica claro que para uma ementa final de disciplina, outros assuntos devem ser abordados, e

que aqui não apareceram por não terem se revelado importantes *para o contexto considerado*. No quadro 7.4 estão listados esses tópicos.

QUADRO 7.4: TÓPICOS DO MÓDULO C – MANUTENIBILIDADE DURANTE O DESENVOLVIMENTO

Tópico	Fundamentação
Documentação (parte 1)	(TF), (OA), (15)
Aquisição e análise de requisitos	(TF), (1), (2), (8)
Definição e aplicação de critérios de testes (parte 1)	(TF), (EC), (2), (3), (4), (9), (17)
Definição e monitoramento de requisitos de qualidade (parte 1)	(TF), (OA), (9)
Verificação e validação de software (parte 1)	(TF), (1), (2), (3), (4), (5), (16), (17)

Legenda (comum aos módulos *B*, *C* e *D*):

TF = Teoria Fundamental
EC = Estudo de Caso
OA = Outros Autores (vide item 5.4)
(1) = Processo de Elicitação de Requisitos (grupo de Engenharia)
(2) = Processo de Análise de Requisitos de Software (grupo de Engenharia)
(3) = Processo de Teste de Software (grupo de Engenharia)
(4) = Processo de Manutenção de Software e Sistema (grupo de Engenharia)
(5) = Processo de Suporte ao Cliente (grupo de Operação)
(6) = Processo de Alinhamento Organizacional (grupo de Gerência)
(7) = Processo de Gerência Organizacional (grupo de Gerência)
(8) = Processo de Gerência de Projeto (grupo de Gerência)
(9) = Processo de Gerência de Qualidade (grupo de Gerência)
(10) = Processo de Gerência de Riscos (grupo de Gerência)
(11) = Processo de Gerência de Recursos Humanos (grupo de Recursos e Infra-estrutura)
(12) = Processo de Treinamento (grupo de Recursos e Infra-estrutura)
(13) = Processo de Gerência do Conhecimento (grupo de Recursos e Infra-estrutura)
(14) = Processo de Infra-estrutura (grupo de Recursos e Infra-estrutura)
(15) = Processo de Documentação (grupo de Gerência de Configuração)
(16) = Processo de Gerência de Configuração (grupo de Gerência de Configuração)
(17) = Processo de Gerência de Solicitações de Mudança (grupo de Gerência de Configuração)

A seguir, são descritos os objetivos gerais de cada tópico.

- *Documentação (parte 1)*: a primeira visão da documentação deve contemplar a necessidade de se construir um projeto bem elaborado do software, incluindo detalhar adequadamente pontos que são de natureza mais crítica ou de difícil compreensão da lógica empregada. Essa postura na confecção da documentação visa especialmente facilitar a manutenção do software, principalmente se essa tarefa for ser realizada por pessoas que não tiveram contato com o projeto inicial. Em suma, é preciso deixar claro que a documentação durante o desenvolvimento deve ser feita sempre se pensando nas pessoas que a utilizarão, por exemplo, dois anos mais tarde.

- *Aquisição e análise de requisitos*: neste tópico deverão ser apresentados os critérios formais necessários à aquisição e mudança de requisitos, bem como sua análise do ponto de vista de viabilidade de desenvolvimento. Isso significa mostrar maneiras de formalizar requisitos e de como validá-los junto ao cliente. O importante é passar aos alunos que antes do desenvolvimento a aquisição e análise de requisitos deve sempre seguir um processo formal que facilitará a percepção de necessidades futuras do cliente. Essa percepção é importante para que se possam prever no projeto do software mudanças evidentes ou potenciais nos requisitos, possibilitando que ainda durante o desenvolvimento sejam embutidas facilidades de manutenção no projeto.

- *Definição e aplicação de critérios de testes (parte 1)*: a idéia central neste tópico é a de mostrar que um produto de software entregue com erros acarretará a antecipação de necessidades de manutenção. Note que corrigir erros em um software já entregue constitui atividade de manutenção, segundo a definição da IEEE, ainda que o software tenha acabado de ser entregue. Por conseqüência, os alunos devem valer-se do conhecimento de engenharia de software para utilizarem técnicas de definição e aplicação de testes eficientemente.

- *Definição e monitoramento de requisitos de qualidade (parte 1)*: visando garantir a qualidade global do produto que será entregue, o aluno precisará tomar conhecimento, neste tópico, da existência de processos de qualidade e padrões para desenvolvimento. Novamente, garantir um produto com qualidade durante o desenvolvimento evitará que muitas manutenções sejam necessárias posteriormente.

- *Verificação e validação de software (parte 1)*: as principais fontes de necessidades de manutenção, nos primeiros momentos após a entrega do software, são deficiências na interpretação do que o cliente desejava e erros de codificação. Essa é a motivação principal para expor procedimentos de verificação (fazer o que de fato o cliente quer) e validação (fazer de maneira correta), no momento do desenvolvimento. É fácil perceber que com a verificação e validação realizada de maneira adequada, menos problemas surgirão no produto de software final. É importante destacar, no entanto, que ainda que fosse possível realizar essas tarefas de forma *perfeita*, isso não anularia as futuras necessidades de manutenção, já que essa é também uma necessidade que surge com a modificação do ambiente no qual o software trabalha.

7.3.6 Módulo D – Manutenção no Produto Final de Software

Finalmente, o módulo de manutenção no produto final de software deve considerar as respostas possíveis para a grande maioria dos problemas de manutenção de software conhecidos, já que é nesse momento que a maior parte desses problemas surge.

No quadro 7.5 estão apresentados os tópicos considerados necessários para esse módulo com suas respectivas fundamentações.

