Luiz Gustavo Rivelo

Adequação de uma área de Manutenção de Sistemas às Boas Práticas sugeridas pela Engenharia de Software: Estudo de Caso

Dissertação apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT, para obtenção do título de mestre em Engenharia de Computação.

Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. Enrico Giulio Franco Polloni

São Paulo

2006

DEDICATÓRIA

Dedico este trabalho a minha avó, Angelina Fumo Rossetti, pelos seus exemplos e coragem diante das dificuldades e obstáculos da vida.

AGRADECIMENTOS

Agradeço a paciência e compreensão do meu filho por aceitar minhas prolongadas ausências, que pela pouca idade, sempre solicitava atenção enquanto o pai estava ‘trabalhando no computador’.

A revisão final de texto do amigo Marcos Zero Iombriller foi essencial para a conclusão do trabalho: muito obrigado.

Também gostaria de agradecer a constante motivação e ajuda do colega de curso Marcelo Almeida e a perseverança do meu orientador, Enrico Polloni: sem o apoio que recebi dos mesmos seria impossível concluir este trabalho.

RESUMO

Aplicação de melhores práticas sugeridas pelo padrão IEEE-1219 (manutenção de software) e pelo ITIL (InformationTecnology Infrastructure Library) numa companhia da área de alimentação, com apresentação de estudo de caso desta empresa.

Para se enquadrar no mercado internacional, uma grande empresa de alimentos separou suas áreas de projeto das áreas de serviço. Desta forma, todas as atividades de manutenção puderam ser tratadas de forma independente das inovações necessárias ao projeto.

A separação ocorreu em todas as áreas de negócio, incluindo a área de tecnologia de informação. Surgiu então a necessidade de gerenciar a área de manutenção de forma específica, tratando as características específicas que distinguem uma área de produção da área de projetos.

Optou-se pela adaptação das melhores práticas aplicáveis ao caso sugeridas pelo padrão IEEE-1219 e pelo ITIL. O padrão IEEE-1219 foca a estrutura de uma área de manutenção, enquanto da norma ITIL foi explorada a organização da equipe dentro da estrutura montada (papéis e responsabilidades dos times de tecnologia de informação).

Este trabalho apresenta estas melhores práticas, a adaptação efetuada para aplicação prática e os resultados obtidos, assim como uma sugestão de continuidade no processo de pesquisa.

Palavras Chaves: Manutenção de Software, Manutenção de Sistemas de Informação, Boas Práticas de Engenharia de Software, IEEE 1219-1998, ITIL

ABSTRACT

Adapting a System Maintenance Area to suggested Software Engineering Best Practicies: Case-Study

Aplication of standard IEEE-1219 (software maintenance) and ITIL (InformationTecnology Infrastructure Library) best practicies in a food's company, presenting a case-study of this company.

With the objective of adapting to the international market, an important alimentation segment company divided its project and maintenance areas. As a result, all the maintenance activities could be treated in a independent way from the inovations (projects) requested by the business areas.

This separation ocurred in all business areas, including the tecnology and information systems. From this, resulted the needs of managing this new createded maintenance area treating this especific caractheristics that distinguesh a production and project areas.

The option was adapt software engineer best practicies aplyied to the case that was suggested by IEEE and ITIL. The IEEE-1219 pattern focus on the maintenance structure, while the ITIL was exploited in the aspects related to the team organization applyied in the structue choosed (roles and responsibilities of information technology teams).

This work presents this best practicies, the adaptation aplyied and the results, as a sugestion of research continuity.

Key Words: Software Maintenance , Information Systems Maintenance, Software Engineering Best Practicies, IEEE 1219-1998, ITIL

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado
ANS	Acordo de Nível de Serviço (mesmo que SLA)
CM	Configuration Management
CMM	Capability Maturity Model
COBIT	Governance, Control and Audit for Information and Related Technology
CRM	Custom Relationship Management
HD1L	Help Desk First Level
IEEE	Institute of Electrical and Electronic Engineers
IPT	Instituto de Pesquisas Tecnológicas
ITIL	InformationTecnology Infrastructure Library
MP	Maintenance Plan
MR	Modification Request
OLA	Operations Level Agreement
PMBOK	Progect Management Book of Knolegment
PMI	Project Management Institute
SCA	Software Change Authorization
SCR	Software Change Request
SCM	Software Configuration Management
SLA	Service Level Agreement
SLM	Service Level Manegement
SMP	Software Maintenance Plan
SMR	Software Modification Request
SQA	Software Quality Assurance
TRR	Test Readiness Review

SUMÁRIO

Capítulo I

1 APRESENTAÇÃO

1.1 Introdução

O aumento da competitividade das empresas tem levado a critérios mais rigorosos para a seleção de investimentos. Orçamentos menores tem restringido gastos, levando a questionamentos mais severos sobre como mensurar os custos e benefícios dos investimentos feitos em projetos de qualquer porte. Esta tendência também afeta a área de Informática.

Em alguns casos, o desenvolvimento de novas tecnologias e produtos de software pode não ser mais o foco para diferenciação entre as empresas dentro deste contexo de controle orçamentário mais rígido. Estes desenvolvimentos ficam restritos à projetos com retorno garantido e quantificável ou quando não há absolutamente outra opção. De acordo com Info Corporate (2005) um estudo feito pela consultoria Frost & Sullivan indica que 71% das corporações vêem o foco no negócio como a principal vantagem competitiva.

Como citado em Simcsik (2006) é necessário reavaliar os custos dos investimentos de TI nas organizações: os modismos criaram a obrigação de novas tecnologias serem adquiridas, o que ocorria sem a preocupação de justificativa destes investimentos. Quando estes recursos tornaram-se escassos surgiu a necessidade de novas alternativas, sempre levando em conta a necessidade de justificar os

investimentos realizados. E a competitividade tem compelido pela busca de custos menores com aplicativos e soluções mais simples e baratas.

Fruto desta tendência, algumas empresas tem se interessado por padrões e melhores práticas vivenciadas por outras empresas, ou pelo mundo acadêmico, como alternativa a estas restrições de investimento, uma vez que montar soluções customizadas sem apoio de nenhuma base ou alicerce prévio acaba sendo muito mais dispendioso.

Adaptar um padrão já existente ou evitar os erros seguindo as melhores práticas estipuladas por órgãos ou institutos de padronização acaba sendo uma opção atraente, ficando a maior parte do investimento na customização e adequação às peculiaridades de cada negócio, respeitando as particulariedades e especificidades de cada ramo de trabalho. Esta é a contribuição pretendida com este trabalho, com a aplicação do modelo proposto a um caso específico.

Seguindo nesta linha, difundem-se conhecimentos antes restritos aos que conheciam a Engenharia de Software na sua forma acadêmica na busca de soluções fundamentadas para problemas encontrados no mercado de trabalho para o ambiente corporativo. A velocidade com que a Engenharia de Software tem amadurecido assim como a consolidação de entidades de pesquisa ou padronização junto ao ambiente corporativo também contribui para este processo, fortalecendo os centros de pesquisa e as empresas.

Este trabalho trata de dois exemplos de melhores práticas propostas, uma pelo padrão IEEE-1219 e outra pelo ITIL adaptados a uma empresa que atua no segmento de varejo de alimentos e não tem como foco de diferenciação dos negócios os investimentos na área de tecnologia de informação.

A falta de soluções prontas (ditas de mercado ou de prateleira) de baixo custo para o gerenciamento de ambientes de manutenção de sistemas, com alta frequência de alterações, levou à adaptação de melhores práticas disponíveis e testadas à realidade vivida por esta empresa quando da separação das suas áreas de projeto e manutenção.

A clara divisão das áreas de projeto e manutenção são essenciais à aplicação dos modelos propostos. Parte dos resultados obtidos, assim como o modelo proposto a esta realidade, são tratados neste trabalho.

1.2 Objetivo

Este trabalho tem como objetivo mostrar a viabilidade da aplicação de modelos consagrados propostos por entidades de padronização, com as devidas adaptações, em uma empresa que não seja da área de tecnologia de informação e que não invista diretamente em pesquisa de novos recursos de sistemas de informação.

1.3 Justificativa

As atuações na manutenção de software concentram a maior parte dos recursos consumidos por um sistema de informação[1].

A medida que a manutenção de sistemas de software torna-se mais difícil de ser realizada quanto maior for o número de pessoas que estiverem usando uma determinada aplicação e quanto maior a complexidade associada a esta manutenção, escolheu-se analisar neste trabalho a aplicação de duas melhores práticas propostas pela Engenharia de Software na área de manutenção de sistemas de uma empresa.

Segundo [SIMCSIK, pág. 117] a Engenharia Software é

> "a arte e a ciência de estruturar e acompanhar um Ciclo de Vida de um Sistema qualquer, no tempo, na forma, no custo e na matéria mais otimizada possível, para que as respostas desses programas/softwares sejam explicitamente aceitos pelo cliente/usuário."

Ou seja, além do fator custo, a Engenharia de Software pode ajudar a evitar que erros repetidos nas etapas de manutenção afetem a credibilidade de toda a área de tecnologia de uma empresa, o que pode colocar em risco a operação (prejuízos financeiros e custos operacionais, por exemplo).

A dificuldade em separar as áreas de projeto e produção de sistemas numa empresa onde o foco não é o desenvolvimento/gerenciamento de sistemas levou à busca pela racionalização destes processos, gerando um modelo de prestação de serviços centralizados na empresa estudada. O processo de separação não é o tema central deste trabalho, mas influenciou sobremaneira na escolha dos padrões adotados.

A área de tecnologia, dentro deste modelo, precisava atuar de forma estruturada e devido a necessidade de alocação e contenção de custos, separar tudo que estivesse associado a novos projetos das necessidades da produção, ou seja, separar as inovações das atividades da administração cotidiana do sistema, mencionados neste trabalho como serviços.

A escolha da empresa utilizada no estudo de caso justifica-se pela característica dos serviços de manutenção possuirem gerenciamento tão complexo quanto o de uma empresa da área de sistemas e pela possibilidade de acompanhamento do processo (escolha dos padrões e implementação dos mesmos) desde o início.

---

[1] É necessário ressaltar que a duração da etapa de projeto em comparação a operação do sistema tende a ser menor. O percentual será apresentado e justificado ao longo do trabalho.

1.4 Organização do Trabalho

Este trabalho apresenta-se organizado, com adaptações, conforme Vieira (2002) na estrutura apresentada a seguir, dividido basicamente nos tópicos apresentação, desenvolvimento, modelagem, estudo de caso, conclusões e referências bibliográficas:

Apresentação - no Capítulo 1 é feita a introdução ao problema a ser estudado: são apresentados os objetivos e a justificativa do trabalho, assim como a proposição de utilizar algumas das melhores práticas apresentadas no padrão IEEE-1219 e no Capítulo dedicado ao ITIL que trata da estruturação dos recursos da área de tecnologia de informação, de forma adaptada, na busca da viabilidade da aplicação de algumas das boas práticas da Engenharia de Software num ambiente corporativo.

Desenvolvimento – os materiais e métodos necessários ao trabalho são apresentados, assim como a forma como foi elaborado o trabalho. Conceitos e referências sobre manutenção de software são apresentados com base nas melhores práticas sugeridas pelo PMI (Project Management Institute), normas e melhores práticas dos institutos e órgãos de padronização. Especificamente, no Capítulo 2 são apresentados os conceitos sobre manutenção de software. No Capítulo 3 são colocados os principais pontos do padrão IEEE-1219 sobre melhores práticas sobre manutenção de software. O Capítulo 4 traz as melhores Práticas sugeridas pelo ITIL na estruturação de equipes de sistemas.

Modelagem - Baseado nos fundamentos anteriores e na experiência prática vivida numa corporação que optou por adotar essa linha de atuação, um modelo de prestação de serviços na área de TI é proposto no Capítulo 5, baseado nas melhores práticas apresentadas anteriormente.

Estudo de Caso – No Capítulo 6 é apresentado um estudo de caso, aplicando o modelo proposto e discutindo os pontos deste modelo.

Conclusões – No Capítulo 7 é apresentada a avaliação do resultado do trabalho e uma conclusão, assim como proposições de continuidade de estudos nesta área.

Referências – Finalizando, as referências bibliográficas e a bibliografia de apoio são apresentadas, assim como um apêndice com os principais acrônimos usados.

Capítulo II

2 CARACTERIZAÇÃO DA MANUTENÇÃO SOFTWARE

2.1 Introdução

As atividades associadas à manutenção de software guardam semelhanças às atividades realizadas nos projetos de software.

Neste trabalho foram usados conceitos que serão desenvolvidos seguindo a ordem de apresentação e o método utilizado, que será descrito neste capítulo.

Serão apresentadas as definições de Manutenção de Software assim como uma classificação para os tipos de manutenção de software.

Durante o processo de execução das manutenções de software, são encontradas dificuldades e consequências, aqui indicadas juntamente com os possíveis efeitos colaterais de manutenções de software executadas sem seguirem as boas práticas indicadas nos capítulos seguintes do trabalho.

Estas definições e conceituações são essenciais para caracterizar a contribuição pretendida neste trabalho, que é o de adaptar um modelo de manutenção de software sugerido pela engenharia de software a uma empresa que não é do ramo de tecnologia de informação.

2.2 Método utilizado na elaboração da Dissertação

Alguns pontos são discutidos a seguir, com o propósito de apresentar como o trabalho desta dissertação foi concebido.

A escolha do tema foi motivada basicamente pela experiência prática do autor na área de informática de uma corporação do ramo alimentício que ocorreu durante os anos que cursava o Mestrado Profissional em Engenharia de Software, o qual exige esta dissertação. Esta foi a oportunidade de aplicar conhecimentos acadêmicos num ambiente corporativo, justificando o propósito deste curso para o autor.

O material básico usado foram boas práticas consagradas da Engenharia de Software, publicadas por institutos e órgãos de padronização. Optou-se pelas melhores práticas sugeridas pelo IEEE e pelo ITIL na área de manutenção de software.

Enquanto o padrão IEEE-1219 trata de todas as etapas do processo, as sugestões do ITIL analisadas auxiliam na adequação das equipes de trabalho ao processo apresentado. A adaptação da solução indicada pelo ITIL ao modelo proposto, viabilizou a implantação na empresa analisada de uma estrutura hierárquica segregada em projetos e manutenção, como pode ser visto no estudo de caso.

Este trabalho discorre sobre um estudo de caso, onde a aplicação das melhores práticas mencionadas acima são aplicadas numa empresa que optou por separar a área de manutenção da área de projetos, levando a total separação do conceito de prestação de serviços da área de tecnologia de informação das novas necessidades, representadas por projetos.

No estudo de caso, foram coletadas informações pontuais antes e depois da aplicação do processo.

2.3 Definição de Manutenção de Software

A manutenção de software é bem mais do que consertar erros, o que justifica esforços da ordem de 60% a 70% das equipes de sistemas. A definição exata do que seja a manutenção de software e as atividades que são compreendidas como manutenção de software ajudam no entendimento do porquê de tanto esforço nesta etapa do ciclo de vida do sistema de software.

Segundo Christensen (2001), manutenção de software é “a atividade de fazer mudanças ao produto de software e à documentação associada”.

O padrão IEEE-1219 define a manutenção de software como a modificação de um produto de software depois da entrega para corrigir falhas, melhorar a performance, melhorar atributos e funcionalidades do produto ou ainda adaptá-lo ao ambiente. O padrão IEEE-1219 também considera como manutenção atividades de suporte a configuração de dados, como por exemplo, o ajuste de parâmetros do sistema, mesmo que não haja alteração no código fonte. As alterações na documentação (manuais, por exemplo) também são consideradas como manutenções pelo padrão IEEE-1219. Ou seja, uma manutenção pode ser demandada pelo usuário-cliente ou pela equipe de desenvolvimento.

Há algumas referências carregadas com um certo preconceito associado às atividades de manutenção. Tais afirmações nem sempre são explícitas, uma vez que é necessário captar e manter bons recursos nas atividades de manutenção e o desmerecimento explícito da mesma pode tornar esta tarefa mais difícil. No entanto pode-se encontrar contra-exemplos, como o citado em (Torvalds 2001, p.194) colocando a manutenção como atividade de menor interesse:

> "Que modo mais realista existe para o código-fonte aberto se firmar do que por meio do patrocínio de empresas ? E existe um modo melhor de conseguir que parte do trabalho menos interessante seja completado – as coisas chatas como manutenção e suporte – do que fazê-lo dentro das empresas ?"

No entanto, o mesmo autor reconhece a importância financeira associada as atividades de manutenção, como pode ser visto em (Torvalds 2001, p.190):

> "Perguntavam o motivo de tanto alvoroço. (...) Raramente a decisão era baseada no não-custo[1] do sistema, porque na verdade o software em si representa uma parte pequena do investimento. O serviço e o suporte custam muito mais caro".

A clara definição e classificação dos tipos de manutenção são essenciais para o gerenciamento de uma área responsável pela manutenção de sistemas. Quando uma alteração corretiva emergencial tem que ser implantada, será necessário coordenar toda a documentação necessária, providenciar uma "janela de implantação", comunicar os usuários de enventual indisponibilidade (mesmo que temporária), etc. E tudo isto impactando as demais áreas de desenvolvimento que passam a trabalhar sob um produto que já foi alterado, sendo necessário realizar uma análise de impacto desta alteração.

[1] O termo 'não-custo' é referenciado por Torvalds (2001) no contexto de software livre

2.4 Tipos de Manutenção de Software

Para melhor categorizar as manutenções de software, Christensen (2001) em seu estudo sobre o padrão IEEE-1219 dividiu as manutenções em três categorias, como mostrado na Figura 1 abaixo:

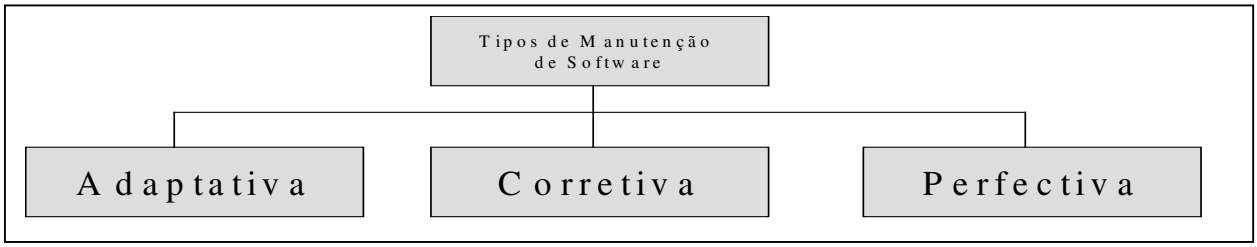

Figura 1: Tipos de Manutenção de Software segundo padrão IEEE-1219

- **Manutenção Adaptativa**

Ocorre devido a rápida mudança que é encontrada em cada aspecto da computação. Segundo Pressman (1996) novas gerações de hardware são anunciadas num ciclo de 24 meses e novos sistemas operacionais e novos lançamentos de versões antigas aparecem regularmente; novos periféricos surgem ou são modificados em prazos menores do que a vida útil média de um sistema que é de cerca de 10 anos.

Desta forma, se a aplicação de software dura mais que o ambiente onde está inserida, é normal que surjam manutenções para acompanhar estas mudanças de ambiente. Uma manutenção é classificada como adaptativa se ela permite que o sistema e o ambiente continuem operando enquanto é realizada.

São exemplos de manutenções adaptativas a mudança de sistema operacional, mudança de hardware, ferramentas de software e alterações formais de dados. As manutenções adaptativas respondem historicamente por cerca de 20% das manutenções.

- **Manutenção Corretiva**

Normalmente de natureza reativa, as manutenções corretivas prestam-se para a correção de falhas e problemas. Entende-se que a fase de testes possa identificar grande parte dos problemas que possam ocorrer quando o sistema for colocado a disposição dos usuários. Mas um sistema de software de grande porte pode conter erros de diversas naturezas tem que ser previstos mesmo após a entrada em produção. Quando um usuário encontra um defeito latente no sistema e comunica o desenvolvedor (responsável pela correção) temos um exemplo de manutenção corretiva.

A criticidade das manutenções corretivas indicam uma subclassificação das manutenções corretivas. Uma manutenção pode ser corretiva emergencial ou corretiva agendada, dependendo da urgência com que a manutenção deva ser

implementada. Historicamente também responde (assim com o a manutenção adaptativa) por cerca de 20% das manutenções de software.

- **Manutenção Perfectiva**

Segundo Pressman (1996) são as manutenções que ocorrem quando um sistema de software foi bem sucedido e recomendações de novas capacidades, de modificações em funções existentes e de ampliações gerais são recebidas dos usuários. São manutenções feitas para aperfeiçoar a performance ou atender novas requisições dos usuários. Respondem por cerca de 60% das manutenções.

Os termos manutenção adaptativa, manutenção corretiva e manutenção perfectiva foram cunhados por Swanson (1976). No entanto uma quarta classificação de manutenção é apresentada por Pressman (1996): manutenção preventiva.

- **Manutenção Preventiva**

Ocorre quando o software é modificado para melhorar a confiabilidade ou a manutenibilidade futura, ou para oferecer uma base melhor para futuras aplicações. É o tipo de manutenção caracterizada pelas técnicas de engenharia reversa[1] e reengenharia[2].

O termo manutenção preventiva é comumente usado na manutenção de hardware e outros sistemas físicos, mas com o avanço da engenharia de software cada vez mais a aplicação aos softwares se torna necessária.

Uma organização de software deve selecionar os programas que provavelmente sofrerão mudanças num futuro próximo e prepará-los para tais alterações.

A manutenção preventiva de um sistema de software cujo programa corresponda a um emaranhado de linhas de código não estruturados e sem módulos ou componentes e também sem documentação devem ser abordados, segundo Pressman (1996) a partir de uma das opções abaixo:

> "1 – Lutar de modificação em modificação, 'combatendo' o projeto implícito e o código-fonte para implementar as mudanças necessárias.
>
> 2 – Entender o funcionamento interior mais amplo do programa num esforço para fazer modificações mais efetivas.
>
> 3 – Redesenhar, recodificar e testar aquelas partes do software que requerem modificação, aplicando a abordagem de engenharia de software a todos os segmentos revisados.

---

[1] Segundo Pressman (1996) A engenharia reversa é um processo de recuperação de projeto, onde um programa é analisado no esforço de conseguir uma representação do programa num nível maior do que o código fonte.

[2] Segundo Pressman (1996) a reengenharia, também chamada renovação ou recuperação, recupera informações do projeto do software e usa essas informações para alterar ou reconstituir o sistema existente num esforço para melhorar sua qualidade global.

> 4 – Redesenhar, recodificar e testar completamente o programa usando ferramentas CASE (ferramentas de engenharia reversa e reengenharia) para ajudar a entender o projeto atual."

