SIMONIA HELENA DE ANDRADE

PROPOSTA DE PROCESSO DE MANUTENÇÃO

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Engenharia de Software.

Área de Concentração:

Engenharia de Software

Orientadora:

Profª. Dra. Jussara Pimenta Matos.

São Paulo

2005

A Deus e a minha família pelo apoio e incentivo.

OK

AGRADECIMENTOS

À Professora Jussara Pimenta Matos, orientadora deste trabalho, pela paciência e dedicação.

Às colegas Fernanda Moreira Sena Gomes e Maria Rosemere Degan Melchert, que me auxiliaram com críticas e sugestões no decorrer deste trabalho.

Aos meus amigos e familiares.

A todos que colaboraram direta ou indiretamente, para a execução deste trabalho.

RESUMO

Este trabalho apresenta uma proposta para um processo de manutenção de sistema de software, onde são descritos as atividades, os responsáveis e os artefatos gerados em cada uma de suas fases. O desenvolvimento de software vem nos últimos anos passando por processo de melhoria de qualidade, com isto as empresas estão alcançando o almejado controle de custo e prazo. Porém, o mesmo não ocorre na manutenção de um sistema de software, pois suas atividades não são claras e com isto não é possível a implementação de um processo de qualidade. Portanto, esta proposta tem como objetivo a obtenção de um melhor controle do gerenciamento dessas atividades e o estabelecimento de melhoria da qualidade.

ABSTRACT

This work presents a proposal of a maintenance process, where they are described: the activities, responsible and the devices generated in each phase. The software development comes in the last years passing for process of quality improvement, with this the companies are reaching the longed for control of cost and stated period. The same it does not occur how much to the process of maintenance of a software system, therefore its activities are not clear and with this the implementation of a quality process is not possible. Therefore, this work has the objective of the attainment of one better control of the management of these activities and establishment of improvement of the quality.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ALI – Arquivo lógico interno.

AIE – Arquivo e interface externa.

CE - Consulta externa.

EE – Entrada externa

EUP – Entreprise Unified Process.

IEEE – Institute of Electrical and Electronics Engineers.

IFPUG - International Function Point Group

ISO – International Organization for Standardization.

LOC – Lines of Code.

SE – Saída externa.

RUP – Rational Unified Process

1. INTRODUÇÃO

1.1 Objetivo do Trabalho

O objetivo deste trabalho é propor um Processo de Manutenção, tendo como base os processos de desenvolvimento de software e as atividades apresentadas na norma NBR-ISO/IEC-12207 (1998). Esta proposta visa proporcionar melhorias no gerenciamento das atividades referente à manutenção, desta forma, é possível estabelecer um melhor controle sobre a execução dessas atividades. Além disso, também é apresentada a definição dos artefatos que devem ser gerados e atualizados, com isto obterem subsídios para a melhoria do produto a ser entregue em prazo mais adequado.

1.2 Motivação do Trabalho

A necessidade de desenvolver software com qualidade e em prazos e custos menores, levou alguns estudiosos e profissionais a elaborar modelos de processos para desenvolvimento de software. Estes modelos auxiliam na padronização das atividades de construção e no controle da execução de um sistema de software, auxiliando no gerenciamento de custo e prazo do projeto.

Após sua construção e validação, o sistema de software é instalado no ambiente operacional e, a partir de então qualquer necessidade de alteração seja para correção ou para implementação de novas funcionalidades, é tratada como manutenção. As

atividades que devem ser executadas para o atendimento da alteração, não são as mesmas do seu desenvolvimento, pois não se está construindo um novo produto e sim alterando um produto que já se encontra em operação. Estas atividades compõem o Processo de Manutenção. Porém, existem poucos estudos sob as atividades necessárias em um processo de manutenção, as normalmente identificadas são adaptações do processo de desenvolvimento.

A falta de controle pela gerência e a ausência de um padrão definido para as atividades a serem executadas, geram inúmeros desgastes entre a equipe técnica, como conseqüência, os prazos e os custos se tornam elevados, assim como, os produtos gerados, normalmente, são de baixa qualidade. Porém, como os propósitos dos processos são diferentes, conseqüentemente, suas atividades e os artefatos gerados não possuem as mesmas finalidades, com isto o processo adaptado não atende as necessidades esperadas, como uma dessas necessidades, pode-se citar o controle do custo. De acordo com Glass (2004), o gasto com a manutenção de software, corresponde entre 40% a 80% do custo de sua construção.

Esses dados foram alguns dos fatores que motivaram a apresentação de uma proposta de um Processo de Manutenção, por meio da definição de uma padronização das atividades, visando permitir um controle gerencial, de forma a alcançar melhorias tanto nos prazos e quanto nos custos.

1.3 Estrutura do Trabalho

A monografia está estruturada em 6 capítulos conforme apresentação a seguir.

O Capítulo 1 apresenta o tema a ser tratado, o objetivo do trabalho, a motivação para escolha do tema e a estrutura dos capítulos da monografia.

O Capitulo 2 apresenta as atividades referentes à manutenção, utilizando como base a norma NBR-ISO/IEC-12207 (1998); que estabelece um conjunto de atividades tanto para o desenvolvimento quanto para a manutenção de software, entre outras. Além disso, também são apresentados os modelos de desenvolvimento que influenciaram no embasamento da proposta e os conceitos de manutenção.

O Capitulo 3 apresenta a importância da atividade de medição de software tanto no processo de desenvolvimento como na manutenção, descrevendo das métricas passíveis de serem utilizadas na medição das alterações.

O Capitulo 4 apresenta a Proposta de Processo de Manutenção, as atividades relativas a cada uma das fases definidas, os artefatos que devem ser gerados ou atualizados e os papéis das pessoas envolvidas.

O Capitulo 5 apresenta a estrutura atual da empresa selecionada para o estudo, quais são as adequações necessárias para a implementação do processo proposto e os resultados obtidos em uma equipe piloto.

O Capitulo 6 apresenta a conclusão, descrevendo os resultados esperados com a implementação, os pontos positivos da proposta e os que merecem melhorias para futuros estudos.

2. MANUTENÇÃO E PROCESSO DE SOFTWARE

2.1 Considerações Iniciais.

O objetivo deste capitulo é apresentar os tipos de manutenção de um sistema de software. Primeiramente é apresentada a Norma NBR-ISO/IEC-12207 (1998) tendo como foco o Processo Fundamental de Manutenção. Em seguida, são apresentados os processos de desenvolvimento adaptados para a manutenção e finalmente, são detalhados os tipos de manutenção e os principais motivos identificados que elevam o custo de sua execução.

2.2–Norma NBR-ISO/IEC 12207:1998 – Processos de ciclo de vida de software

Para estabelecer uma padronização em relação à definição das atividades e das tarefas a serem aplicadas durante aquisição, desenvolvimento ou prestação de serviço de software a *International Organization for Standardization* (ISO) definiu a Norma NBR-ISO/IEC-12207 (1998) – Processos de ciclo de vida de software.

A norma destaca três conjuntos de processos como principais que, consistem dos Processos Fundamentais, Processos de Apoio e Processos Organizacionais, como mostra a figura 2.1, sendo que cada um é composto por seus processos específicos.

O Processo de Manutenção é considerado como um dos Processos Fundamentais.

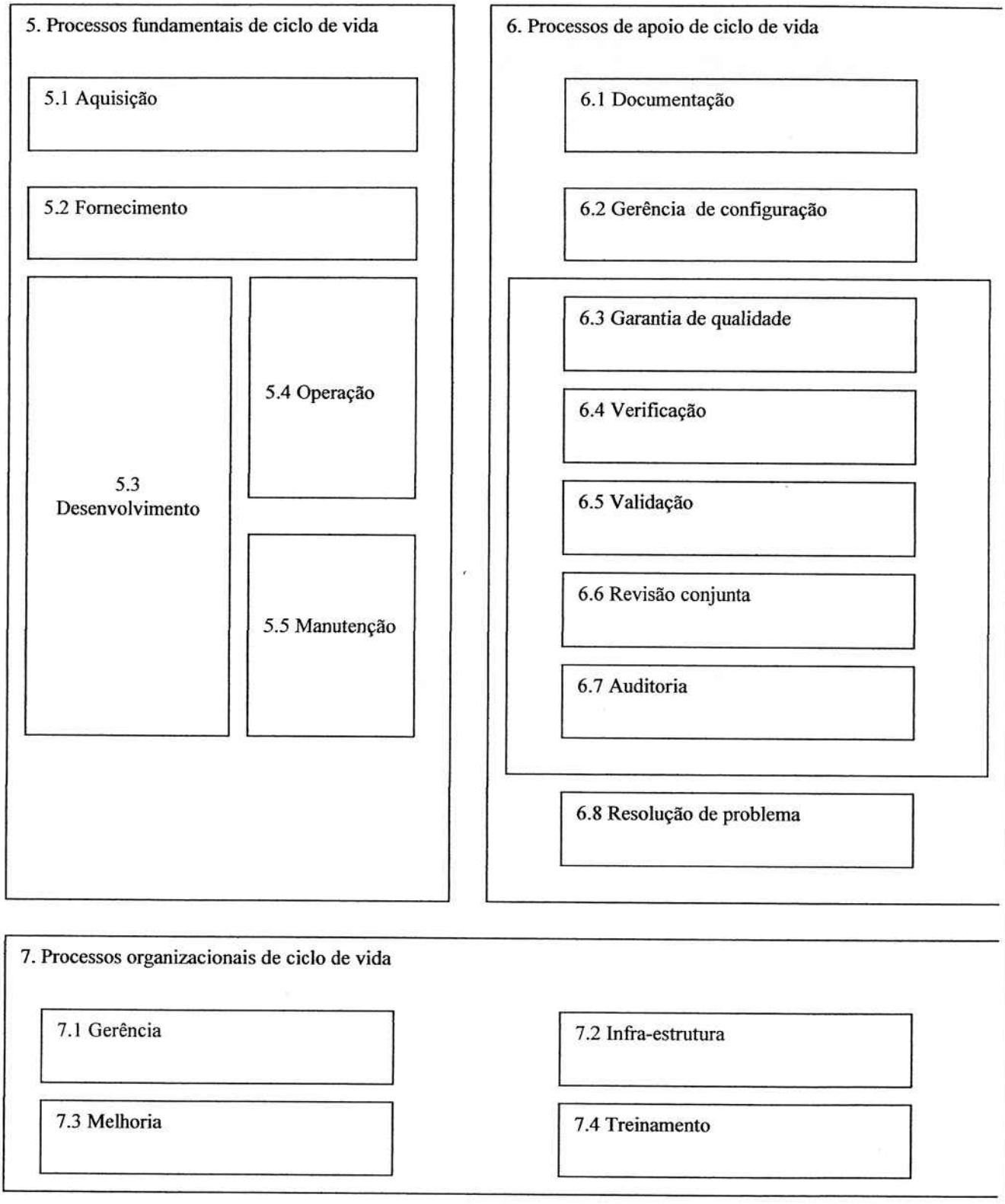


Figura 2.1 - Estrutura da Norma NBR ISO/ IEC 12207 (1998)

2.2.1 Processos Fundamentais

Os Processos Fundamentais são compostos de cinco processos que abrange a aquisição, o fornecimento, o desenvolvimento, a operação e a manutenção. A seguir são apresentados, sucintamente, cada um desses processos:

- **Processo de aquisição:** define as atividades do cliente (comprador) de um produto de software ou serviço.

- **Processo de fornecimento:** define as atividades do fornecedor, quem provê o sistema, produto ou serviço do software.

- **Processo de desenvolvimento:** define as atividades que devem ser seguidas pelos desenvolvedores de software. Estas atividades são baseadas nos diversos modelos de processo de desenvolvimento, onde foram estabelecidas às principais atividades desta fase.

- **Processo de operação:** define as atividades que a empresa, responsável pela operacionalização do software, deverá executar e controlar.

- **Processo de manutenção:** define as atividades que devem ser executadas pelas empresas responsáveis em manter o software, sendo que o principal é manter a documentação atualizada.

2.2.2 Processos de Apoio

Os Processos de Apoio ao Ciclo de Vida são compostos de oito processos, que auxiliam na execução dos processos fundamentais, para garantir a qualidade e o controle do projeto de software. A seguir são apresentados, sucintamente, cada um desses processos:

- **Processo de documentação:** define as atividades para o registro da informação produzida nos processos fundamentais.

- **Processo de Gerência de Configuração:** define as atividades da gerência de configuração, para garantir a estabilidade do produto.

- **Processo de Garantia de Qualidade:** define as atividades que garantem que o produto e processo de software estejam em conformidade com os planos e requisitos especificados.

- **Processo de Verificação:** define as atividades para avaliação do produto, quanto ao atendimento de seus requisitos básicos.

- **Processo de Validação:** define as atividades para a validação do produto junto ao cliente.

- **Processo de Revisão:** define as atividades para avaliação da situação e dos produtos gerados nas atividades dos processos fundamentais.

- **Processo de Auditoria:** define as atividades para determinar a conformidade com os requisitos, planos e contratos.

- **Processo de Resolução de Problemas:** define as atividades para a análise e remoção dos problemas de qualquer natureza ou origem, descoberto durante as atividades dos processos de desenvolvimento, operação e manutenção.

2.2.3 Processos Organizacionais

Os Processos Organizacionais do ciclo de vida são compostos de quatro processos que são empregados para estabelecer e implementar uma estrutura subjacente, constituída dos processos de ciclo de vida e do pessoal associado. A seguir são apresentados sucintamente, cada um desses processos:

- **Processo de gerência:** define as atividades básicas da gerência.
- **Processo de infra-estrutura:** define as atividades básicas para estabelecimento das estruturas de apoio.
- **Processo de melhoria:** define as atividades básicas para executar o controle e melhoria dos processos de ciclo de vida.
- **Processo de treinamento:** define as atividades para prover o pessoal adequadamente treinado.

Os processos organizacionais e de apoio auxiliam no controle e na melhoria dos processos fundamentais de ciclo de vida.

2.2.4 Processo de Manutenção

Um processo de manutenção é um dos processos fundamentais do ciclo de vida, suas atividades determinam a padronização da execução das tarefas que são efetuadas no decorrer de uma alteração no software. A seguir são descritas as atividades a serem exercidas no processo de manutenção:

- **Implementação do Processo**: esta atividade visa documentar e registrar as ocorrências de alterações e ou implementações que o produto venha a sofrer. Para isto é necessário estabelecer alguns procedimentos de documentação ou a utilização de ferramentas que auxiliem nesta tarefa.