QUADRO 7.5: TÓPICOS DO MÓDULO D – MANUTENÇÃO NO PRODUTO FINAL DE SOFTWARE

Tópico	Fundamentação
Documentação (parte 2)	(TF), (OA), (15)
Definição e aplicação de critérios de testes (parte 2)	(TF), (EC), (2), (3), (4), (9), (17)
Definição e monitoramento de requisitos de qualidade (parte 2)	(TF), (OA), (9)
Verificação e validação de software (parte 2)	(TF), (1), (2), (3), (4), (5), (16), (17)
Delegação e administração de tarefas	(TF), (EC), (OA), (8)
Acompanhamento das manutenções	(TF), (8), (9)
Obtenção e análise de solicitações de mudança	(TF), (1), (2), (4), (5), (8), (17)
Planejamento de administração de riscos	(TF), (10)
Manutenção preventiva	(TF), (9)
Planejamento de manutenção	(TF), (EC), (OA), (4), (8), (17)
Definição de política de troca de versões	(TF), (4)
Medição de nível de satisfação do cliente	(TF), (5), (9)

Legenda (comum aos módulos *B*, *C* e *D*):

TF = Teoria Fundamental
EC = Estudo de Caso
OA = Outros Autores (vide item 5.4)
(1) = Processo de Elicitação de Requisitos (grupo de Engenharia)
(2) = Processo de Análise de Requisitos de Software (grupo de Engenharia)
(3) = Processo de Teste de Software (grupo de Engenharia)
(4) = Processo de Manutenção de Software e Sistema (grupo de Engenharia)
(5) = Processo de Suporte ao Cliente (grupo de Operação)
(6) = Processo de Alinhamento Organizacional (grupo de Gerência)
(7) = Processo de Gerência Organizacional (grupo de Gerência)
(8) = Processo de Gerência de Projeto (grupo de Gerência)
(9) = Processo de Gerência de Qualidade (grupo de Gerência)
(10) = Processo de Gerência de Riscos (grupo de Gerência)
(11) = Processo de Gerência de Recursos Humanos (grupo de Recursos e Infra-estrutura)
(12) = Processo de Treinamento (grupo de Recursos e Infra-estrutura)
(13) = Processo de Gerência do Conhecimento (grupo de Recursos e Infra-estrutura)

(14) = Processo de Infra-estrutura (grupo de Recursos e Infra-estrutura)
(15) = Processo de Documentação (grupo de Gerência de Configuração)
(16) = Processo de Gerência de Configuração (grupo de Gerência de Configuração)
(17) = Processo de Gerência de Solicitações de Mudança (grupo de Gerência de Configuração)

Da mesma forma que nos módulos anteriores, a seguir apresenta-se a descrição geral do conteúdo de cada tópico.

- *Documentação (parte 2)*: na segunda visão sobre a documentação, algumas considerações diferentes devem ser apresentadas com relação à maneira como ela é abordada durante o desenvolvimento. Durante a manutenção, é preciso considerar alternativas de documentação para alterações pontuais simples, facilitando essa tarefa, já que esse tipo de manutenção é normalmente o mais freqüente. Para alterações maiores, processos formais precisam ser considerados como no desenvolvimento. Outro ponto fundamental é decidir por qual processo a documentação geral do projeto será atualizada, e quem serão os responsáveis por essa tarefa. É importante destacar que a desatualização da documentação causará dificuldades progressivas para a manutenção.
- *Definição e aplicação de critérios de testes (parte 2)*: o objetivo neste tópico é apresentar a necessidade de se estabelecer critérios bem definidos para testes em manutenções efetuadas. Os testes pontuais na alteração é apenas uma das obrigações. Essa condição se justifica pelo fato de que manutenções freqüentemente produzem efeitos colaterais em outras partes do software, e os alunos precisam estar cientes disso para estabelecerem tanto testes pontuais, como de integração no produto alterado. É interessante esclarecer que o efeito cascata de alterações é um dos principais motivos de geração de desconfiança do cliente em relação ao software.
- *Definição e monitoramento de requisitos de qualidade (parte 2)*: neste tópico deve ser tratada a exposição de quais são os requisitos desejáveis de qualidade nas manutenções e como acompanhar seu efetivo cumprimento, propondo-se correções conforme necessárias. Novamente, é importante que o aluno esteja ciente que manutenções mal validadas gerarão novas necessidades de manutenção, assim como desconfiança do cliente.
- *Verificação e validação de software (parte 2)*: neste tópico a verificação e validação de software é retomada, agora sob a perspectiva de um produto final de software em manutenção. O principal intuito é mostrar que o entendimento incompleto ou errôneo da necessidade de manutenção do cliente pode ocasionar esbanjamento de tempo e

recursos. Aqui a verificação e validação assume um papel ainda mais relevante do que durante o desenvolvimento, isso porque o impacto negativo de alterações fora de conformidade com o que o cliente precisa pode produzir efeitos negativos muito grandes, uma vez que o cliente provavelmente já apresenta dependência em relação às funcionalidades do software. Essa visão de dependência em relação ao produto de software estável e correto deve ser trabalhada junto aos alunos.

- *Delegação e administração de tarefas*: neste tópico o intuito é expor as conseqüências negativas da sobrecarga de tarefas, principalmente quando tarefas de desenvolvimento e de manutenção são atribuídas às mesmas pessoas. O aluno deve aprender que a manutenção precisa ter equipe e ambiente computacional separado do de desenvolvimento. O uso de ferramentas para auxiliar a classificação da complexidade das alterações é uma alternativa viável para melhorar a distribuição de tarefas.

- *Acompanhamento das manutenções*: a intenção do acompanhamento de manutenções está centrada no estabelecimento de mecanismos de controle das alterações em andamento, principalmente do ponto de vista de cumprimento de cronogramas. O acompanhamento deve propor medidas de correção ou alternativas quando uma manutenção se mostrar inviável em meio à codificação ou os prazos forem comprometidos por atividades paralelas ou complexidade superior à estimada. O controle deve considerar o tempo necessário também para a realização de testes, já que as manutenções não poderão ser entregues sem os testes adequados.

- *Obtenção e análise de solicitações de mudança:* neste tópico deve ser abordada, em um primeiro momento, a discussão de alternativas para a obtenção dos requisitos de mudanças do cliente. Devem ser citados os problemas de se utilizar e-mail, telefone, fax etc. Documentos padronizados pode ser uma alternativa, porém é necessária a avaliação de como será a troca desses documentos com o cliente (envio por e-mail seguido de preenchimento e retorno via fax não é uma boa opção - isso é justamente uma das fontes de problemas, pois é muito fácil a perda dessas informações). O ideal é o uso de ferramentas com registro em bases de dados, principalmente fornecendo uma forma de o cliente acompanhar o status de seus pedidos (normalmente com o uso de sistemas baseados na *web*). Em um segundo momento devem ser apresentados critérios de como analisar a viabilidade dos pedidos do cliente. Os alunos devem entender que a possibilidade de se perder a estabilidade do produto de software é alta quando manutenções mal validadas são efetuadas, portanto uma boa política é a de sempre considerar alternativas antes de proceder-se com as modificações no código-

fonte. É claro que isso nem sempre será possível e os casos que de fato exigirem manutenções precisam ser bem entendidos e analisados do ponto de vista de efeitos colaterais em outros módulos do software.