Não é escopo deste trabalho detalhar os critérios que orientariam na escolha das opções apresentadas; deve ser levado em conta critérios econômicos e o momento em que a manutenção será efetuada. No entanto a primeira opção, a menos desejável do ponto de vista técnico, pode ser a melhor numa determinada conjuntura.

2.5 Características e Dificuldades encontradas na Manutenção de Software

Em (Christensen, 2001 apud Boehm, 1976, p.503-511) onde a manutenção de software é indicada como o maior elemento simples e também como uma das mais difíceis etapas no ciclo de vida do software. Embora não seja devidamente reconhecida, o que ocorre comumente por três razões.

A primeira é que o processo de manutenção normalmente não está associado a novas oportunidades de negócio e não atrai a atenção da alta gerência. A segunda é a visão de que o negócio está perdendo dinheiro em atividades que deveriam ter sido previstas ou evitadas e de que as novas versões são uma forma de exploração. E a terceira é a de que muitos engenheiros de sistemas preferem partir do zero a ter que suportar sistemas com possíveis deficiências de documentação ou desenvolvidos em plataformas tecnologicamente defasadas.

Além disso, a manutenção de software é mais difícil de lidar tecnicamente e gerencialmente. Uma alteração não pode inserir novos problemas e tem que respeitar as restrições já existentes, mas que nem sempre são claras no início da manutenção. Desta forma, as opções de análise e projeto ficam mais restritas. Para evitar que novos problemas sejam inseridos, pode ser necessário conhecer o sistema como um todo. E em todos os casos, os testes são mais complexos, pois tem que validar, além das alterações, a continuidade da estabilidade do sistema.

Outro ponto a ser considerado é quando manutenções corretivas, adaptativas e de melhoramentos são feitas simultaneamente. A coordenação e controle do projeto, implementação e testes tornam-se mais complexas do que se tivessem sendo executadas isoladamente.

Segundo Pressman (1996) a manutenção de software é a fase mais negligenciada do processo de engenharia de software, com literatura muito inferior se comparada às etapas de projeto (incluindo desenvolvimento). Poucas abordagens e métodos técnicos, comparativamente a projetos de software, foram propostos. A manutenção de software deve lidar com a organização (ou a falta dela) na coordenação das atividades (que são características desta etapa), custos específicos e os problemas que podem ocorrem especificamente quando da manutenção de um sistema de software.

2.6 Efeitos colaterais da Manutenção de Software

A complexidade do processo de manutenção de software ocorre tanto no aspecto gerencial, dada a dificuldade de coordenar atividades críticas em curto espaço de tempo, quanto no aspecto técnico. Toda vez que uma mudança é introduzida num procedimento lógico complexo o potencial de erros cresce. Por melhores que sejam os testes e atualizações de documentação, surgirão efeitos colaterais (indesejados...) associados às manutenções.

Segundo Pressman (1996) existem três categorias de efeitos colaterais:

- **Efeitos colaterais na codificação**

A comunicação com a máquina é feita a partir de uma linguagem de programação; erros de sintaxe são identificados por estas linguagens, mas sintaxes válidas podem conter erros de semântica (lógica) que podem ser de difícil detecção.

- **Efeitos colaterais nos dados**

Ocorrem como resultado de modificações feitas na estrutura de informações do software (em elementos individuais de uma estrutura de dados ou na própria estrutura).

- **Efeitos colaterais na documentação**

Ocorrem quando as alterações no código fonte não são refletidas na documentação do projeto ou nos manuais (técnicos e de usuários).

2.7 Projeto de Softwares e Manutenção de Software

Uma categorização das atividades desenvolvidas na área de engenharia de software pode ser feita de forma a dividir as atividades associadas à projeto e as associadas à manutenção.

Segundo (Heldman 2003,p. 3), o PMBOK (2005)[1] traz a definição de um projeto como sendo caracterizado por:

> "a) Ser temporário: por mais longo que seja um projeto, ele tem início e fim bem definidos.
>
> b) Gerar produtos, serviços ou resultados únicos: gera algo que já não tenha sido feito anteriormente.
>
> c) Ser elaborado de forma progressiva: seguindo passos de forma incremental."

[1] PMBOK – Progect Management Book of Knolegment – referência para boas práticas no gerenciamento de projetos.

Ou seja, um projeto é um empreendimento temporário que gera produtos/serviços únicos. Uma vez terminado, a área de projetos "entrega" o produto/serviço para ser gerido por uma equipe de manutenção, que será encarregada de realizar as manutenções e atualizações na documentação.

Ao contrário do que ocorre durante um projeto, uma manutenção pode afetar um grande número de usuários e eventuais falhas que ocorram durante uma manutenção geram consequências que impactam diretamente o cotidiano dos usuários destes produtos.

Muitas vezes, pequenos projetos acabam sendo executados como manutenções, o que agrava o risco de enventuais falhas/problemas com a alteração.[1]

Desta forma, a distinção entre projeto e manutenção é crítica para a aplicabilidade do modelo sugerido no padrão IEEE-1219.

Segundo Christensen (2001) mais de 60% dos gastos com um produto de software[2] ocorrem na fase de manutenção. Como os projetos geralmente tem ciclo de vida menores do que as manutenções, existe maior facilidade em obtenção de recursos a serem alocados em projetos, dada a visibilidade e maior rapidez na obtenção de resultados tangíveis (retorno "rápido"). No entanto, uma organização precisa tratar a manutenção de forma estruturada sob o risco de diminuir a vida útil do produto de software com manutenções ineficientes e pequenos projetos "emergenciais" para correções pontuais e não coordenadas.

Nesta linha, a aplicação do padrão IEEE-1219 surgiu como uma opção estruturada para servir como base para o modelo adotado no estudo de caso.

2.8 Gerenciamento de Projetos e Gerenciamento de Produtos

Uma vez definido o que é um projeto e a sua distinção das atividades de manutenção, pode-se definir os conceitos de gerenciamento de projeto e o gerenciamento de produto.

Os projetos podem ser gerenciados usando um grupo comum de processos de gerenciamento de projetos. Este grupo de processos utilizado para o gerenciamento de projetos pode ser aplicado a outros tipos de projetos, inclusive de áreas distintas.

---

[1] Se, por exemplo, um relatório funcionou corretamente ao longo de onze meses, mas na transição de um ano para outro um totalizador não está correto, trata-se de uma manutenção. Eventualmente esta manutenção pode envolver aspectos externos ao sistema ou de maior complexidade. Neste caso, esta alteração precisa ser tratada como um projeto. No exemplo do relatório, o ano em questão poderia ser 1999 e o problema ser uma consequência do 'bug' do milênio.
Por outro lado, atividades que são tecnicamente classificados como simples pelos técnicos/programadores devem ser consideradas como pequenos projetos. Um exemplo seria a disponibilização de um determinado relatório, hipoteticamente já utilizado num módulo de um sistema, em outro módulo. Apesar da simplicidade técnica de replicação de um relatório em outro módulo, outros aspectos precisam ser considerados, tais como segurança (permissionamento) ou performance: o número de usuários pode aumentar ou horário de utilização pode passar a coincidir com alguma outra rotina crítica.

[2] O Custo de Manutenção será discutido a parte, em tópico específico, onde também serão apresentados dados indicados por Pressman (1996)

Todos os projetos precisam ser definidos e planejados e todos os projetos devem gerenciar o escopo, a comunicação, os riscos, a qualidade e outros quesitos que caracterizam a boa prática do gerenciamento de projetos.

O resultado gerado pelos projetos é um produto, ou seja, o gerenciamento de projetos pode ser pensado como um processo e este processo ser utilizado para entregar os produtos.

Este produto criado a partir de um projeto pode ter sido desenvolvido internamente ou comprado de um fornecedor. No caso de ser comprado de um terceiro, o projeto foi desenvolvido por uma entidade e será utilizado por outra. Nos dois casos (projetos internos ou externos), é necessário que se tenha a gerência deste produto, assim como ocorreu a gerência do projeto.

Segundo Tenstep (2005) o gerenciamento de produtos é uma abordagem para coordenar centralmente as atividades que dão apoio ao longo prazo, os aperfeiçoamentos e as novas otimizações de um produto. O profissional que executa estas responsabilidades é chamado de Gerente de Produtos.

O processo de gerenciamento de produtos pode começar durante o projeto que criou o produto. No caso da compra, o gerenciamento do produto começa no processo de avaliação e seleção do produto.

Enquanto o papel de um gerente de projeto é planejar e gerenciar um projeto o gerente de produto é focado no apoio em longo prazo do produto dentro da organização.

Capítulo III

3 PADRÃO IEEE-1219 1998 PARA MANUTENÇÃO DE SOFTWARE

3.1 Introdução

Neste capítulo é apresentada no padrão IEEE-1219, que trata da manutenção de software. Esta norma é fundamental no modelo utilizado neste trabalho e além das definições são apresentados tópicos sobre a manutenção que abrange o gerenciamento, planejamento, implementação e controle.

As fases da manutenção implementadas pelo padrão IEEE-1219 são detalhadas, assim como a comparação com o processo de software da ISO/IEC 14764.

3.2 Definição do padrão IEEE-1219

A adoção do padrão IEEE-1219 foi peça chave na transição de uma estrutura integrada de sistemas de informação para uma estrutura com áreas distintas de projeto e manutenção na empresa analisada no estudo de caso. Os pontos relevantes do padrão IEEE-1219 apresentados no estudo de caso são apresentados neste capítulo.

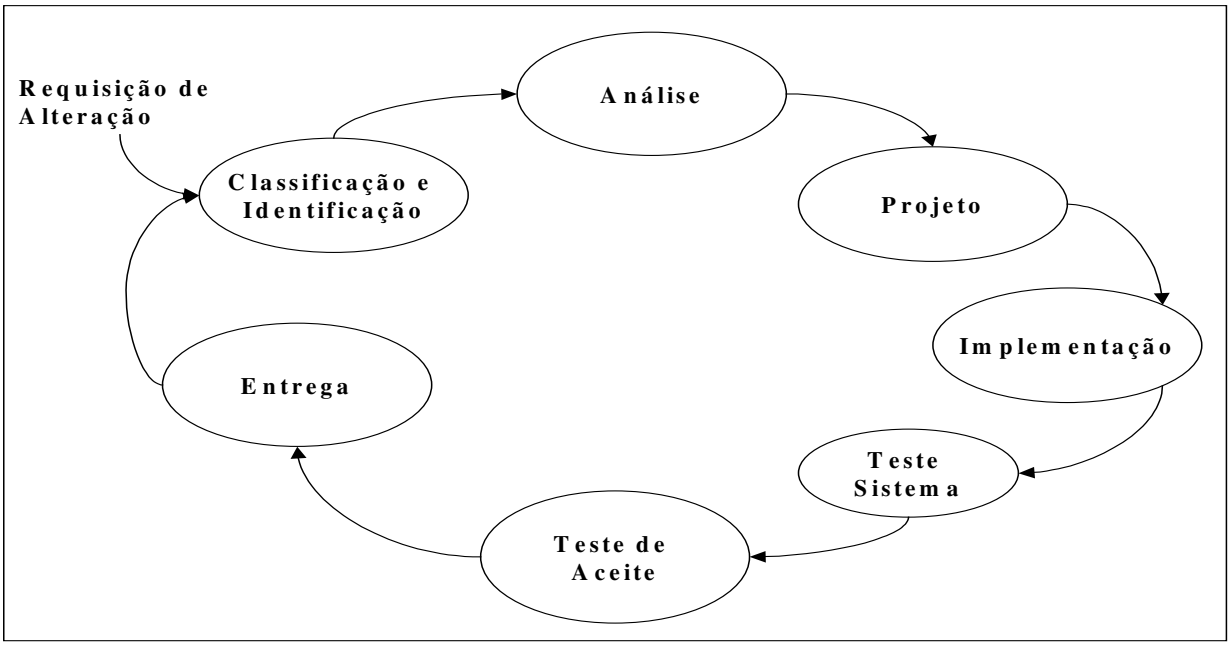

Figura 2: Atividades do Processo de Manutenção de Software - adaptado de SWEBOK (2004)

Segundo Christensen (2001), o padrão IEEE-1219 (IEEE Standard for Software Maintenance, 1219-1998) provê guias para gerenciamento e execução de atividades de manutenção de software. Este padrão pode ser aplicado a qualquer produto de software e não é restrito a tamanho, criticidade ou complexidade da aplicação. O padrão IEEE-1219 não prevê a utilização de nenhum modelo particular de desenvolvimento, como cascata ou espiral.

Ainda segundo Christensen (2001) o padrão IEEE-1219, apresenta cinco etapas, sendo o foco do estudo de caso na classificação por fases da manutenção de software:

a) Gerenciamento da manutenção de software

b) Planejamento da manutenção de software

c) Implementação da manutenção de software

d) Controle da manutenção de software

e) As sete fases da manutenção de Software

3.3 Gerenciamento da manutenção de software segundo padrão IEEE-1229

O gerenciamento da manutenção de software consiste nas atividades necessárias para guiar a implementação de mudanças do início até que a versão final seja entregue (implantada). Estas atividades não diferem em essência das atividades de gerenciamento de projeto, no entanto o gerenciamento de manutenção de software deveria usar mais intensamente as funções de **planejamento** e **controle** do que num projeto de software. A complexidade ocorre devido a menor maneabilidade dos prazos: como múltiplas atividades ocorrem de forma concorrente, o planejamento se torna mais complexo.

Uma vez iniciada a manutenção, o controle dos esforços sofrem um incremento similar de complexidade, com equipes trabalhando em bases distintas com focos diferentes assim como diferentes orçamentos. O ideal seria manter a administração destes custos de forma independente, mas certas circunstâncias requerem sinergia das equipes o que pode dificultar esta separação.

Mesmo quando dentro da manutenção ocorre um novo desenvolvimento, este deve ser tratado como uma das etapas do processo de gerenciamento da manutenção de software.

3.4 Planejamento da manutenção de software segundo padrão IEEE-1229

O planejamento da manutenção de software está subdividido em cinco atividades, descritas a seguir.

- **Determinação dos esforços**

Para a determinação dos esforços, o primeiro passo é a determinação das atividades de manutenção previstas num futuro imediato. As ações derivadas destas atividades devem ser identificadas, documentadas e categorizadas permitindo ao gerente de projeto de engenharia de software dimensionar com clareza os recursos e processos necessários para a manutenção.

Este primeiro conjunto de mudanças deve ser submetido aos impactos que podem trazer para a organização, suporte aos clientes/usuários e sobre o time de desenvolvimento. Estas mudanças devem ser avaliadas e classificadas (desejáveis, necessárias) e este ciclo pode se repetir algumas vezes. Uma vez aceita, a análise deve ser registrada num 'Software de Gerenciamento de Configuração' - SCM (*Software Configuration Management*) pois a complexidade destas alterações, novas requisições e dificuldade de atualização de documentação requerem um sistema de software que auxilie neste processo, sem o qual seria muito difícil ou até mesmo impossível executar os controles da manutenção.

- **Determinação dos processos necessários**

Uma vez determinado o que deve ser feito, deve-se focar em quem irá fazê-lo. O projeto deve ter um processo de manutenção de software, mesmo que não seja reconhecido como tal. Em projetos maiores onde são introduzidas alterações durante estágios posteriores (após alguns testes, por exemplo) ou onde há a

integração de módulos distintos de software ocorrem atividades de manutenção de software.

Estas atividades deveriam, idealmente estar descritas no plano do processo modelo do ciclo de vida do sistema. Caso este documento, ou similar, não exista deve-se recorrer à entrevistas com os empregados. Outra fonte de informação seria o arquiteto de software, quando este fizer parte da equipe. Um problema comum com os processos “herdados” é a falta de rastreabilidade dos requisitos de sistema.

Problemas encontrados nos processos existentes devem ser analisados de forma pragmática e verificados quanto a sua viabilidade econômica.

- **Quantificação dos esforços**

Neste ponto pode-se estimar o orçamento e o cronograma de forma quantitativa. Os custos, pessoal, e cronogramas necessários para cada possível alteração identificada anteriormente devem ser avaliadas usando o processo de manutenção obtido previamente.

As alterações devem ser classificadas de acordo com o seu risco, dificuldade, criticidade de necessidade , necessidades especiais de recursos e a área de impacto do sistema. Esta classificação talvez tenha que ser repetida diversas vezes por causa de novas idéias que possam surgir e especial atenção deve ser dedicada à alterações onde a solução necessite de conhecimento especializado e recursos de retaguarda, pois estes podem ser difíceis de agendar.

- **Antecipação das necessidades**

Para planejar esforços de manutenção futura é necessário antecipar o volume e tipo de manutenções desejáveis no futuro. O orçamento, cronograma e recursos necessários destas alterações devem ser baseadas em valores médios, corrigidos adequadamente (inflação).

- **Desenvolvimento do plano**

O padrão IEEE-1219 provê um guia para o 'Plano de Manutenção de Software' - *SMP (Software Maintenance Plan*), cujo formato geral segue o apresentado abaixo, subdividido por seções:

Seção 1: Introdução

Seção 2: Referências

Seção 3: Definições

Seção 4: Visão geral da manutenção do software

Seção 5: Ferramentas, Técnicas e métodos

Seção 6: Necessidades de reporte da manutenção de software

Seção 7: Necessidades de controle da manutenção de software

Seção 8: Necessidades de documentação da manutenção de software

Não faz parte do escopo deste trabalho analisar o plano de manutenção, mesmo no estudo de caso. No entanto, qualquer iniciativa prática no sentido de seguir as melhores práticas deve considerar a etapa de planejamento.

3.5 Implementação da manutenção de software segundo padrão IEEE-1229

Os fatores que devem ser considerados quando da implementação da manutenção de software são:

a) Prioridade das alterações;

b) O custo e restrições de cronograma da alteração;

c) Restrições de pessoal para a alteração, especialmente se algum recurso com habilidades especiais for necessário;

d) As facilidades físicas necessárias para implementação dos testes associados à manutenção;

e) As áreas do produto as quais as alterações serão aplicadas;

f) As restrições dos testes de regressão necessárias à manutenção;

g) Qualquer documentação ou equipamento de suporte; e

h) Restrições especiais de versão.

Segundo Christensen (2001) estes fatores devem ser avaliados isoladamente e em conjunto, o que fará surgir grupos de alterações que poderão ser executadas com ganho de sinergia (pacotes de manutenção). Uma vez identificados estes pacotes de manutenção, estes devem ser formalmente aplicados a um cronograma.

3.6 Controle da manutenção de software segundo padrão IEEE-1229

O controle da manutenção de software é visto segundo Christensen (2001) em quatro níveis:

a) Garantia de Qualidade do software - *SQA (Software Quality Assurance*)

b) Configuração de software

c) Métricas

d) Documentação

Estes níveis são detalhados a seguir, segundo os papéis que são desempenhados, ou seja, sob o ponto de vista do papel do controle de qualidade, gerenciamento de configuração e assim por diante.

3.6.1 Controle da qualidade do software na manutenção de sistemas de software

A função do controle da qualidade do software é garantir que o trabalho está sendo feito de acordo com o publicado no 'Plano de Manutenção de Software' - SM (*Software Maintenance Plan*) e que este trabalho está sendo executado usando os métodos, processos e procedimentos prescritos. Se a atividade de manutenção está sendo executada sob contrato, as funções de garantia de qualidade de software também podem ser usadas para garantir a qualidade prevista no contrato.

3.6.2 Gerenciamento de configuração de software na manutenção de sistemas de software

A 'Gerência de Configuração de Software' - *SCM (Software Configuration Management*) é a disciplina mais importante usada no controle das atividades de manutenção de software.

De início, uma disciplina confiável de controle de alterações de sistema é necessária para que a manutenção de software possa ser identificada, aprovada, planejada, implementada, testada e versionada. Durante o processo de execução das alterações, uma ferramenta de gerenciamento de mudança confiável, flexível é necessária para que as implementações ocorram de maneira ordenada, pois vários esforços de alterações podem estar ocorrendo simultaneamente e agindo sobre os mesmos itens de software.

Segundo Christensen (2001) um sistema de gerenciamento de configuração é a única maneira de garantir a qualidade e objetivamente determinar que os itens alterados foram os itens testados e aprovados em determinada versão.

3.6.3 Uso de métricas na manutenção de software

Para comparar linhas de código desenvolvido existem parâmetros definidos e utilizados mundialmente. É o caso, por exemplo, da utilização da análise por pontos de função. Segundo [TIBOR, pág. 78]:

> "métricas por Pontos de Função são usadas por um grande número de companhias em uma base mundial, classificando sistemas com precisão segundo o tamanho, em um esforço para atender a demanda do cliente na hora certa, com o serviço ou produto correto e dentro do orçamento (isto é Eficácia na prontidão). "

No entanto, não existe uma metodologia consagrada dentro da Metrologia que possa ser usada para comparar se o processo de desenvolvimento de projetos e manutenção de software deva ser executado por uma mesma equipe ou por equipes distintas. O que pode ser feito é aplicar métricas padrões para comparações.

As métricas aplicadas aos esforços de manutenção de software tornam o processo de gerenciamento das manutenções de software aceitável e objetivo. Existem dois tipos de métricas aplicadas: gerenciais e técnicas.

As métricas gerenciais são usualmente gráficos representando duas linhas, uma indicando a performance esperada e outra indicando a performance atual. São exemplos deste tipo de métricas:

- Trabalho esperado versus trabalho planejado,
- Testes completados versus testes agendados, e
- Esforço despendido para completar tarefas versus esforços planejados

Os dados necessário para as métricas gerenciais podem vir dos sistemas de custo e controle de cronogramas da empresa. Estes gráficos permitem, além de uma visão geral de como estão as manutenções, uma oportunidade para revisões baseadas nas últimas performances obtidas.

As métricas técnicas são usadas para monitorar e controlar a performance técnica e qualidade do produto e devem ser apropriadas ao tipo de produto em questão. São exemplos deste tipo de métrica para avaliação da manutenção de software:

- Percentual de capacidade de processamento disponível usada;
- Percentual de memória disponível usada (volátil e perene);
- Percentual de banda de transmissão utilizada;
- Linhas de código por módulo;
- Complexidade dos módulos;
- Erros por 1000 linhas de código; e
- Erros associados ao software reportados em 1000 linhas de código

3.6.4 Usando e mantendo a documentação

Considera-se segundo Christensen (2001) documentação toda a informação sobre o software. São exemplos de documentação: requisitos, projeto, código, testes, manuais operacionais, manuais de usuários, histórico das alterações e cadernos de anotações dos engenheiros.

Como a documentação é essencial para a manutenção, qualquer área de manutenção precisa ter controle sobre a documentação existente e manter esforços em atualizar esta documentação, de forma clara, sem ambigüidades e completa.