- **Análise do Problema e da Modificação**: esta atividade visa avaliar os impactos que a correção ou implementação venha ocasionar no produto. Para esta avaliação é sugerido que seja aplicada métrica de mensuração, para dimensionar o tamanho da mudança solicitada.

- **Implementação da Modificação**: esta atividade visa a implementação física das mudanças solicitadas. Deve ser efetuadas análise e correção e/ou implementação no produto e em seus artefatos. A validação das alterações realizada no produto deve ser executada detalhadamente, para garantir a integridade do produto que entrará em operação.

- **Revisão e Aceitação da Manutenção**: esta atividade visa a obtenção da revisão pelo cliente das alterações efetuadas e assim obter a autorização para a implementação do novo produto em operação

- **Migração**: esta atividade visa executar a migração do produto alterado para o ambiente operacional. É fundamental nesta atividade a garantia da configuração do produto antigo com o novo, para não perder a integridade.

- **Descontinuidade do Software**: esta atividade visa a desativar o software, operacionalmente a pedido do proprietário, necessitando de toda a documentação associada a esta atividade.

A norma ISO/IEC-12207 (1998) estabelece o conjunto de atividades para cada um dos processos apresentados, cabendo ao desenvolvedor sua adaptação.

2.3 Processos de Software

A norma ISO/IEC-12207 (1998) define que processo é um conjunto de atividades inter-relacionadas, que transforma entradas em saídas. Pressman (2003) define como "um roteiro que ajuda a criar a tempo um resultado de alta qualidade".

Portanto, o estabelecimento de um processo padrão é importante, pois fornece estabilidade e controle para o desenrolar de uma atividade, que sem este controle pode-se tornar caótica (SOMERVILLE, 2004).

Na engenharia de software existem diversos modelos de processo de desenvolvimento, alguns são mais conhecidos e utilizados pelos profissionais da área, pois apresenta um roteiro fácil e mais adaptável nas organizações. A seguir são apresentados três destes modelos:

a) Modelo Cascata: este é um dos primeiros processos descrito pela Engenharia de Software. Proposto por Royce em 1970 (PFLEEGER;FRANKLIN, 2004), ele é definido como uma seqüência de atividades conforme mostra a figura 2.2, que devem ser executadas uma após a outra, sendo que a próxima atividade não pode iniciar sem

que a anterior tenha sido concluída. Este tipo de comportamento ressalta a qualidade de um modelo rígido e linear, no sentido que somente se inicia uma fase ao termino da anterior, sem a possibilidade de procedimento de ajustes nas atividades efetuadas nas fases anteriores.

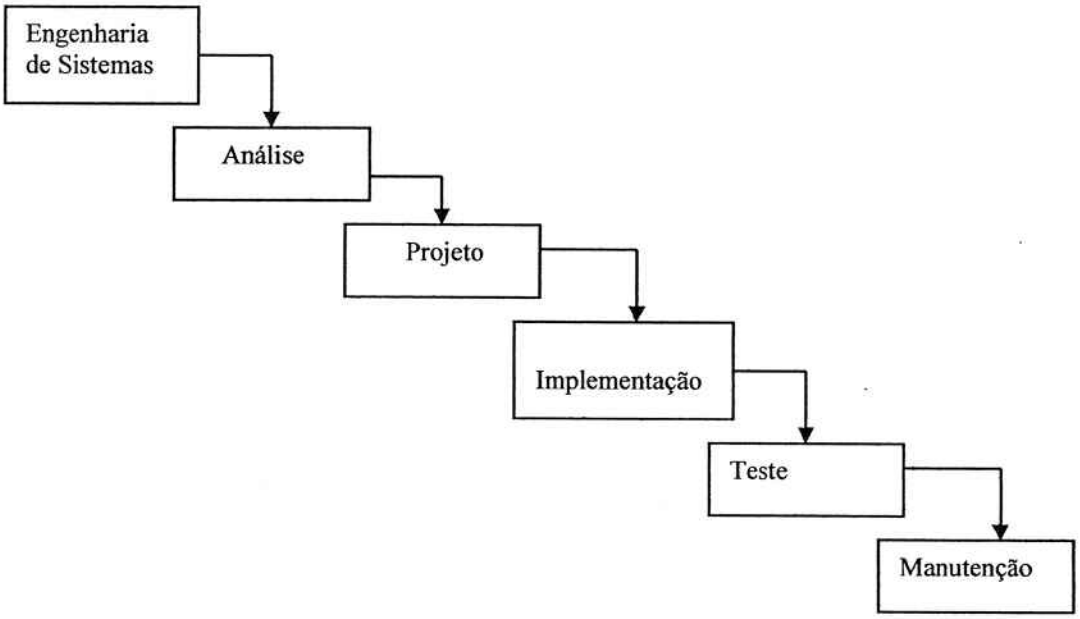


Figura 2.2 - Modelo Cascata (PRESSMAN, 2003).

Este modelo é muito utilizado até hoje, por possuir atividades bem claras e específicas que normalmente são seguidas nos roteiros de desenvolvimento, mas devida a sua estrutura, recebe muitas críticas, que fizeram seus mais ativos defensores questionarem sua aplicabilidade (PRESMMAN, 2003). Alguns dos pontos apontados são:

- projetos reais não seguem o fluxo seqüencial que o modelo propõe, iterações sempre ocorrem e isto traz problemas na aplicação do modelo.

- raramente o cliente declara todas as informações explicitamente no início do projeto, neste modelo as adequações em fases mais adiantas geram um custo e um retrabalho muito grande.

Mesmo assim, suas fases são utilizadas como base para outros modelos que surgiram posteriormente. Conforme apresentado na figura 2.2, a manutenção é tratada neste modelo como a última fase do ciclo de vida do software.

b) **Modelo Espiral:** este modelo recomenda que todas as fases descritas no modelo cascata sejam executadas diversas vezes ao longo do projeto, produzindo ciclo que se repetem ao longo de todo o desenvolvimento. Cada ciclo, que compreende desde a identificação de requisitos até implementação, recebe o nome de iteração, conforme representado na figura 2.3. No desenvolvimento iterativo, o software cresce a cada iteração, isto é, o resultado de cada iteração é um software pronto, testado e aprovado, sendo que a primeira contém poucas funcionalidades, enquanto a última contém todas as funcionalidades do sistema (TELES, 2004).

Neste modelo não existe a referência a fase de manutenção, mas seu conceito de iteração e a aplicação da análise de risco são os fatores mais importantes que este modelo contribui.

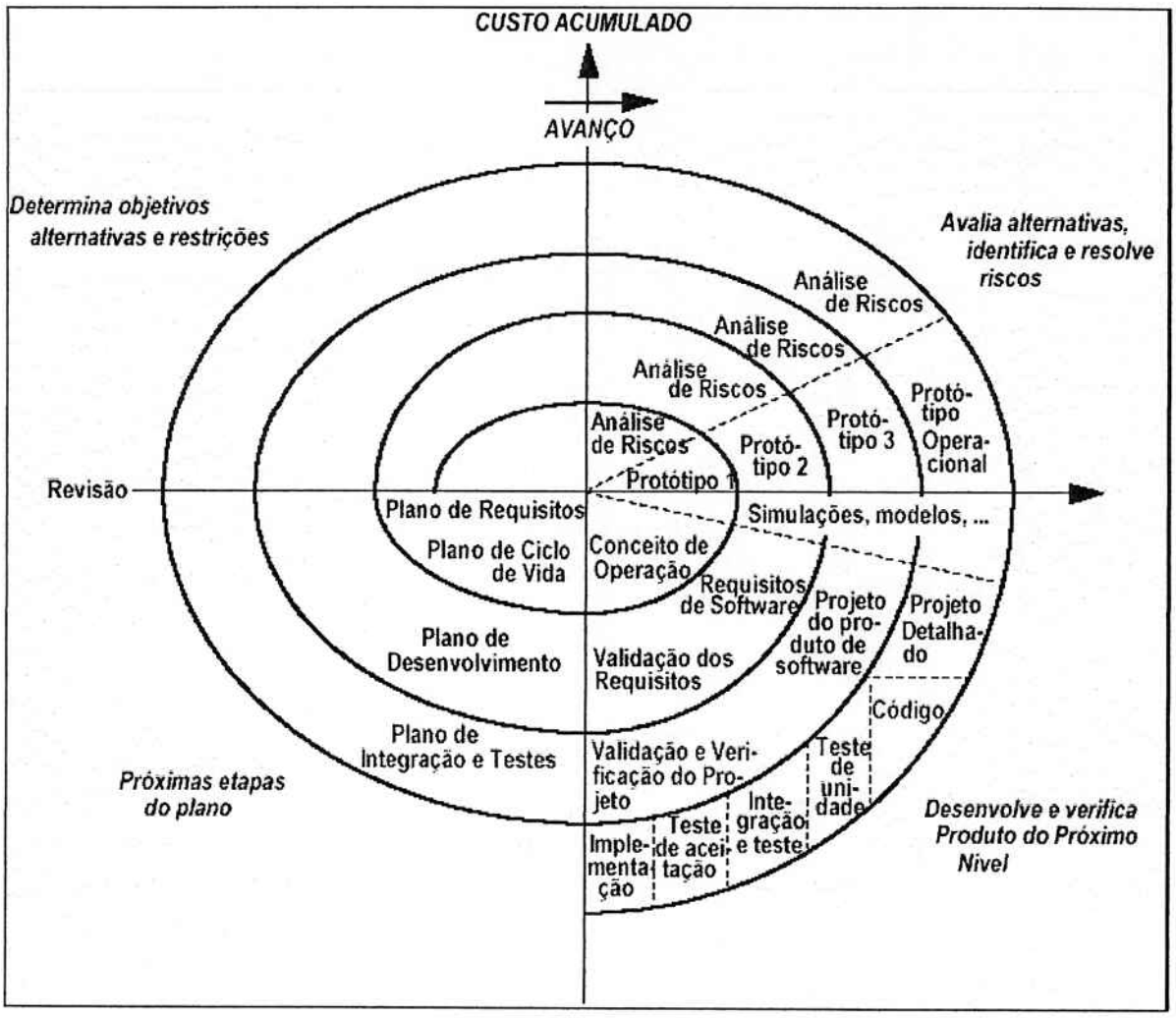


Figura 2.3 - Modelo Espiral (JALOTE, 1997).

c) Processo Unificado: este processo é apresentado em Jacobson, Booch e Rumbaugh (1999), tem como objetivo a melhora do gerenciamento através de um processo que possa ser configurável para diversos tipos de projeto. O processo está baseado em de três conceitos básicos que são apresentados a seguir:

- **Baseado em casos de uso**: os casos de uso representam as interações entre o sistema e seus usuários. Os usuários podem ser pessoas ou outros sistemas. Cada caso de uso representa uma parte da funcionalidade do sistema, e o conjunto de

todos os casos de uso deve descrever por completo toda a funcionalidade do sistema.

- **Centrado em arquitetura:** o conceito de arquitetura de software engloba os aspectos dinâmicos e estáticos mais significativos do sistema, esses aspectos são representados através do conjunto de modelos gerados durante o desenvolvimento.

- **Iterativo e incremental**: todo projeto pode ser dividido em partes menores, que somados representam o projeto inteiro. Cada parte menor do projeto é chamada de iteração, e seu resultado, um incremento. Portanto, um processo iterativo e incremental é composto de fases, cujos produtos convergem em direção ao produto final do processo.

No Processo Unificado, o desenvolvimento é dividido em quatro fases, como mostra a figura 2.4, dentro das quais podem ocorrer uma ou mais iterações:

- **Iniciação**: tem como objetivo iniciar o projeto, construindo o modelo de negócio, que inclui a viabilidade técnica e econômica, e justifica a continuidade do projeto. Deve definir a visão e o escopo do sistema, estabelecer arquiteturas candidatas, identificar os principais riscos e estimar de maneira aproximada os custos e recursos necessários para o projeto.

- **Elaboração**: tem como objetivo capturar os requisitos, formular os mesmos em casos de uso, estabelecer os fundamentos da arquitetura do sistema, monitorar alterações nos riscos identificados e identificar novos riscos, e detalhar o plano de projeto.

- **Construção**: tem como objetivo desenvolver, em iterações e incrementos, um sistema pronto para a operação inicial em seu ambiente final, o ambiente do usuário. Os componentes da arquitetura são implementados, integrados e testados.

- **Transição**: tem como objetivo estabelecer o produto em seu ambiente operacional final. Se o produto vai ser disponibilizado ao mercado, ou se será instalado apenas em um único cliente, por exemplo, é nesta fase que isto acontece.

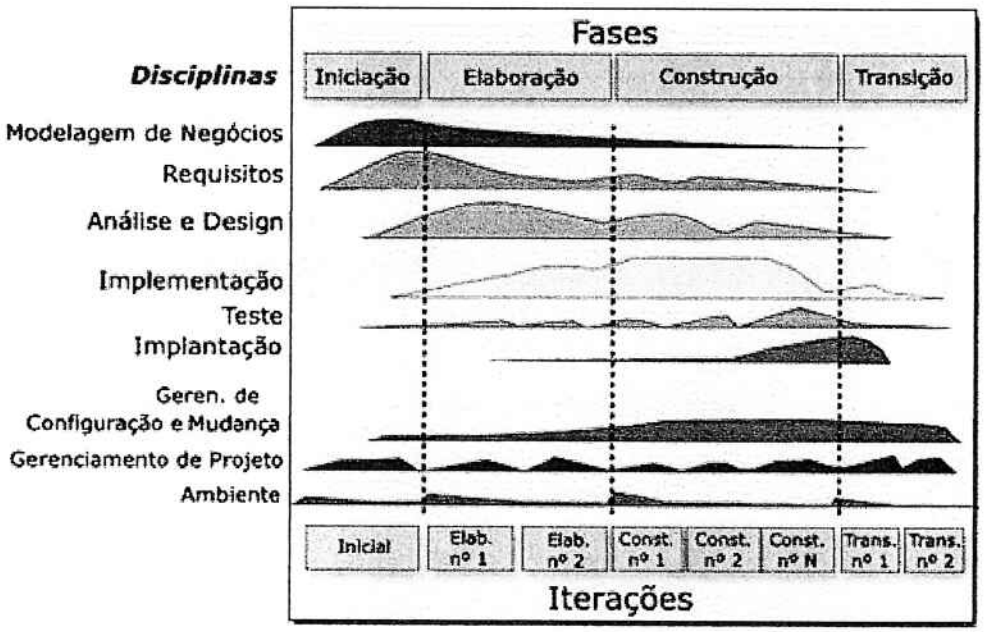


Figura 2.4 - Estrutura Estática do RUP (JACOBSON; BOOCH;RUMBAUGH, 1999)

Este processo é direcionado para desenvolvimento orientado a objeto.

