ANDRÉ FONSECA AMÂNCIO

UMA PROPOSTA DE PROCESSO DE MANUTENÇÃO DISTRIBUÍDA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2008

ANDRÉ FONSECA AMÂNCIO

UMA PROPOSTA DE PROCESSO DE MANUTENÇÃO DISTRIBUÍDA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:

Engenharia e Qualidade de Software

Orientador:

Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS – BRASIL
2008

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Amâncio, André Fonseca

Uma Proposta de Processo de Manutenção Distribuída de Software / André Fonseca Amâncio. Lavras – Minas Gerais, 2008. p. 65.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Manutenção de Software. 2. Desenvolvimento Distribuído. 3. Processos. I. AMANCIO, A. F. II. Universidade Federal de Lavras. III. Uma Proposta de Processo de Manutenção Distribuída de Software.

ANDRÉ FONSECA AMÂNCIO

UMA PROPOSTA DE PROCESSO DE MANUTENÇÃO DISTRIBUÍDA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 18/11/2008

Prof. Dr. Antônio Maria Pereira de Resende

Prof. Dr. Valter Vieira de Camargo

Prof. Dr. Heitor Augustus Xavier Costa

(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

Muita gente acredita que os grandes vencedores possuem um segredo
que lhes proporciona a conquista e o êxito.
Tal segredo, no entanto, todos nós podemos possuir.
Qualquer pessoa que pode chegar aos mais altos lugares,
sempre que reúna os conhecimentos adequados,
uma inabalável firmeza de vontade,
uma grande tenacidade
e muita confiança no próprio valor

[MONTALVÃO, (1979)].

Não consigo encontrar as palavras certas para expressar os sinceros agradecimentos que tenho para todos que me ajudaram nessa fase da minha vida.

Nem consigo citar todos que me ajudaram a concretizar meus objetivos, pois a memória sempre me prega peças.

No entanto, existem algumas pessoas que me acompanharam de perto e me deram força pra continuar.

Em especial agradeço meu Pai pelas lições de realidade e de vida.

Um outro agradecimento especial para minhas duas mãezonas, para a mãe de Lavras que me ensinou a ser como sou e que faz uma comida muito boa e para a mãe de Cuiabá que mesmo longe mora em meu coração.

Um outro especial para minha irmã curuja que é brava mais é super legal.

Outra pessoa especial que eu gostaria de agradecer é meu grande Professor e Orientador, Heitor, pela ajuda na elaboração da monografia e pelo caminho de sucesso e realizações que eu nunca teria conseguido traçar sozinho.

Não posso esquecer de agradecer, a todos os colegas de turma que viveram junto a mim nessa fase de aprendizado que é graduação.

Agradeço aos meus amigos que moram no meu coração e que tomam chimarrão e tereré comigo: Batata, Julinho , Frangu D'agua, Thamior, Lekão, Lorrany, Mangolino, Finganfor,Mario,Toshi, Hommer, Marlon, Rubãum, Bábala, Drops, Bortolino ...

Agradeço as noites de Magic, RPG, Dota e Baladas que me fizeram desligar um pouco da faculdade e esfriar a cachola.

Sei que devo estar esquecendo alguém, mas pode ter certeza que eu também te agradeço pelo convívio que me ensina cada dia mais sobre a vida.

Assim como eu consegui essa conquista desejo muitas realizações a todos.

OBRIGADO A TODOS

SUMÁRIO

Lista de Figuras

Lista de Tabelas

LISTA DE FIGURAS

LISTA DE TABELAS

UMA PROPOSTA DE PROCESSO DE MANUTENÇÃO DISTRIBUÍDA DE SOFTWARE

RESUMO

A área de Engenharia de Software tem sido alvo de novas demandas em função de mudanças decorrentes da internacionalização e, mais profundamente, da globalização na sociedade atual. Na mesma medida em que as organizações distribuem suas operações globalmente, o processo de desenvolvimento de software sofre pressões nesta mesma direção. Este trabalho visa contribuir com o crescimento da Tecnologia de Processos de Software, com a elaboração de um Processo de Manutenção Distribuída que auxilie na gerência de desenvolvimento distribuído de software. Para isso, foram analisados diversas técnicas e métodos que compõem os processos de manutenção de software contidos na norma NBR ISO/IEC 12207, no Processo da IEEE e em demais trabalhos científicos, que contribuíram para o entendimento das principais atividades, atores e artefatos existentes nos processos de manutenção de software. Além disso, literaturas sobre Desenvolvimento Distribuído de Software e o PMBoK foram importantes para o entendimento do desenvolvimento distribuído de software e para o modelo de construção de processo.