As recomendações são a de usar o sistema de 'Controle de Alterações/Versões' – CM (*Change Management)* – para controlar os trabalhos planejados de documentação, monitorando a execução desta atividade. Desta forma, a manutenção de software é uma área produtora e consumidora de documentação.

3.7 As sete fases da manutenção de Software segundo padrão IEEE-1229

Conforme indicado no desenvolvimento do plano de manutenção de software, são indicadas ferramentas, técnicas e métodos. Esta seção é detalhada no padrão IEEE-1219 com um exemplo de como a manutenção de software pode ser executada e gerenciada usando um processo básico para descrever cada fase da manutenção de software.

Estas fases detalhadas no padrão são apresentadas a seguir e serviram de base para a aplicação apresentada no estudo de caso.

Tabela 1: Fases da Manutenção de Software

Fase	Objetivos
1. Identificação Problemas	Identificar, classificar e priorizar requisitos de modificação
2. Análise	Determinar a possibilidade de realização, caminhos alternativos de solução e escopo da alteração
3. Projeto	Desenvolver um projeto para as modificações aprovadas
4. Implementação	Executar as mudanças requeridas no software
5. Teste Sistema	Verificar se as modificações foram satisfeitas e nenhuma nova falha foi introduzida
6. Teste Aceitação	Aplica teste de aceite nos sistemas de software modificados certificando-se da satisfação do cliente-usuário
7. Entrega	Entrega as modificações nos sistemas de software e atualizações nas documentações.

As atividades apresentadas na Tabela 1 apresentam certa semelhança às atividades desenvolvidas durante as atividades de teste no desenvolvimento inicial do sistema de software. No entanto o ambiente sob a qual estas atividades estão sendo desenvolvidas é muito distinto, impondo restrições à manutenção. Um exemplo é o caso de vários grupos trabalhando simultaneamente. Cada uma das sete etapas será analisada a seguir, comparando-se quando possível às etapas que ocorrem no desenvolvimento.

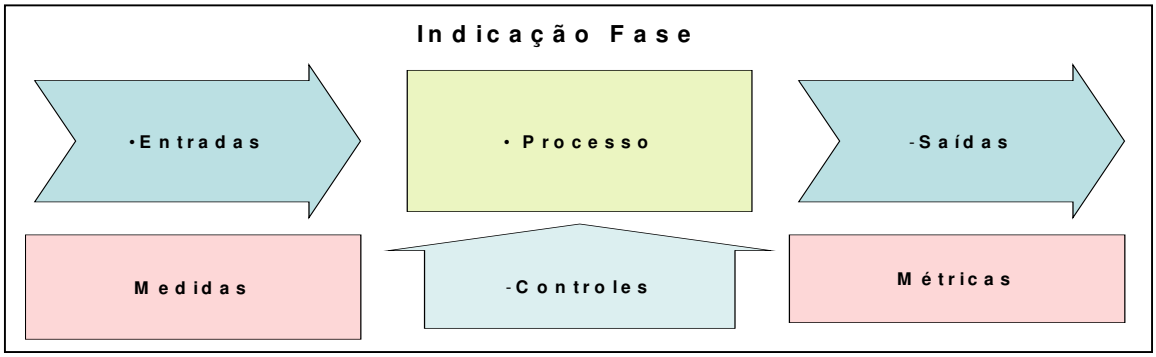

Figura 3: Modelo para representação de fases do processo de manutenção

Para facilitar a visualização, as Entradas, o Processo, as Saídas, os Controles, as Medidas e as Métricas foram representadas como ilustrado na Figura 3, onde o conteúdo do padrão IEEE-1219 é apresentado graficamente.

A seguir as sete etapas do padrão IEEE-1229 representando o processo de manutenção de sistemas.

3.7.1 Fase de Identificação do Problema

A fase de identificação do problema quando é caracterizada uma 'Requisição de Alteração no Software' (*Software Change Request – SCR*) ou 'Requisição de Mudança no Software' (*Software Modification Request – SMR*) é recebida pela área de manutenção de sistemas de software. Vale ressaltar a importância da formalidade desta solicitação, que deve conter informações padronizadas, garantindo os requisitos mínimos para avaliar a solicitação.

Os objetivos desta fase são:

- Garantir que o problema esteja apropriadamente documentado no formulário de requisição de manutenção de software.
- Obter os dados necessários para validar e verificar a solução.
- Garantir que a requisição de manutenção de software está devidamente armazenada e disponível para consultas no futuro.
- Garantir que se trata de um problema real e que os seus impactos foram compreendidos.
- Desenvolver uma estimativa preliminar dos esforços necessários à implementação da manutenção.
- Identificar preliminarmente uma versão à qual a manutenção possa ser executada.
- Obter visibilidade adequada (métricas) nesta fase de forma que a manutenção possa ser gerenciada.

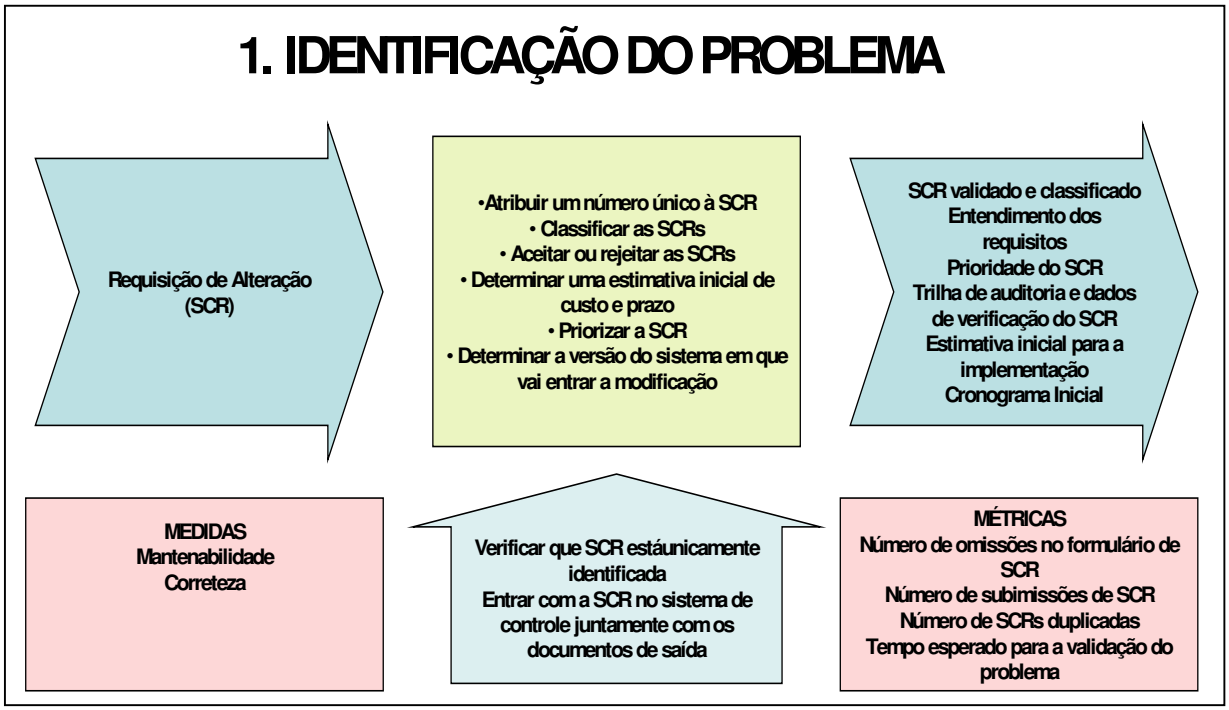

Figura 4: Fase de Identificação do Problema - baseado IEEE-1219

3.7.2 Fase de Análise

O objetivo da fase de análise é o de determinar o escopo e a capacidade de realização da manutenção documentada na requisição de manutenção, baseados na validação da requisição de manutenção, nos outros produtos geradas na fase inicial e na documentação disponível do sistema.

Uma análise de risco deve ser executada como parte da fase de análise. Com os resultados da análise as estimativas de esforço são revistas e é tomada a decisão de passar ou não para a fase seguinte.

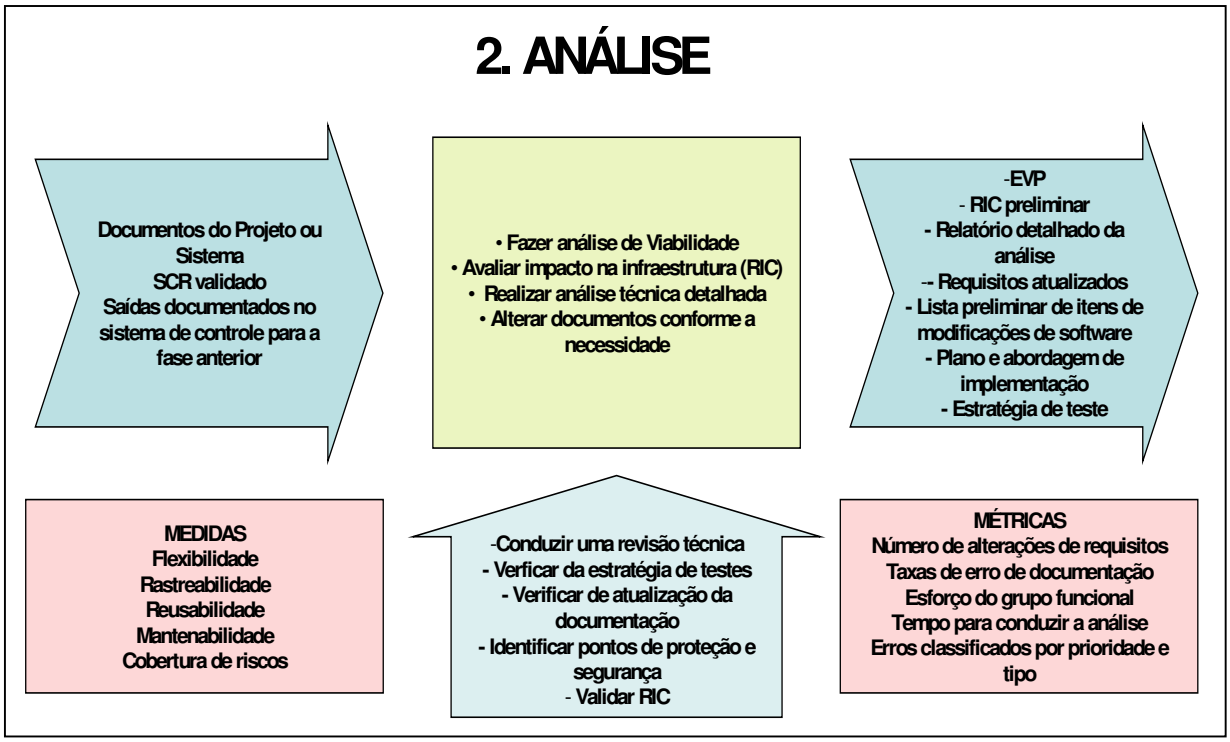

Figura 5: Fase de Análise - baseado no IEEE-1219[1]

3.7.3 Fase de Projeto

O objetivo primário da fase de projeto é a de criar um projeto para realizar as modificações no sistema a partir das saídas obtidas na fase de análise mais as documentações existentes, software e informações de banco de dados.

Os detalhes exatos do processo usado no projeto dependem da representação e metodologia originalmente usados no desenvolvimento, assim como do tamanho da modificação, do tamanho dos sistemas existentes, dos ambientes de desenvolvimento e teste disponíveis e os impactos nos usuários caso defeitos secundários sejam introduzidos durante a alteração.

A seleção e criação de testes de regressão devem ser considerados na fase de projeto. Algumas opções de projeto podem ser mais demoradas para implementar, mas muito mais fáceis de testar.

Todas as características do produto devem ser analisadas e consideradas durante a fase de projeto, permitindo que o produto continue confiável e passível de novas alterações após a implementação das alterações solicitadas.

[1] RIC - Request Infraestructure Contract

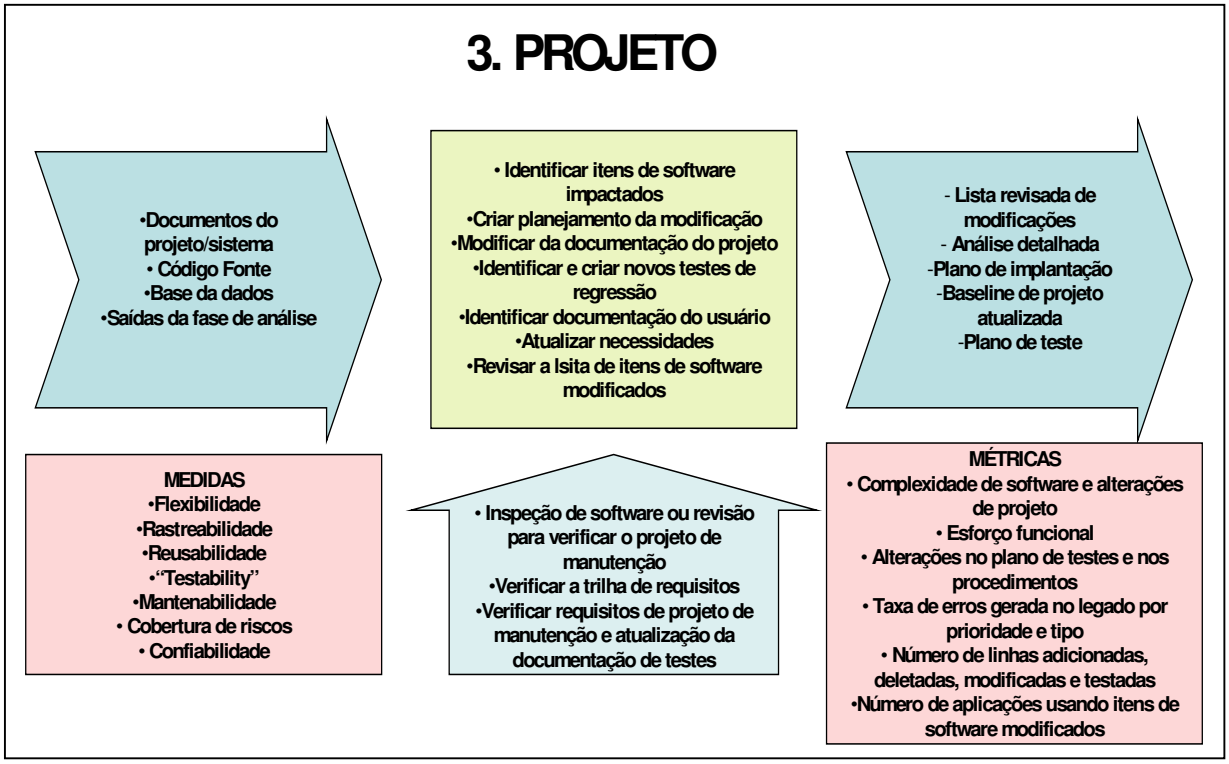

Figura 6: Fase de Projeto - baseado no IEEE-1219

3.7.4 Fase de Implementação

Na fase de implementação as mudanças do projeto são executadas para os itens de software identificados. Os resultados da fase de projeto juntamente com o código e documentação gerados são as entradas para esta etapa. Durante esta etapa o código fonte, os planos de teste, projeto e documentação usada pelos usuários são modificados a partir do projeto.

O código novo e o código que foi alterado deve ser inspecionado e revisado e passados pelo teste de unidade. Estes itens devem ser integrados ao produto final e novamente testados. Estes passos são usualmente desenvolvidos usando métodos e procedimentos usados durante a fase inicial de desenvolvimento. Qualquer anomalia (tipicamente defeitos presentes na documentação inicial ou aqueles introduzidos durante as atividades de manutenção) detectada durante o desenvolvimento destas atividades devem ser documentados e adicionados ao SMP (Software Maintenance Plan).

Após a conclusão da fase de implementação, realiza-se um 'Teste de Revisão após a Conclusão' - *TRR (Test Readiness Review*) para agrupar e analisar as evidências objetivas, usualmente dados do teste de integração, demonstrando que o sistema modificado está pronto para a próxima fase de teste, ou seja o teste de sistema. O

time de implementação apresenta estes dados para uma audiência composta pelo time de teste de sistemas (a qual executará a próxima fase da atividade de manutenção), o gerente de manutenção e se necessário outros participantes. Caso não seja obtido sucesso no 'Teste de Revisão após a Conclusão', deve-se voltar ao ponto apropriado da fase de manutenção, corrigir o problema e refazer os passos.

Como a fase de implementação é uma das maiores da implementação, grande parte do orçamento e do cronograma envolvidos são gastos nesta etapa. Da mesma forma, os maiores riscos e problemas de performance também ocorrerão nesta etapa. Portanto os esforços desta etapa devem ser monitorados de perto. Conclusão de tarefas, relatórios de progresso no cronograma e custos devem ser usados para quantificar os riscos. A abordagem deve ser baseada no controle dos mecanismos de custos e do cronograma.

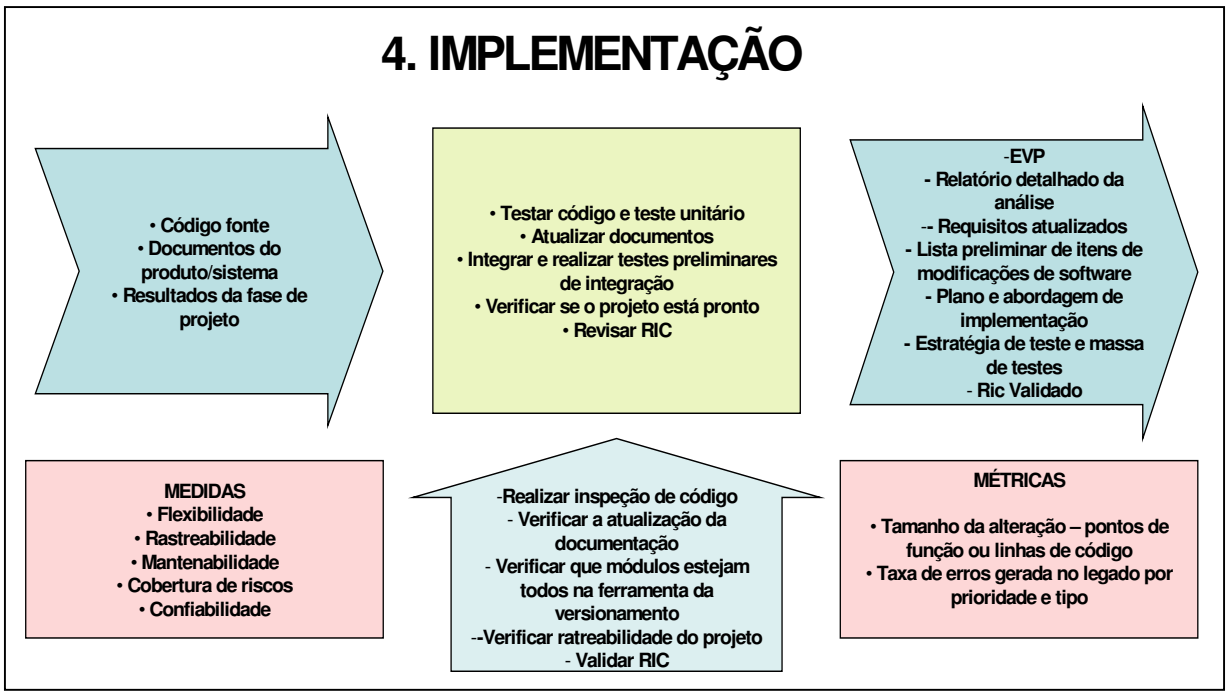

Figura 7: Fase de Implementação - baseado no IEEE-1219

3.7.5 Fase de Teste de Sistema

O objetivo da fase de testes do sistema é a de garantir que os requisitos mencionados na manutenção foram cumpridos e validar que nenhum erro foi introduzido durante as atividades de manutenção remanescentes. Os produtos modificados durante a fase de implementação são testados, seguindo a documentação do plano de teste revisado.

Os testes de sistema são idealmente realizados em um ambiente completamente operacional, com configuração semelhante a encontrada no ambiente de produção. Caso seja necessário, várias configurações devem ser testadas. O sistema é testado funcionalmente da entrada para a saída. Cada configuração deve ser alvo de uma variedade distinta de dados de entrada. Estes dados de entrada devem conter

informações esperadas e também inesperadas (imprevistas) retratando situações atípicas de uso. A quantidade de *stress test* necessária para um determinado sistema dependerá dos requisitos de confiabilidade do sistema, da própria complexidade do sistema e da complexidade do ambiente onde se encontra o sistema e do desenho planejado para as entradas.

Simulações devem ser usadas nos casos em que não é possível ter o ambiente de teste completo. Dependendo das condições contratuais, o cliente ou o usuário final talvez testemunhem o teste de sistema.

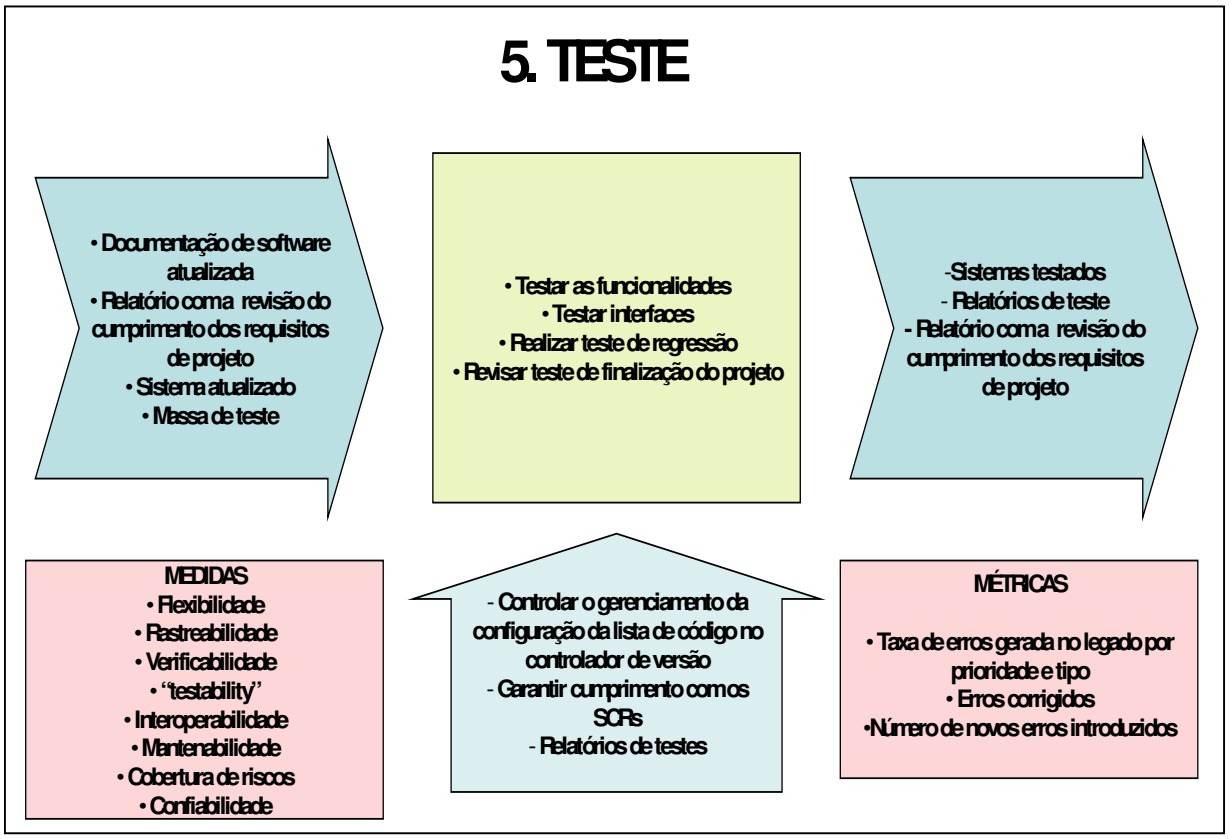

Figura 8: Fase de Teste - baseado no IEEE-1219

3.7.6 Fase de Teste de Aceite (Homologação)

O objetivo do teste de aceite é o de validar e garantir que as alterações no sistema e na documentação estão satisfatórias para o usuário final e para o cliente. O sistema produzido durante o teste de sistema, junto com todos os dados e documentação relevante são usados durante o teste de aceitação (homologação).