O Processo Unificado contempla apenas as atividades referentes ao desenvolvimento, em Ambler, Nalbone e Vizdos (2005), são apresentadas complementações em relação as fases e as disciplinas desse processo, onde duas novas fases e oito novas disciplinas são acrescentadas, buscando a visão total do ciclo de vida do software e atender as perspectivas da Tecnologia da Informação. Este processo recebeu o nome de Enterprise Unified Process (EUP)[1], conforme é apresentado na figura 2.5.

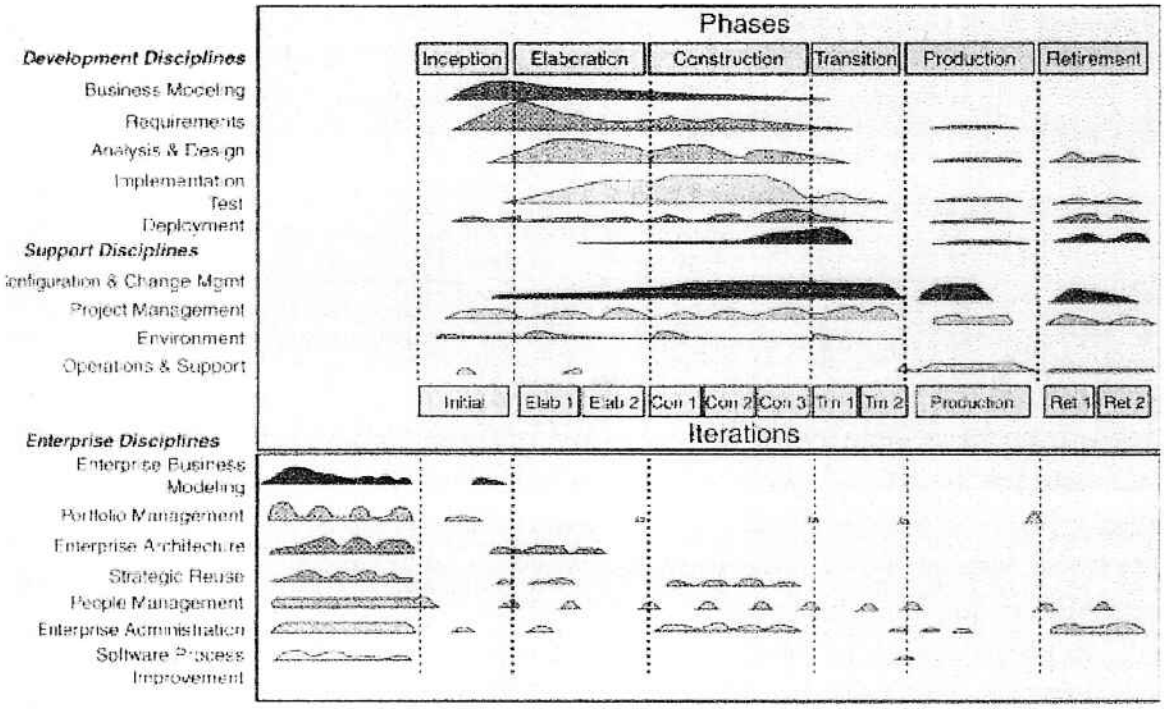


Figura 2.5 EUP ciclo de vida (AMBLER;NALBONE;VIZDOS, 2005)

A fase de Produção (*Production*) atende as manutenções que o software sofrerá após sua instalação em ambiente operacional. A fase de Descontinuidade (*Retirement*) foi adicionada para tratar o encerramento ou retirada do software do ambiente operacional produtivo.

[1] Os termos apresentados para o processo "*Enterprise Unified Process*" estão em idioma original, pois não foram identificadas as traduções oficiais.

2.4 – Conceitos de Manutenção

No ciclo de desenvolvimento de software, a manutenção é considerada a última fase do processo, isto é, ocorre após a implantação operacional, quando o sistema está em utilização pelos usuários em um ambiente real de produção. Qualquer alteração efetuada em relação ao sistema implantado, após o início de sua operação, é considerada como manutenção (PFLEEGER;FRANKLIN, 2004).

Essas modificações podem ser geradas por diversos fatores tais como, correções de defeitos, falhas, inclusão de novas funcionalidades, mudanças ocorridas no mundo externo, passíveis de afetar o resultado apresentado pelo software, assim como, mudança do ambiente operacional e alterações institucionais.

Segundo Vehvilaimen (2000) o *Institute of Electrical and Electronics Engineers* (IEEE), definiu quatro categorias para manutenção de software, conforme apresentado a seguir:

- **Correção ou Manutenção Corretiva:** consiste da atividade de correção de erros observados durante a operação do sistema;

- **Adaptação ou Manutenção Adaptativa:** consiste na realização das alterações no software, para que ele possa ser executado sobre um novo ambiente (CPU, arquitetura, novos dispositivos de hardware, novo sistema operacional, dentre outros);

- **Aperfeiçoamento** ou **Manutenção Perfectiva:** consiste na realização das alterações para melhorar alguns aspectos do software, como por exemplo, o seu desempenho, a sua interface, a introdução de novas funções.

- **Manutenção preventiva:** consiste na ocorrência de uma modificação do software, para melhorar a confiabilidade ou a manutenibilidade futura, ou para oferecer uma base melhor para futuras ampliações. Esta atividade é caracterizada pelas técnicas de engenharia reversa e reengenharia.

A princípio, a atividade de manutenção pode ser classificada individualmente. Na prática, não há uma distinção nítida entre esses diferentes tipos de manutenção, pois em uma correção de defeito, pode-se implementar uma rotina para prevenção ou até mesmo uma nova funcionalidade (SOMMEVILLE, 2003).

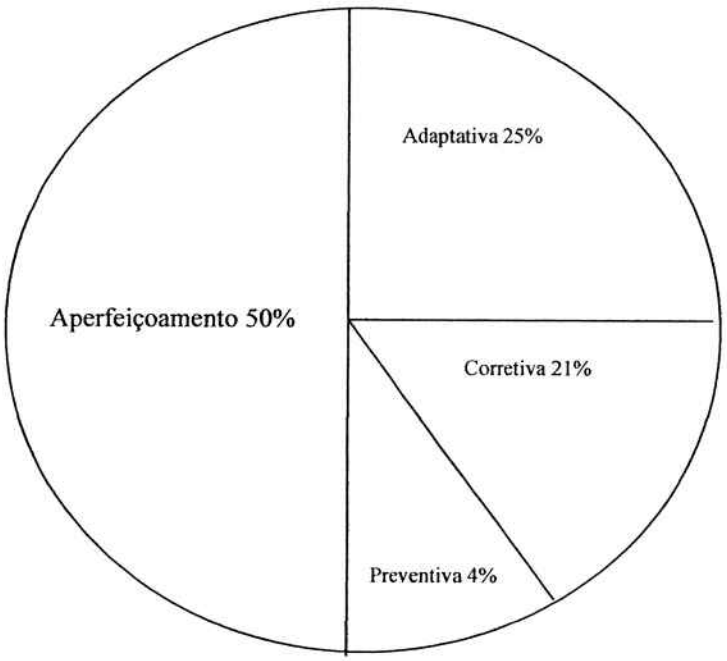


Figura 2.6 Distribuição do Esforço (PFLEEGER, 2004)

Como pode ser verificado na figura 2.6, os maiores esforços são despendidos na manutenção para melhoria do software, isto é, novas funcionalidades ou adaptações que não foram previstas na fase de desenvolvimento ou também mudanças nas regras do negócio.

Um dos motivos para que esta categoria tenha a percentagem apresentada é devido ao pouco conhecimento do cliente, das necessidades do negócio quando solicita o desenvolvimento do software. Muitas vezes, estes clientes partem de conhecimentos obtidos no mercado e que desejam implementar na sua empresa, porém sem consultar o usuário, aquele que trabalha no negócio diariamente, criando então um software com muitos recursos, mas que não atendem as necessidades reais do negócio, utilizadas pelos usuários. Para que o software não cause maiores prejuízos são solicitadas alterações, para atender as necessidades dos usuários.

Um outro motivo é a falta de entendimento do desenvolvedor na solicitação do cliente. Requisitos que não estejam claros e que tenham duplo entendimento é a causa principal desta ocorrência, que poderia ser identificada se houvesse melhoria na fase de levantamento de requisitos ou nas fases de testes e/ou homologação pelo cliente, mas normalmente isto não ocorre, pois as fases citadas são quase sempre ignoradas, principalmente quando o desenvolvimento está atrasado. Somente após sua implantação operacional é que se identifica a discrepância entre o solicitado e o construído, e para tentar minimizar os prejuízos são solicitadas às mudanças para adequar o software.

A mudança que mais ocorre é a inclusão de novas funcionalidades, pois com o tempo de utilização e as evoluções do negócio, fazem com que o software passe a necessitar de adequações para contemplar essas novas necessidades.

As alterações em um software serão melhores executadas, se durante o seu desenvolvimento houver a preocupação em se desenvolver um produto adaptável a futuras mudanças. Isto implica em ter uma documentação atualizada e códigos fontes bem construídos. Mas, se no desenvolvimento isto não for previsto, uma pequena alteração poderá gerar o custo de um novo desenvolvimento do software. Portanto, para diminuir o custo gasto na manutenção é importante construir software com qualidade e com uma boa estrutura, estando preparado para as futuras adaptações que irão ocorrer.

2.5 Considerações Finais

Para este trabalho são destacadas as atividades apresentadas pela norma NBR-ISO/IEC-12207 (1998) que estabelece um processo de manutenção, nos demais modelos de processo apresentados, a manutenção é tratada com uma fase que complementa o ciclo de vida do software, não sendo estabelecidas às atividades que a ela devem ser aplicadas.

O custo gasto com a manutenção pode ser considerado muito maior do que desenvolver um novo software, portanto, é necessário que se tenha estratégias para minimizar estes gastos. A definição de um processo com atividades claras e especificas pode ser uma destas estratégias, visando a melhoria da qualidade.

3 MENSURAÇÃO

3.1 Considerações Iniciais.

O objetivo deste capitulo é apresentar a importância da mensuração para o processo de software, tanto no desenvolvimento quanto na manutenção, visando a melhoria da qualidade. Para isto são apresentados os conceitos de métricas de software, suas classificações, como utilizar e as dificuldades encontradas de sua implementação. Além disso, são apresentadas as duas métricas utilizadas para a obtenção do tamanho do software, isto é, Pontos de Função e Linhas de Código Fonte. Estas métricas também são utilizadas para calcular o tamanho e o prazo para as alterações ocorridas no software.

3.2 – Conceitos de Métricas e a importância da mensuração

Para que se possa medir algo, é necessário que se saiba como fazê-lo e a esse conceito é que chamamos de **métrica** (PRESMMAN, 2003).

A definição dada por DeMarco apud Silva (2005), onde "Métrica é o número que você vincula a uma idéia, mais precisamente, é uma indicação mensurável de algum aspecto quantitativo do sistema". Esta definição explica que a métrica é um padrão de medida, utilizada para julgar os atributos de algo que está sendo medido.

Tom DeMarco (1989) define que uma métrica pode ser considerada útil quando for diferenciada e possuir quatro características:

- **Mensurável**: quando é obtido como resultado, um valor que possa representar uma quantidade que se possa medir.

- **Independente**: quando a obtenção de seus dados não sofre influência direta, das pessoas que estão interagindo na medição.

- **Coletável**: os dados obtidos devem ser armazenados, para servirem como base para outras atividades;

- **Precisa**: seus dados devem apresentar uma função de exatidão, isto é, os dados coletados devem ser avaliados quanto a sua exatidão para que sirvam de base de informação para o futuro.

Desde modo, uma métrica que atenda as características citadas, deverá ser implementada com facilidade e seus resultados devem alcançar os objetivos previstos para a sua aplicação. A simplicidade de entendimento também facilita na implementação e utilização da métrica.

A utilização da métrica auxilia o planejamento do projeto, pois, quando se obtém um valor absoluto do que se irá fazer, pode-se calcular prazo e custo com maior probabilidade de acerto, no desenvolvimento do produto e na melhoria contínua da qualidade do processo (DEMARCO, 1989).

Não se consegue avaliar os benefícios de uma métrica de forma imediata; por isso, há a necessidade da criação de uma base com os dados históricos de medições realizadas, que serve também para avaliação e refinamento da métrica utilizada.

Dentre as dificuldades que foram relatadas na utilização de uma métrica de software, as mais citadas são: é a falta de experimentos para validação, a falta de ferramentas de apoio, a falta de base conceitual e a falta de base histórica (DEMARCO, 1989).

Assim podemos ressaltar que as métricas auxiliam no gerenciamento e no planejamento do desenvolvimento e manutenção dos sistemas de software, além de oferecer uma compreensão sobre o processo de engenharia de software e de seu produto.

3.3 Classificação das Medidas e Métricas de Software

Conforme a norma NBR ISO/IEC 9126-1 (2003) as medidas são divididas em duas categorias, conforme exposto a seguir:

- **Medidas Diretas:** são aquelas medidas de um atributo que não depende da medida de qualquer outro atributo.
- **Medidas Indiretas:** são aquelas medidas de um atributo, a qual é derivada de medidas de um ou de vários outros atributos.

As métricas de software também se enquadram nestas categorias e são classificadas sob diferentes formas, considerando o tipo de dado a ser coletado, os objetivos e o nível de utilização delas, conforme apresentado a seguir.

- **Métricas de Tamanho**: medida direta do software e do processo por meio do qual ele é desenvolvido. São medidas de fácil obtenção, mas que ainda não estão universalmente aceita, pois falta precisão e dependem do que está sendo medido.