- *Planejamento de administração de riscos*: neste tópico busca-se mostrar que, assim como no desenvolvimento, na atividade de manutenção riscos também existem e o planejamento de resposta aos mais conhecidos representa uma garantia de maior estabilidade para a organização. Dificuldades inesperadas de manutenção, atrasos, mudanças de prioridades, manutenções urgentes não previstas e rotatividade são exemplos de riscos potenciais para a manutenção de software que devem ser considerados em um planejamento por parte da gerência.

- *Manutenção preventiva*: neste tópico deve ser apresentado aos alunos o conceito de manutenção preventiva e a exposição de suas principais técnicas. Além disso, os objetivos desse tópico incluem criar uma visão crítica do aluno sobre o tema. Isso significa explicar a necessidade de recursos humanos extras e a complexidade para trabalhar a manutenção preventiva, tornando necessário sempre avaliar os objetivos estratégicos da organização em relação ao seu produto de software, para a partir desse entendimento decidir sobre preocupar-se ou não com manutenção preventiva.

- *Planejamento de manutenção*: neste tópico os conceitos de planejamento de manutenção devem ser apresentados. Tais conceitos incluem considerar a atualização da documentação, a delegação de tarefas, o controle da atividade e a realização de testes. Além do planejamento técnico, é necessário também o planejamento da comunicação com o cliente, da definição de cronograma e de possíveis versões de teste. Todos os conceitos e características relevantes a esses pontos devem ser trabalhados com os alunos.

- *Definição de política de troca de versões*: neste tópico deve ser exposta a necessidade de previamente se estabelecer qual processo será utilizado para a disponibilização e troca de versões, informando o cliente o que ficará a seu cargo e o que será responsabilidade da organização. Várias formas de tratar essa disponibilização podem ser aplicadas, sendo as mais comuns a disponibilização de *download* via internet ou o uso de *CD-ROM*. Dependendo da infra-estrutura do cliente, a opção por download pode não ser viável, devendo ser consideradas outras alternativas. O aluno precisa tomar conhecimento dos inúmeros problemas que podem surgir com a troca de versões, como por exemplo, o cliente não saber como proceder e a organização não possuir alguém para apoiá-lo nessa tarefa. Problemas como esse podem dar a

impressão ao cliente de que é a sua solicitação de manutenção que não está sendo disponibilizada, quando na verdade se trata de uma dificuldade com a atualização de seu software.

- *Medição de nível de satisfação do cliente*: neste tópico o aluno deve perceber que a manutenção de software pode ser considerada também como um fator de competitividade. A esperada satisfação do cliente com a entrega do produto final após desenvolvimento seguindo padrões e processos bem definidos, somente poderá ser conservada com o suporte, e conseqüentes manutenções necessárias. Estabelecer formas de medir o nível de satisfação do cliente após meses, ou anos, da entrega do software favorecerá a adoção de medidas, principalmente relacionadas com manutenção e suporte, para manter uma relação positiva da organização com o cliente. Uma forma de medir essa satisfação pode ser com um sistema baseado em *web* que permita ao cliente marcar, para cada pedido de manutenção efetuado, a sua impressão com relação à solução oferecida. Por exemplo, para cada manutenção registrada e concluída, o cliente poderia ter a sua disposição a possibilidade de selecionar entre algumas opções de resposta, a que melhor refletisse seu sentimento com relação ao atendimento oferecido para a manutenção.

7.3.7 Resumo da Estrutura

A estruturação dos assuntos e tópicos considerados relevantes para uma disciplina de manutenção de software, em face da metodologia de resposta aos problemas práticos de manutenção, está resumida na figura 7.2.