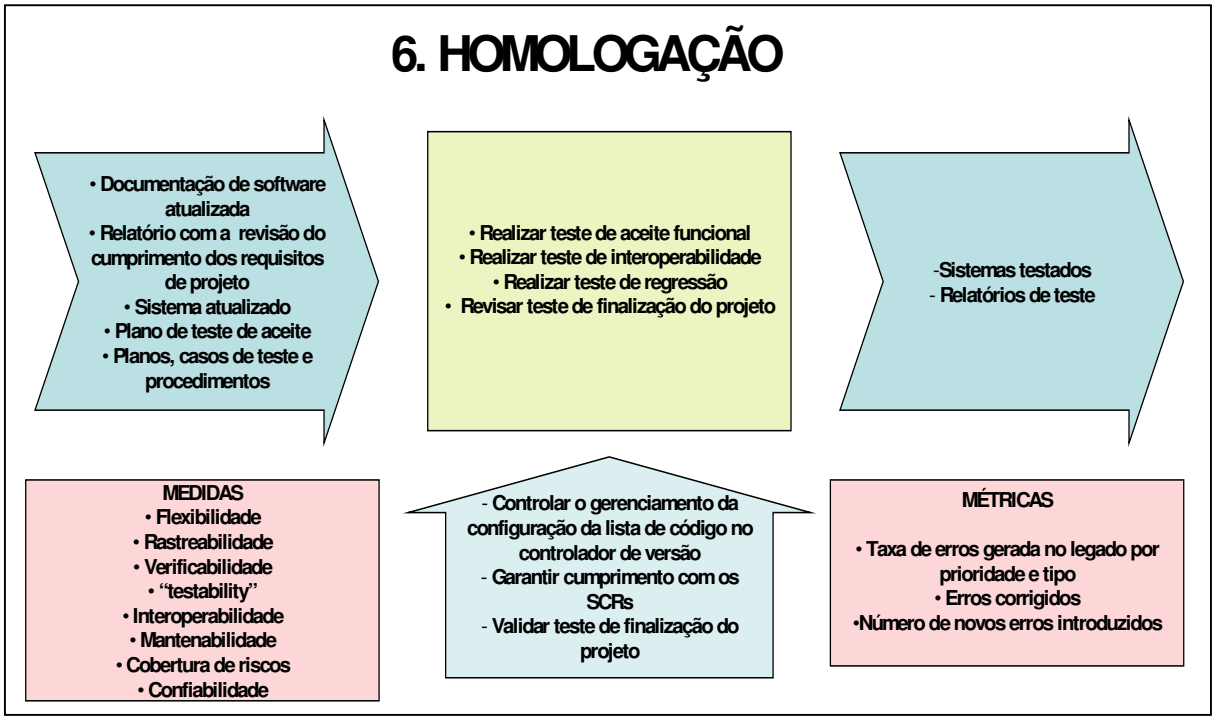

Figura 9: Fase de Homologação - baseado no IEEE-1219

3.7.7 Fase de Entrega

A fase de entrega tem por objetivo implantar as modificações de software e alterar a documentação para o cliente. A entrega pode substituir o sistema existente pela nova versão, sincronizar os controles e providenciar a unicidade para todos os usuários (inclusive os remotos).

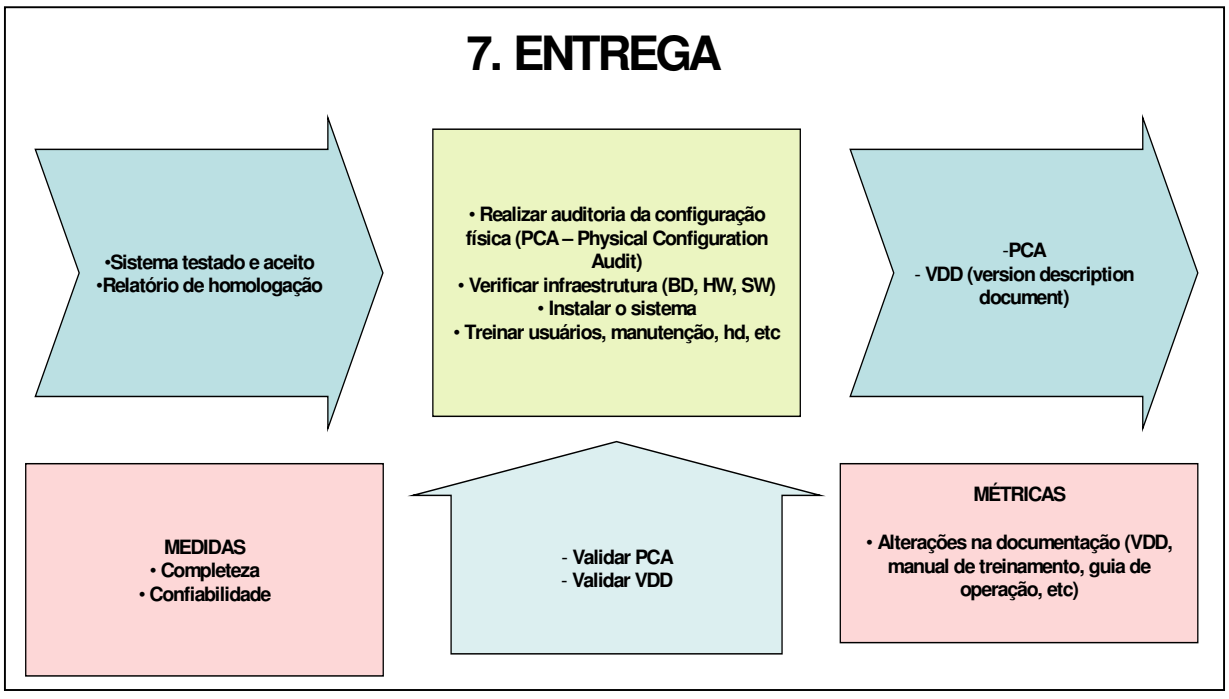

Figura 10: Fase de Entrega - baseado no IEEE-1219

3.8 Processo de Manutenção de Software ISO/IEC 14764

A padronização é uma linha seguida pelo mundo do software e estes esforços são guiados pela ISO/IEC 12007 - International Standard Life Cycle Processes. Para organizar os processos de manutenção de software foi publicada o padrão ISO/IEC 14764 - International Standard for Software Maintenance.

A norma trata questões de sua própria implementação, as diferentes estratégias de implementação das atividades de manutenção, o processo de manutenção além do detalhamento de cada atividade e tarefa de manutenção.

Desta forma, não há como lidar com manutenção de software sem analisar o ISO/IEC 14764, que pode ser resumido como na Figura 11, a seguir:

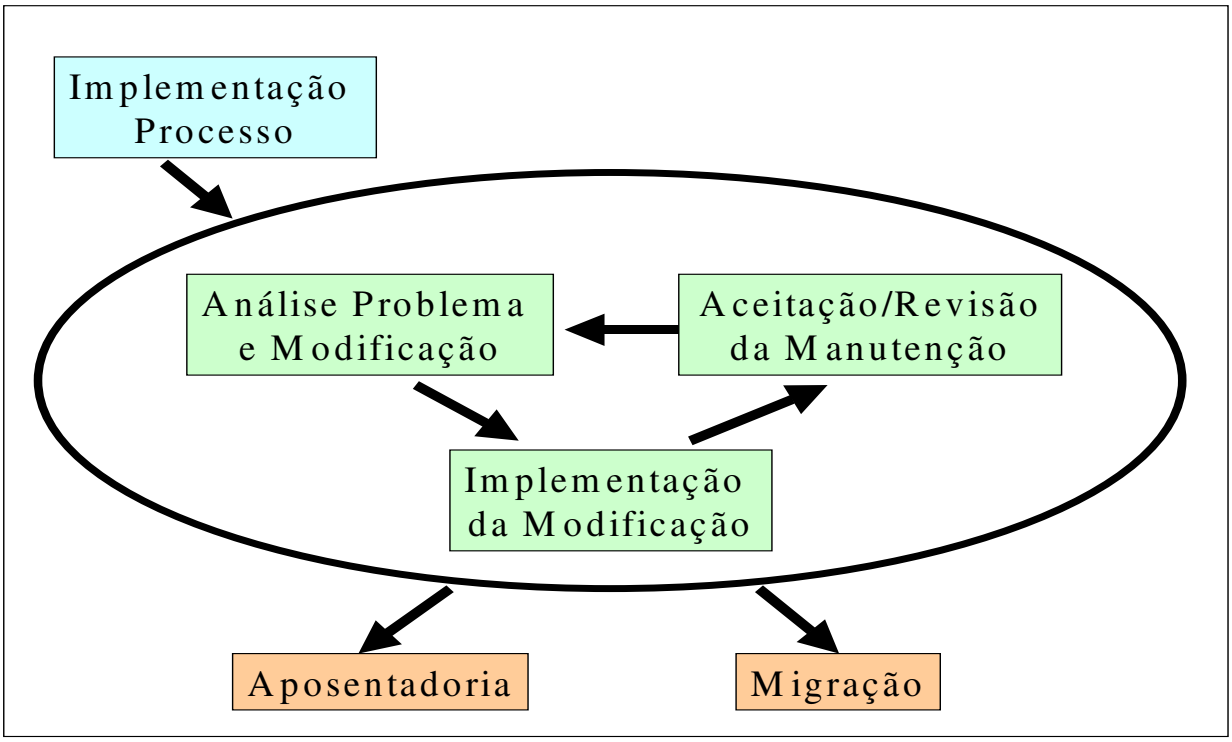

Figura 11: Processo ISO/IEC 14764 - Adaptado de SWEBOK (2004)

Como a base do trabalho está nas boas práticas sugeridas pelo padrão IEEE-1219, é importante relacionar as duas normas.

Segundo Pigoski (2003), as principais diferenças são:

a) O padrão IEEE-1219 provê as melhores práticas

b) ISO/IEC 14764 provê um modelo de processo

c) Os anexos do padrão IEEE-1219 são em sua maioria informativos, com apenas um normativo.

Existem trabalhos de harmonização das duas normas, o que no entanto foge do escopo inicial deste trabalho.

Capítulo IV

4 ORGANIZAÇÃO DE EQUIPES SEGUNDO O ITIL

4.1 Introdução

Devido à necessidade de adaptação dos recursos após a implementação das sugestões adaptadas do modelo sugerido no padrão IEEE-1219, buscou-se suporte no ITIL (IT Infrastructure Library) como base na organização das equipes de trabalho.

De acordo com Fernandes (1999) a implantação de uma metodologia de sistemas deve abranger toda a estrutura organizacional de tecnologia da qual depende a administração de sistemas, o que implica, muitas vezes, em mudanças de planejamento e controle praticados pela corporação, passando por mudanças nos procedimentos operacionais, nas descrições das funções e na própria estrutura organizacional.

A definição de novos papéis e a realocação dos integrantes das equipes pode ser fundamental numa empreitada que visa otimizar a área de manutenção de sistemas.

4.2 Definição

Segundo a definição encontrada em The ITIL and ITSM Directory1 o ITIL (IT Infrastructure Library) ou biblioteca de infraestrutura de tecnologia de informação é uma série de documentos que são usados para ajudar na implementação de um modelo de gerenciamento de serviços de manutenção na área de tecnologia de informação. Este modelo define como os serviços de gerenciamento são aplicados em organizações específicas. Por se tratar de um modelo, é totalmente adaptável a qualquer tipo de negócio ou organização que possua demandas na infraestrutura dos serviços de tecnologia de informação.

Assim que foi criado pelo governo do Reino Unido, o ITIL foi rapidamente adotado por todo o mundo como um padrão de melhores práticas para prover serviços de tecnologia de informação.

O ITIL é constituído por sete conjuntos:

- Suporte à Serviços
- Serviços de Entrega
- Planejamento para implementação de gerenciamento de serviços
- Gerenciamento de Infraestrutura de ICT (Information and Communications Technology)
- Gerenciamento de Aplicações
- Gerenciamento de Segurança
- Perspectiva do Negócio

O principal foco do gerenciamento de serviços de tecnologia de informação (IT Service Management -ITSM) está concentrado em duas grandes áreas: Suporte à Serviços e Serviços de Entrega. Juntas, estas áreas contém as disciplinas que tratam de provisão e gerenciamento efetivos dos serviços de tecnologia de informação.

4.3 Estruturação da equipe de manutenção de software baseado no modelo proposto no ITIL

Como encontrado em COBIT (2005) é essencial ter as pessoas certas executando as atividades certas no momento correto para obter o melhor rendimento das equipes dentro do modelo esperado.

Uma das dificuldades encontradas após a implantação de mudanças, é a adequação das pessoas às novas atividades que tem que ser executadas.

Buscou-se soluções já existentes para adaptá-las a uma área de manutenção estruturada pelo modelo proposto no padrão IEEE-1219.

---

[1] Texto encontrado no siste oficial do ITIL, referenciado na bibliografia.

A opção adotada foi derivada de sugestões apresentadas encontradas no ITIL – Information Technology Infrastructure Library. A estrutura avaliada foi apresentada no documento ITIL Organization Structure – Versão 2.0 de Julho de 2002.

As boas práticas do ITIL dão pouco suporte nas estruturas de tecnologia de informação adotadas nas organizações, principalmente porque cada organização é única e qualquer padronização teria que ser muito genérica. No entanto, um guia foi elaborado para dar assistência a estruturação de uma organização média, com cerca de 40 a 50 recursos, de uma área de Serviços de Suporte e Entrega, com sugestões de como introduzir esta estrutura. A questão de escalabilidade desta estrutura não é avaliada em detalhe neste trabalho, sendo feitas observações apenas em pontos considerados pertinentes.

Este guia, não pôde levar em conta que cada organização possui especificidades, com características peculiares do negócio que executa. E estas características também podem ser afetadas num dado momento por fatores externos à organização, tais como crises econômicas ou alterações na concorrência.

Mesmo assim, a estrutura sugerida foi muito útil na definição dos papéis e responsabilidades necessários. Este tipo de aproveitamento ficou explícito após a identificação da dificuldade assinalada no documento do ITIL indicando que alguns títulos atribuídos aos cargos podem, em alguns casos, serem substituídos por equivalentes funcionais. Por exemplo, o termo gerente poderia ser substituído por supervisor ou líder de equipe em alguns casos.

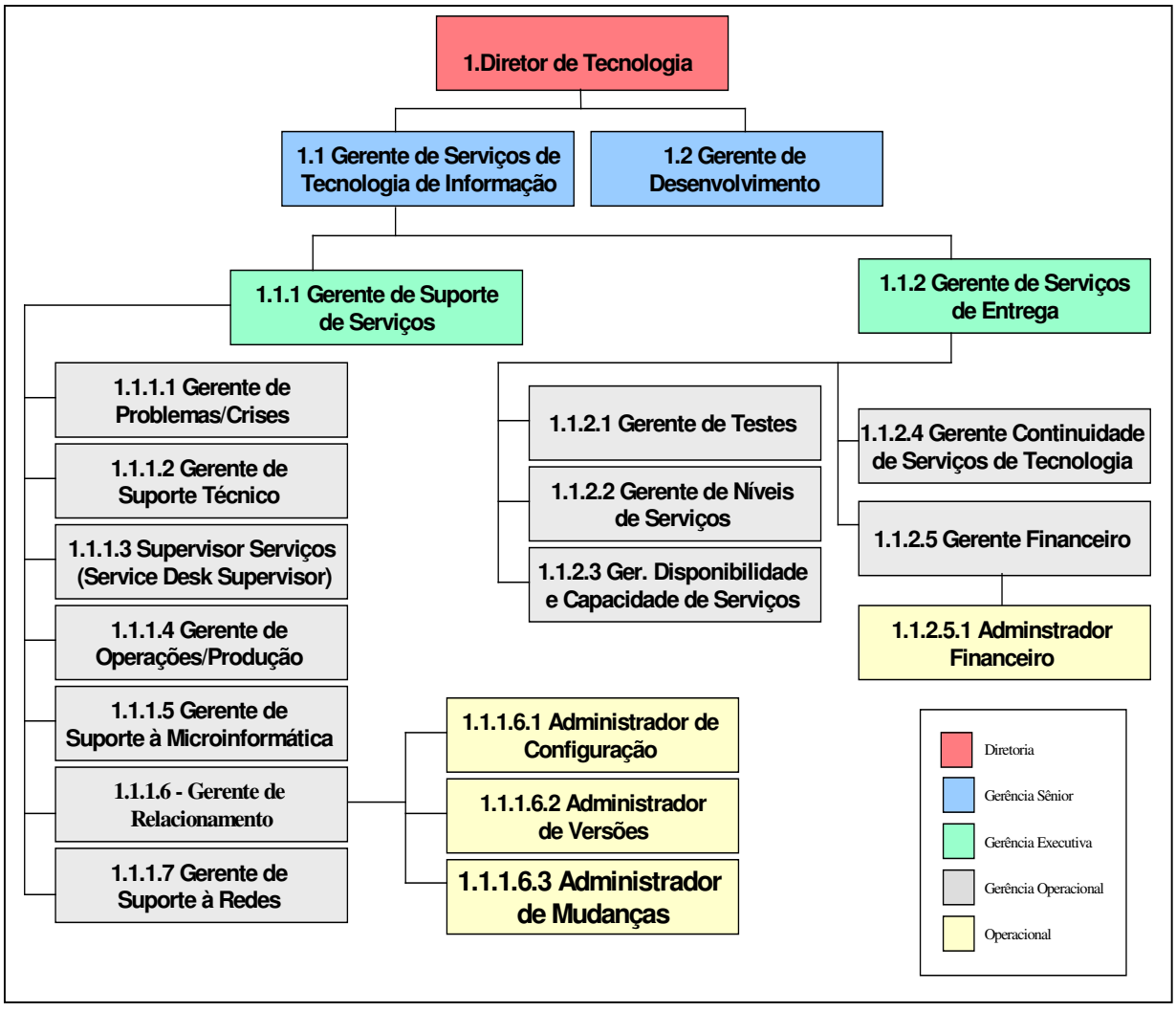

Figura 12: Organograma base - baseado no modelo proposto pelo ITIL

As principais atribuições da Figura 12 (organograma sugerido pela ITIL) serão detalhadas a seguir.

1.Diretor de Tecnologia

Deve ter acesso direta ou indiretamente à estrutura administrativa sênior da organização. Deve lidar com questões estratégicas demandadas desta estrutura ou dos gerentes de negócios seniores.

1.1 Gerente de Serviços de Tecnologia de Informação

O papel desenvolvido pelo gerente de serviços de TI é mais operacional do que o do diretor de tecnologia. Responsável por todas as atividades relacionadas aos serviços prestados pela área, sendo o principal foco deste organograma.

1.1.1 Gerente de Suporte de Serviços

Responsável pelos serviços de suporte, prestados de forma tipicamente reativa. Como possui um percentual grande de subordinados, as tarefas de gerenciamento de pessoal também são destacadas.

1.1.1.1 Gerente de Problemas/Crises

Responsável pela análise das causas dos problemas e pela implementação de medidas de prevenção para situações que possam levar à risco. Em organizações menores a atribuição pode ser compartilhada pelo gerente de dimensionamento ou pelo gerente de continuidade de serviços de tecnologia de informação (nota de rodapé indicando que estes cargos serão explicados posteriormente). Em organizações maiores o gerente de problemas/crises pode precisar de uma equipe proporcional à demanda e o quanto os problemas estão associado à fragilidade da estrutura adotada.

1.1.1.2 Gerente de Suporte Técnico

Responsável pelo suporte técnico e manutenção dos sistemas centrais: servidores, sistemas operacionais, subsistemas de disco. Os membros destas equipes precisam ser especialistas e desejavelmente possuir habilidades de outras áreas (*cross-skilling*), tais como medição de capacidade e disponibilidade, principalmente se estas funcionalidades não estiverem implementadas em outras gerências.

1.1.1.3 Supervisor Serviços (*Service Desk Supervisor*)

Responsável por todas as atividades de atendimento, também conhecidos como *'help desk'*. Dependendo da organização, são necessários mais de um supervisor de

atendimento para lidar com as questões de escalonamento e gerenciamento de equipe. Equipe esta que numericamente depende da natureza dos incidentes, tempo médio de atendimento e padrões de horário de trabalho.

Os Operadores de *Help Desk* e os Suportes de Segundo Nível são os responsáveis por lidar inicialmente com as ocorrências relatadas à Central de Atendimento, ficando para o pessoal do segundo nível a resolução de questões que exijam maior detalhamento técnico. Sugere-se a rotatividade de pessoal nestas duas funções.

1.1.1.4 Gerente de Operações/Produção

Responsável pelas atividades operacionais, tais como agendamento de horário para atividades, central de impressão, cópias de segurança de servidores (*back-up*). O número de recursos varia em função do horário de funcionamento, além do volume demandado.

1.1.1.5 Gerente de Suporte à Microinformática

A responsabilidade do gerente de suporte à microinformática depende da localização geográfica deste com relação aos usuários, mas passa pela garantia de bom funcionamento do equipamento usado pelo usuário final. Outra variante é o quanto da assistência ao hardware é terceirizada.