- **Métricas de Esforço**: medida indireta que são pré-requisitos para medidas confiáveis de custo de projetos em desenvolvimento e que auxiliam as organizações na melhoria do processo de desenvolvimento.

- **Métricas de Qualidade**: medida indireta que proporcionam um indicador de como se ajusta o software aos requisitos implícitos do cliente.

- **Métricas de Funcionalidade**: medida indireta do software e do processo pelo qual se desenvolve. Utiliza-se o ponto de vista do usuário. Estas medidas se centralizam na funcionalidade do produto.

- **Métricas de Desempenho**: medida indireta que avaliam o desempenho do produto, como tempo de resposta.

- **Métricas de Confiabilidade**: medida indireta em que são indicadas probabilidades do software de realizar suas tarefas sob determinadas condições em um período de tempo.

- **Métricas de Produtividade**: medida indireta que se concentram na saída do processo de engenharia de software.

- **Métricas Orientadas às pessoas**: medida indireta que compilam informações sobre a maneira segundo a qual as pessoas desenvolvem software e percepções humanas sobre a efetividade das ferramentas e métodos.

- **Métricas de Custo**: medida indireta que principalmente envolvem o custo de recursos humanos pelo tempo decorrido e ferramentas utilizadas.

As métricas de funcionalidade e de tamanho são as mais conhecidas e utilizadas, pois, com base na informação gerada por elas, é possível estabelecer parâmetros a serem

utilizados em outras métricas, tais como qualidade e produtividade; usadas no planejamento e na avaliação da qualidade.

Como a medição deve ser uma atividade constante no processo de desenvolvimento e manutenção, devemos selecionar as métricas mais adequadas ao projeto dentro das metas estabelecidas pela organização, alcançando assim um controle efetivo do que está sendo construído, visando a um software com melhor qualidade.

Para que os resultados gerados pelas métricas escolhidas se apresentem coerentes, temos que estabelecer os objetivos e as metas, da medição, antes de iniciarmos a coleta de dados, definir cada métrica de forma não ambígua e avaliando o resultado com dados anteriormente coletados ou em bases disponíveis no mercado (DEMARCO, 1989).

O conhecimento mais aprofundado das métricas, de suas características e tipos, facilita a implementação desta atividade, isto também facilita a seleção das métricas ideais para as metas almejadas pela empresa.

3.4 Implementação da Atividade de Medição.

Conhecendo as métricas existentes é possível selecionar quais são as mais adequadas para a implementação da atividade de medição, esta atividade não é uma tarefa fácil. Segundo Presmman (2003), a maior dificuldade para implementar a atividade de medição é o convencimento das pessoas envolvidas da sua necessidade. Existe uma resistência natural, pois o medo que a utilização das informações geradas, seja utilizada

para avaliação de desempenho pessoal, acaba tornando-se um grande dificultador desta atividade.

Segundo Tom DeMarco (1989), quando uma empresa for implantar a atividade de medição, é necessário que seja composta uma equipe especializada para essa função, que não sofra interferência das equipes de projeto. Esta equipe deverá determinar as metas e os objetivos a serem mensurados nos projetos, bem como ser responsável pela criação de uma base de dados, gerados a partir dos dados obtidos das medições.

Portanto, é importante que seja definido quais são as metas da empresa para utilização da atividade de medição, estas devem ser claras e difundidas nas equipes.

A equipe designada para iniciar esta atividade deverá também efetuar a medição do software já existente criando uma base de dados histórica. Este trabalho é difícil, pois dependerá de informações antigas, que nem sempre estão disponíveis. Este levantamento é muito importante, pois com nas informações obtidas, é possível a avaliação e definição de métricas, para avaliação do desempenho e da qualidade, como também servirá para a medição das alterações que o software venha a sofrer.

Caberá a esta equipe, o suporte e treinamento necessários às demais equipes da empresa, esclarecendo sempre a importância desta atividade e conscientizando a todos das metas principais.

3.5 Métrica de Funcionalidade – Pontos de Função

Ponto de função é uma medida funcional de tamanho de software, cujo conceito foi inicialmente introduzido por Allan J. Albrecht (IBM White Plains) em 1979 (AGUIAR, 2003). Posteriormente, esse conceito foi aprimorado em metodologia formal, tornando-se de domínio público em 1984. Com a utilização desta metodologia, foi formado um grupo de usuários que resolveram efetuar padronizações adicionais às regras de contagem; este grupo formou a partir de 1986 o Grupo Internacional de Usuários de Ponto de Função, organização internacional sem fins lucrativos, sediada nos Estados Unidos da América (DEEKERS, 1998). Hoje este grupo é responsável pelas publicações do Manual de Práticas de Contagem, atualmente em sua versão 4.2, que estabelece os padrões para o cálculo dos pontos de função. O grupo também responde pelas publicações de artigos, certificações de especialistas e treinamento, ele realiza também, duas conferências anuais. Possui membros em mais de 13 países, inclusive no Brasil, que participam nos comitês, conferências e treinamentos. Em 2002 esta métrica passou à condição de padrão internacional, através da norma ISO/IEC 20926 de 2002 apud Dekkers (2003).

A métrica de Ponto de Função é considerada uma medida funcional, pois é baseada em uma avaliação padronizada dos requisitos lógicos dos usuários. Quando utilizada com outras métricas, auxilia o planejamento dos projetos de software.

Uma das maiores vantagens desta métrica é que podemos utilizá-la em qualquer tipo de projeto do software, independentemente de linguagem, método de desenvolvimento ou modelo de projeto.

3.5.1 Identificação dos parâmetros

Para iniciar o processo de contagem de Pontos de Função, temos que estabelecer alguns parâmetros como tipo de contagem e a definição das fronteiras da aplicação.

O Tipo de contagem determina as funcionalidades que serão incluídas numa contagem específica de Ponto de Função.

Existem três tipos de contagem de ponto de função:

- Contagem de ponto de função de projetos em desenvolvimento;
- Contagem de ponto de função para projeto finalizado;
- Contagem de ponto de função para projeto de manutenção

A fronteira da aplicação indica a linha entre o software que está sendo medido e o usuário, servindo como membrana através da qual os dados processados, passam para dentro e para fora do sistema de software. Ajuda também a identificar os dados lógicos referenciados, mas não mantido pelo sistema de software.

3.5.2 Contagem de Ponto de Função.

A contagem de Pontos de Função está dividida em duas fases:

- Contagem de Pontos de Função não ajustados;
- Fator de Ajustamento.

A contagem de Ponto de Função não ajustado reflete especificamente a contagem das funcionalidades providas para o usuário através do projeto. Somente o que foi requisitado e definido pelo usuário é contado.

A contagem de pontos de função não ajustados, conforme figura 3.1, compreende dois tipos de funções:

- Funções de dados: constituem arquivos lógicos internos e arquivos de interface externa;
- Funções transacionais: constituem entradas externas, saídas externas e consultas externas.

A contagem das funções de dados representa as funcionalidades disponíveis para o usuário, através de dados internos e externos que foram requeridos. Estas funções de dados são tratadas como Arquivo Lógico Interno (ALI) e Arquivo de Interface Externa (AIE).

Um arquivo lógico interno é um grupo de dados lógicos relacionados ou informações de controle, identificados pelo usuário e que sejam mantidos pelo sistema de software.

Um arquivo de interface externa é um grupo de dados lógicos relacionados ou informações de controle, que são utilizadas pelo sistema de software medido, mas que não sofra manutenção, isto é, é um arquivo lógico interno de outro sistema.

A contagem das funções transacional representa as funcionalidades disponíveis para o usuário processar os dados. Estas funções são chamadas de Entrada Externa (EE), Consulta Externa(CE) e Saída Externa (SE).

Entrada Externa é um processo elementar, trata os dados ou informações de controle que entram no sistema, mantendo (incluindo/alterando/excluídos) dados de um ou mais ALI.

Consulta Externa é um processo elementar que envia dados ou informações de controle para fora da aplicação, isto é, apresenta os dados mantidos no ALI e AIE sem efetuar nenhum processamento lógico (cálculo ou fórmulas matemáticos).

Saída Externa é um processo elementar que envia os dados mantidos pelo sistema nos ALIs, efetuando algum tipo de processamento lógico (cálculo ou fórmulas matemáticas), criando dados derivados.

Pode-se calcular os pontos de função não ajustados a partir da identificação das funções do projeto.

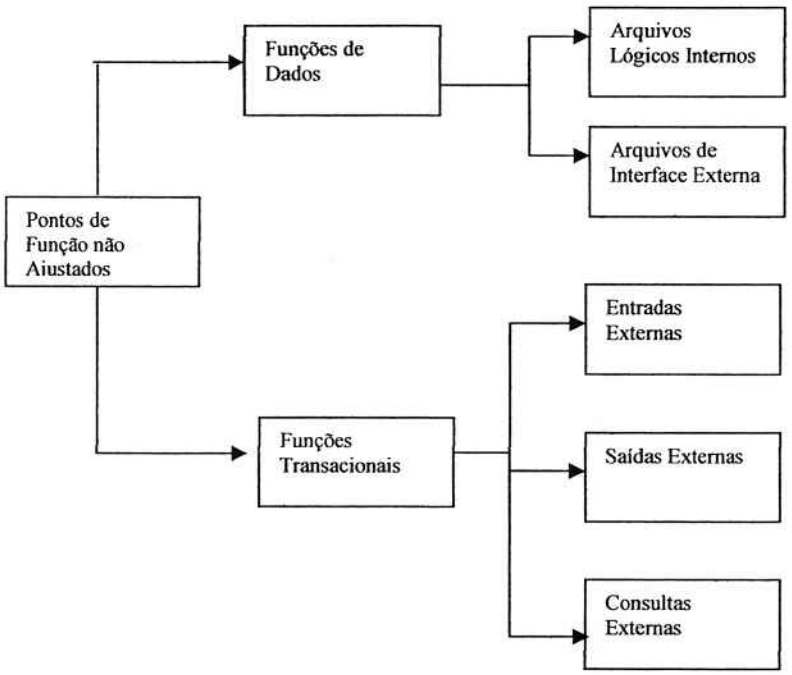


Figura 3.1 - Fases da Contagem de Pontos de Função não Ajustados (IFPUG)

Após essa identificação, devemos classificar cada uma das funções conforme seu nível de complexidade baixa, média ou alta.

O nível de complexidade das funções de dados é determinado pelos números de elementos de dados e tipos de elementos de registro associados com os ALI e AIE.

O nível de complexidade das funções transacionais é determinado pelos números de tipos de arquivos referenciados e tipos de elementos de dados associados com os EE, SE e CE.

A tabela 3.1 apresenta o número de pontos de função atribuídos conforme as complexidades das funções.

Para se obter o resultado do Total de Pontos de Função não Ajustados (TPFNA), utiliza-se a seguinte fórmula:

$$TPFNA = \sum (ALI \times Fi) + \sum(AIE \times Fi) + \sum(EE \times Fi) + \sum(SE \times Fi) + \sum(CE \times Fi)$$

Tabela 3.1 - Tipo de Função/Complexidade (DEKKERS, 1998).

Tipo de Função \ Complexidade	Baixa	Média	Complexa
Entrada Externa (EE)	3	4	6
Saída Externa (SE)	4	5	7
Consulta Externa (CE)	3	4	6
Arquivo Lógico Interno (ALI)	7	10	15
Arquivo Interface Externa (AIE)	5	7	10

Após a obtenção do TPFNA, temos que determinar o fator de ajustamento (FA) que indica a funcionalidade geral proporcionada ao usuário. O FA consiste de 14

características, que devem ser avaliadas através do nível de influência que é dado em uma escala de 0 a 5 conforme descrito abaixo:

0 - não está presente ou não tem influência

1 - pouca influência

2 - moderada influência

3 - média influência

4 - significante influência

5 - forte influência

As 14 características são descritas a seguir:

1ª - Comunicação de Dados: descreve o grau em que a aplicação se comunica diretamente com o processador.

2ª - Processamento de Dados Distribuído: descreve o grau em que a aplicação transfere dados entre os componentes da aplicação.

3ª - Desempenho: descreve o grau em que as considerações de tempo de resposta e do tempo de processamento influenciaram o desenvolvimento da aplicação.

4ª – Utilização de Equipamento Configuração altamente utilizada: descreve o grau em que as restrições de recursos de computador influenciaram o desenvolvimento da aplicação.

5ª – Volume de Transação: descreve o grau da quantidade de transações previstas no negócio que influência o desenvolvimento da aplicação.

6ª - Entrada de Dados On-Line: descreve o grau em que os dados são fornecidos através de transações interativas.

7ª - Eficiência do Usuário Final: descreve o grau de consideração em relação a fatores humanos e a facilidade de uso, para o usuário, da aplicação objeto de contagem.

8ª - Atualização On-Line: descreve o grau em que a aplicação possibilita atualização on-line dos arquivos lógicos internos.

9ª - Processamento Complexo: descreve o grau em que o processamento lógico tem influenciado o desenvolvimento da aplicação.

10ª – Reutilização de Código: descreve o grau em que a aplicação e os códigos têm sido especificamente projetados, desenvolvidos e suportados para serem utilizáveis para outras aplicações.

11ª - Facilidade de Instalação: descreve o grau em que a conversão de ambiente anterior tem influenciado o desenvolvimento da aplicação.

12ª - Facilidade Operacional: descreve uma característica da aplicação. Uma aplicação minimiza a necessidade de atividades manuais, tais como: montagem de fitas, manuseio de papéis e intervenção manual direta do operador.

13ª - Múltiplas Instalações: descreve o grau em que a aplicação tem sido desenvolvida para várias localizações e organizações dos usuários.

14ª - Facilidade de Mudanças: descreve o grau em que a aplicação tenha sido desenvolvida de modo que permita facilidade de modificação do processamento lógico ou da estrutura de dados.

Após a aplicação dos níveis de influência em cada uma das 14 características, totalizando, teremos o nível de influência total (TNI), que utilizaremos para determinar o FA conforme a fórmula a seguir:

FA = 0.65 + (0.01 x TNI) onde:

TNI : é o somatório dos níveis de influência.

O valor 0,01 é uma constante para simplificar a divisão por 100.