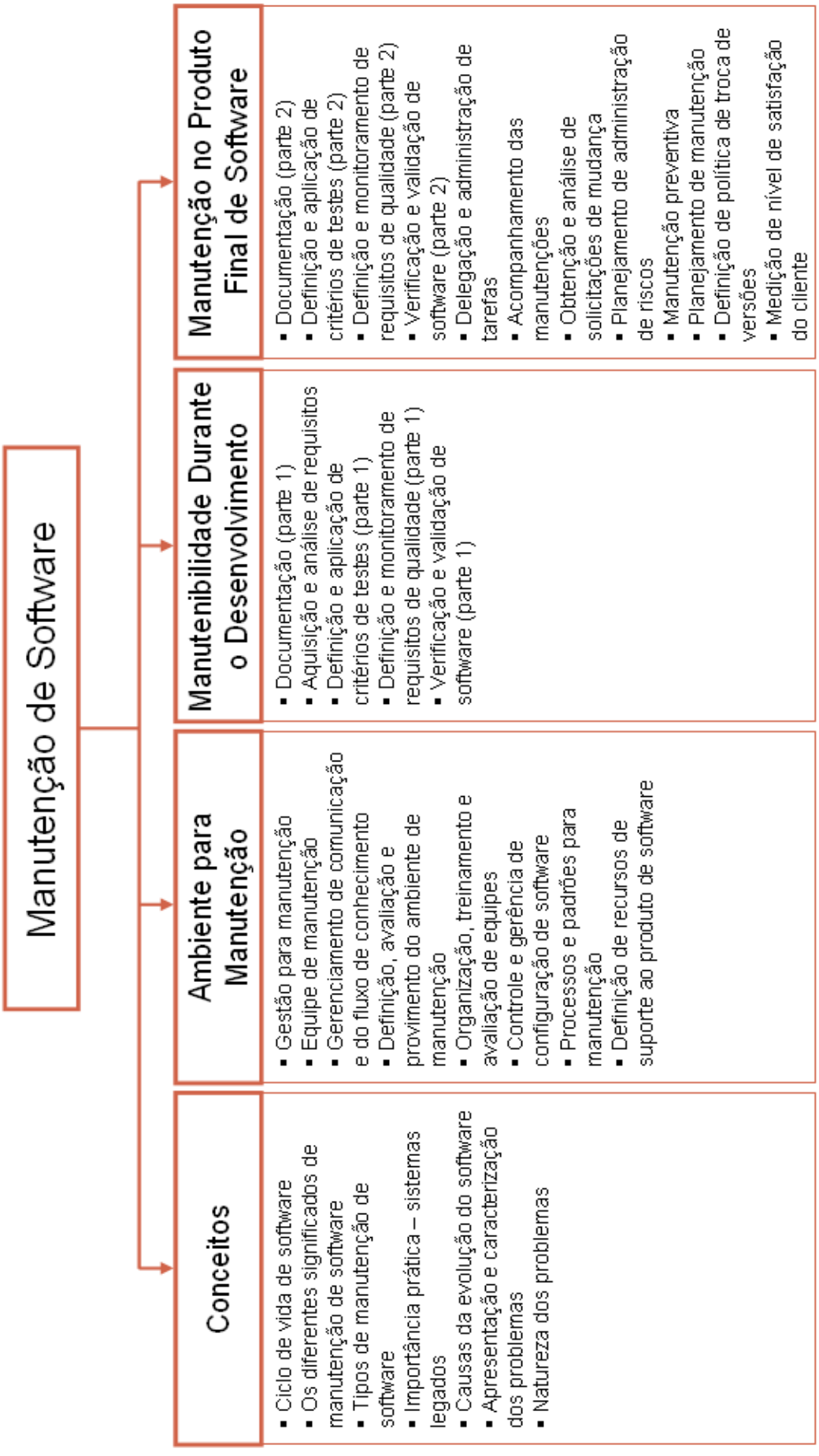

FIGURA 7.2: VISÃO GERAL DA ESTRUTURA DE MÓDULOS E TÓPICOS

7.4 Considerações Finais

Uma das características fundamentais da estrutura de módulos e tópicos apresentada refere-se ao fato de que ela expõe aqueles assuntos que foram considerados *essenciais* a uma futura disciplina de manutenção de software baseada na resposta a problemas. A estrutura não corresponde, portanto, a uma ementa definitiva, sendo tão somente um guia de quais assuntos devem constar em uma ementa final para essa disciplina.

A opção por dividir alguns tópicos em *parte1* e *parte2* resultou da observação de que em diferentes etapas do ciclo de vida de software, diferentes abordagens devem ser atribuídas a um mesmo assunto, mostrando-se necessário discutir com os alunos esse fato. Por exemplo, apresentar conceitos de documentação de maneira geral parece ser menos eficiente do que apresentar abordagens diferenciadas levando-se em conta o fato de o software ainda estar em desenvolvimento ou já ter sido entregue ao usuário final.

Um ponto não considerado foi o da adoção de projetos práticos para a fixação e treinamento dos conceitos expostos durante a disciplina. Seguramente o uso de projetos práticos é recomendável e já se mostrou adequado em muitos momentos para o ensino de engenharia de software. No entanto, a estipulação de quais trabalhos desenvolver e como organizar equipes de alunos para esse propósito foge do escopo deste trabalho, que se fundamenta principalmente no diagnóstico dos assuntos necessários para o contexto de disciplina centrada na resposta a problemas de manutenção.

CAPÍTULO

8

Conclusão

8.1 Considerações Gerais

O crescimento do número de sistemas legados e a dificuldade para mantê-los adequados às necessidades das organizações foram a principal observação que guiou a elaboração deste trabalho.

Percebe-se mais do que nunca como o mundo dos softwares é dinâmico e as abordagens modificam-se através dos tempos. Partindo de uma euforia inicial por desenvolvimento e formas de melhorar esse desenvolvimento, atinge-se agora um estágio a mais, um crescente equilíbrio com outra atividade do ciclo de vida de software: a manutenção.

Isso não significa que o desenvolvimento esteja com sua importância sendo reduzida. Pelo contrário, não param de surgir novas idéias, ferramentas e metodologias, algumas vezes com propostas inovadoras, para facilitar o desenvolvimento e a própria evolução de software. A idéia de valer-se de componentes de software reutilizáveis em diferentes projetos é um exemplo da tentativa da comunidade de pesquisadores de favorecer ainda mais o desenvolvimento de software.

O fato é que se percebe agora o surgimento de trabalhos e pesquisas voltadas exclusivamente para manutenção de software, o que inclui o surgimento de eventos (como congressos) específicos para esse fim. E isso com certeza é fruto da verificação de que já se conta com um volume grande de software em pleno funcionamento pelo mundo e que precisará continuar em funcionamento por tempo indeterminado. Como prover vitalidade a

esse legado que a evolução da computação produziu, e que hoje são tratados como bens das organizações, representa o foco de muitas pesquisas recentes.

8.2 Contribuições

As contribuições deste trabalho incluem a apresentação e estudo de fatos relacionados à atividade de manutenção de software, principalmente dos problemas ligados a essa atividade, fornecendo parâmetros inicias para guiar a confecção de uma disciplina de manutenção de software em graduações na área de computação.

A apresentação dos problemas típicos de manutenção de software e seu confronto com a norma ISO/IEC 12207 é um dos pontos fortes deste trabalho, pois se a norma é considerada um excelente guia para a prática controlada e ajustável do processo de desenvolvimento de software, defrontá-la com os problemas de manutenção é uma forma de identificar o que vem sendo feito errado nas organizações.

A esse entendimento das causas dos problemas de manutenção une-se a proposta dos assuntos que devem ser tratados em uma futura disciplina de manutenção de software. Esses assuntos são reflexos imediatos das constatações das razões principais dos problemas de manutenção apresentados, e acredita-se que devam ser o ponto de partida para testes com uma disciplina de manutenção de software.

Mais do que a disciplina em si, o propósito maior deste trabalho consiste em motivar e conscientizar seus futuros leitores para a importância da manutenção de software, que ironicamente ainda é colocada de lado em muitas organizações, apesar de elas mesmas sofrerem com a falta de controle sobre essa atividade.

8.3 Trabalhos Futuros

O trabalho futuro mais evidente é o de implementar na prática uma disciplina de manutenção de software que use como parâmetros as indicações e observações apresentadas. Uma análise posterior, tanto da receptividade dos alunos, como dos resultados finais, inicialmente com uma pesquisa entre os egressos, produziria uma nova fonte para análise, possibilitando um caminho para o estabelecimento em definitivo de uma ementa adequada para a disciplina.