1.1.1.6 Gerente de CCR - Relacionamentos

Responsável pelo relacionamento com clientes e usuários.

1.1.1.6.1 Administrador de Configuração, Mudanças e Versões

Em grandes organizações é necessário um recurso (ou mais) para cada um destes papéis, mas devido a ligação forte entre estes processos, a linha de reporte deve ser única e quando possível executada sob a mesma supervisão. Cabe ao executor destas funções (1.1.1.6.2 Administrador de Versões e 1.1.1.6.3 Administrador de Mudanças) garantir a estabilidade do ambiente de produção a partir de qualquer evento que possa alterar este ambiente (nova versão, mudança de horário, alteração de versão de sistema operacional ou de aplicativo, etc.)

1.1.1.7 Gerente de Suporte à Redes

Responsável pela manutenção às diversas redes (*LAN, WAN*) disponibilizadas. Existe forte tendência na terceirização desta manutenção, principalmente quando há grande variedade de produtos (tipos de rede) no ambiente, requerendo especialização técnica para suporte à diferentes fornecedores.

1.1.2 Gerente de Serviços de Entrega

Responsável por prover novas atividades no ambiente de produção, com um papel fortemente pró-ativo com planejamento de médio e longo prazo. A gerência de serviços de entrega deve lidar com os próprios serviços da tecnologia da informação envolvendo práticas de gerenciamento para garantir que os serviços são prestados como acordado entre o provedor do serviço e o usuário (cliente ou consumidor).

1.1.2.1 Gerente de Testes

Responsável pela garantia da qualidade dos produtos entregues (implantados). A atividade de teste deve ser independente e não sofrer pressões externas, como prazo e adequação dos níveis de qualidade pré-estabelecidos.

1.1.2.2 Gerente de Níveis de Serviços

Responsável pelo controle dos acordos de nível de serviço (*SLA – Service Level Agreement*) e pelos acordos de nível de operação (*OLA – Operations Level Agreement*), garantindo que os serviços acordados com os usuários sejam executados segundo o que foi acordado.

1.1.2.3 Gerente de Disponibilidade e Capacidade de Serviços

Responsável pelo planejamento de longo prazo da continuidade dos serviços prestados. Por exemplo, o dimensionamento de equipamentos de armazenamento e a disponibilização de novos canais de comunicação para o crescimento vegetativo. A atualização tecnológica é indispensável, assim como a avaliação das soluções disponíveis no mercado.

1.1.2.4 Gerente de Continuidade de Serviços de Tecnologia

Trabalha muito próximo ao arquiteto de TI da organização. As atividades devem estar de acordo com as expectativas das áreas de negócio, atendendo às previsões de demanda

1.1.2.5 Gerente Financeiro

Responsável pelo orçamento da área de tecnologia de informação. Novas demandas não previstas, quebra nos contratos de nível de serviços e tarefas que não fazem parte da rotina da área de TI (um relatório extraordinário para auditoria externa, por exemplo) são exemplos de atividades que devem ser compartilhadas com o gerente financeiro, validando se o orçamento disponível é suficiente para cobrir estas demandas ou se será necessário novos aportes de recursos. O reporte pode ser matricial, incluindo o diretor de tecnologia e conta com o suporte de um *1.1.2.5.1 Administrador Financeiro* que pode cuidar dos aspectos técnicos da interface com a tesouraria.

1.2 Gerente de Desenvolvimento

Os papéis e responsabilidades das áreas de desenvolvimento e suporte de aplicações, assim como o gerenciamento de projetos não são cobertos pelo ITIL. Dependendo do porte da organização, um gerente de desenvolvimento pode fazer a ponte (ligação) entre as áreas de projetos e novas demandas com a área de entrega, facilitando o gerenciamento de novas demandas, testes e garantia de qualidade, garantindo que a transição para a produção ocorra de forma suave.

4.4 Outros papéis e responsabilidades sugeridos pelo ITIL

Além dos papéis e responsabilidades exibidos no organograma acima, outros são apresentados e podem ser adequados, segundo o modelo apresentado pelo ITIL, dependendo do tipo e do momento vivido por cada organização. Segue breve descrição de cada uma destas novas atribuições.

Gerente Estratégico

Papel mais adequado à grandes organizações, onde as responsabilidades estratégicas de curto e médio prazo podem ser compartilhadas com o diretor de tecnologia, a quem deve se reportar.

Gerente de Conta – Gerente de Relacionamento com Clientes

Responsável pelo contato entre os clientes (usuários ou não) e a área de tecnologia de informação. Em organizações maiores pode-se ter um gerente de relacionamento para cada grupo representativo de clientes. Auxilia na definição e acompanhamento dos níveis de serviço acordados. Se a senioridade do gerente de conta for compatível, algumas atividades do cotidiano podem ser acompanhadas, havendo divisão de responsabilidade com a área de *Help Desk*, provendo uma interface com o cliente-usuário.

Segurança e Auditoria

Segurança e Auditoria não são cobertos especificamente pela ITIL, ficando as sugestões restritas a participação de recursos junto ao gerente de disponibilidade. Hierarquicamente é feita a sugestão de subordinar este cargo ao serviço de entrega.

Arquiteto de Tecnologia de Informação

Responsável pelo planejamento e definição técnica da infraestrutura da organização, podendo em alguns casos ser responsável também pela elaboração de alguns padrões e avaliação de novas tecnologias.

Projetos Especiais

Hierarquicamente subordinada á área de serviços de entrega, os recursos alocados para projetos especiais precisam ter seus papéis e responsabilidades muito bem definidos. Se as atribuições não são da área de manutenção, estes recursos devem estar sobre a supervisão do gerente de projeto.

Transição para Produção

Existem vários pontos no ciclo de desenvolvimento de aplicações onde são necessárias considerações especiais para a transição para a produção. Este papel pode estar associado à área de desenvolvimento de aplicações garantindo que todos os envolvidos no processo de desenvolvimento foram envolvidos. Também pode assumir responsabilidades anteriores à implantação, tais como treinamento, familiarização dos usuários com novas ferramentas, categorização de riscos e elaboração de roteiros para atendimento (help desk).

Capítulo V

5 MODELO PROPOSTO

5.1 Introdução

O modelo proposto é baseado principalmente no padrão IEEE-1219, replicando e adaptando as fases propostas a uma empresa onde a manutenção de software esteja separada dos novos desenvolvimentos (projetos).

Ou seja, a priorização de uma manutenção emergencial (por exemplo) não pode afetar o prazo, qualidade ou o custo de um projeto, mesmo que seja um projeto fortemente acoplado a este sistema que esteja sofrendo a manutenção emergencial.

5.2 Apresentação Modelo Proposto

A principal adaptação no padrão IEEE-1219 para a criação do modelo proposto consiste na unificação das fases de Análise e Projeto apresentados na norma, consolidadas na fase de 'Construção'. Estas fases são tratadas num único bloco conceitual. A motivação para esta aglutinação foi a necessidade do gerenciamento destas atividades de manutenção no formato de *outsourcing*, envolvendo empresas terceirizadas (prestadores de serviço) e em alguns casos empresas quarteirizadas (prestadores de serviço para os prestadores de serviço).

Como citado em SWEBOK (2004), a terceirização da manutenção tem se tornado uma indústria de porte considerável. Muitas empresas terceirizam a manutenção dos seus sistemas tentando manter o foco no seu negócio principal, para ter diferencial competitivo no que sabe fazer de melhor.

No entanto, muitas vezes esta terceirização ocorre sem os pontos de controle adequados, mesmo sendo estes controles essenciais para manter o acompanhamento estratégico sobre estes sistemas.

Acrescente-se a este cenário sistemas de diferentes tipos, como alguns dos exemplos abaixo:

- pacotes de software totalmente fechados;
- pacotes com customizações;
- desenvolvimentos internos cujos fontes serão repassados para manutenção;
- fontes 'herdados' ou 'comprados', onde não houve a participação de nenhuma equipe de desenvolvimento conhecida.

Além da diversidade do tipo de contrato que o fonte se encontra e das diferentes formas de documentação de cada um, há também o problema crítico do porte destes sistemas, que podem ser pequenas aplicações especializadas desenvolvidas por pequenos parceiros que se especializam num nicho de negócio específico, assim como grandes pacotes internacionais que cuidam desde a planta fabril até o financeiro, passando pelas serviços prestados as áreas fiscais e contábil.

O desenvolvimento de controles eficientes e eficazes para o controle desta gama de sistemas pode facilmente fugir das prioridades estratégicas de uma empresa, principalmente se seu negócio principal não for sistemas.

Esta foi a grande motivação para unificar as fases do modelo proposto pelo padrão IEEE-1219 que diretamente lidam com a necessidade de codificação. Se por um lado a solução proposta pode parecer uma simplificação muito grande das melhores práticas sugeridas, ela viabiliza a implantação das demais etapas e permite que cada parceiro/fornecedor se adeque as regras solicitadas.

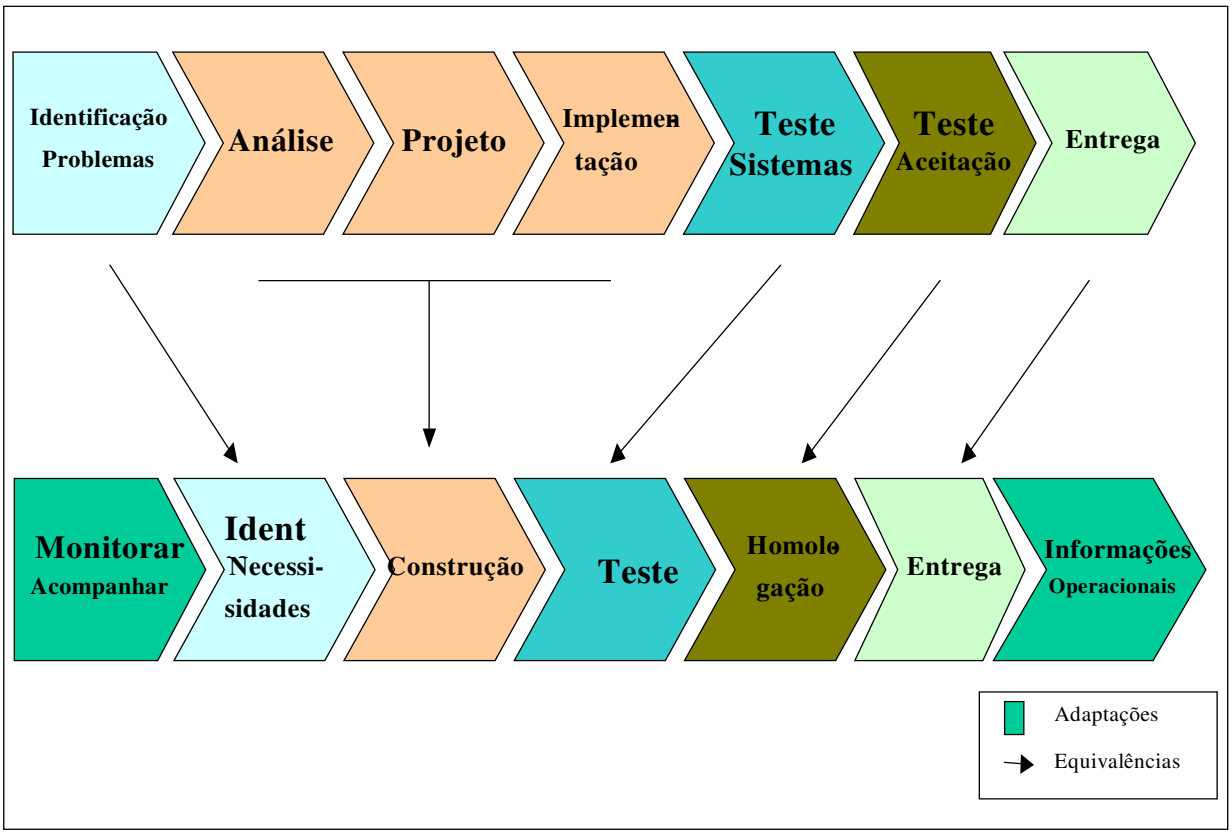

Figura 13: Comparação do Padrão IEEE 1219 com o Modelo Proposto

Monitorar e Acompanhar

A necessidade de regulamentar o elo entre projetos e produção justifica as atividades de monitoramento e acompanhamento, apresentando de forma clara um método para a transição da produção para a manutenção.

Um processo de absorção de atividades de projeto para a produção se justifica baseado em:

- Minimizar os riscos na absorção de um novo projeto;

  Estes riscos podem estar associados a implantação ou mesmo a interoperabilidade com os demais sistemas já em produção.

- Buscar sinergia com produtos já anteriormente ou simultaneamente;

  Esta sinergia pode ser representada por exemplo pela definição de implantações em janelas simultâneas.

- Minimizar o impacto na operação atual;

  Com a definição de todos os aspectos e recursos necessários à implantação pode-se planejar com antecedência o que será executado.

- Garantir a correta absorção, com atualização das documentações necessárias

  Por exemplo o ANS (acordo de nível de serviço e a Matriz de Risco, definida no PMBOK (2005) atualizada.

Identificação de Necessidades

Basicamente, a área de identificação de necessidades permanece a mesma sugerida pelo padrão IEEE-1219.

Apenas a abrangência e a forma de atuação são adaptadas, em função da alteração do modelo sugerido no padrão IEEE-1219 como um todo.

Se uma necessidade implica em um novo projeto, a área de projeto é acionada e são providenciados os subsídios (no caso requisitos) para montar o escopo, definir o patrocinador do projeto e demais boas práticas sugeridas pelo PMI (Project Management Institute).

Além das atividades endereçadas via PMBOK (2005), quando uma especificação de requisitos for necessária, esta deve seguir os parâmetros indicados pelo padrão definido em IEEE 830[1]:

a) Completude

b) Consistência

c) Classificação por importância e/ou estabilidade

d) Verificabilidade

e) Modificabilidade

f) Rastreabilidade

A área de identificação de necessidades não precisa dominar todas as técnicas e ferramentas, mas precisa coletar as informações importantes para alimentá-las. Mesmo que sejam necessárias várias iterações com os demandantes.

Construção

Ao aglutinar a Análise, o Projeto e a Implementação o papel da equipe de construção é a de gerenciar estas atividades que passam a ser desenvolvidas por terceiros.

Para o caso de diversos sistemas de fornecedores distintos, estes podem ser tratados sob um único responsável. Surge o conceito de integrador de sistemas: um integrador é uma empresa de software, parceira da empresa que aplica o modelo proposto e que já possui um relacionamento estreito com esta empresa.

O papel do integrador é o de estabelecer a ponte entre diversos outros fornecedores de menor porte mas de áreas de atuação semelhante. Por exemplo, se uma determinada empresa cuida da manutenção do CPD, esta também se envolverá com a empresa que fornece o link entre o CPD e os demais pontos de acesso. Desta forma, o integrador passa a ser o responsável pelo Acordo de Nível de Serviço (SLA - Service Level Agreement) do serviço.

Em contrapartida, a terceirização associada ao papel do integrador gera um distanciamento maior da metodologia de desenvolvimento utilizada nos sistemas.

---

[1] Std 830-1998: Recommended Practice for Software Requirements Specifications

Segundo Simcsik (2006), pode-se definir a metodologia de desenvolvimento de um sistema de informação como:

> " os diferentes processos com o emprego de inúmeras ferramentas para orientar o desenvolvedor e o usuário final na procura
>
> a) da Qualidade e Produtividade do software;
>
> b) de custos compatíveis com os valores financeiros da empresa;
>
> c) de termos previstos e dominados;
>
> d) de equipes sólidas, participantes e comprometidas com os usuários".

Uma vez que cada empresa pode usar uma metodologia de desenvolvimento distinta, esse distanciamento pode comprometer a qualidade, produtividade, custos, prazos e integração equipe-usuário. É o preço por ter os serviços encapsulados em uma 'caixa preta' (sem acesso aos detalhes do serviço prestado). Ou seja, ganha-se em generalidade nos quesitos administrativos, mas perde-se em especificidade para cada tipo de ocorrência de manutenção de software.

Teste, Homologação

Não foram feitas alterações significativas nestas etapas do processo sugerido pelo padrão IEEE 1219, apenas foram utilizados os conceitos originais de acordo com as alterações nas demais fases.

Para o modelo proposto foram incorporados alguns índices de qualidade para avaliar, se não comparativamente, mas pelo menos de forma objetiva características consideradas relevantes para cada tipo de sistema.

Muitas das informações sobre a construção dos sistemas poderiam ser inacessíveis, num modelo onde a terceirização seja fortemente utilizada. No entanto, outros parâmetros podem ser avaliados de forma quantitativa e constituir base para análises futuras.

Na busca por parâmetros consagrados academicamente e que pudessem ser utilizados no cotidiano dos sistemas utilizados, os indicadores de qualidade sugeridos por Gil (1999) foram adaptados, a partir do modelo completo indicado na Tabela 2, onde são apresentados os parâmetros de qualidade (eficiência, eficácia, etc.) para os estados 'Sistema em Operação Normal' e 'Sistema em Desenvolvimento', onde foi definido o critério de desenvolvimento de uma manutenção de software ou de projeto de software.

Para medir o item de qualidade 'eficiência' é sugerida a utilização de 'quantidade de compilações'. A partir da utilização de uma ferramenta de versionamento, pode-se medir não somente as compilações que serão efetivamente implantadas em produção, mas todas as vezes que um fonte tem que ser recompilado. Dessa forma pode-se medir a influência que outras versões podem ter numa determinada versão em específico.

Tabela 2: Indicadores de Qualidade - adaptado de Gil(1999)

Parâmetros Qualidade X Momentos Ambiente Informática	Sistema Operação Normal	Sistema em Desenvolvimento	Centro de Computação
Eficiência	Quantidade de Reprocessamento	Quantidade de Compilações	Tempo total de reprocessamento
Eficácia	Quantidade de relatórios entregues com atraso	Tempo final de atraso na implantação projeto	Tempo médio de resposta para todos sistemas aplicativos
Segurança Física	Quantidade de arquivos *back-up*	Quantidade de terminais para análise e programação	Quantidade de queda de energia elétrica
Segurança Lógica	Quantidade de arquivos protegidos por senha	Quantidade de rotinas de controle inseridas no sistema	Quantidade de senhas e cartões de acesso
Produtividade	Quantidade de relatórios produzidos por tempo de computador despendido	Tempo médio despendido por programa criado	Quantidade de horas de uso computador pela quantidade de profissionais do centro computação

Para medir a 'eficácia' de um sistema em 'operação normal', foi adaptado o item de controle 'número de notas fiscais emitidas em atraso', por ser um indicador já avaliado anteriormente.

Entrega

Os principais objetivos para a melhora nos processos de implantação de novas versões vieram das sugestões do padrão IEEE-1219 conjugada a necessidade de adequação de alguns destes processos às melhores práticas de Governança Corporativa de Tecnologia de Informação, sugeridas pelo COBIT (Governance, Control and Audit for Information and Related Technology).

O papel do COBIT é o de ser um guia, com as melhores práticas, para a gestão de TI com ênfase em controles e métricas, orientado para a área de negócios. O COBIT abrange diretrizes de gerenciamento, objetivo de controle e diretrizes de auditoria, sendo aceito internacionalmente como melhor prática para controle sobre a informação, tecnologia de informação e riscos relacionados.

A área de entrega é a última barreira para a implantação: precisa garantir que haja planejamento e o que foi planejado seja executado. Além do principal: lidar com as exceções, garantindo a integridade do ambiente de produção em todos esses casos.

No que diz respeito ao planejamento, o conceito de régua de versionamento é a principal ferramenta a ser utilizada e o tratamento das exceções deve ser previsto. Ou seja, deve-se ter uma regra geral e uma uniformização para tratar os casos que não se enquadram às regras.

Entende-se por Régua de Versionamento, a programação de mudanças em sistemas ou recursos, conforme calendário previamente estipulado, e em concordância com os padrões definidos pela área de Gestão de Mudanças.

Os objetivos da régua de versionamento são:

- Reduzir a quantidade de mudanças de versão dos sistemas: cada implantação pode gerar problemas que não foram mapeados e introduzir riscos que não tem ações específicas para mitigá-los.

- Melhorar a qualidade da documentação e dos testes / validações, permitindo a rastreabilidade de cada uma das alterações. Dessa forma, pode-se associar documentos de implantação, *scripts* de execução e relatórios por uma data específica, facilitando a organização e localização dessas informações no futuro.

- Reduzir a quantidade de falha nas mudanças ou indisponibilidade de sistema, uma vez que pode-se planejar com antecedência e evitar indisponibilidade de recursos (humanos ou tecnológicos).

- Agilidade no processo de planejamento.

- Segurança da informação, somente disponibilizando dados e informações críticas em momentos adequados.

- Permitir que empresas terceirizadas (*outsourcing*) possam participar do modelo de qualidade e entrega.

- Aderência aos padrão de mudança vigentes, atendendo as exigências de fiscalização e regulatórias, tanto de auditorias externas quanto internas.

Para participar da régua de versionamento, um sistema precisa seguir algumas premissas, que o qualificam a ser um sistema implantável com qualidade:

- Possuir um histórico favorável nas últimas três implantações (implantação de sucesso: sem necessidade de restaurar a versão anterior via *back-up*). Exceções devem ser consideradas e tratadas como implantações de risco.

- Cada sistema deve respeitar o limite máximo de implantações num determinado período. Sistemas críticos podem ter uma implantação por mês previstas na régua. Novos sistemas, que precisam de ajustes durante a fase de acompanhamento podem ter até 3 implantações por mês. Mais importante do que a frequência (e consequentemente o intervalo entre as implantações) é o respeito a regra imposta para cada classe de sistemas.

- O planejamento da régua deve ser feito com antecedência pré-estipulada, considerando-se o ideal o planejamento anual. No entanto, o modelo pode ser implantado com períodos menores (bimestres, por exemplo) até encontrar o ponto de equilíbrio. O planejamento deve levar em conta períodos atípicos,

como feriados, períodos de maior demanda pela área de negócio e outras sazonalidades conhecidas.

● A régua de versionamento deve impactar o ambiente de teste, direcionando os recursos usados em cada momento, permitindo a otimização de ambientes comuns.

● Toda e qualquer implantação deve estar coberta por um chamado técnico (ou demanda técnica) que a caracterize unicamente e permita associar todos os eventos (técnicos ou não) a esta identificação. Este registro deve permanecer aberto até o final da última etapa da implantação da régua, o que pode não ser necessariamente a implantação do próprio sistema ou recurso. Por exemplo, um remanejamento de banco de dados para um sistema pode ter sido um sucesso, mas se o banco não puder ser reiniciado na nova máquina, todo o processo tem que ser desfeito: o sucesso na implantação só ocorre no final de toda a mudança.