O valor 0,65 é para produzir uma variação de +/- 35%. Esta variação foi definida por Albretch na época da apresentação de Análise de Pontos de Função e deveria representar a variação das 14 características para a realidade dos sistemas da época. Recentemente o IFPUG tornou o uso do fator de ajuste opcional como uma adequação necessária ao padrão ISO/IEC de medição funcional (VAZQUEZ;SIMÕES;ALBERT, 2003).

Para calcular o total de pontos de função do projeto, é necessário multiplicar o Fator de Ajuste (FA) pelo Total de Pontos de Função Não Ajustados (TPFNA).

TPF = FA x PFNA

O total de pontos de função indica o tamanho estimado do projeto a ser desenvolvido. Com estes dados, podem ser estimados o prazo e o custo, utilizando-se métricas de esforço.

3.5.3 Diferença entre os tipos de contagem

Existem três tipos de contagem aceitos pela IFPUG conforme é apresentado a seguir:

- **Projeto de Desenvolvimento**: onde são medidas as funcionalidades fornecidas aos usuários finais do software quando da sua primeira instalação.

- **Projeto de Melhoria**: mede as funções adicionadas, modificadas ou excluídas em uma aplicação já instalada, e também eventuais funções de conversão de dados.

- **Aplicação**: mede as funcionalidades fornecidas aos usuários para uma aplicação instalada.

A contagem do Projeto de Desenvolvimento é utilizado na fase de planejamento, para que seja obtida uma medição estimada do projeto, e assim calcular o custo e prazo estimado. Esta contagem é baseada nas estimativas das funções de dados e funções transacionais

A contagem de Projeto de Melhoria é utilizado para a medição das alterações ou inclusões que são efetuadas durante a Manutenção. Cada organização estabelece como utilizar este tipo, obedecendo às regras do IFPUG, visto que as mesmas podem ser adaptadas. Esta contagem é obtida pontuando as alterações sofridas nas funções de dados e funções transacionais.

A contagem da Aplicação é utilizada para medição do software após sua finalização.

Estes conceitos são os definidos pelo IFPUG em seu manual de Contagem de Pontos de Função.

3.6 Métricas de Tamanho – Linhas de Código

A métricas de tamanho são medidas diretas que se obtém totalizando a quantidade de linhas descritas nos códigos fontes do software. Moller e Paulish (GRUBB;TAKANG, 2003) definiram linhas de código (LOC) como "a quantidade de linhas do código excluindo comentários e linhas em branco".

Embora esta métrica possa parecer simples, existe discordância sobre o que constitui uma linha de código (VAZQUEZ;SIMÕES;ALBERT, 2003), pois nem todos seguem as definições de Moller e Paulish e totalizam linhas em branco e comentários, o que geram inúmeras controvérsias na sua utilização. Outra crítica a esta métrica é que por ser simples ela não reflete o custo ou produtividade do projeto (GRUBB;TAKANG, 2003).

Para a utilização de LOC, o ideal é ter um padrão de programação por linguagem na empresa, assim, a partir deste padrão, torna-se mais fácil à utilização de ferramentas para a obtenção dos dados.

Uma das vantagens desta métrica e que ela é de fácil obtenção e podendo ser utilizada para avaliar a qualidade, como por exemplo, quantidade de erros encontrados por linhas de código (erros/kloc).

LOC pode ser utilizada no processo de manutenção, desde que se tenha a quantidade de linhas que a aplicação tinha antes da alteração ou da inclusão de nova funcionalidade e verificar a quantidade de linhas que foram incluídas, excluídas ou alteradas pela mudança. Com isto se obtêm o tamanho da manutenção.

3.7. Modelo de Custo

Boehm projetou um modelo do custo chamado *Constructive Cost Model* (COCOMO) como modelo estático de valor simples que contabiliza o esforço e custo de desenvolvimento de software em função do tamanho do software, sua calibração foi efetuada a partir dos dados obtidos nos projetos da empresa TRW, uma consultoria situada na Califórnia (FENTON;FFLEGER, 1997). COCOMO é um modelo relativamente direto baseado nas entradas que se relacionam ao tamanho do software com o custo que afeta a produtividade. O modelo original de COCOMO foi publicado primeiramente em 1981 (Boehm, 1981). Boehm fez evoluções no modelo para abranger novas tecnologias, que é chamado COCOMO II (BOEHM, 2000).

O COCOMO pode ser aplicado em três classes de projetos de software, conforme é apresentado a seguir:

- **Modelo orgânico:** projeto de software simples, equipe pequena com boa experiência;
- **Modo semidestacado:** projeto de software intermediário, em tamanho e complexidade, no qual equipes com níveis de experiência mistos devem atingir uma combinação de requisitos rígidos e não tão rígidos;
- **Modo embutido:** projeto de software que deve ser desenvolvido dentro de um conjunto rígido de restrições operacionais, de hardware e de software (por exemplo, software para controle de vôo de aeronaves).

O modelo básico é ampliado para considerar um conjunto de "atributos direcionadores do custo" que podem ser agrupados em quatro grandes categorias:

- **Atributos do produto**
  - a. Confiabilidade exigida do software
  - b. Tamanho do banco de dados da aplicação.
  - c. Complexidade do produto.
- **Atributos do hardware**
  - a. Restrições ao desempenho em tempo de execução.
  - b. Restrições de memória.
  - c. Volatilidade do ambiente de máquina virtual.
  - d. Tempo para completar o ciclo exigido.
- **Atributos de pessoal**
  - a. Capacidade de análise.
  - b. Capacidade em engenharia de software.
  - c. Experiência em aplicações.
  - d. Experiência em máquina virtual.
  - e. Experiência em linguagens de programação.
- **Atributos de projeto**
  - a. Uso de ferramentas de software.
  - b. Aplicação de métodos de engenharia de software.
  - c. Cronograma de atividades de desenvolvimento exigido

Cada um dos 15 atributos é classificado de acordo com uma escala de 6 pontos que varia de "muito baixo" a "extremamente elevado", em importância e valor.

Baseando-se na classificação, um multiplicador de esforços é determinado a partir das tabelas publicadas por Boehm (Boehm, 1981) e o produto de todos os resultados de multiplicadores de esforços torna-se um fator de ajustamento de esforços (EAF). Os valores típicos do EAF variam de 0,9 a 1,4.

3.8 Considerações Finais.

Para obter o controle do que está sendo executado é necessário que se saiba o tamanho do produto que está sendo gerado ou alterado. As métricas auxiliam os gerentes neste controle, pois fornecem dados para a avaliação de custo, prazos e qualidade.

Dentre as métricas utilizadas para medição do tamanho, as mais utilizadas são KLOC e Pontos de Função, sendo que a última por definição mede as funcionalidades do software.

Na manutenção estas métricas também podem ser utilizadas, seus dados podem estimar o tamanho da alteração e assim através de dados históricos são calculados o custo e prazo da execução.

4 MODELO PROPOSTO DE PROCESSO DE MANUTENÇÃO

4.1 Considerações Iniciais

O objetivo deste capitulo é apresentar o processo proposto para a Manutenção de Software. Este modelo visa estabelecer as fases e as atividades voltadas à manutenção, também apresenta os artefatos que devem ser gerados e atualizados em cada uma das fases e os responsáveis por cada atividade. Os modelos de desenvolvimento tiveram as seguintes influências nesta proposta: as fases foram baseadas na Norma NBR ISO/IEC-12207 e no Modelo Cascata; do Modelo Espiral o embasamento se deu pelos conceitos de interatividade com o cliente e análise de risco, responsável pela solicitação da alteração, que pode decidir pela continuidade ou não da execução da alteração; a iteração das fases e o controle do gerenciamento e a definição dos papéis dos responsáveis são os conceitos apresentados no Processo Unificado e do *Enterprise Unified Process.*

4.2 Modelo de Processo de Manutenção.

Após a instalação do software em operação, qualquer solicitação de mudança seja uma inclusão de nova funcionalidade, correção de um defeito ou melhoria de desempenho; o desenvolvimento desta alteração deve ser tratado no Processo de Manutenção. O

modelo proposto apresentado a seguir, define as fases e as atividades previstas para a manutenção.

a) Fase 1: Análise da Manutenção

Nesta fase deve ser efetuada a pré-análise da alteração avaliando sua viabilidade, a estimativa de custo e de prazo para a execução. A seguir são apresentadas as atividades previstas para esta fase:

- **Recebimento da Alteração**: esta atividade inicia o processo de manutenção. Normalmente, a solicitação é repassada através de um documento formal ou através de e-mail ou uma chamada telefônica. Porém, independente da forma como é efetuada, a solicitação deve ser registrada em um documento próprio, denominado Termo de Abertura para Manutenção.

- **Avaliação da Viabilidade da Alteração**: esta atividade deve ser efetuada baseada no Termo de Abertura de Manutenção, sendo que a análise deve ser efetuada pela equipe de manutenção, avaliando a viabilidade de sua execução. Caso seja considerado não viável deve-se retornar ao cliente um documento com a explicação detalhada dos riscos que impedem a realização da alteração e proceder ao arquivamento do documento.

- **Rastreamento da Alteração**: no caso de ser considerada uma alteração viável, deve-se então, avaliar dentro da versão do software em operação, quais são as funções que serão afetadas com a mudança, executando um rastreamento nos requisitos já existentes e avaliar os possíveis riscos.

- **Estimativa de Tamanho**: após o rastreamento é estimado o tamanho da alteração para obter prazo e o custo estimado da manutenção. Para efetuar esta

estimativa, devem ser utilizadas métricas de software apropriadas, como Pontos de Função.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- **Gerente de Manutenção**: é o responsável pelo recebimento da solicitação de alteração e de encaminhar a equipe de manutenção correspondente.

- **Analista de Negócio**: é o responsável no entendimento da alteração avaliando junto ao cliente e a equipe de manutenção a sua viabilidade e no rastreamento dos requisitos a fim de verificar quais funções serão afetadas com a mudança.

- **Analista de Manutenção**: é o responsável em conjunto com o analista de negócio, da avaliação da alteração quanto a sua viabilidade, deve executar o rastreamento dos requisitos e executar uma prévia estimativa de prazo para a execução da alteração.

- **Cliente**: é o responsável pela solicitação da alteração, que deverá estar descrita de forma clara. Auxiliando nos esclarecimentos das dúvidas da equipe técnica. É o responsável pela autorização para continuidade da alteração.

Os artefatos que são gerados nesta fase:

- Termo de Abertura de Manutenção: é onde se descreve a solicitação do cliente.

- Documento de avaliação da alteração: é onde se descreve sucintamente a alteração que será executada, bem como as funções e requisitos que serão afetados e os possíveis riscos que a ela poderá ocasionar no software em operação.

- Documento de Prazo e Custo: onde é apresentado ao cliente o prazo e o custo da alteração. Estas informações devem estar baseadas em uma métrica como Ponto de Função.

Tabela 4.1 - Matriz de Responsabilidade – Fase 1

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	**Quem Coordena**	**Quem Executa**	**Quem Participa**	**Quem Aprova**	**Artefatos Entrada**	**Artefatos Saída**
Fase 1 - Análise de Manutenção						
1.1 Recebimento da Alteração	Gerente de Manutenção	Gerente de Manutenção	Analista de Negócio		Documento de Solicitação	Termo de Abertura de Manutenção
1.2. Avaliação da Viabilidade da Alteração	Gerente de Manutenção	Analista de Negócio	Analista de Negócio e Cliente	Cliente	Termo de Abertura Homologação	Avaliação Preliminar da Alteração
1.3. Rastreamento da Alteração	Gerente de Manutenção	Analista de Manutenção	Analista de Negócio		Avaliação Preliminar da Alteração	Avaliação Final da Alteração
1.3. Estimativa de Tamanho	Gerente de Manutenção	Analista de Manutenção	Gerente de Manutenção	Cliente	Avaliação Final da Alteração	Doc.Custo e Prazo

b) Fase 2: Projeto de Manutenção

Nesta fase é iniciada a partir da aceitação do cliente, e deve ser executado o detalhamento da alteração, o rastreamento dos documentos e códigos que serão alterados. A seguir são apresentadas as atividades previstas para esta fase:

- **Avaliação detalhada da alteração**: a partir da aceitação do cliente deve ser detalhada a alteração, avaliando o impacto que causará nos registros de requisitos e no modelo de dados. Caso haja uma equipe específica para tratar do modelo de dados, esta avaliação deve ser executada em conjunto com a mesma.

- **Alteração do Modelo de Dados**: caso no detalhamento tenha sido identificado que a mudança solicitada implique em alterações no modelo de dados, a mesma deverá ser executada pela equipe especifica, caso haja, ou pela própria equipe de manutenção.

- **Alteração nos Registros de Requisitos:** após o detalhamento da alteração, se deve avaliar os registros de requisitos que serão alterados ou incluídos e proceder a sua alteração.

- **Rastreamento dos Códigos Fontes:** com o modelo de dados e os registros de requisitos alterados, deve ser executado o rastreamento nos códigos fontes para avaliação de quais serão afetados com a mudança, separando uma cópia dos códigos a serem alterados em uma biblioteca própria para manutenção. Esta atividade é muito importante, pois visa a integridade das versões dos códigos em operação.

- **Preparação do Ambiente para Teste**: com o modelo de dados e os registros de requisitos alterados, a equipe de teste pode iniciar a preparação do plano de teste e dos casos de teste, para a validação das alterações.

- **Estimativa de tamanho detalhada**: a nova medição é efetuada com os registros de requisitos e modelo de dados alterados, com esses dados é possível obter a estimativa de tamanho da alteração e assim calcular o custo e prazo estimado para sua execução. Pode se então gerar o cronograma detalhado das demais atividades da alteração.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- **Gerente de Manutenção**: é o responsável pelo envio da aceitação do cliente a equipe de manutenção, da elaboração do cronograma e deve também acompanhar e controlar a execução da alteração.

- **Analista de Banco de Dados**: é o responsável em conjunto com a equipe de manutenção pela alteração no modelo de dados e sua atualização.

- **Analista de Negócio**: é o responsável no esclarecimento de dúvidas entre a equipe de manutenção e o cliente, auxilia o Gerente na elaboração do cronograma.

- **Analista de Manutenção**: é o responsável pelo rastreamento e atualização dos requisitos; em conjunto com Analista de Banco de Dados em alterar o modelo de dados; em conjunto com Arquiteto de Teste da criação do plano e caso de teste e de controlar as versões dos códigos fontes que serão alterados.