Outra possibilidade seria reproduzir os passos do atual trabalho, mas focando-se em alguma característica do processo de desenvolvimento de software ou de seu ciclo de vida. Um exemplo seria reproduzir este trabalho aplicando-o à qualidade de software. Primeiramente, seriam elencados os critérios de qualidade de software desejáveis, e isso pode incluir tanto critérios funcionais como não funcionais, como por exemplo, a usabilidade, manutenibilidade, portabilidade, etc. A partir dos requisitos desejáveis, o passo seguinte seria

investigar quais deles são comumente não atendidos, ou atendidos de maneira insuficiente, nos softwares criados para um determinado domínio. De forma semelhante a este trabalho, essa investigação também traria indícios de como tratar o ensino, dessa vez de qualidade de software, que seria considerado de maneira focada nos problemas de qualidade verificados na indústria, e não apenas baseado em teoria e conceitos acadêmicos.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS, J. H.; LUTFIYYA, H. L. (2000) "Experiences with a software maintenance project course", *IEEE Transactions on Software Engineering (TSE)*, v. 43, n. 4, p. 383-388.

BASILI, V. (1990) "Viewing Maintenance as Reuse-Oriented Software Development", *IEEE Software*, v. 7, n. 1, p. 19-25.

BECK, K. (1999) "Extreme Programming Explained", First Edition. Addison-Wesley.

BENNETT, K. H.; RAJLICH, V. T. (2000) "Software maintenance and evolution: a roadmap", In: *Conference on The Future of Software Engineering*, Limerick, Ireland, June.

BENNETT, K. H.; RAMAGE, M.; MUNRO, M. (1999) "Decision Model for Legacy Systems", *IEEE Proceedings on Software (TSE)*, v.146, n. 3, p. 153-159.

BHATT, P.; SHROFF, G.; MISRA, A. K. (2004) "Dynamics of software maintenance", *ACM SIGSOFT Software Engineering Notes*, v. 29, n. 5, p. 1-5.

BHATT, P.; WILLIAMS, K.; SHROFF, G.; MISRA, A. K. (2006) "Influencing Factors in Outsourced Software Maintenance", *ACM SIGSOFT Software Engineering Notes*, v. 31, n. 3, p. 1-6.

BLASI, P. J. (1999) "Overview of ACM". Disponível em: <http://www.acm.org/about_acm/ov.html>. Acesso em: 09 mai. 2006.

BOEHM, B.; BASILI, V. (2001) "Software Defect Reduction Top 10 List", *Computer*, v. 34, n. 1.

CARDOW, J.E. (1992) "Can software maintenance be taught?", In: *Conference on Software Maintenance*, Orlando, FL, USA, November.

CASTRO, J. F. B.; GIMENES, I. M. S.; MALDONADO, J. C. (2000) "Uma proposta de Plano Pedagógico para a matéria Engenharia de Software", In: *II Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática*, Curitiba, PR, Brasil, Julho.

CHAPIN, N. (1986) "Software maintenance: A different view", In: *Conference on Software Maintenance*, Orlando, FL, USA, November.

COSTA, C. M.; AUDY, J. L. N.; MAZZUCCO, J.; FURTADO, J. V. (2000) "Plano Pedagógico para cursos de Bacharelado em Sistemas de Informação", *Sociedade Brasileira de Computação* (SBC). Disponível em <http://www.sbc.org.br/index.php?subject=39>. Acesso em: 09 mai. 2006.

DAHMER, A.; SANTOS, B. S. DOS; OGIBA, S.; KIST, T. (2002) "Uma Proposta de Plano Pedagógico para o Curso de Licenciatura em Computação", *Sociedade Brasileira de Computação* (SBC). Disponível em: <http://www.sbc.org.br/index.php?subject=39>. Acesso em: 14 mai. 2006.

DART, S.; CHRISTIE, A. M.; BROWN, A. W. (2001) "A Case Study in Software Maintenance", *Technical Report*, Carnegie Mellon University.

DEKLEVA, S. M. (1990) "Annual Software Maintenance Survey: Survey Results", *Software Maintenance Association*, Vallejo, California, USA.

_____. (1992) "Delphi study of software maintenance problems", In: *Conference on Software Maintenance*, Orlando, FL, USA, November.

DE LUCIA, A., POMPELLA, E., STEFANUCCI, S. (2004) "Effort estimation for corrective software maintenance". In: *14th International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy, July.

DIAS, M. G. B. (2004) "Uma experiência no ensino de manutenção de software", In: *Workshop de Manutenção de Software Moderna (WMSWM'04)*, Brasília, DF, Brasil, Outubro.

FERREIRA, A. P. L.; BATTAIOLA, A. L.; SOUZA, F. F.; TORI, R. (2001) "Proposta de Plano Pedagógico: Bacharelado em Ciência da Computação", *Sociedade Brasileira de Computação* (SBC). Disponível em: <http://www.sbc.org.br/index.php?subject=39>. Acesso em: 09 mai. 2006.

GHEZZI, C.; MANDRIOLI, D. (2005) "The Challenges of Software Engineering Education", In: *27th International Conference on Software Engineering*, Saint-Louis, Missouri, USA, May.

HAZZAN, O.; DUBINSKY, Y. (2003) "Teaching a Software Development Methodology: The case of Extreme Programming", In: *16th Conference on Software Engineering Education and Training (CSEE&T 2003)*, Madrid, Spain, March.

IEEE (1998) "Std 1219 – IEEE Standard for Software Maintenance", *Institute of Electrical and Eletronic Engineers*, New York, NY, USA.

ISO/IEC 12207 (1998) "Standard for Information Technology - Software Lifecycle Processes", *International Standard Organization*, New York, NY, USA.

ISO/IEC 12207 (2002) "Standard for Information Technology - Software Lifecycle Processes/Amd.1", *International Standard Organization*, New York, NY, USA.

ISO/IEC 12207 (2004) "Standard for Information Technology - Software Lifecycle Processes/Amd.2", *International Standard Organization*, New York, NY, USA.

ISO/IEC 15504 (2003) "Software Process Assessment", *International Standard Organization*, New York, NY, USA.

KOSKINEN, J.; SALMINEN, A.; PAAKKI, J. (2004) "Hypertext support for the information needs of software maintainers", *Journal of Software Maintenance and Evolution: Research and Practice*, v. 16, n. 3 , p. 187-215.