No entanto, não são considerados para efeito de Régua de Versionamento, sistemas parados (não estão mais parcial ou totalmente funcionando) e as correções de problemas que obrigatoriamente estejam impactando alguma das áreas de negócio e que não possam aguardar a próxima data de implantação.

Informações Operacionais

Da mesma forma que surgiu a necessidade de se montar um elo com a equipe de projeto, também ficou evidente ao longo do processo de implantação das melhores práticas sugeridas pelo padrão IEEE-1219 que seria necessário um canal único de divulgação das informações da equipe de manutenção.

Estas informações podem ser distribuídas para as diversas áreas: tanto para os usuários (para avisá-los da interrupção de um sistema para implantação de nova versão) quanto para as áreas que tomam decisões estratégicas (diretoria executiva).

Capítulo VI

6 ESTUDO DE CASO

6.1 Introdução

O modelo proposto é exercitado na adequação da área de manutenção de serviços de Tecnologia de Informação de uma empresa do ramo alimentício.

A contextualização da empresa é essencial para a aplicação do modelo, devido a implementação prévia à aplicação do modelo da separação das áreas de manutenção e projeto de sistemas. A empresa em questão adota a separação entre as áreas de produção (manutenção de software) e de projetos, caracterizando a aplicação dos padrões sugeridos no padrão IEEE-1219 e se enquadra também às melhores práticas sugeridas pelo ITIL para organização dos recursos de TI.

6.2 Contexto empresa apresentada no estudo de caso

A contextualização das características da área de tecnologia da empresa é importante para referenciar e justificar algumas escolhas e alternativas seguidas.

Num primeiro momento, a área de tecnologia concentrava todas as ações relativas a projetos e manutenção numa mesma equipe. Esta é a regra e não a exceção, sendo a realidade da maior parte das empresas brasileiras. As divisões costumam ocorrer nas especificidades técnicas, como por exemplo equipes de desenvolvimento de sistemas, equipes de suporte e equipes de hardware. Para as equipes de hardware ainda é comum a subdivisão nas áreas de telecomunicações e equipamentos de informática propriamente ditos (como servidores e redes).

Dentro desta estrutura unificada, quando um novo projeto se tornava necessário, alguns membros das equipes eram destacados para colaborarem nestes projetos. Muitas vezes estes profissionais acabavam acumulando suas atividades rotineiras de manutenção com as novas responsabilidades associadas aos projetos.

Mesmo quando consultorias externas eram contratadas para gerenciar os projetos, os recursos destes projetos tinham compromissos com a continuidade do negócio, que acabavam assumindo uma prioridade maior, em detrimento das atividades associadas ao projeto. Quase sempre,este tipo de comportamento estava associado às restrições de tempo: por exemplo, um erro num sistema em produção não pode esperar tanto quanto um relatório preliminar que proverá informações para a decisão de uma entre várias alternativas. Neste exemplo, o recurso do projeto foi omisso enquanto que o recurso da manutenção cumpriu seu papel. O ponto chave é que estes dois papéis eram executados pela mesma pessoa. E por mais que horas extras fossem acumuladas para tentar compensar estas ocorrências, o stress, cansaço e desmotivação acabavam comprometendo tanto as atividades de manutenção quanto os projetos.

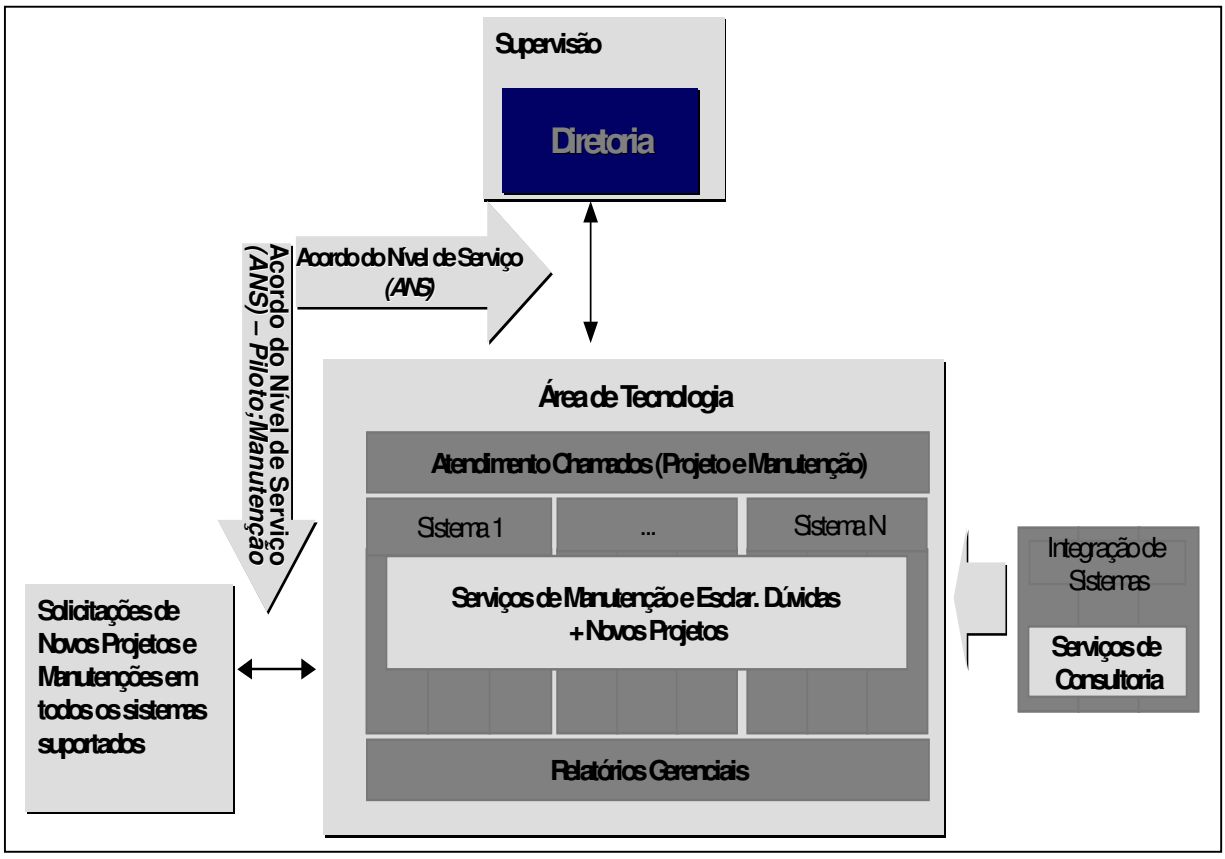

Figura 14: Estrutura área de TI (antes da separação projetos e manutenção)

Os principais problemas que motivaram que alguma ação fosse tomada dentro do contexto de TI foram:

- Constante atraso nos prazos estimados para os projetos com a justificativa de que os recursos estavam alocados em atividades de manutenção.
- Subutilização de recursos em algumas áreas com menor demanda que poderiam ser aproveitados em outras (projetos ou manutenções).
- Adequação de perfis nas atividades corretas: nem sempre um bom técnico apresentava as características necessárias para a condução de projetos e vice-versa.
- Queda na qualidade das manutenções (levando a constantes retrabalhos) sob as justificativas de que o tempo disponível para estas manutenções era limitado pelas atividades dos projetos.

A decisão tomada em busca de uma solução foi a contratação de uma consultoria internacional, que após estudos e análises sugeriu a separação das equipes de projeto e manutenção.

As principais estratégias por trás desta decisão assim como a justificativa tomada fogem do escopo deste trabalho, que é o de apresentar como foram obtidas melhorias das atividades associadas à manutenção de sistemas.

A Figura 15 a seguir apresenta a subdivisão em áreas distintas: Projetos e Manutenção.

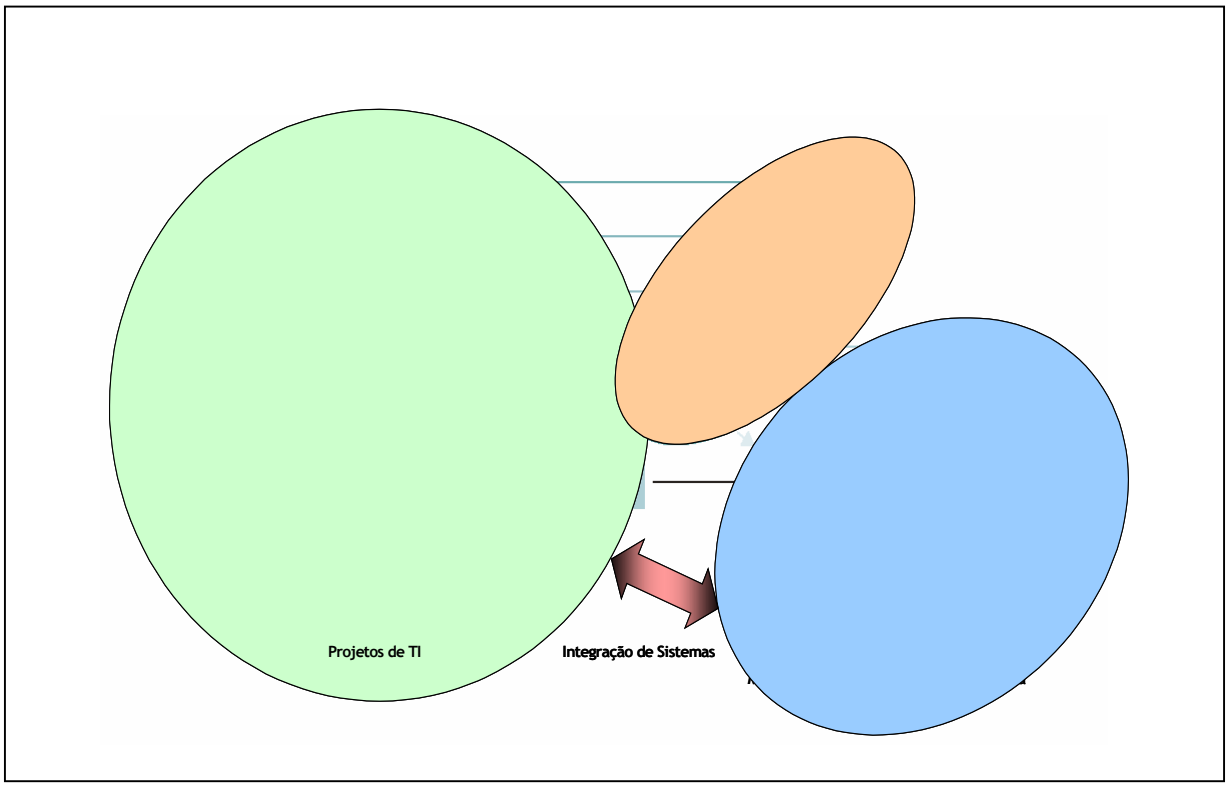

Figura 15: Representação macro das atividades de TI após separação das áreas de Projeto e Manutenção

A separação das áreas trouxe uma série de benefícios para a área de projetos:

- Os prazos dos projetos passaram a ser cumpridos ou os erros estavam associados a outros fatores que não a divisão de recursos com a área de manutenção.
- Otimização dos recursos alocados em projetos, sendo possível a alocação de um mesmo profissional para diversos projetos simultaneamente.
- Os profissionais da área de projeto foram treinados/contratados segundo as boas práticas sugeridas pelo PMI (Program Management Institute), permitindo a homogeneidade das diversas equipes de projeto e maior facilidade na padronização das ações.

No entanto, a área de manutenção não foi contemplada com ações específicas que contemplassem a nova realidade. Projetos eram inseridos no ambiente de produção sem terem sido assimilados totalmente, e principalmente, sem haver treinamento específico das equipes de manutenção.

Outro fator que gerava constantes problemas era a necessidade de trabalho em conjunto das equipes de manutenção e projeto - como não houve uma adequação e reformulação dos processos associados às equipes de manutenção os bons resultados não podiam ser repassados às demais equipes e os maus resultados não eram catalogados e usados para que não ocorressem novamente.

6.3 Uso das boas práticas à área de Manutenção de Tecnologia de Informação

Uma vez constatada a necessidade de atuação na área de manutenção, diversas abordagens foram apresentadas. No entanto, pela limitação orçamentária (os recursos foram concentrados na área de projetos) optou-se pela adoção de boas práticas sugeridas pela engenharia de software, com uma equipe interna para o desenvolvimento de um projeto para a área de manutenção de tecnologia de informação. Este projeto visava tornar as ações, processos e procedimentos mais eficientes, com o objetivo final de reduzir custos.

Um dos problemas encontrados foi a falta de recursos com o perfil de condução de projetos dentro da própria área de manutenção, necessários para o desenvolvimento das atividades relacionadas a esse projeto.

Devido as limitações de tempo, o foco foi mantido no padrão IEEE-1219. A proposta inicial era a de usar o modelo como proposto, minimizando as adaptações o que facilitaria a divulgação do processo.

No entanto, devido ao modelo de terceirização dos sistemas e equipamentos ser amplamente utilizado, optou-se por adaptar a etapa inicial, mais especificamente no que diz respeito as etapas centrais do processo de manutenção. Foi uma opção entre o "ótimo" e o "bom", devido as características da empresa (e das empresas parceiras) assim como limitações de tempo e custo existentes para a obtenção de resultados.

Outro fator decisivo foi a natureza da operação base da empresa e da postura da área de informática como área prestadora de serviços e não como área fim da empresa.

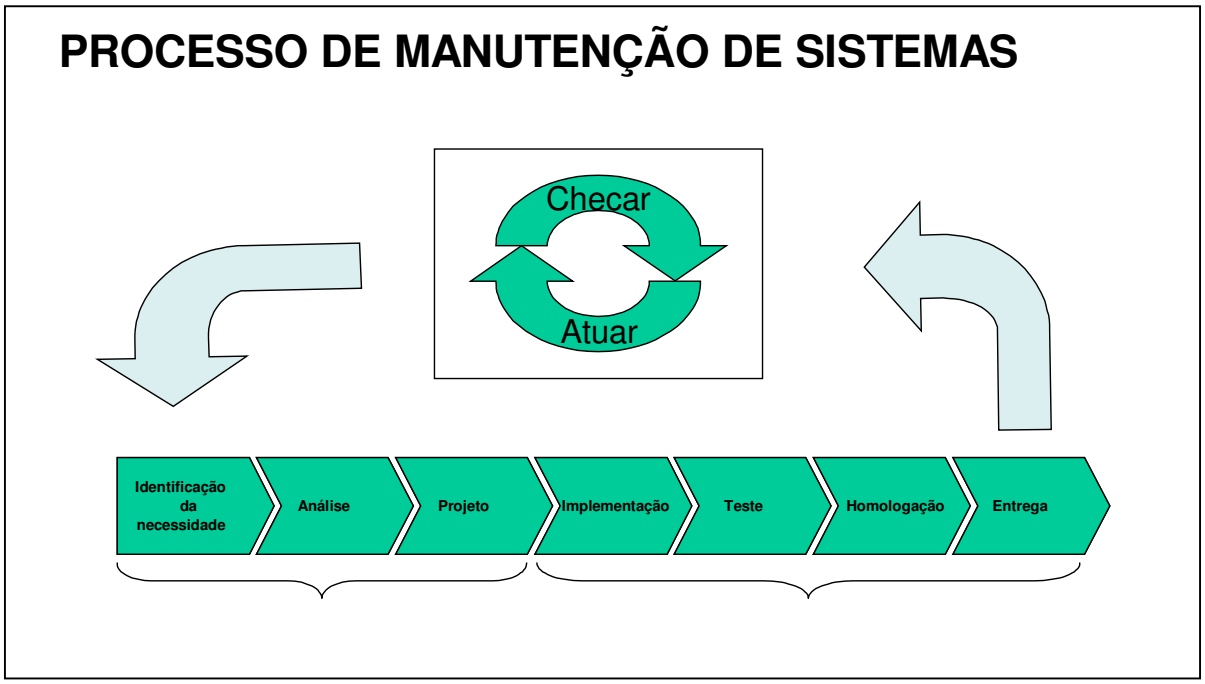

Figura 16: Processo de manutenção de sistemas - baseado no IEEE-1219: visão inicial

Desta forma a análise, projeto[1] e implementação foram 'encapsulados' num processo chamado de 'Construção'. O nome 'Construção' veio originalmente do conceito de fabricação, ou seja, onde as manutenções seriam confeccionadas.

A aplicação do modelo proposto foi feito de forma gradual, em etapas. Durante a implantação constatou-se a necessidade de definir claramente um processo de ligação entre os projetos e a área de manutenção. Ao conjunto destas atividades e procedimentos deu-se o nome de 'Monitorar e Acompanhar' e sua representação está, graficamente (como pode ser visto na representação abaixo) no início dos processos de manutenção, mas permeia todas as demais etapas.

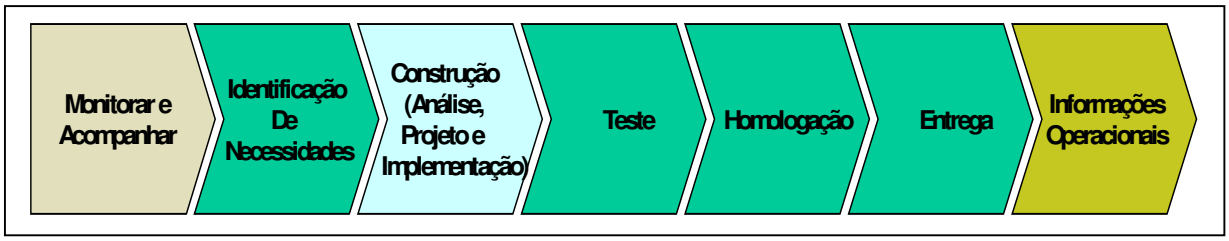

Figura 17: Processo Manutenção Sistemas - Adaptado IEEE-1219

Monitorar e Acompanhar

O maior desafio do processo de monitoramento foi a definição de regras para a transição de sistemas no estágio de projeto para a produção.

Justifica-se então a necessidade de proposição clara de um processo de absorção de sistemas com a definição dos documentos utilizados durante o processo de transferência de sistemas e recursos (infra-estrutura, por exemplo) da fase de projeto para a produção. Este processo pode ser aplicado a qualquer sistema novo, correção ou item de infra-estrutura.

A definição do processo de absorção passa pela relação dos documentos necessários para a transição assim como a definição da responsabilidade de quem deve preenchê-los. Foram identificados os seguintes documentos como importantes para o processo de absorção:

a) Formulário de aprovação de demanda

b) Matriz de risco do projeto

c) Planilha com as atividades necessárias à implantação

d) Formulário de aprovação da demanda

e) Termo de aceite das equipes técnicas da documentação das manutenções realizadas (manuais atualizados, alterações em procedimentos de instalação, necessidades de hardware)

f) Termo de homologação do gestor do sistema

---

[1] O termo 'projeto' neste contexto diz respeito ao projeto do ciclo de vida do sistema, especificamente associado ao 'como' fazer e não ao projeto como uma atividade única, com prazo específico (como definido no PMBOOK).

Estes documentos foram introduzidos gradualmente e foram definidas as regras de quando a aplicação de cada um deles fazia ou não diferença na qualidade do processo. Projetos de grande porte, que envolvem muitas equipes ou que são suportados por empresas com estruturas apropriadas a processos estruturados tiveram uma adaptação às novas exigências de documentação mais fácil. No entanto, nos pequenos fornecedores (usualmente associados a pequenos projetos) houve maior resistência, e foi justamente junto a estes fornecedores que houve melhora no processo de implantação.

Identificação de Necessidades

Como o objetivo principal desta equipe era o de apontar, classificar e priorizar todas as ocorrências que poderiam ser potenciais problemas, a padronização na forma como estas necessidades precisam ser catalogadas é essencial.

A equipe responsável pela identificação das necessidades adotou o modelo proposto pelo padrão IEEE-1229 para as requisições de alteração de software, sendo este o modelo adotado inicialmente para catalogar as necessidades e problemas encontrados.

Uma vez definida a categoria de manutenção, vários procedimentos poderiam ser adotados. Por exemplo, se fosse necessária montar uma especificação de requisitos para corrigir uma funcionalidade, a área de construção já seria acionada e trabalharia em conjunto com a área de identificação de necessidades.

Tabela 3: Modelo de Requisição de Manutenção de Software - Adaptado IEEE-1219

Requisição de Alteração de Software		
Identificação do Alterador		___ Nro. Requisição
Tipo Requisição	Deficiência Requisitos Alteração Ambiente Melhoria	
Identificação da Alteração		
Identificaçação do Solicitante Nome Título Organização Data	Urgência Crítico Muito Importante Importante Incoveniente Interessante	Identificação da Baseline Nome do Projeto SCI Ident. Configurador Ident. Versão
Alteração Solicitada		
Descrição da alteração solicitada	Necessidades para a alteração solicitada	
Validação da Requisição		
Aceita para Validação	Rejeitada para validação por: Incompleta Inválida Duplicada	
Assinatura	**Data Validação**	

Construção

Na busca de incremento na produtividade, a empresa utilizada no estudo de caso buscou manter o foco em sua competência básica, e como citado por Simcsik (2006) busca o binômio "custo x benefício" maximizado.

Se o gerenciamento de recursos internos da área de tecnologia e informação já possa estar completamente fora do conceito de "*core business*", a necessidade de planejamentos, controle e comparação (para eventuais recontratações) continua existindo.

Neste sentido, sem entrar no mérito de custos, a implantação da área de construção vem a atender a esta necessidade. Ou seja, o custo total de propriedade (TCO – *Total Cost Of Ownership*) deve englobar as despesas (operacionais e administrativas) dos recursos da área de 'Construção'.

Por englobar as fases de Análise, Projeto e Implementação a ’Construção' é sem dúvida a etapa mais crítica de todo o processo.

Se por um lado existe um ponto de contato único para vários fornecedores, para o usuário final ocorre um distanciamento entre a sua ferramenta e quem presta o serviço de manutenção, o que levou a protestos dos usuários quando das primeiras manutenções.

Uma das razões para o modelo de integradoras ter apresentado um alto índice de rejeição entre os usuários está associado ao fato dos usuários não terem mais acesso direto à empresa que vai efetivamente resolver o seu problema quando uma manutenção de software for necessária. Ao ser constatado um erro, este passou a ser notificado a uma das empresas integradoras, que por sua vez se responsabilizava pela solução do mesmo, mesmo que envolvendo uma outra empresa.

Em alguns casos este tipo de abordagem (via empresa integradora) pode ser mais rápida do que a intervenção direta da empresa responsável por um subsistema específico. Por exemplo, quando se trata de um erro no ambiente, em que todos os sistemas estão sendo afetados (uma queda ou diminuição de performance no link de comunicação) um comunicado geral pode informar os usuários, evitando por exemplo abertura de chamados de manutenção em outros sistemas.