Palavras-chave: Processo, Manutenção, Desenvolvimento Distribuído

A PROPOSAL OF A SOFTWARE DISTRIBUTED MAINTENANCE PROCESS

ABSTRACT

The area of software maintenance have been require new demands due changes elapsed by internalization and, more deeply, by the globalization in actual society. In the same pound that organizations distribute their operations around the word, the software development process suffers pressure in the same direction. This work intends to contribute with the growth of software process technology, with the development of distributed maintenance software process that assist with distributed software manager. To the development of the work many techniques and methods of software maintenance, like NBR ISO/IEC 12207, the IEE maintenance process and the massive adaptive process were analyzed. They constructed the knowledge about actors, activists and artifacts of a software maintenance process. Besides, literatures about distributed software development and the PMBoK were important to understanding for the distributed software development and modes of process.

Keywords: Process, Maintenance, Distributed Development

1. INTRODUÇÃO

Em engenharia de software, estudos a cerca de mecanismos para gerenciar processos de software constituem uma subárea denominada Tecnologia de Processos de Software [FREITAS, 2005]. A Tecnologia de Processos de Software envolve um conjunto de linguagens, de métodos, de arquiteturas e de ferramentas para apoiar a gerência de processos de software [TOTLAND; CONRADI, 1995].

1.1. Motivação

Desde o início dos anos 90, uma tendência no desenvolvimento de software tem despertado a atenção dos pesquisadores [CARMEL, 2005]: a distribuição do desenvolvimento. Este fenômeno é um reflexo de mudanças sociais e econômicas, que têm levado organizações a distribuírem geograficamente seus recursos e seus investimentos, visando aumentar a produtividade, melhorar a qualidade e reduzir os custos no desenvolvimento de software [AUDY, 2007].

A distribuição física das equipes agrava problemas existentes e inerentes à gerência do processo de desenvolvimento. Diferenças culturais, de linguagem, de fuso-horário entre outros aspectos aumentam a complexidade na comunicação, na coordenação e no controle durante o processo de desenvolvimento [LANUBILE *et al*., 2003; PILATTI *et al*., 2006].

Embora vários problemas encontrados no desenvolvimento distribuído de software sejam causados por razões sociais e culturais, o uso de recursos tecnológicos pode contribuir para solução dos problemas [MAIDANTCHICK; ROCHA, 2002].

Segundo Pfleeger (2001), a engenharia de software pode ser definida pelo estabelecimento e uso de sólidos princípios de engenharia para obter, economicamente, um software confiável e funcional. Mas, como qualquer área do saber, a engenharia de software envolve aspectos técnicos (próprios da área) e não-técnicos, que agregam conhecimento de áreas complementares.

1.2. Objetivo

O objetivo deste trabalho é apresentar uma proposta de processo de manutenção distribuída. Para isso, foram estudados técnicas e métodos de desenvolvimento distribuído

de software, compilando resultados presentes na literatura, bem como os processos de manutenção tradicionais.

1.3. Metodologia de Desenvolvimento

A metodologia de pesquisa adotada foi, a pesquisa aplicada, com objetivos de caráter exploratório. Foram usados procedimentos operacionais e a fundamentação foi feita através de pesquisas bibliográficas.

1 De acordo com Jung (2004), a finalidade da pesquisa de objetivo exploratório é a descoberta de teorias e de práticas que modificarão as existentes, a obtenção de alternativas ao conhecimento científico convalidado e, principalmente, inovações tecnológicas (produto ou processos).

Foi utilizada a pesquisa bibliográfica que foi realizada no período de março de 2008 a outubro de 2008, iniciando-se por um levantamento bibliográfico, na internet e em bibliotecas digitais e impressas, de artigos científicos relacionados ao tema. O objetivo da pesquisa foi revisar a literatura existente de modo a não redundar o tema de estudo e criar assim um banco de dados sobre as publicações relacionadas ás temáticas do trabalho.