- **Arquiteto de Teste**: é o responsável em conjunto com a equipe de manutenção da criação do plano e caso de teste.

Os artefatos que são gerados nesta fase:

- Modelo de banco de dados atualizado: é onde constam as tabelas e o relacionamento entre elas, apresentando assim a estrutura de armazenamento das informações mantidas pelo software. Este modelo deve ser atualizado a cada alteração nos atributos das tabelas ou em seus relacionamentos ou também na inclusão de novas tabelas.

- Registro de Requisitos atualizado: é o documento onde devem ser registrados os requisitos do software, devendo ser atualizado a cada alteração.

- Plano de Teste: é o documento onde constam os tipos de testes a serem realizados e como deve ser sua execução.

- Casos de Teste: é o documento onde constam as funcionalidades que devem ser testadas.

- Cronograma: é o documento onde constam as atividades, prazos, recursos alocados e custos do projeto.

Tabela 4.2 - Matriz de Responsabilidade Fase 2

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	**Quem Coordena**	**Quem Executa**	**Quem Participa**	**Quem Aprova**	**Artefatos Entrada**	**Artefatos Saída**
Fase 2 - Projeto de Manutenção						
2.1 Avaliação detalhada da Alteração	Gerente de Manutenção	Analista de Manutenção	Analista de Negócio		Avaliação Preliminar da Alteração, Registro de Requisitos e Modelo de Dados	
2.2 - Alteração no Modelo de Dados	Gerente de Manutenção	Analista de Banco Dados	Analista de Manutenção	Analista de Negócio	Avaliação Final da Alteração e Modelo Dados Anterior	Modelo de Dados Atualizado
2.3 - Alteração dos Registros de Requisitos	Gerente de Manutenção	Analista de Manutenção	Analista de Manutenção	Analista de Negócio	Registro de Requisitos anterior	Registro de Requisitos Atualizado
2.4 -Rastreamento dos códigos fontes	Gerente de Manutenção	Analista de Manutenção			Códigos Fontes	Lista de Códigos Fontes a serem alterados
2.5 - Preparação do Ambiente de Teste	Gerente de Manutenção	Arquiteto de Teste	Analista de Manutenção	Analista de Negócio	Modelo de Dados Atualizado e Registro de Requisitos Atualizado	Plano de Testes e Casos de Testes
2.6. Estimativa da Alteração	Gerente de Manutenção	Analista de Manutenção	Analista de Negócio e Gerente de Manutenção	Cliente	Modelo de Dados Atualizado e Requisitos Atualizado	Cronograma e Custo

c) Fase 3: Construção

Nesta fase são executadas as alterações nas especificações de programas, as alterações nos bancos de dados, as alterações e implementações nos códigos fontes. O importante desta fase é manter a integridade dos códigos em operação com os que estão em manutenção. A seguir são apresentadas as atividades previstas para esta fase:

- **Alteração nas Especificações de Programas**: com o rastreamento dos códigos fontes a serem alterados, e com base nos registros de requisitos e modelo de dados alterados, deve se proceder à alteração nas especificações de programas existentes e a especificação dos novos programas. É importante manter registrado no documento: o motivo da alteração, a data e o responsável pela solicitação.

- **Alteração dos Bancos de Dados**: com o modelo de dados alterado, deve se proceder à alteração nos bancos de dados. Em empresas que mantêm equipes especialistas em Banco de Dados, esta atividade é efetuada por essa equipe.

- **Alteração nos Códigos Fontes**: com as especificações de programas a serem alterados e das novas especificações, são executadas as alterações nas cópias dos códigos fontes que estão na biblioteca própria para a manutenção.

- **Testes Unitários**: a cada finalização das alterações nos códigos fonte deve ser executada os testes unitários no programa, para validar a alteração com o que foi descrito na especificação.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- **Gerente de Manutenção**: é o responsável em acompanhar e controlar a execução da alteração.

- **Analista de Manutenção**: é o responsável pelo controle de configuração, especificação dos programas e auxilia os programadores nos testes.

- **Programadores**: é o responsável pela codificação dos códigos fontes e dos testes unitários.

Os artefatos que são gerados nesta fase:

- Especificações dos códigos fontes alteradas e novas: é o documento onde constam as definições das funcionalidades de um programa.

- Banco de dados alterados: são as tabelas que mantêm as informações do software.

- Códigos fontes alterados e novos.

Tabela 4.3 - Matriz de Responsabilidade Fase 3

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	**Quem Coordena**	**Quem Executa**	**Quem Participa**	**Quem Aprova**	**Artefatos Entrada**	**Artefatos Saída**
Fase 3 - Construção						
3.1 - Alteração nas Especificações de Programas	Gerente de Manutenção	Analista de Manutenção			Modelo de Dados Atual, Registro de Requisitos Atual e Especificações dos Programas Anterior	Especificações de programas alteradas
3.2 - Alteração dos Bancos de Dados	Gerente de Manutenção	Analista de Banco de Dados	Analista de Manutenção		Modelo de Dados Atualizado	Modelo de Dados Físico Atualizado
3.2 - Alteração dos Programas	Gerente de Manutenção	Programador	Analista de Manutenção		Especificações de programas alterados e novos	Programas Fontes Alterados e novos
3.3 – Testes Unitários	Gerente de Manutenção	Programador	Analista de Manutenção		Especificações de programas alteradas	Doc. de Evidência de Testes

d) Fase 4: Teste

Nesta fase devem ser realizados os testes de verificação das alterações, não somente os códigos fontes, mas também devem ser avaliados as integrações do software com os demais softwares que fazem interface. Assim, é recomendado que as atividades desta fase sejam executadas por uma equipe especializadas em teste, que tenha o conhecimento de como efetuar a validação seguindo o plano de teste que foi idealizado na fase de projeto. Essa equipe participa na fase de projeto de manutenção, elaborando o plano de teste tendo como base os registros de requisitos e prepara os casos de testes para verificação e validação do software. A seguir são apresentadas as atividades previstas para esta fase:

- **Elaboração dos Casos de Teste**: a partir do plano de teste elaborado na fase de Projeto de Manutenção são criados os casos de testes para a validação do software alterado.

- **Criação da Massa de Teste**: com os casos de testes e preparada a massa de teste para a execução das validações e verificações da nova versão;

- **Execução dos Testes:** com a massa de teste pronta deve ser executado o teste de integração, de regressão e de sistema, para se evitar possíveis falhas que podem ocorrer, avaliando os resultados das alterações nas integrações junto aos demais softwares.Todos os resultados devem ser anotados, pois farão parte da documentação da manutenção.

- **Correção dos erros**: os erros encontrados deverão ser encaminhados a equipe de manutenção para a correção. A equipe de manutenção deve corrigir os erros e retornar o software a uma segunda bateria de testes.

- **Avaliação dos resultados dos testes**: as equipes de teste e manutenção devem avaliar o término dos testes, disponibilizado para a validação junto ao cliente.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- Gerente de Manutenção: é o responsável em acompanhar e controlar a execução da alteração.

- Analista de Manutenção: é o responsável em conjunto com o analista de teste da execução é avaliação dos resultados dos testes

- Arquiteto de Teste: é o responsável pela criação dos casos de teste e execução dos testes.

- Analista de Teste: é o responsável em conjunto com o Analista de Manutenção na execução dos testes e na avaliação.

- Programador: é a responsável pela correção nos códigos fonte dos erros encontrados.

O artefato que é gerado nesta fase:

- Documento de Evidência de Teste: é onde devem constar os resultados obtidos com os testes.

Tabela 4.4 - Matriz de Responsabilidade Fase 4

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	Quem Coordena	Quem Executa	Quem Participa	Quem Aprova	Artefatos Entrada	Artefatos Saída
Fase 4 – Teste						
4.1- Elaboração dos Casos de Testes	Gerente de Manutenção	Arquiteto de Teste	Analista de Teste e Manutenção	Analista de Teste e Manutenção	Plano de Teste	Casos de Teste
4.2 - Criação da Massa de Teste	Gerente de Manutenção	Arquiteto de Teste	Analista de Teste e Manutenção	Analista de Teste e Manutenção	Modelo de Dados Atualizado e E-R Atualizado	Massa de Teste
4.3 - Execução dos Testes	Gerente de Manutenção	Arquiteto de Teste	Analista de Teste e Manutenção	Analista de Teste e Manutenção	Massa de Testes	
4.4 – Correção dos Erros Encontrados	Gerente de Manutenção	Programador	Analista de Teste e Manutenção	Analista de Teste e Manutenção	Códigos Fontes	Códigos Fontes Alterados
4.5– Avaliação dos Resultados dos Testes	Gerente de Manutenção	Analista de Teste e Manutenção	Analista de Negocio e Manutenção	Analista de Negocio e Manutenção		Doc. de Evidência dos Resultados dos Testes

e) Fase 5: Homologação

Nesta fase, o software é disponibilizado ao cliente para a validação das mudanças solicitadas. Esta disponibilização deve ocorrer em um ambiente intermediário entre o desenvolvimento e a operação, denominado Ambiente de Homologação. A seguir são apresentadas as atividades previstas para esta fase:

- **Preparação do Ambiente de Homologação**: antes de iniciar a homologação é necessário preparar o ambiente para a execução da nova versão. Preparando a rotina de execução, alterando os bancos de dados e outras tarefas que sejam inevitáveis para a boa execução do software, também deve ser instalada a nova versão.

- **Validação Operacional**: após a preparação do ambiente e da instalação da nova versão, deve ser verificado se o software está sendo executado corretamente, quanto a sua parte operacional, isto é necessário, para se evitar a ocorrência de um erro operacional durante a homologação do cliente.

- **Validação Funcional**: após a validação operacional o software será disponível ao cliente para que ele possa proceder a sua validação. O cliente deve criar seus casos de testes ou utilizar informações operacionais para validar a alteração solicitada. Ele deve avaliar as alterações e seus reflexos no software e se o resultado esperado foi atendido. Após a avaliação dos resultados da homologação e estando o software validado pelo cliente, ele deverá proceder a autorização da migração da versão para o ambiente operacional, através de um documento especifico a ser repassado a equipe de operação.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- **Gerente de Manutenção**: é o responsável em acompanhar junto ao cliente a validação do software.

- **Analista de Negócio**: é o responsável em conjunto com o cliente de criar os casos de teste e executar a validação.

- **Analista de Manutenção**: é o responsável em prestar orientação ao cliente das alterações efetuadas.

- **Cliente**: é o responsável em conjunto com o analista de negocio de criar os casos de teste e executar a validação das alterações do software. Também responde pela autorização da migração da nova versão ao ambiente operacional.

O artefato que é gerado nesta fase:

- termo de homologação: documento que deve ser emitido pelo cliente autorizando a operacionalização do software.

Tabela 4.5 - Matriz de Responsabilidade Fase 5

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	**Quem Coordena**	**Quem executa**	**Quem Participa**	**Quem Aprova**	**Artefatos Entrada**	**Artefatos Saída**
Fase 5 – Homologação						
5.1 - Preparação Ambiente de Homologação	Gerente de Manutenção	Analista de Teste e Manutenção	Analista de Teste ,Manutenção e Negocio	Cliente	Modelo de Dados Atualizado e Registro de Requisitos Atualizado	Casos de Validação
5.2 - Validação Operacional	Gerente de Manutenção	Analista de Teste e Manutenção		Analista de Manutenção		
5.3 - Validação Funcional	Gerente de Manutenção	Cliente	Analista de Negocio e Manutenção	Cliente		Termo de Homologação

f) Fase 6: Implantação

Nesta fase o software é implementado em ambiente operacional. Esta migração deve ocorrer do ambiente intermediário para o operacional, com a filosofia de pacotes que contenham todas as versões executáveis dos códigos alterados, visando sempre a integridade do software. Caso ocorra algum problema de catalogação das versões no ambiente operacional, deve-se ter uma rotina de contingência para voltar o software à versão anterior, sem prejuízo ao que está sendo executado em operação.O software será considerado implementado em operação, quando todas as versões dos códigos que foram alterados, forem migradas e estiverem em execução com geração de dados

consistentes com o esperado. A seguir são apresentadas as atividades previstas para esta fase:

- **Preparação do Ambiente Operacional**: após receber a autorização para a instalação da nova versão em ambiente operacional, a equipe de operação deve proceder às alterações necessárias para a instalação da versão. As tarefas são as mesmas executadas na atividade de Preparação do Ambiente de Homologação.

- **Migração do Software**: a instalação da versão em operação deve ser executada em horário definido entre a equipe de manutenção, cliente e equipe de operação, para não causar grandes impactos nas rotinas existentes, que utilizam informações geradas pelo software.

- **Controle de Configuração**: após a instalação da nova versão, deve ser executada uma verificação se todos os programas e versões estão instalados corretamente.

- **Avaliação da Implantação**: após a implantação da nova versão o cliente deve validar o comportamento da nova versão do software em ambiente operacional.

Em cada atividade nesta fase deve envolver os seguintes perfis:

- **Gerente de Manutenção**: é o responsável em acompanhar junto à equipe de operação e do cliente a validação do software.

- **Analista de Manutenção**: é o responsável em prestar orientação a equipe de operação das alterações efetuadas

- **Analista de Operação**: é o responsável em preparar o ambiente operacional, instalar a nova versão do software e acompanhar a sua execução, executando as validações e acertos necessários.