No entanto, para os casos em que uma segunda empresa tem que ser acionada pela mantenedora acrescenta-se uma etapa ao processo, o que invariavelmente aumenta o tempo de atendimento pois existem dois Acordos de Nível de Serviço (SLA) a serem cumpridos: um com a integradora e outro com a empresa responsável por um subsistema de informática específico.

Agindo como um “intermediário” o integrador cumpre as melhores práticas da engenharia de software, (exigidas por contrato) não permitindo por exemplo alterações nos requisitos sem que seja feita uma formalização desta alteração. Estas ações podem ser contrárias as prioridades das companhias que tem outro negócio que não o da criação de sistemas de software (por mais que seja amplamente divulgado pela engenharia de software que os métodos e processo melhoram os tempos de resposta). Se uma especificação foi mal feita por um usuário, este espera corrigí-la com um simples telefonema, acrescentando os “detalhes” que faltaram na especificação inicial. Do ponto de vista deste usuário, seu trabalho não é manter um sistema funcionando, mas operá-lo produzindo um resultado de forma mais eficiente se não estivesse usando ajuda computacional.

O distanciamento do usuário final das equipes de sistemas é relatado por (Weinberg 1990 p.140 e p.142), e ilustra no nível de análise e projeto os efeitos deste distanciamento:

> “É apenas humano resistir a comprometer nossos interesses; portanto, se quisermos alterar especificações, devemos tornar isto o mais fácil possível. (...) Quando a especificação e o projeto ficam separados com demasiada rigidez, os usuários passam o controle de suas especificações para o analista e o projetista. As vezes os usuários lamentam amargamente as especificações que foram feitas no seu interesse. Isto é bastante ruim, mas o pior são aqueles usuários que simplesmente aceitam essa perda do controle como parte inevitável da maneira pela qual o sistema tem de ser desenvolvido.”

De certa forma, o atendimento 'personalizado', feito informalmente foi substituído por um processo burocrático (no melhor sentido da palavra), onde o usuário se via forçado a registrar inequivocamente qual é o problema que deve ser solucionado, uma vez que será exatamente isto que será produzido (e validado na fase de homologação).

Desta forma, os problemas relacionados a especificações mal feitas puderam ser separados dos problemas oriundos dos próprios fornecedores, como prazos (não cumprimento do SLA) e baixa qualidade.

Teste

Os testes são realizados pela área de sistemas e também pelos usuários. Quando cabível são executados testes integrados, em ambientes específicos.

A garantia de que a versão que está sendo testada será a mesma versão que será implantada somente foi conseguida com a implantação de uma ferramenta de versionamento, essencial nos testes e na homologação.

Mesmo em implantações iminentemente técnicas os usuários participam obrigatoriamente, atestando que todas as condições necessárias ao funcionamento dos sistemas que o afetam continuam sendo satisfeitas. São desenvolvidos planos de teste com maior ou menor rigor, de acordo com o tipo de sistema. Por exemplo, se a implantação de um novo grupo de ramais para todas as filiais exige a participação dos usuários fazendo testes, será elaborado um plano que deva ser executado por pelo menos um usuário de cada filial, e o resultado comunicado oficialmente.

Com os testes, muitas versões foram 'barradas', ou seja, impedidas de serem implantadas. E de acordo com o conceito de versão utilizada pela ferramenta de versionamento, todas as demais versões que estivessem sendo desenvolvidas precisam ser reavaliadas. Se por exemplo a versão 1.7 foi impedida de ser implantada, a versão 1.6 que estava em teste e a 1.5 que estava na fila aguardando publicação voltam todas para desenvolvimento: o desenvolvedor tem que verificar se o erro da versão 1.7 não irá impactar nas versões 1.6 e 1.5, que iriam ser testadas e implantadas.

Dessa forma, poupa-se o usuário de encontrar o mesmo erro mais de uma vez, ou pior, de implantar uma versão que traz de volta um erro que já havia sido corrigido.

No entanto, para os sistemas onde as alterações eram frequentes (manutenções corretivas, preventivas ou mesmo de regra de negócio) uma versão poderia demorar muito mais para ser implantada/disponibilizada para o usuário.

Uma forma de avaliar a qualidade dos sistemas foi a de medir o número de *baselines* extornadas, ou seja, quantas vezes tentou-se implantar uma versão e esta foi impedida por existirem erros considerados impeditivos para implantação pelo usuário.

Estes dados foram compilados e apresentados na Tabela 4 a seguir, onde vale ressaltar que uma *baseline* pode gerar extorno em cascata de outras alterações no mesmo sistema que estivessem aguardando para serem publicadas. Considerou-se também nestes dados apenas os extornos por erros associados a qualidade - foram

filtrados os estornos por impedimento de data (régua de versionamento) ou necessidade de conciliação de implantação com outro sistema ou projeto.

Tabela 4: Estornos de Baselines por Sistemas

Categoria Sistema	Id.Sistema	Nro. Estornos por Baseline						
		jun/04	jul/04	ago/04	set/04	out/04	Nov/o4	dez/04
Plataforma Baixa	Sistema 1	17	9	11	13	8	18	21
	Sistema 2	7	4	2	3	1	5	8
	Sistema 3	23	14	9	16	8	17	20
	Sistema 4	8	3	2	5	2	6	5
	Sistema 5	19	15	19	13	12	14	14
	Sistema 6	7	6	7	3	5	4	4
	Sistema 7	9	9	8	9	7	16	7
	Sistema 8	10	3	5	3	3	4	7
	Sistema 9	8	7	14	18	12	23	18
	Sistema 10	19	15	15	19	23	12	7
Plataforma Alta	Sistema 11	5	3	3	2	4	4	4
	Sistema 12	7	5	4	9	4	2	6
	Sistema 13	3	2	3	1	0	2	2
	Sistema 14	2	0	0	1	2	3	3
	Sistema 15	1	1	0	2	0	0	1

É importante avaliar na tabela os dados dos meses de novembro e dezembro, onde ocorre um aumento considerável no volume de produção o que impacta os sistemas de duas formas: aumento na 'pressão' por manutenções mais rápidas para diminuir o impacto nos negócios e maior rigor nos testes (usuários e área de sistemas): um erro em produção neste período traz impactos muito maiores.

De forma análoga, o impacto nos erros nas emissões de notas fiscais pode ser acompanhado na Tabela 5. Historicamente o valor esteve na casa dos 2%, e apesar de refletir a maior eficácia de um único sistema, outros sistemas são envolvidos e podem contribuir de forma negativa com este indicador de qualidade, daí sua importância.

Tabela 5: Atraso na emissão de N/F (em até 2 dias)

Atraso em Emissão Notas Fiscais						
jun/04	jul/04	ago/04	set/04	out/04	Nov/o4	dez/04
2,3%	2,1%	1,6%	0,7%	0,9%	1,2%	1,1%

Homologação

O processo de homologação já era executado em alguns sistemas, mas não havia um padrão definido. A uniformização para todos os sistemas e adoção da ferramenta de versionamento como caminho único para a implantação foram as grandes mudanças no processo de homologação, que precisa contar com o aceite formal do gestor do projeto e da obrigatoriedade na realização dos testes de sistema.

Entrega

O processo de implantação não era padronizado: cada sistema era implantado a medida que surgia a necessidade de instalação de uma nova versão ou correção de algum tipo de erro.

As equipes responsáveis pelos sistemas não se falavam e era frequente que recursos comuns, como administradores de banco de dados fossem alocados de maneira desorganizada.

Foi implantado o conceito de régua de versionamento para centralizar e otimizar as implantações, sempre de acordo com as demais áreas de tecnologia, principalmente Teste e Homologação.

Ficam fora da régua de versionamento as alterações consideradas emergenciais, ou seja que impactam o negócio fim da empresa e não podem esperar pela próxima *janela*[1] de implantação (por exemplo quando um sistema deixa de funcionar).

Um dos principais ganhos com a implantação da janela de versionamento foi facilitar o papel dos integradores, declarando regras claras para todos os sub-fornecedores (aglutinados sob um mesmo integrador). Dessa forma pode-se aumentar a agilidade nas decisões e planejamento das implantações desses fornecedores reduzindo a quantidade de mudanças nos sistemas. Outro ganho, foi a partir da análise de dados objetivos das alterações, diminuição no tempo total de indisponibilidade de sistema quando das implantações, pois podia-se prever alguns problemas recorrentes e divulgar essas previsões de forma mais ágil.

O processo de entrega foi viabilizado com a utilização de uma ferramenta de controle de fluxo e aprovação das versões (ferramenta de versionamento) que também funciona como repositório de dados. Sem o uso desta ferramenta seria inviável administrar o volume de dados envolvidos.

[1] Alusão às janelas de lançamento e reentrada em órbita terrestre

Tabela 6: Exemplo de Régua de Versionamento

Data prevista da janela de implantação: 28/Ago/04 06:00 - 29/Ago/04 23:00					
Sistemas implantados dentro da janela prevista: Sistema 1, Sistema 2, Reorganização Banco Dados Sistema 3; Sistema 3, Upgrade Sistema Telefonia Filial SP					
Responsável pela Janela de Versionamento:			Nome Responsável JV		
Identificação Sistema	Sistema 1	Sistema 2	Sistema 3	Sistema 3	Telefonia *SPOT*
Identificação Baseline	Sis1 Bsl 1 V3.4	Sis2 Bsl 2 V7.1	Sis3 Bsl 3	Sis3 Bsl 4	Tel1 Bsl 5
Número problemas últimas 3 implantações	0	0	2	2	0
Afeta outra Baseline ?	NÃO	NÃO	NÃO	NÃO	NÃO
Data planejamento Implantação	30/jul/04	30/jul/04	22/ago/04	22/ago/04	30/jul/04
Responsável Negócios / Substituto	Nome Responsável	Nome Responsável	Nome Responsável	Nome Responsável	Nome Responsável
Responsável Técnico / Substituto plantão	Nome Responsável Sistema 1 (fone / cel)	Nome Responsável Sistema 2 (fone / cel)	Nome Responsável Sistema 3 (fone / cel)	Nome Responsável Sistema 3 (fone / cel)	Nome Responsável Sistema Tel (fone / cel)
Data/Hora término implantação	28/8/2004 / 15:35:00	28/8/2004 / 16:00:00	28/8/2004 / 12:45:00	28/8/2004 / 13:00:00	28/8/2004 / 08:00:00
Ordem Prevista	1	2	3	4	5
Ordem Realizada	4	5	2	3	1

Informações Operacionais

A responsabilidade de comunicação ficou a carga da equipe de informações operacionais, que gera informações relevantes e atualiza as bases de conhecimento (estatísticas).

Muitas vezes bons trabalhos simplesmente não eram divulgados adequadamente, o que, além do fator motivacional para as equipes, também não contribuia para a imagem de eficiência da área de tecnologia como um todo.

Uma equipe responsável por estas informações garantia a publicação de forma adequada das evidências de que problemas estavam sendo solucionados ou pelo menos o quão longe da situação ideal a área de tecnologia se encontrava. Mais do que uma ferramenta de *marketing* interno, a equipe de informações operacionais é responsável por divulgar os índices de qualidade acordados para os sistemas (o que nem sempre pode corresponder a informações positivas).

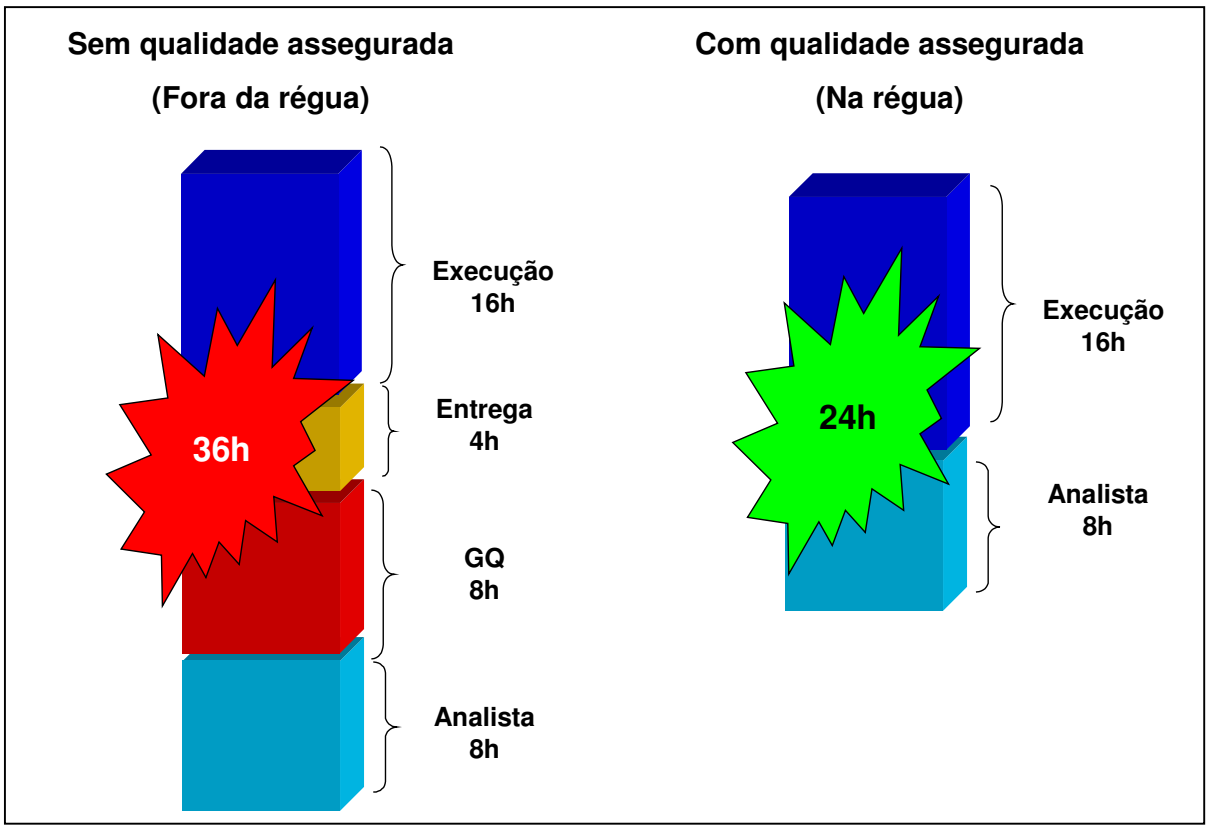

Figura 18: Exemplo de material divulgado equipe de informações operacionais

6.4 Papel das melhores práticas sugeridas pelo ITIL no Estudo de Caso

Antes da separação das equipes de projeto e tecnologia, o atendimento era feito por sistema. Cada equipe era responsável por todas as atividades (projeto, manutenção) dentro do próprio sistema, atendendo áreas funcionais (de negócio) específicas.

Existiam uma ou mais equipes para cada área de negócio, como apresentado abaixo, na Figura 19:

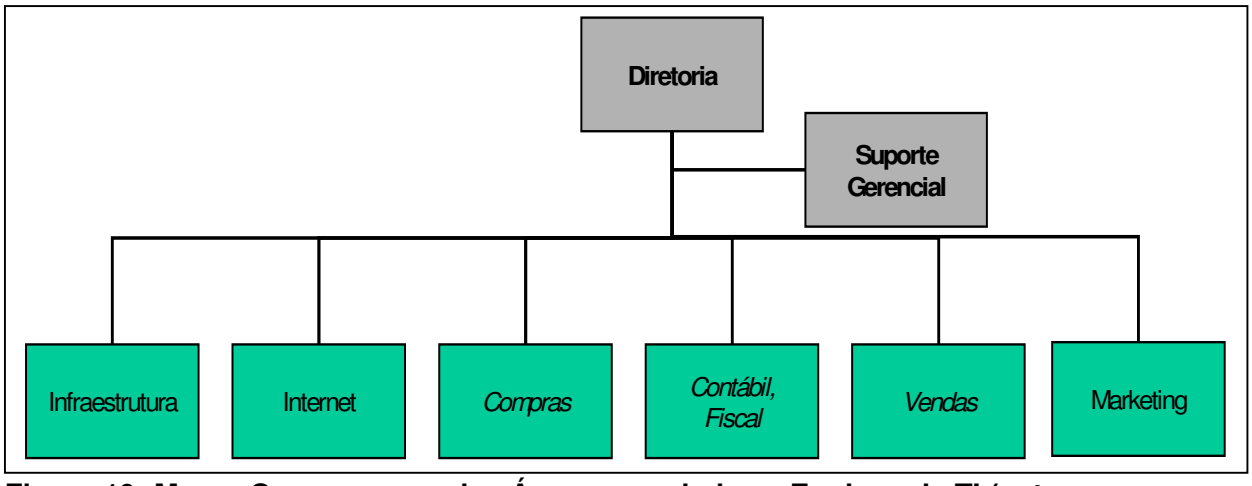

Figura 19: Macro Organograma das Áreas associadas a Equipes de TI (antes separação Projetos e Manutenção)

Com a separação das áreas de produção e manutenção, a responsabilidade pelas atividades também se dividiram:

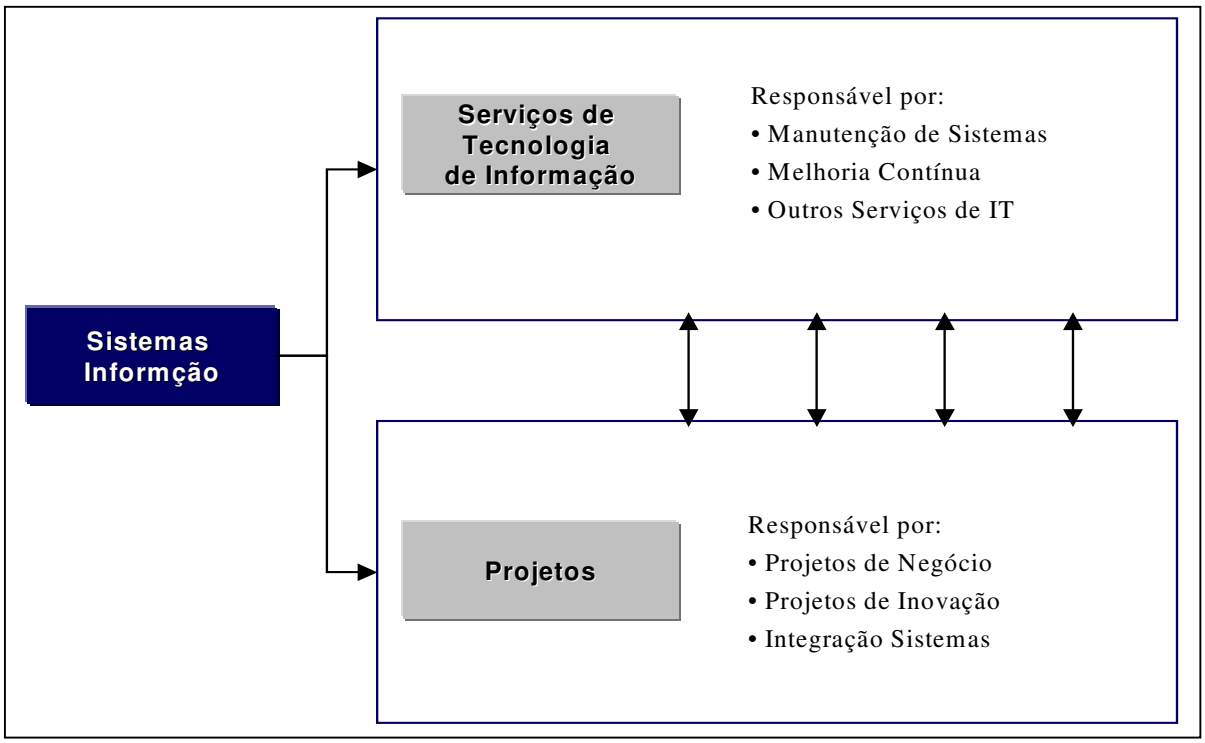

Figura 20: Responsabilidade das áreas de projeto e manutenção (após divisão Projetos e Manutenção)

6.5 Criação do Papel de Gerente de Produto

A partir do modelo ITIL apresentado anteriormente e das necessidades e dificuldades encontradas foi instituído o papel do gerente de produto, responsável pela manutenção dos projetos desenvolvidos. Os produtos tratados são ferramentas de software terceirizadas. Desta forma, fundiu-se a necessidade de gerenciamento e controle com a responsabilidade de manutenção no papel de gerente de produto, que passou a ser o principal ponto focal de cada parte da equipe de 'Construção'.

O papel do gerente de produtos é apresentado a seguir, adaptado de TEN STEP (2005)[1].

Planejamento do Produto

● Coordenar os problemas do produto e toda a comunicação com o vendedor do produto como contato preliminar.

● Monitorar a direção do produto com o vendedor, e comunicar esta informação aos *Stakeholders* da empresa.

● Determinar que componentes do produto devem ser usados na organização.

● Identificar as oportunidades para o uso do produto.

O planejamento do produto, descrito acima, deve ser acompanhado pelo gerente de produto antes da implantação efetiva, ou seja, o gerente de produto deve participar do processo de compra e seleção dos produtos que serão colocados na produção e que terão a supervisão sob sua responsabilidade. Mesmo após a implantação, o gerente de produto deve maximizar o aproveitamento deste produto por outras áreas da organização, visando o maior reaproveitamento e o menor custo, sempre mantendo ou aumentando a qualidade envolvida.

Testes

● Coordenar os Testes dos produtos novos e as novas liberações, incluir coordenação dos pilotos com os usuários potenciais do produto.

● Determinar quando um produto está pronto para produção "*production-ready*" baseado nos testes e nos projetos pilotos.

● Coordenar a certificação dos produtos novos e as novas liberações.

O gerente do produto deve agir como o filtro da área de produção, certificando-se da qualidade do que será implantado.

---

[1] **TenStep, Inc. -** O TenStep, Inc., é uma empresa de consultoria independente focada no desenvolvimento das metodologias em Gerenciamento de Projetos, fornecedora de treinamento e consultoria.

Gerenciamento Financeiro

- Coordenar as negociações dos contratos dos produtos, acordos de compras, e acordos da manutenção dos produtos.
- Assegure-se de que o orçamento esteja disponível para compras e manutenção dos produtos.
- Determinar quando começar a consideração de cancelar ou reduzir os pagamentos da manutenção baseados na direção do produto.

Implementação e Distribuição do Produto

- Coordenar o Desenvolvimento de um plano da distribuição do produto.
- Gerenciar e monitorar a distribuição do produto ou das novas liberações
- Distribuir (instalar) o produto.
- Manter o inventário do produto (onde o produto foi distribuído).
- Receber pedidos contínuos da equipe de funcionários para a distribuição individual do produto.
- Integrar e adicionar produtos novos e novas liberações na arquitetura.