2 De acordo com Jung (2004), o procedimento de pesquisa operacional tem por princípio a investigação de forma sistemática e racional dos processos envolvidos na realização de uma atividade produtiva, com a finalidade de orientar a melhor opção para a tomada de decisões.

A etapa seguinte consistiu em identificar características de Processo de Manutenção de Software analisando a norma NBR ISO/IEC 12207, o processo de manutenção da IEEE e o trabalho de Lucia *et al*. (2001). Logo, a etapa posterior consistiu em analisar as características do Desenvolvimento Distribuído de Software.

A última etapa consistiu em elaborar um Processo de Manutenção Distribuída de Software respeitando as características dos processos co-localizados analisados e acrescentando a eles as características do Desenvolvimento Distribuído de Software.

1.4. Estrutura do Trabalho

O Capítulo 2 discorre sobre processos em um âmbito geral e sua contextualização junto ao processo de desenvolvimento de software. Além disso, uma análise das características de processos de desenvolvimento de software e algumas descrições da manutenção de software são apresentadas contextualizando essas descrições nos ciclos de vida de software.

O Capítulo 3 apresenta a descrição de processos de manutenção. São descritos o processo de manutenção da norma NBR ISO/IEC 12207, o processo de manutenção da IEEE e o processo massivo adaptativo de manutenção.

O Capítulo 4 contextualiza e descreve o desenvolvimento distribuído de software (DDS). São descritas razões que levam ao seu uso e mostrados os seus níveis de dispersão e conceitos relacionados ao DDS. Além disso, os desafios do DDS são apresentados e categorizados. Por fim, é apresentado o modelo MuNNDos.

O Capítulo 5 apresenta resultados e discussões sobre o trabalho. Nele, é apresentada a sugestão do processo de manutenção distribuída de software, mostrando uma visão geral do processo. Por fim, é descrita cada atividade do processo, enfatizando as entradas, as tarefas e as saídas de cada etapa do processo.

O Capítulo 6 apresenta considerações finais sobre o trabalho, mostrando conclusões e contribuições acerca do trabalho desenvolvido e sugestões de trabalhos futuros.

2. PROCESSO DE MANUTENÇÃO

2.1. Considerações Iniciais

Este capítulo apresenta definições sobre processos em um âmbito geral e processos de software. Além disso, ele relata sobre manutenção de software e finaliza com a descrição de processos de manutenção de software.

A seção 2.2 apresenta descrições sobre modelos de processos em um âmbito geral e uma contextualização desses modelos junto ao processo de desenvolvimento de software. Além disso, é realizada uma análise sobre características de processos de desenvolvimento de software e são citados exemplos de abordagens que surgiram para guiar a construção de processo de desenvolvimento de software. A seção 2.3 apresenta descrições sobre manutenção de software com uma visão voltada para a engenharia de software e essas descrições são contextualizadas no ciclo de vida de software. Posteriormente, é feita uma discussão sobre qualidade de software. A seção 2.4 apresenta uma visão geral sobre processo de manutenção de software e descreve as etapas básicas de um processo de manutenção.

2.2. Processo

Esta seção apresenta uma descrição sobre modelos de processos em um âmbito geral e os contextualiza junto ao processo de desenvolvimento de software. Além disso, são apresentadas características sobre processos de desenvolvimento de software e cita exemplos de abordagens que surgiram para guiar a construção de processo de desenvolvimento de software.

2.2.1. Visão Geral

Quando um projeto é criado, mesmo que não seja para um software, sua criação segue uma seqüência de passos para que ele seja concluído. Os passos são usualmente realizados na mesma ordem a cada nova produção. Para Pfleeger (2001), um processo pode ser definido como uma série de passos envolvendo atividades, prazos e recursos que produzem uma saída esperada.