KUNG, H.; HSU, C. (1998) "Software Maintenance Life Cycle Model", In: *International Conference on Software Maintenance*, Bethesda, Maryland, USA.

LEHMAN, M. M. (1974) "Programs, Cities, Students, Limits to Growth?", In: *Imperial College of Science Technology*, London, England, May.

_____. (1980) "On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle", *Journal of Systems and Software (JSS)*, v. 1, n. 3, p. 213-221.

_____. (1991) "Software Engineering, the Software Process and their Support", *IEEE Software Engineering Journal: Special Issues on Software Environments and Factories*, v. 1, n. 3, p. 213-221.

_____. (1996) "Laws of Software Evolution Revisited". In: *5th European Workshop on Software Process Technology*, Nancy, France, October.

LIENTZ, B. P.; SWANSON, E. B. (1980) "Software Maintenance Management", *Reading, MA*, Addison-Wesley.

NAUR, P.; RANDELL, B. (1968) "Software Engineering: Report", In: *Conference of NATO Science Committee*, Garmisch, Germany, October.

NIESSINK, F. (1999) "Software Maintenance Research in the Mire?", In: *Annual Workshop on Empirical Studies of Software Maintenance (WESS'99)*, Oxford, United Kingdom, September.

NUNES, J. D. (2005) "Projetos de Planos Pedagógicos Orientados a Problemas", *Biblioteca Digital da SBC*. Disponível em << http://www.sbc.org.br/bibliotecadigital/download.php?paper=218>>. Acesso em: 20 jan. 2007.

PADUELLI, M.; SANCHES, R. (2006) "Problemas em manutenção de software: caracterização e evolução", In: *III Workshop de Manutenção de Software Moderna*, Vila Velha, ES, Brasil, Maio.

PFLEEGER, S. L. (2001) "Software Engineering: theory and practice". Second Edition, New Jersey, Prentice Hall.

PIGOSKI, T. M. (1996) "Practical Software Maintenance: Best Practices for Managing Your Software Investment", Willey Computer Publishing.

POLO, M.; PIATTINI, M.; RUIZ, F.; CALERO, C. (1999) "Roles in the maintenance process", *ACM SIGSOFT Software Engineering Notes*, v. 24, n. 4, p. 84-86.

POLO, M.; PIATTINI, M.; RUIZ, F. (2003) "Using a qualitative research method for building a software maintenance methodology", *Software – Practice and Experience*, v.32, n. 13, p. 1239–1260.

PRESSMAN, R. S. (2005) "Software Engineering: a practitioner's approach", 6.ed., McGrawHill Higher Education.

RODRIGUEZ, E. (2004). "Overview of ACM / Education". Disponível em: <http://www.acm.org/about_acm/ov_edu.html>. Acesso em: 09 mai. 2006.

SHACKELFORD, R.; CROSS, J. H.; DAVIES, G.; IMPAGLIAZZO, J.; KAMALI, R.; LEBLANC, R.; LUNT, B.; MCGETTRICK, A.; SLOAN, R.; TOPI, H. (2004) "Computing Curricula 2004: A Guide to Undergraduate Degree Programs in Computing", *Joint Task Force for Computing Curricula 2004*, The Association for Computing (ACM), November.

SILVA, L.; SAYÃO, M.; LEITE, J. C. S. P.; BREITMAN, K. (2003) "Enriquecendo o código com cenários", In: *Simpósio Brasileiro de Engenharia de Software (SBES)*, Manaus, AM, Brasil, Outubro.

SILVA, L. DE P.; SANTANDER, V. F. A. (2004) "Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software", In: *Workshop em Engenharia de Requisitos*, Tandil, Argentina, Dezembro.

SINGH, R. (1996). "International Standard ISO/IEC 12207 Software Life Cycle Processes", *Software Process Improvement and Practice*, vol. 2, p. 35–50.

SNEED, H. M. (2003) "Critical Success Factors in Software Maintenance", In: *International Conference on Software Maintenance*, Amsterdam, The Netherlands, September.

SOARES, M. DOS S. (2005) "Uma experiência no ensino de Engenharia de Software orientada a trabalhos práticos", In: *Workshop de Educação em Informática (WEI2005),* Rio de Janeiro, RJ, Brasil, Novembro.

SOMMERVILLE, I. (2003) "Engenharia de Software", 6.ed., São Paulo, Addison Wesley.

SOUZA, S. C. B. DE; NEVES, W. C. G. DAS; ANQUETIL, N.; OLIVEIRA, K. M. DE. (2004) "Documentação Essencial para Manutenção de Software II", In: *I Workshop de Manutenção de Software Moderna*, Brasília, DF, Brasil, Outubro.

STILLER, E.; LEBLANC, C. (2002) "Effective Software Engineering Pedagogy", *Journal of Computing Sciences in Colleges*, v. 17, n. 6, p. 124-134.

SWEBOK (2004) "The Institute of Electrical and Electronics Engineers, Inc – IEEE", *Guide to the Software Engineering Body of Knowledge*, Version 2004.

TEIXEIRA, C. A. C.; MARTINS, J. S. B.; PRADO, A. F.; JÚNIOR, O. M.; GEYER, C. F. R.; AZEREDO, P. A. (2002) "Um Plano Pedagógico de Referência para Cursos de Engenharia de Computação", *Sociedade Brasileira de Computação* (SBC). Disponível em <http://www.sbc.org.br/index.php?subject=39>. Acesso em: 09 mai. 2006.

TOMAYKO, J. E. (1987) "Teaching a Project-Intensive Introduction to Software Engineering", *Technical Report*, Carnegie Mellon Institute.

ULRICH, W. M. (1990) "The evolutionary growth of software reengineering and the decade ahead". *American Programmer*, v. 3, n. 10, p. 14-20.

VISAGGIO, G. (2001) "Assessing the Maintenance Process Through Replicated, Controlled Experiment". *Journal of Systems and Software*, v. 44, n. 3, p. 187-197.

_____. (2001) "Ageing of a data-intensive legacy system: symptoms and remedies". *Journal of Software Maintenance and Evolution: Research and Practice*, v. 13, n. 5, p. 281-308.

ZARIFIAN, P. (2001) "Objetivo competência: por uma nova lógica". Tradução Maria Helena C. V. Trylinski, São Paulo, Atlas.

ZVEGINTZOV, N.; PARIKH, G. (2005) "60 years of Software Maintenance: Lessons Learned", In: *21st IEEE International Conference on Software Maintenance (ICSM 2005)*, Budapest, Hungary, September.