Gerenciamento de Liberações do Produto

- Decidir quando promover uma nova liberação.
- Planejar e gerenciar a implementação das novas liberações.
- Assegurar que as novas liberações foram adicionadas na arquitetura se requerido.

Retirada de Circulação do Produto

- Determinar quando o produto deve ser retirado de circulação.
- Planejar e gerenciar a retirada de circulação do produto.
- Retirar (desinstalar) o produto do ambiente.
- Assegurar que o produto foi removido da arquitetura.

Além do gerente de produto, que basicamente age na área de construção, os demais membros das equipes que antes atendiam funcionalmente, foram realocados com papéis claros dentro da estrutura adaptada da IEEE-1219. Os papéis passam a ser definidos pela área onde o recurso está alocado (Teste ou Entrega, por exemplo) e não mais pela função (analista, programador, etc.).

Desta forma, de acordo com os perfis, um recurso da área de manutenção passou a ser, por exemplo, um especialista em qualidade na área de Testes, atendendo a todos os sistemas e não mais um profissional genérico, responsável por todas as ações de um sistema ou um grupo de sistemas. Não existe mais o analista ou programador de um produto ou sistema, mas sim especialistas em cada etapa do ciclo de vida de cada sistema.

Capítulo VII

7 CONCLUSÕES

7.1 Conclusões

A escolha das boas práticas sugeridas pelo padrão IEEE-1219 e pela ITIL foram uma opção interessante e se mostraram uma alternativa viável uma vez que não existiam recursos (tempo e financeiro) específicos para a reestruturação da área de manutenção de sistemas.

As simplificações e adaptações às boas práticas foram necessárias, o que gerou impactos, inclusive em outras fases que foram utilizadas de forma mais próxima ao sugerido. Foi uma opção entre o "ótimo" e o "bom", que mostrou sua eficácia, principalmente com os sistemas que não vinham recebendo tratamento algum do ponto de vista metodológico.

A necessidade de adequação das equipes, com a definição dos novos papéis, somente foi possível com a clara definição das áreas de atuação, mesmo que simplificadas a partir do padrão IEEE-1219. Da mesma forma que surgiu a necessidade, quando da apresentação do estudo de caso, da atuação de equipes de interface, ou seja da equipe que se comunica, basicamente recebendo informações, da área de projetos e da equipe que distribui as informações da área de manutenção para as demais áreas da empresa (informações operacionais).

O modelo apresentado reflete esta realidade, específica para a empresa apresentada no estudo de caso mas sem perder a generalidade podendo ser usado como guia em outras empresas. No entanto, como cada empresa traz características próprias, principalmente relacionadas à cultura corporativa, a adaptação das equipes seria a parte do trabalho que talvez tenha que sofrer maiores adaptações, enquanto que as fases do processo sugerido pelo padrão IEEE-1219 dependeriam basicamente do grau de terceirização destas empresas.

Pode-se destacar do estudo de caso:

● O controle dos projetos e das atividades de manutenção passam a ser feito de forma independente uma da outra, sendo avaliados pelas áreas de "interface" ('Monitorar e Acompanhar' e 'Informações Operacionais') entre as equipes de projeto e a área de produção, permitindo a identificação exata de "gargalos" nas equipes, sem mascarar problemas de produtividade pela execução concomitantes de atividades de projeto e manutenção, algumas vezes sendo 'cobradas' por responsáveis distintos.

● A obrigatoriedade do planejamento dos projetos para serem incorporados à produção traz disciplina ao cumprimento do processo.

● Melhora no aproveitamento dos recursos humanos onde há melhor adaptação: um bom membro da equipe de projeto pode não ter o perfil adequado para a área de manutenção e vice-versa.

7.2 Contribuição do Trabalho

Basicamente, pode-se resumir as contribuições como adaptações de boas práticas encontradas na engenharia de software a um ambiente real, construindo um modelo que pode ser usado em outras circunstânicas com algumas revisões.

A principal foi a criação de uma área responsável pela 'Construção' das manutenções. Esta área engloba as atividades de análise e projeto dos sistemas que estão passando por um processo de manutenção, com foco nos controles gerenciais destas atividades sob um modelo de *outsourcing*. Ou seja, as etapas de análise e projeto tradicionais foram aglutinadas numa fase de 'Construção' permitindo o gerenciamento das manutenções realizadas num modelo de terceirização.

A criação de equipes específicas para 'Monitorar e Acompanhar' e gerar 'Informações Operacionais' permite encapsular as atividades de manutenção e blindar os integrantes das demais equipes das influências externas. Funcionam como áreas de interface para todas as equipes de manutenção.

A equipe de 'Monitoramento e Acompanhamento' faz o filtro na entrada do processo, selecionando e priorizando os recebimento de novas solicitações de manutenção, possibilitando que estas demandas sejam tratadas de forma ordenada e documentada. Funciona com a interface entre projetos e manutenção. Nenhuma solicitação ou demanda técnica pode ocorrer sem um chamado ou registro equivalente, por mais urgente que seja. Com um ponto único de entrada das manutenções, fica viabilizado o controle e planejamento.

Já a equipe de 'Informações Operacionais' implementa de forma sistemática a coleta e divulgação de informações de forma padronizada, evitando que recursos técnicos especializados exerçam atividades que fogem do seu perfil de atuação.

Dentro das outras áreas sugeridas pelo padrão IEEE-1219, foram feitas contribuições auxiliares, relacionadas as equipes de interface e a equipe de construção. Foram contribuições associadas a interação com as outras áreas sugeridas pelo modelo.

A primeira dessas contribuições foi no estabelecimento de um documento padrão de 'Requisição de Manutenção de Software' como comunicação entre a equipe de 'Monitorar e Acompanhar' e a equipe de 'Identificação de Necessidades'. Desta forma, garantia-se que a equipe de 'Construção' recebesse sempre as demandas técnicas de forma padronizada (além de facilitar o registro e controle).

Da área de 'Teste' foram utilizados dados já coletados por esta equipe como indicadores de qualidade para a área de 'Construção'.

Para medir o índice de qualidade 'Eficiência', ao invés de utilizar-se da quantidade de compilações, como sugerido por Gil (1999) foi usado o número de Estornos de Baselines por Sistemas, fornecido pela ferramenta de versionamento.

Já a 'Eficácia' foi medida pelo atraso na emissão de N/F, uma adaptação do índice 'Quantidade de relatórios entregues com atraso'.

Dessa forma, as empresas contratadas e geridas pela área de construção puderam ser avaliadas de forma objetiva.

O grande diferencial implementado na equipe de 'Homologação' foi a implantação de uma ferramenta de Versionamento, mas não houve interação direta, como contribuição, com o trabalho realizado nesta pesquisa, apesar da implantação de uma ferramenta de versionamento ter sido fundamental para a implementação do modelo.

A principal ligação entre as equipes de 'Construção' e 'Entrega' foi a criação de uma 'Régua de Versionamento'.

Apesar do nome, a 'régua' não trata apenas as novas versões, mas todas as alterações feitas pela área de manutenção e projeto que serão implantadas na produção. Dessa forma, trata-se de um trabalho desenvolvido a 'quatro mãos', envolvendo a área de projeto e a área de manutenção de software, onde foi feita a contribuição específica.

Com a régua de versionamento, as alterações no ambiente de produção podem ser planejadas e otimizadas, além de evidenciarem problemas reincidentes permitindo a sinergia entre as equipes técnicas envolvidas nos processos de mudança.

Especificamente dentro da área de construção e Monitoramento foram realocados recursos específicos das equipes de acordo com as sugestões sugeridas pelo ITIL, adaptando papéis e responsabilidade e flexibilizando laços de hierarquia aproveitando melhor os recursos destas equipes.

As contribuições para a adequação dos recursos das equipes na área de manutenção de sistemas foram mais difíceis devido a regras de cargos e salários vigentes na empresa, e foram implementas apenas parcialmente. A exceção foi a equipe de 'Informações Operacionais' que teve todos os recursos realocados de forma a se adaptarem ao perfil da equipe.

7.3 Sugestões de continuidade do trabalho

Devido a limitação do escopo do estudo de caso e aos recursos disponíveis, este trabalho foi focado em algumas da melhores práticas encontradas sugeridas pela Engenharia de Software.

Um caminho natural para continuidade dos trabalhos seria o aproveitamento de outras práticas a situações reais, com estudos e análises estatísticas direcionadas.

Como encontrado no artigo “Best Practices ganham força na gestão de SLM/SLA” Intermanagers (2005) pode-se constatar que O ITIL e o COBIT são documentações completas e consistentes das “melhores práticas” para o gerenciamento de serviços de Tecnologia de Informação. Utilizar esses modelos de referência permite que as organizações tenham o controle necessário para absorver, de maneira mais rápida e menos traumática, a dinâmica própria de ambientes de tecnologia de alta criticidade:

> “Os modelos de referência ITIL e COBIT ganham força a cada dia na gestão de SLA (Service Level Agreement - acordo de nível de serviço). Isso acontece porque há uma grande necessidade de gestão e integração de processos. Os executivos necessitam falar a 'mesma língua' ao estabelecerem as negociações e essas metodologias têm se mostrado muito eficientes nesse processo. Diante desse cenário traçado, os executivos realmente voltam suas preocupações para o SLA e os modelos de referência utilizados nesse processo, como o ITIL e o COBIT, que ajudam no relacionamento da TI com seus clientes internos e externos.”

Das melhores práticas sugeridas pelo ITIL, foi aproveitado neste trabalho apenas a sugestão de estruturação das equipes de tecnologia e informação, sendo fonte importante para trabalhos posteriores, assim como as melhores práticas sugeridas pelo COBIT e CMM (Capability Maturity Model).

BIBLIOGRAFIA

BOEHM. B;. **Software Engineering - R&D Trends and Defense Needs**, in Research Directions in Software Evaluations, Acadmic Press, 1972

CHRISTENSEN, M.J.; THAYER, R.H. **The Project Manager´s Guide to Software Engineerings´s Best Practices – IEEE Computer Society,** Los Alamitos, California, 2001

COBIT - **Control Objetives for IT and Related Technology** - site: www.itgi.org, mantido pelo IT Governance Institute, acessado em Jun/2005.

DIAS, Márcio Greyck Batista; **Uma experiência no ensino de manutenção de software** (Artigo) - Faculdade Alves Faria, Centro Universitário de Goiás, 2004

FERNANDES, Daniel B. Metodologia Dinâmica para desenvolvimento de Sistemas Versáteis, Editora Érica 1ª Edição, 1999.

HELDMAN, Kim. **Gerência de Projetos – Guia Oficial para o exame oficial do PMI**, Editora Campus, SP, 2003

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**, Editora Atlas 4ª. edição, 2002.

GIL, Antonio Loureiro. **Qualidade Total em Informática**, Editora Atlas 3ª edição, 1999

Info Corporate, Nro 19, Abril 2005, Editora Abril - Artigo: Outsourcing Latino, página 18

InformationTecnology Infrastructure Library. **ITIL Organization Structure;** versão 2.0 de Julho de 2002.

Institute of Electrical and Electronic Engineers. **Std 1219-1998:** Software Maintenance, 1998.

_____ **. Std 830-1998:** Recommended Practice for Software Requirements Specifications, 1998.

INTERMANAGERS, site: Intermanagers.com.br – artigo publicado em 16/03/05, acessado em Abril/2005.

PRESSMAN, Roger S. **Engenharia de Software**, Makron Books 3ª Edição, 1996

PMBOK, **A Guide to the Project Management Body of Knowledge**, 4ª. Edição

TORVALDS, Linus ; DIAMOND, D. **Só por prazer – Linux os bastidores da sua criação,** Editora Campus, 2001

WEINBERG, Gerald M.; **Redefinindo a Análise e o Projeto de Sistemas**, McGrawHill, São Paulo, 1990

SWANSON, E.B., **The Dimensions of Maintenance**, Proc. 2nd Intl. Conf. Software Engineering, IEEE, 1976

SIMCSIK, Tibor e Polloni, Enrico; **Código GQPS^2**, Prosoft Tecnologia 1a. Edição, São Paulo, 2006

SWEBOK - **Guide to the Software Engineering Body of Knowledge IEEE – 2004 Version** - Versão no site http://computer.org

TENSTEP, **Boletim Informativo Brasil,** site www.tenstep.com.br acessado em Abril/2005

The ITIL and ITSM Directory - site http://www.itil-itsm-world.com, acessado em Set/2004

VIEIRA, Sonia. **Como Escrever uma Tese**, Editora Pioneira Thomson Learning 5ª. edição, 2002.

Referências Consultadas (Bibliografia Auxiliar)

ANQUETIL, Káthia Marçal et al. **Aplicação da Análise Postmortem no Processo de Manutenção de Software**; Universidade Católica de Brasília; EQPS - Brasília, Dezembro 2003.

BOGHI, C; Shitsuka, R. **Sistemas de Informação - Um enfoque dinâmico**, Editora Érica

CHRISTOPH, Roberto de Holanda, **Engenharia de software para software livre**. Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2004.

CRUZ, Tadeu **Sistemas de Informações Gerenciais**, Editora Atlas 2ª edição, 2000

Microsoft Solutions Framework, site www.microsoft.com/technet/itsolutions/msf acessado em Março/2005

PIGOSKI , Thomas M.; Croll , Paul R. - **IEEE Computer Society** - Software Maintenance, Montreal, May 2003.

Project Manager Network Magazine, edição de Janeiro 2005

ROSINI, A.M; Palmisano, A. **Sistemas de Informação e a gestão do Conhecimento**, 1ª Edição,Editora Pioneira Thomson, 2003.

YORDON, E. **Declínio e Queda dos Analistas e dos Programadores**, Makron Books, 1995

ANEXOS

ANEXO A

DEFINIÇÕES E ACRÔNIMOS DA NORMA IEEE-1219 1998[1]

[1] Acrônimos retirados do site http://www.redetronic.com.br/glossario_i.asp em 13/03/2005 - IEEE (Institute of Electrical and Electronic Engineers) é uma sociedade profissional internacional que edita seus próprios padrões e é membro participante da ANSI e ISO

Definições e Acrônimos do Padrão IEEE-1219 1998

As definições e acrônimos usados foram retirados do padrão IEEE-1219 e são aqui apresentadas como facilitadores para a leitura deste texto. As definições aqui apresentadas devem ser entendidas no contexto do padrão IEEE-1219. São definições conceituais utilizadas para argumentar a compreensão das atividades de manutenção de software descritas no padrão.

Manutenção adaptativa:

Modificação de produto de software executada depois da entrega da versão inicial para manter um programa de computador utilizável num ambiente que sofreu ou está sofrendo alterações.

Manutenção corretiva:

Modificação reativa num produto de software executada após a entrega das correções dos erros encontrados.

Cliente:

Pessoa ou pessoas para as quais o produto foi produzido, as quais usualmente (mas não necessariamente) decidem os requisitos de uma manutenção.

Manutenção emergencial:

Correção não agendada executada para manter o sistema operacional.

Teste de interoperabilidade:

Teste conduzido para garantir que um sistema modificado mantém a capacidade de trocar informações com sistemas de diferentes tipos e continuar usando estas informações.

Solicitação de Manutenção (*Modification Request – MR*):

Um termo genérico que inclui os formulários associados aos relatórios dos problemas informados (por exemplo relatório de incidentes e relatório de anomalias) e o controle dos documentos de configuração (por exemplo a solicitação de manutenção de software – SCR – *Software Configuration Request*).

Manutenção perfectiva:

Modificação de um produto de software após a entrega para melhorar a performance ou capacidade de manutenção.

Projeto:

Um subsistema que é objeto da atividade de manutenção[1]

---

[1] A diferenciação entre Projeto e Manutenção é feita ao longo do texto. A definição aqui apresentada faz sentido no contexto de um trabalho realizado na manutenção de software.

Teste de regressão:

Testar novamente para detectar falhas introduzidas por modificações.

Repositório:

(A) Uma coleção de todos os artefatos relacionados ao software (por exemplo, o ambiente de engenharia de software) que pertence ao sistema. (B) A localização/formato na qual tal coleção está armazenada.

Engenharia Reversa:

Processo de extração de informações sobre sistemas de software (incluindo documentação) a partir do código fonte.

Manutenção de software:

Modificação de um produto de software que ocorre após a entrega para correção de falhas, melhora dos seus atributos ou para adaptar este produto a um ambiente modificado.

Sistema:

Um conjunto de unidades interligadas organizadas para realizar com sucesso uma ou mais funções específicas.

Usuário:

Pessoa ou pessoas que operam ou interagem diretamente com o sistema.

ANEXO B

DEFINIÇÕES DAS ATRIBUIÇÕES DE UM GERENTE DE PRODUTO

As definições das atribuições do cargo 'Gerente de Produto' são detalhadas no formato de quadro, conforme apresentado no capítulo sexto.

Ação	Fase Execução	Implementação	Objetivo	Não é atribuição do gerente de produto
Coordenar problemas produto e comunicação com vendedor	Planejamento do produto	Contato preliminar com vendedor do produto, indicando os problemas e qual a situação ideal de solução	Existência de um ponto focal único para atuação nos problemas	Indicar a solução técnica detalhada ao vendedor - responsabilidade da equipe técnica do vendedor
Monitorar a direção do produto		Acompanhando no mercado e com concorrentes a evolução do produto	Manter o stakeholder atualizado sobre os rumos do produto (novas versões, necessidades de atualização)	Conhecer os detalhes de projeto e planejamento estratégico envolvidos na troca/aquisição de produtos - responsabilidade do gerente de projetos
Determinar que componentes do produto serão usados		Avaliando o custo benefício da implantação de todos os módulos diante do parque de soluções já instaladas	Evitando conflitos entre produtos já existentes minimizando a redundância (principalmente de dados cadastrais) e maximizando a comunicação entre os produtos	Responder pelos outros produtos já implantados e pela infra estrutura para a comunicação destes produtos. Responsabilidade de cada gerente de produto (pode ser de outra área, inclusive de negócio) e das equipes técnicas (internas ou externas)
Identificar as oportunidades para o uso do produto		Observando em outras áreas necessidades semelhantes que possam ser atendidas pelo produto em questão.	Oferer solução para outras áreas, diluindo o custo da licença de uso do produto	Planejamento estratégico futuro - responsabilidade dos gerentes executivos
Coordenação testes	Testes	Exigência de um plano de teste atualizado para a implantação do produto ou de nova versão	Garantir a qualidade do produto a ser implantado evitando problemas no ambiente de produção.	Realização dos testes. Os testes devem ser feitos pelos desenvolvedores, pelos usuários e pela equipe técnica interna (integração)
Liberar para produção um novo produto ou versão		Aprovar (ferramenta de versionamento ou por e-mail) a implantação	Oficialização do momento da implantação.	Implementar a implantação ou coordenar equipes técnicas da implantação. Responsabilidade do gerente de implantação
Coordenar as certificações dos produtos		Seguindo diretivas da organização (plano de obtenção CMM, normas internas)	Aderência aos padrões internacionais de qualidade e/ou da Matriz-Filiais Regionais	Montar os planos de certificação - responsabilidade da área de recursos humanos
Coordenar as negociações dos contratos dos produtos	Gerenciamento Financeiro	Participando das negociações de aquisição, acordos de compras e manutenção	Redução custo de manutenção da base implantada	Distribuição do orçamento para a área de manutenção; responsabilidade do conselho administrativo
Gerenciamento orçamento		Controlando as entradas e saídas do centro de custo (ou centro de responsabilidade) específico, via autorizações	Garantir o cumprimento do orçamento anual e controle dos gastos com manutenções fora dos contratos (imprevistos como enchentes, panes elétricas, incêndios)	Obtenção de recursos para manutenções não previstas; responsabilidade conselho administrativo.
Controle dos pagamentos das manutenções		Autorizações e planejamento da continuidade no uso dos produtos	Garantir aderência à estratégia de manutenção definida e a transição segura entre fornecedores	Definir a estratégia de manutenção; responsabilidade do diretor TI (plano diretor).

Ação	Fase Execução	Implementação	Objetivo	Não é atribuição do gerente de produto
Coordenar o Desenvolvimento de um plano de distribuição do produto	Implementação e Distribuição do Produto	Organizando frentes de trabalho associadas aos usuários e equipes técnicas, integrando prazos e recursos	Atender de forma técnica e política as necessidades de priorização da implantação de novas distribuições	Definição das necessidades políticas; responsabilidade do diretor de TI.
Gerenciar e monitorar a distribuição do produto		Controlando as *baselines, via aprovação*, da ferramenta de versionamento.	Controlar se o plano de distribuição está sendo cumprindo a contento	Renegociar o plano de distribuição do produto; responsabilidade do diretor de TI.
Distribuir (instalar) o produto		Garantindo a infraestrutura necessária (por meio de outras equipes de trabalho) para a instação de acordo com o plano de distribuição	Disponibilizar o produto ao usuário final	Instalação física dos produtos; responsabilidade das equipes técnicas de infra (internas ou externas).
Manter o inventário do produto		Controlando as licenças	Mapear onde o produto foi distribuido	Contabilizar as licenças; responsabilidade do depto contábil.
Tratar pedidos para a distribuição individual do produto		Garantindo que todos as instalações fora do plano de distribuição sejam efetivadas	Garantir aderência à estratégia de manutenção definida	Instalação física dos produtos; responsabilidade das equipes técnicas de infra (internas ou externas).
Integrar e adicionar produtos novos e novas liberações na arquitetura		Garantindo a integração das equipes técnicas das diversas soluções usadas, e a intermediação entre o arquiteto de software	Não gerar incompatilbilidades tecnológicas que possam gerar custos elevados de manutenção	Questionar a qualidade das ferramentas usadas em cada pacote externo. Responsabilidade dos fornecedores.
Determinar quando o produto deve ser retirado de circulação.	Retirada de Circulação do Produto	Indicar à área de Projetos que será necessário substituir um determinado produto (prazo, razões), via um ´plano de substituição.	Garantir a transição segura entre fornecedores	Disponibilizar verba para a transição; responsabilidade conjunta das áreas de TI e negócio.
Planejar e gerenciar a retirada de circulação do produto		Montagem do plano de substituição	Garantir a transição segura entre fornecedores	Desistação física do produto; responsabilidade das equipes técnicas.
Retirar (desinstalar) o produto do ambiente		Controlando as licenças	Manter a equivalência entre licenças e a base instalada	Desistação física do produto; responsabilidade das equipes técnicas.
Assegurar que o produto foi removido da arquitetura		Garantindo a atualização da documentação da arquitetura de sistemas, com o envolvimento do arquiteto de software.	Manter a compatibilidade tecnológica da base instalada	Redefinir as políticas de arquitetura de TI da empresa; responsabilidade do arquiteto de software.

Os quadros foram divididos em duas partes apenas por questões de apresentação.