Tabela 4.6 - Matriz de Responsabilidade Fase 6

MATRIZ DE RESPONSABILIDADE						
ATIVIDADES	**Quem Coordena**	**Quem executa**	**Quem Participa**	**Quem Aprova**	**Artefatos Entrada**	**Artefatos Saída**
6.1 - Preparação do Ambiente Operacional	Gerente de Manutenção	Analista Operacional	Analista de Manutenção		Termo de Homologação	
6.2 - Migração das Versões	Gerente de Manutenção	Analista Operacional	Analista de Manutenção			
6.3 - Avaliação da Implantação	Gerente de Manutenção	Cliente e Analista de Manutenção		Cliente		Doc de avaliação da implantação

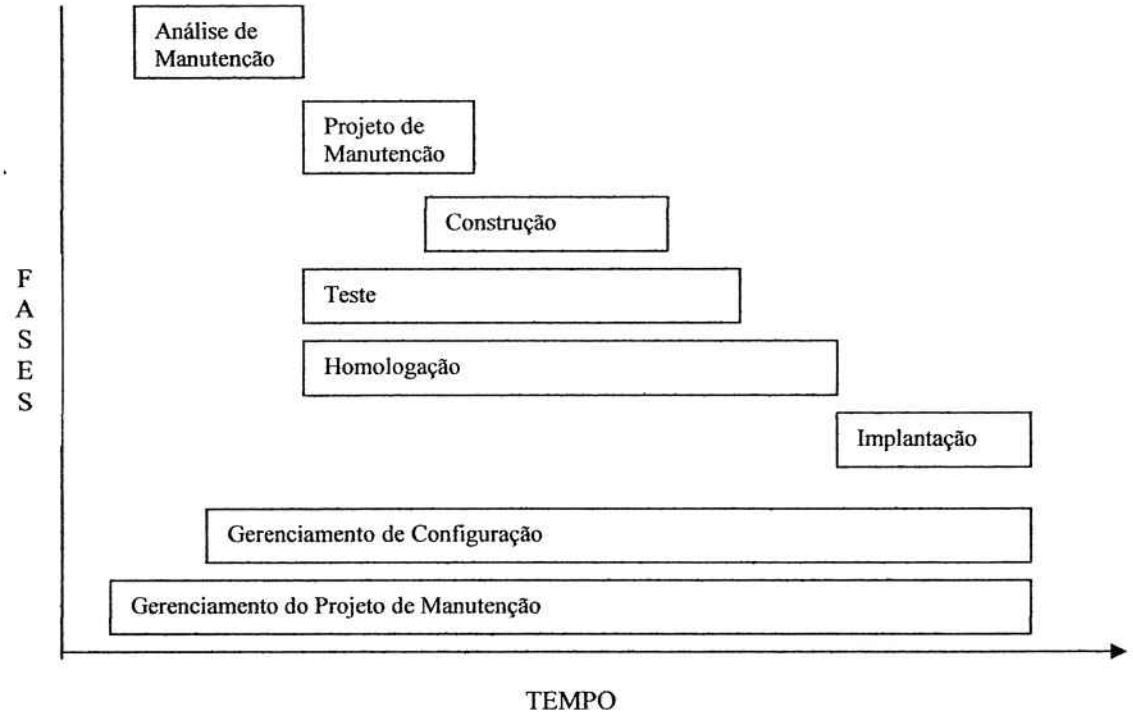


Figura 4.1 – Processo de Manutenção -Fase x Tempo

4.3 – Considerações Finais

Neste capitulo, é apresentada uma proposta para um processo de manutenção, que tem por finalidade estabelecer um roteiro básico, estabelecendo as fases e as atividades a serem executadas durante uma alteração de software. Este processo também deve ser utilizado para auxiliar a gerência no controle dos custos e prazo da manutenção.

O processo proposto é composto por fases que tem dependência, iteração e paralelismo, conforme apresentado na figura 4.1. A fase de Projeto de Manutenção é dependente da fase anterior, Análise de Manutenção, e da aprovação do cliente para sua continuação. A fase Construção pode ser iniciada durante a fase Projeto de Manutenção. As fases Teste e Homologação devem ser iniciadas durante a de Projeto de Manutenção. Assim é apresentado o paralelismo que podem ocorrer entre as fases. A fase de Implantação é dependente direta da fase anterior, Homologação, e da aprovação do cliente.

O acompanhamento das gerências de manutenção e de configuração durante todo o processo tem a finalidade de assegurar a execução das atividades e garantir a integridade das versões nos ambientes.

Este processo foi proposto a partir de um caso de estudo real e da dificuldade encontrada no controle das execuções das tarefas.

5 IMPLEMENTAÇÃO DO PROCESSO DE MANUTENÇÃO

5.1 Considerações Iniciais

O objetivo deste capitulo é apresentar a implementação do processo proposto em uma empresa que desenvolve e mantém software, primeiramente, é apresentada a situação atual da empresa no que se refere à infra-estrutura do ambiente e o nível organizacional, pois ambos interagem no fluxo da execução das alterações dos softwares ou na sua construção.

5.2 Visão Geral do Processo Atual

Para a implementação do processo é necessário fazer uma análise da situação atual da empresa, buscando ressaltar os pontos que merecem melhorias e aqueles que devem ser adaptados ao processo.

5.2.1 Estrutura do Ambiente

Devido à determinação dos Órgãos Regularizadores foi necessária a criação de ambientes separados para desenvolvimento/manutenção, homologação e operação, e foi

também estabelecido um prazo para que todos os softwares legados, isto é, aqueles que já estão em operação, fossem instalados no ambiente de homologação, para que somente pudessem ser validados neste ambiente. Atualmente já existem softwares legados e novos que são validados neste ambiente.

Quando ocorre a implementação de nova versão destes softwares, antes de migrar para o ambiente operacional o mesmo deve ser validado na homologação pelo cliente e somente após a sua autorização é que se migra a versão para a operação. Todos os softwares novos devem ser validados em ambiente de homologação.

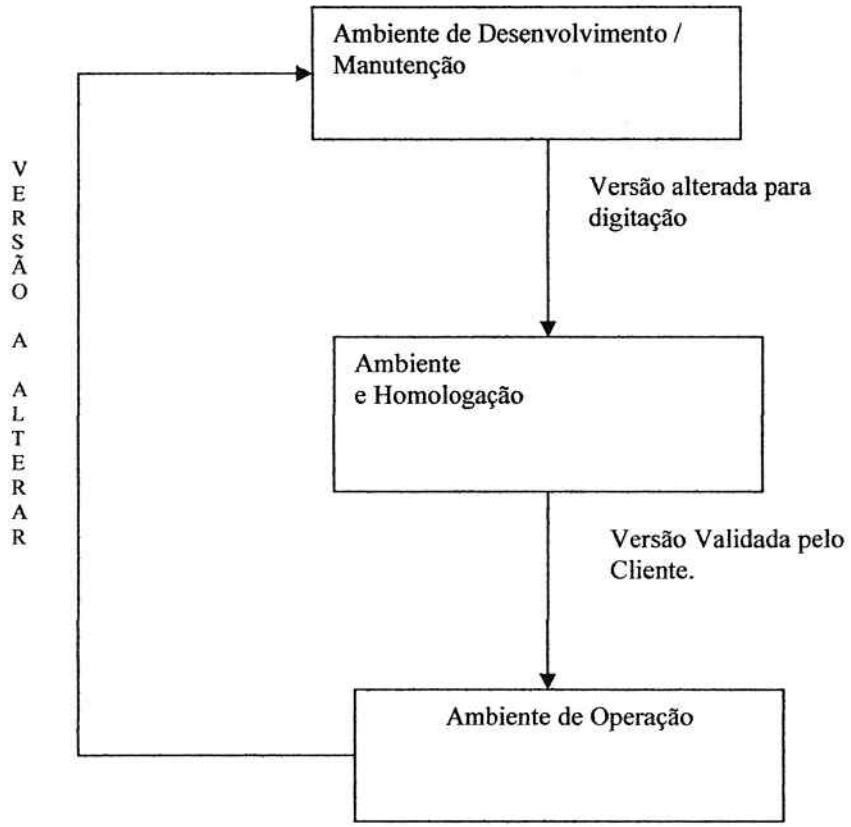


Figura 5.1 - Fluxo da Migração de Versões

As versões dos softwares são controladas por uma ferramenta. A ferramenta mantém um histórico do código fonte com todas as alterações efetuadas, sendo possível obter versões anteriores a atual que está em operação. A versão do software executada em homologação deve ser a mesma que a executada em operação, exceto quando o software está sendo homologado pelo cliente.

Quando é efetuada a manutenção do software, deve ser obtida uma cópia do código fonte da versão que está em operação. Migra-la para o ambiente de desenvolvimento/manutenção e após sua alteração, a versão é transferida para o ambiente de homologação. Depois de validada e autorizada pelo cliente à nova versão é instalada no ambiente operacional, conforme mostra a figura 5.1.

Uma outra determinação dos órgãos regularizadores foi à implementação da documentação de todos os softwares novos e legados. Houve um prazo para que fosse implementada esta documentação e eventualmente auditores destes órgãos selecionam alguns softwares para avaliação.

Para que fosse disseminado este conceito em todas as equipes, foi estabelecida uma metodologia de desenvolvimento de sistemas, que padroniza e estabelece a documentação necessária no desenvolvimento do software.

Esta metodologia está baseada no Modelo Cascata é contém sete fases:

- Anteprojeto: nesta fase, é executada uma pré-análise do software solicitado, gerando um documento que contenha os parâmetros técnicos e estimativas de tamanho, esforço, duração e custo, visando estabelecer um contrato entre o cliente e o desenvolvimento.

- Planejamento: nesta fase, é definido o escopo do projeto, o cronograma de atividades, custos e recursos humanos e análise de risco.

- Análise da Área de Negócio: nesta fase, é efetuado o detalhamento dos documentos gerados na fase anterior.

- Projeto de Sistema de Negócio: nesta fase, são especificados os procedimentos e programas que devem implementar o software.

- Projeto Técnico e Construção do Sistema de Informação: nesta fase, é construído o software.

- Homologação: nesta fase, é efetuada a validação do software antes do mesmo passar para o ambiente de operação.

- Implantação: nesta fase, é instalado o software no ambiente operacional.

Em cada fase são gerados os documentos, conforme tabela 5.1, que ao final do processo comporão a documentação do software. As fases são dependentes, isto é, não se pode iniciar uma fase sem a conclusão da anterior.

Tabela 5.1 - Documentos por Fase

Fases	Documentos Gerados
Anteprojeto	Modelo de Dados Preliminar
	Diagrama de Contexto
	Especificação Preliminar de Requisitos
	Modelo de Arquitetura de Solução
	Cálculo de Estimativa de Tamanho-Preliminar
	Proposta de Solução
Planejamento	Plano de Desenvolvimento de Sistema
	Lista de Requisitos
	Registro de Requisitos
	Protótipo
	Modelo de Dados
	Cálculo de Estimativa de Tamanho
	Cronograma do Projeto
Análise da Área de Negócio	Modelo de Dados Refinado
	Diagrama de Fluxo de Dados
	Plano de Implementação
	Plano de Teste
	Estratégia de Teste
	Cenário de Teste
Projeto do Sistema de Negócio	Layout das interfaces
	Arquitetura de Solução Refinada
	Modelo de Dados Final
	Especificação de Programas
	Roteiros de Testes
	Casos de Testes
Projeto Técnico e Construção	Códigos Fontes
	Bases de Dados
	Manual de Usuário
	Manual de Operação
Homologação do Sistema	Termo de Homologação
Implantação do Sistema	

A metodologia vem auxiliando na melhoria do controle dos projetos. Alguns pontos são importantes e devem ser adaptados para o processo de manutenção, conforme destacamos a seguir:

- Foram criadas equipes de qualidade para executar o acompanhamento dos projetos novos, auxiliando as equipes de desenvolvimento nas atividades da metodologia, na geração da documentação e no controle dos projetos.

- Foi adquirida a ferramenta de controle de configuração, que mantém a integridade das versões dos softwares nos três ambientes: desenvolvimento, homologação e operação.

- Foi estabelecido um processo para validação que é aplicado para os softwares novos e para alguns legados que estão instalados no ambiente de Homologação.

- Foi adquirida uma ferramenta, e estabelecido um processo que auxilia no controle das mudanças das versões dos softwares entre os ambientes de desenvolvimento e operação.

- Foi desenvolvida uma ferramenta para controlar as solicitações dos clientes quanto as alterações no software ou no pedido de novos softwares.

- É utilizada a métrica de Pontos de Função para dimensionar os softwares novos, para os softwares em manutenção dependendo da avaliação da equipe a alteração pode ou não ser pontuada. Já existe uma base histórica para a estimativa dos prazos e dos custos. Quando o software é desenvolvido pela fábrica de software, o custo e prazo são obtidos através das informações que constam no contrato, estabelecido com a empresa contratada para desenvolvimento e manutenção de software.

O processo atual é voltado ao desenvolvimento, sendo que as atividades que se refere à manutenção não é tratada, a seguir é apresentados os pontos que merecem melhorias:

- A metodologia para o desenvolvimento é muito rígida, sendo que suas atividades não são adaptáveis no tratamento da manutenção.

- A equipe de qualidade não auxilia no controle dos projetos de manutenção, com isto não há um controle de custo e prazo no atendimento.

- Não existe uma equipe especializada que auxilie as demais equipes na confecção dos testes de verificação.

- Cada equipe de manutenção trabalha de uma forma diferente, não existindo uma padronização das atividades.

- Não existe um controle quanto à atualização da documentação existente, quando ocorre alteração nos softwares.

A implementação de um processo para a Manutenção auxiliará na melhoria dos seguintes pontos:

- Padronização das atividades em todas as equipes de manutenção da empresa.

- Definição dos documentos que devem ser atualizados.

- Definição dos responsáveis por cada atividade.

- Controle por parte da gerência das atividades que estão sendo executadas.

5.2.2 - Estrutura do Organizacional

A estrutura organizacional da empresa, está relacionada diretamente com o fluxo de trabalho no desenvolvimento e manutenção dos softwares. A organização possui três departamentos:

- Departamento de Análise de Negócio: responsável pelo contato com o cliente e encaminhamento das demandas de manutenção e novos softwares
- Departamento de Desenvolvimento/Manutenção: responsável pelas manutenções nos softwares e desenvolvimento de novos softwares.
- Departamento de Operação: responsável pela execução dos softwares no ambiente operacional e pelo ambiente de homologação.

Cada departamento é responsável por uma parte da demanda, o relacionamento entre eles é regido por Acordos de Nível de Serviço estabelecido entre os mesmos.