YOURDON, E. (1992) "Análise Estruturada Moderna", tradução da terceira edição, Rio de Janeiro, Campus.

APÊNDICE A

Questionários utilizados no estudo de caso

A seguir são apresentados os questionários utilizados para obter parte dos dados analisados no estudo de caso do *capítulo 3*.

Por não se conhecer previamente o tamanho das respostas, e visando evitar que a estrutura do questionário pudesse limitá-las, as questões foram listadas em uma página e as respostas eram dadas em páginas seguintes.

Para a confecção dos questionários, os seguintes pontos foram considerados:

- Apresentação dos objetivos de cada um;
- Perguntas diretas, evitando-se textos longos;
- Questionários com número reduzido de questões.

No quadro A.1 é apresentado o questionário utilizado para a coleta de dados da primeira etapa do estudo de caso (características da atividade de manutenção de software na organização e principais problemas envolvidos).

QUADRO A.1: QUESTIONÁRIO DA PRIMEIRA ETAPA DO ESTUDO DE CASO

QUESTIONÁRIO

OBJETIVOS: Coletar dados referentes à maneira como a equipe de manutenção de software trabalha, obtendo respostas que influenciarão diretamente a elaboração de uma relação de problemas característicos da atividade de manutenção de software na organização.

PARTE 1: Características do gerenciamento da atividade de manutenção.

1. Como um cliente se manifesta para apresentar uma necessidade de manutenção de software? (*e-mail, telefone, algum sistema específico de help-desk etc.*)

2. Uma vez que uma alteração de software foi requisitada pelo cliente, qual o processo usado para avaliar tal solicitação? (*todo pedido do cliente é atendido?*)

3. Para as solicitações de manutenção, como é tratado o acordo de datas de entrega com o cliente? (*existe algum tipo de programação diferenciado por nível de urgência da solicitação ou por importância do cliente?*)

4. Considerando que nem sempre é clara a comunicação entre o responsável por obter os requisitos de software e o cliente, como são tratados os problemas de má especificação de manutenções? (*exemplo, quando o profissional que for tratar a manutenção não entender o que deve ser feito de fato*)

5. É comum que prazos acordados com o cliente não sejam cumpridos? Caso positivo, a que se devem, principalmente, os atrasos?

6. Supondo que já exista um cronograma de manutenções a entregar, como são distribuídas internamente as tarefas? (*ou seja, existe algum critério para determinar qual profissional atenderá qual solicitação?*)

7. Considerando agora que a manutenção do código-fonte já foi efetuada, quais os critérios para testes antes de fornecer a versão modificada ao cliente?

8. Estando a modificação implementada e testada, como ela é fornecida ao cliente? (*por meio de download em site, CD de atualização etc.*)

PARTE 2: Características técnicas da abordagem das manutenções.

9. De maneira simples, como é tratada a documentação das manutenções? (*considere questões como revisão geral da documentação de projeto*).

10. Considerando a grande quantidade de arquivos associados a um projeto de software, como é tratado o controle de versão desses arquivos, quando manutenções simultâneas precisam ser feitas em arquivos comuns? (*alguma ferramenta de controle de versão é usada? É comum terem problemas com a ferramenta?*)

11. De maneira geral, os profissionais empenhados na tarefa de manutenção possuem um conhecimento suficiente da tecnologia utilizada? (*linguagens, banco de dados etc.*) Existe algum programa de reciclagem desses profissionais?

Considerações Finais

12. Caso você seja responsável por decisões na área de manutenção de software, poderia descrever, livremente, quais as principais dificuldades que sente para garantir uma manutenção eficiente e a satisfação do cliente?

A seguir é mostrado o questionário utilizado na segunda parte do estudo de caso, agora focado na obtenção de soluções adotadas pela organização para tratar seus problemas de manutenção de software. Vale lembrar que esse questionário possui poucas questões justamente porque se esperava que os entrevistados espontaneamente citassem as soluções durante a entrevista, as quais seriam registradas pelo entrevistador em material a parte. Nessa segunda parte do estudo de caso não seria possível estipular um conjunto prévio de questões, porque somente a organização seria capaz de listar suas soluções eficientes adotadas.

QUADRO A.2: QUESTIONÁRIO DA SEGUNDA ETAPA: QUESTÕES GERAIS PARA DISCUSSÃO

QUESTIONÁRIO

OBJETIVOS: Identificar soluções eficientes adotadas pela organização para tratar seus problemas de manutenção de software.

1. Os problemas com a atividade de manutenção de software são geralmente muitos. Partindo desse fato, como sua empresa vem tratando a questão: (*assinale apenas uma alternativa*):

☐ Normalmente apenas resolvemos os problemas à medida que surgem.

☐ Buscamos, além de resolver os problemas à medida que surgem, adotar medidas para preveni-los.

2. Alguma ferramenta é utilizada para mapear ou registrar soluções normalmente muito parecidas para problemas que ocorrem com freqüência? (*considere problemas comuns de help-desk e suporte ao cliente*)

3. Do seu ponto de vista, os problemas de manutenção de software vêm diminuindo na organização? (*deixe em branco caso não saiba avaliar*)

4. Descreva, livremente, atitudes gerenciais, hábitos ou outros fatores que do seu ponto de vista já contribuíram para melhorar a abordagem dos problemas de manutenção, citando outras mudanças que acredita poderem auxiliar nesse sentido.

APÊNDICE B

Tabela de Atividades (CBO)

A *Classificação Brasileira de Ocupações* (CBO) corresponde ao documento que reconhece, nomeia e codifica os títulos bem como descreve as características das ocupações do mercado de trabalho brasileiro[1].

No quadro B.1 estão listadas as denominações principais e os respectivos sinônimos das ocupações ligadas à análise de sistemas (o quadro exclui funções de técnicos e auxiliares – normalmente sem exigência de curso superior).