Um processo geralmente envolve um conjunto de ferramentas e técnicas e possui as seguintes características [PFLEEGER, 2001]:

- O processo prescreve as atividades do processo;
- O processo usa recursos, se sujeita a uma série de prazos (seguindo um cronograma) e produz produtos intermediários e finais;
- O processo pode ser composto por sub-processos ligados ao processo de alguma forma. O processo precisa ser definido como uma hierarquia de processos, organizado de forma que cada sub-processo tenha seu próprio modelo;
- Cada atividade do processo tem seus critérios de entrada e saída; assim, é possível saber quando uma atividade começa e termina;
- As atividades são organizadas em seqüência, de modo que fique claro quando uma atividade está relacionada com outra;
- Cada processo possui um guia de princípios que explica os objetivos de cada atividade;
- Prazos e controles podem ser aplicados às atividades, aos recursos ou ao produto. Por exemplo, o prazo pode delimitar o tempo que uma atividade pode gastar ou uma ferramenta pode limitar como um recurso é usado.

Quando um processo envolve a criação de um produto, ele normalmente é referenciado como ciclo de vida. Desta maneira, o processo de software pode ser referenciado como ciclo de vida de software, porque ele descreve a sua vida desde a concepção até a manutenção.

Segundo Pfleeger (2001), os processos são importantes, pois eles acrescentam consistência e estrutura no grupo de atividades. Essas características são úteis quando alguma coisa precisa ser produzida bem e os demais produtos sigam um padrão de produção. Um processo estrutura as ações permitindo examinar, entender, controlar e aprimorar as atividades contidas no processo. Os processos são importantes, pois possibilitam a captura de experiências e permitem que elas sejam transmitidas para futuros do processo.

2.2.2. Processo de Desenvolvimento de Software

Segundo Audy (2007), a essência do uso de processos está na resolução de problemas por meio da criação e da instanciação de descrições de processo. Ao invés de resolver repetidamente um grupo de problemas, é preferível criar uma especificação generalizada da solução e torná-la disponível para futuras resoluções do mesmo grupo de problemas.

Ainda de acordo com Audy (2007), um processo efetivo deve considerar a relação das tarefas necessárias, as ferramentas, os métodos, as habilidades, os treinamentos e a motivação das pessoas envolvidas.

Para Oliveira (2007), a qualidade do processo de software é determinada pelo grau de flexibilidade para incorporar características implícitas de qualidade de produto e novos métodos, técnicas e ferramentas ao processo de desenvolvimento de software. De acordo com Pessoa (2003), este processo é a seqüência de passos para construção de um software e abrange as relações com o fornecedor e com o cliente, gerenciamentos (de projeto, de qualidade, de configuração, de requisitos, de custo, de tempo e de risco) e a engenharia do produto. A qualidade de um software é obtida de forma consistente, em longo prazo, a partir da qualidade do processo.

Sommerville (2003) diz que processos de software são complexos e abrangem número alto de atividades inter-relacionadas. Com isso, melhorar o processo de software de uma empresa não significa simplesmente admitir métodos ou ferramentas específicas ou algum modelo de processo que tenha sido utilizado em outro projeto. Assim como os produtos, os processos têm atributos ou características (Tabela 2 -1).

Tabela 2-1 – Características do Processo (Fonte: Sommerville (2003))

Característica do Processo	Descrição
Facilidade de Compreensão	Até que ponto o processo está explicitamente definido e com que facilidade se pode compreender a definição do processo?
Visibilidade	As atividades de processo culminam em resultados nítidos, de modo que o progresso do processo seja externamente visível?
Facilidade de Suporte	Até que ponto as atividades do processo podem ser ajudadas por ferramentas CASE (*Computer Aided Software Engineering*)?
Aceitabilidade	O processo definido é aceitável e utilizável pelos engenheiros responsáveis pela produção do software?
Confiabilidade	O processo é projetado de tal maneira que erros sejam evitados ou identificados antes que resultem em erros no produto?
Robustez	O processo pode continuar, mesmo que surjam problemas inesperados?
Facilidade de Manutenção	O processo pode evoluir para refletir os requisitos mutáveis da organização ou melhorias de processo identificadas?
Rapidez	Com que rapidez pode ser concluído o processo de entrega de um software, a partir de uma determinada especificação?