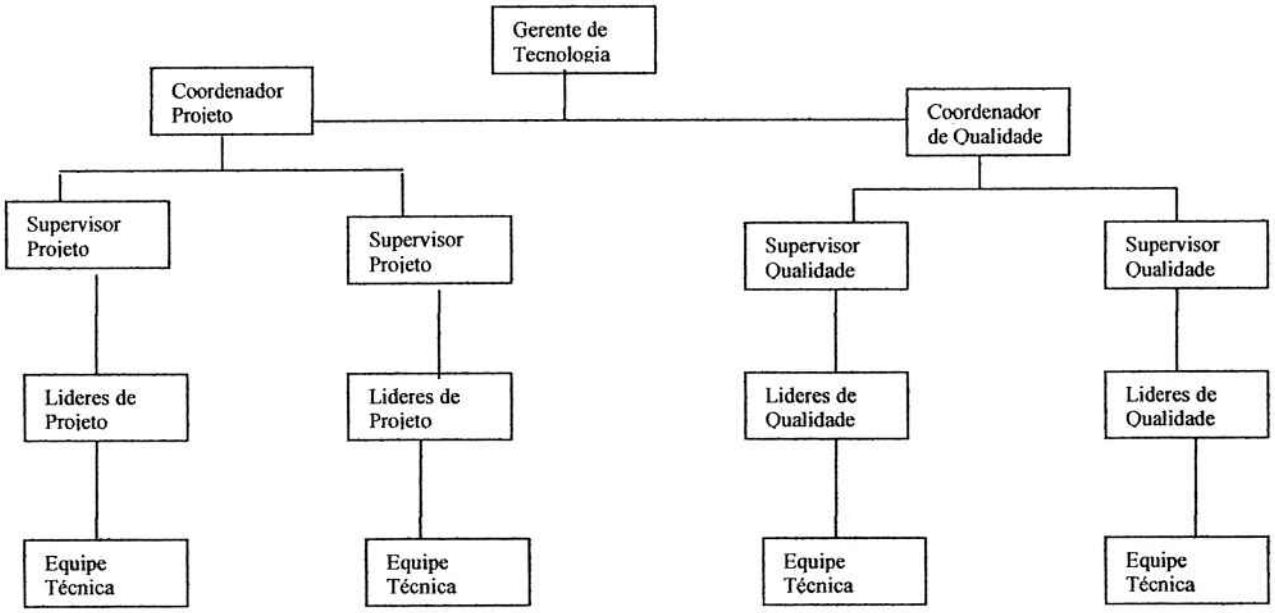


Figura 5.2 - Estrutura Hierárquica.

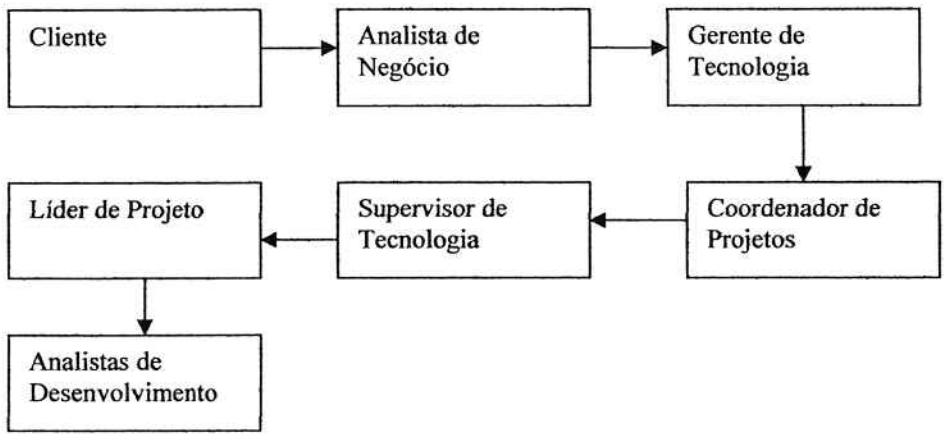
No Departamento de Desenvolvimento/Manutenção são executadas as atividades de alterações e desenvolvimento de software, sua representação hierárquica está apresentada na figura 5.2.

As equipes abaixo do Coordenador de Projeto são responsáveis pelo desenvolvimento de projetos novos e manutenção dos softwares legados.

As equipes abaixo do Coordenador de Qualidade são responsáveis pelo acompanhamento dos softwares novos junto às equipes de projeto e de treinar as equipes na metodologia utilizada.

As equipes técnicas são compostas por analistas e prestadores de serviços, eles são gerenciados pelo Líder de Projeto. Cada Líder de Projeto é responsável por um grupo de software que pode estar em desenvolvimento (softwares novos) ou em manutenção (software legado).

As figuras 5.3 e 5.4 representam o fluxo de encaminhamento das solicitações de desenvolvimento e manutenção de software.



.Figura 5.3 - Fluxo para Softwares novos

O Cliente encaminha para o Analista de Negócio a solicitação de alteração ou novo software, que executa uma pré-avaliação e encaminha para o Departamento de Desenvolvimento/Manutenção utilizando a ferramenta de solicitação de demanda, que tem a finalidade de registrar a solicitação. As equipes somente podem executar uma alteração se receber a solicitação através da ferramenta

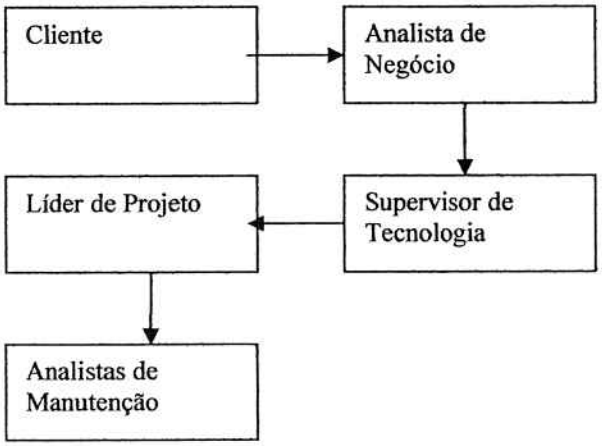


Figura 5.4 - Fluxo para Manutenção de Software

Após o recebimento da solicitação no departamento de desenvolvimento/manutenção, conforme figuras 5.3 e 5.4, ela é encaminhada à equipe técnica que a executa e registra na ferramenta de solicitação, seus entendimentos, prazo e custo da execução. Após a finalização da demanda é solicitado ao cliente que seja efetuada a homologação. Concluindo a homologação o cliente emite o Termo de Homologação que é a autorização para o Departamento de Operação instalar o software.

5.3 Implementação do Processo.

Para a implementação do processo, foi necessário selecionar uma demanda que não implicasse em prejuízo ao negócio, pois o prazo para a execução da manutenção poderia ser maior para a adaptação da equipe ao processo. Também foi selecionado um software que tivesse documentação um pouco mais atualizada, pois em software sem documentação ou com documentação desatualizada, as atividades que executam avaliação da documentação seriam prejudicadas. Outro detalhe importante, é que a equipe escolhida como piloto, possuía integrantes com um bom conhecimento de processo, isto facilitou o entendimento e a aplicação do processo.

Primeiramente foi necessário efetuar uma apresentação do processo proposto a equipe de manutenção, nesta apresentação foram expostas algumas sugestões de melhoria ao processo, mas que não foram adaptados a ele no momento.

Seguindo a estrutura do processo proposto e da adaptação do mesmo com a utilização das ferramentas e rotinas do processo atual, a implementação foi bem aceita e o entendimento da equipe foi adequado. Os resultados apresentados estão a seguir:

- atualização da documentação existente: a equipe avaliou, que a existência de uma atividade especifica para esta tarefa, facilita a compreensão dos desenvolvedores da necessidade de sua execução, principalmente, quando eles precisaram avaliar a demanda a partir dos documentos existentes.

- rastreamento dos códigos fontes a serem alterados: a equipe avaliou, que esta atividade é muito importante, pois evita o esquecimento de algum código no decorrer da execução da alteração.

- testes: a equipe avaliou, que mesmo sem ter uma equipe especializada em teste, esta atividade deve ser sempre executada, independente do tamanho da alteração.

- gerenciamento: o líder pode acompanhar a execução da demanda, identificando as atividades mais complexas, isto é, aquelas onde foram gasto um maior tempo em sua execução. Ele também pode avaliar e controlar os prazos e custos com maior eficácia.

5.4 Considerações Finais.

A empresa citada neste processo de implementação já possui algumas regras que auxiliam na aplicação deste novo processo, o que difere de muitas outras organizações no mercado.

Este processo visa conscientizar as equipes a não somente proceder à alteração na versão do software, mas em todos os artefatos por ele gerados que necessita manter atualizada a versão em operação.

Outro fator importante é a padronização nos procedimentos de uma alteração, estabelecer atividades e controlar sua execução, auxilia para que todas equipes independentes do tipo de software, ou das pessoas envolvidas, executem as mesmas atividades. Assim o nível gerencial, Coordenadores, Supervisores e Lideres, pode controlar as alterações e encaminhar dados reais quando solicitado para outras áreas.

A adaptação do modelo proposto nas atividades que atualmente já são executadas pelas equipes, pode facilitar sua implementação nas demais equipes.

Os resultados esperados com este processo são: melhoria no gerenciamento da alteração; estabelecimento de prazos viáveis e atingíveis, e redução de custo; atualização da documentação com a versão do software em operação; e melhora da qualidade das alterações no software.

6. CONCLUSÃO

Neste capitulo são apresentados os pontos positivos do processo proposto e aqueles que necessita de melhoria, inclusive os apontados pela equipe piloto. Este processo foi elaborado para uma empresa que trabalha com manutenção em ambiente de grande porte, mas pode ser adaptado a outros tipos de empresas que trabalhem em outras plataformas. A aplicação deste processo em outros ambientes pode contribuir para a sua melhoria como acrescentar pontos não abordados no ambiente da proposta.

6.1 Pontos Positivos.

Dentro do objetivo para o desenvolvimento desta proposta, são apresentados a seguir os pontos relevantes obtidos com sua implementação:

- Proporcionou uma clara definição de fases, de papéis e de responsabilidades, além da definição dos documentos gerados e alterados, durante o processo de manutenção.

- Padronização das atividades de manutenção entre todas equipes da empresa.

- Possibilita ao Gerente ou Líder de Projeto uma visão mais clara das atividades de manutenção, podendo assim obter com maior segurança o controle de prazo e custo.

- Destaca pontos que necessitam de controles especiais, como a gerência de configuração.

- Sugere melhorias em pontos como infra-estrutura física para os ambientes de desenvolvimento e operacional.

- Sua concepção foi proposta de forma flexível para ser adaptável em outras organizações.

6.2 Pontos que necessitam de melhoria

Para concluir este estudo, são apresentados alguns possíveis caminhos de melhoria no processo, inclusive os apontados pela equipe piloto, como forma de incentivar trabalhos futuros, que dêem continuidade ao que foi aqui proposto e aplicado:

- O processo propõe a necessidade de atualização da documentação, mas não vincula a isto a manutenção da mesma em um repositório único.

- O processo não apresenta uma atividade que execute a revisão da documentação gerada.

- O processo não apresenta um plano de comunicação entre as equipes das alterações.

- O processo não apresenta uma atividade de medição do software, ao final da manutenção, para a obtenção do tamanho final do software.

Desta forma, é possível que pesquisadores e outros profissionais que trabalhem nesta área, possam avaliar e dar prosseguimento a este trabalho, contribuindo com a Engenharia de Software nos estudos sobre a Manutenção de Software.

REFÊRENCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC-12207, Tecnologia da Informação - Processos de ciclo de vida de software, ABNT, 1998.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126-1, Engenharia de Software – Qualidade de Produto Parte 1: Modelo de qualidade. ABNT. 2003.

AGUIAR, M. Pontos de função ou pontos de caso de uso? Como estimar projetos orientados a objetos. Developers´magazine. V. 7, n. 77. Jan., 2003.

AMBLER,S.;NALBONE,J.;VIZDOS,M.: The Enterprise Unified Process – Extending the Rational Unified Process. Pearson Education. Indiana-USA. 2005.

ARNOLD, M. PEDROSS,P.: Software Size Measurement and Productivity Ratingin a Large-Scale Software Development Department, IEEE, 1998

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. The unified software development: ADDISON-WESLEY. 1998.

BOEHM, B.: Software Engineering Economics, Prentice-Hall, 1981

BOEHM, B.: Software Cost Estimation With COCOMO II. Prentice Hall, Englewood Cliffs, NJ, U.S.A., 2000.

DEKKERS, C. Measuring the 'logical' or 'functional' size of software projects and software application. ISO BULLETIN. May, 2003.

DEKKERS, C. Pontos de Função e Medidas. O que é um Ponto de Função? –no QA Journal, dez de 1998,. Disponível em: <http: www.bfpug.com.br> Acesso em: 21 out 2005

DEMARCO, T., Controle de Projetos de Software, Norma P Carvalho, - Rio de Janeiro, Editora Campus, 1989.

FENTON; PFLEEGER, S. Software metrics: a rigorous & practical approach. Boston: PWS Publishing Company, 1997.

GLASS, R.: Learning to Distinguish a Solution from a Problem, IEEE Software, p.112, 113. MAY/JUNE 2004.

GRUBB, P.;TAKANG, A.: Software Maintenance, 2. ed, World Scientific, 2003

IFPUG. International Function Point Users Group. Function Point Counting Practices Manual: Release 4.2, 2005

JALOTE, P. An integrated approach to software engineering. 2 ed. New York:Springer-Verlag, 1997.

PLEEGER, S.; FRANKLIN, D.: Engenharia de Software: Teoria e Prática. Prentice Hall. 2004.

PRESMMAN, Roger S., Engenharia de software – 5ª ed., McGraw-Hill, 2003.

STEPHEN, S; ROBIN, A.: An Integrated Lyfe-Cicle Model for Software Maintenance, IEEE Transaction Software Engineering, vol 14, 8 AUG 1988.

SILVA, R.P. ENGENHARIA DE SOFTWARE SEGURO, Monografia apresentada para o curso de Especialização em Engenharia de Software da Universidade Candido Mendes, Rio de Janeiro, 2005, disponível em

<www.riosoft.softex.br/arquivos/engenhariasoftwareseguro-renatopessanha.pdf>, em 21 de out 2005.

SOMERVILLE, I. : Engenharia de Software. 6.ed Addison Wesley, 2003.

VAZQUEZ,C.;SIMÕES,G.; ALBERT,R.; Análise de Pontos de Função, 1. ed. São Paulo, Érica, 2003

VEHVILAINEN,R.: What Is Preventive Software Maintenance?. IEEE Computer Society p.18 e 19. 2000.

TELES,V.: Extreme Programam: Aprenda Como Encantar seus Usuários. Novatec. São Paulo. 2004.

ZVEGINTZOV,N.;PARIKH,G. : 60 years of Software Maintenance: Lessons Learned. IEEE Computer Society. 2005.