QUADRO B.1: DENOMINAÇÕES PRINCIPAIS E RESPECTIVOS SINÔNIMOS DAS OCUPAÇÕES (CBO)

Denominação principal	Sinônimos
Analista de desenvolvimento de sistemas	*- Analista de comércio eletrônico (e-commerce)* *- Analista de sistemas de informática administrativa* *- Analista de sistemas web (webmaster)* *- Analista de tecnologia de informação* *- Consultor de tecnologia da informação*
Analista de redes e de comunicação de dados	*- Analista de comunicação (tele-processamento)* *- Analista de rede* *- Analista de telecomunicação*
Analista de sistemas de automação	
Analista de suporte computacional	*- Analista de suporte de banco de dados* *- Analista de suporte de sistema* *- Analista de suporte técnico*
Administrador de banco de dados	*- Administrador de banco de dados e de sistemas computacionais*

[1] http://www.mtecbo.gov.br

Administrador de redes	*- Administrador de rede e de sistemas computacionais* *- Administrador de sistema operacional de rede* *- Analista de suporte de rede*
Administrador de sistemas operacionais	*- Administrador de sistemas computacionais* *- Analista de aplicativo básico (software)*
Engenheiro de aplicativos em computação	*- Engenheiro de sistemas computacionais – aplicativos* *- Engenheiro de softwares computacionais*
Engenheiro de equipamentos em computação	*- Engenheiro de hardware computacional* *- Engenheiro de sistemas computacionais – equipamentos*
Engenheiros de sistemas operacionais em computação	*- Engenheiro de software computacional básico* *- Engenheiro de suporte de sistemas operacionais em computação*
Programador de internet	
Programador de sistemas de informação	*- Programador de computador* *- Programador de processamento de dados* *- Programador de sistemas de computador* *- Técnico de aplicação (computação)* *- Técnico em programação de computador*

Para os profissionais enquadrados dentro dessas denominações, a CBO informa que são atribuições gerais típicas deles as mostradas a seguir. Percebe-se que em momento algum existe uma referência específica para manutenção (de software).

ÁREAS	ATIVIDADES					
DESENVOLVER SISTEMAS INFORMATIZADOS	Estudar as regras de negócio inerentes aos objetivos e abrangência de sistema	Dimensionar requisitos e funcionalidade de sistema	Fazer levantamento de dados	Prever taxa de crescimento do sistema	Definir alternativas físicas de implantação	Especificar a arquitetura do sistema
	Escolher ferramentas de desenvolvimento	Modelar dados	Especificar programas	Codificar aplicativos	Montar protótipo do sistema	Testar sistema
	Definir infra-estrutura de hardware, software e rede	Aprovar infra-estrutura de hardware, software e rede	Implantar sistemas			
ADMINISTRAR AMBIENTE INFORMATIZADO	Monitorar performance do sistema	Administrar recursos de rede	Administrar banco de dados	Administrar ambiente operacional	Executar procedimentos para melhoria de performance de sistema	Identificar falhas no sistema
	Corrigir falhas no sistema	Controlar acesso aos dados e recursos	Administrar perfil de acesso às informações	Realizar auditoria de sistema		
PRESTAR SUPORTE TÉCNICO AO CLIENTE	Orientar áreas de apoio	Consultar documentação técnica	Consultar fontes alternativas de informações	Simular problema em ambiente controlado	Acionar suporte de terceiros	Instalar software e hardware
	Configurar software e hardware					
TREINAR CLIENTE	Consultar referências bibliográficas	Preparar conteúdo programático	Preparar material didático	Preparar instrumentos para avaliação de treinamento	Determinar os pré-requisitos do treinando	Determinar recursos áudio-visuais, hardware e software
	Configurar ambiente de treinamento	Validar ambiente de treinamento	Ministrar treinamento			

ELABORAR DOCUMENTAÇÃO PARA AMBIENTE INFORMATIZADO	Descrever processos	Desenhar diagrama de fluxos de informações	Elaborar dicionário de dados	Elaborar manuais do sistema	Elaborar relatórios técnicos	Emitir pareceres técnicos
	Inventariar software e hardware	Documentar estrutura da rede	Documentar níveis de serviços	Documentar capacidade e performance	Documentar soluções disponíveis	Divulgar documentação
	Elaborar propostas técnicas e comerciais	Elaborar estudos de viabilidade técnica e econômica	Elaborar especificação técnica			
ESTABELECER PADRÕES PARA AMBIENTE INFORMATIZADO	Estabelecer padrão de hardware e software	Criar normas de segurança	Definir requisitos técnicos para contratação de produtos e serviços	Padronizar nomenclatura	Instituir padrão de interface com usuário	Divulgar utilização de novos padrões
	Definir metodologias a serem adotadas	Especificar procedimentos para recuperação de ambiente operacional				
COORDENAR PROJETOS EM AMBIENTE INFORMATIZADO	Selecionar equipe de trabalho	Preparar cronograma de atividades e financeiro	Administrar recursos internos e externos	Delegar funções à equipe	Acompanhar execução do projeto	Realizar revisões técnicas
	Avaliar qualidade de produtos gerados	Validar produtos junto a clientes em cada etapa				
OFERECER SOLUÇÕES PARA AMBIENTES INFORMATIZADOS	Propor mudanças de processos e funções	Prestar consultoria técnica	Identificar necessidade do cliente	Avaliar proposta de fornecedores	Negociar alternativas de solução com cliente	Adequar soluções a necessidade do cliente
	Negociar com fornecedor	Demonstrar alternativas de solução	Divulgar solução	Propor adoção de novos métodos e técnicas	Organizar fóruns de discussão	

PESQUISAR TECNOLOGIAS EM INFORMÁTICA	Pesquisar padrões, técnicas e ferramentas disponíveis no mercado	Identificar fornecedores	Solicitar demonstrações de produto	Avaliar novas tecnologias por meio de visitas técnicas	Construir plataforma de testes	Analisar funcionalidade do produto
	Comparar alternativas tecnológicas	Participar de eventos para qualificação profissional				
COMUNICAR-SE	Desenvolver expressão oral	Desenvolver expressão escrita	Assimilar idéias	Desenvolver expressão em forma gráfica	Interpretar gráficos, diagramas e símbolos	Adaptar linguagem
	Trabalhar em equipe	Desenvolver capacidade de negociação				
DEMONSTRAR COMPETÊNCIAS PESSOAIS	Desenvolver raciocínio abstrato	Desenvolver raciocínio numérico	Desenvolver raciocínio lógico	Demonstrar capacidade de síntese	Demonstrar senso analítico	Evidenciar concentração
	Demonstrar flexibilidade	Cultivar criatividade	Demonstrar iniciativa	Inovar	Desenvolver capacidade de memorização	Observar detalhes