Em Liebman (2006), executando as atividades de um processo corretamente, ocorrerá a melhoria do processo. Estas atividades são:

- Análise de processo: estuda os processos existentes e cria um modelo específico para documentar e entender o processo;
- Identificação de melhoria: a melhoria do processo deve visar à eliminação dos problemas que se opõem à qualidade do produto encontrados no estágio anterior de análise de processo. A melhoria também deve expor novos procedimentos, métodos e ferramentas para a resolução destes problemas;
- Introdução de mudanças de processo: novos procedimentos, métodos, ferramentas são estabelecidos e organizados em outras atividades de processo. No entanto, é preciso ter tempo para implantar estas alterações e garantir que sejam conciliáveis às outras atividades do processo, métodos e padrões da empresa;
- Treinamento em mudanças de processo: melhorias impostas sem um treinamento adequado têm resultado contrário à qualidade do produto. Esta fase é considerada essencial ao processo de melhoria;
- Ajuste de mudanças: as alterações serão eficientes após os problemas encontrados com a mudança no processo forem eliminados. A fase de ajuste propõe e aplica novas modificações com os erros gerados na mudança.

Para Oliveira (2007), a implementação de programas de melhoria de processo tem se manifestado como um ponto principal para o sucesso das empresas desenvolvedoras de software. Mas, isto depende do cumprimento com as metas determinadas, da disponibilidade de recursos e do apoio e da participação dos colaboradores da empresa.

A partir da década de 1990, diversas abordagens surgiram para guiar a construção de processo de desenvolvimento de software. Entre elas, estão: i) OPEN [OPEN, 2003]; ii) RUP [KRUCHTEN, 2001]; iii) MSF [MSF, 2007] e iv) as metodologias ágeis (por exemplo, *Extreme Programming* – XP [BECK, 2000]). De forma paralela, a comunidade de engenharia de software também viu surgir abordagens para guiar o gerenciamento desses projetos: i) PMBoK (*Project Management Body of Knowledge*) [PMBOK, 2000]; ii) PRINCE2 [PRINCE2, 2005]; e iii) metodologias ágeis (por exemplo, SCRUM [SCHWABER, 2004]).

Segundo Audy (2007), estas abordagens desempenham papel importante na construção de processos aderentes à realidade da empresa, de forma alinhada com os

objetivos organizacionais e independentes se serem uma abordagem ágil, uma abordagem tradicional ou a união das duas.

2.3. Manutenção de Software

Esta seção apresenta uma descrição de manutenção de software com uma visão voltada para a engenharia de software. São contextualizados os ciclos de vida de software e são apresentados os principais tipos e graus de ocorrência de manutenção de software. Posteriormente, é realizada breve discussão sobre qualidade de software.

2.3.1. Visão da Manutenção

Para Costa (2005), o processo de desenvolvimento de software está completo quando o software foi entregue e está sendo usado pelos usuários. Qualquer mudança, após o software estar operacional, é considerada manutenção. Como mudanças são inevitáveis ao longo da sua vida, mecanismos devem ser previstos para avaliar, controlar e fazer essas modificações.

Existe, ainda, a idéia equivocada da manutenção de software ser semelhante à manutenção de hardware que consiste em substituir peças gastas e/ou usar técnicas para prolongar a sua vida útil. No entanto, software não degrada com o tempo e a manutenção de software está associada com alteração do software [COSTA, (2005)].

A manutenção é considera a fase mais dispendiosa do ciclo de vida de um software sendo que, muitas vezes, a qualidade do código-fonte reparado e atualizado pode ser baixa e comprometer o seu desempenho. A demanda pelo esforço e tempo decorre, principalmente, da falta de disciplina e de controle nas atividades iniciais do processo de desenvolvimento de software. As decisões não adequadas de projeto podem influenciar negativamente no software e, conseqüentemente, aumentar a complexidade de sua manutenção [ARAÚJO (1993)].

De um modo geral, pode-se organizar o processo de desenvolvimento de um software a partir de três grandes fases [SOMMEVILLE (2000); PRESSMAN (2006); PFLEEGER (2001); EISENMANN *et al*. (2005)]:

- Definição. Esta fase está associada à determinação do que vai ser feito. Assim, o profissional encarregado do desenvolvimento de software deve identificar as informações manipuladas, as funções processadas, o nível de desempenho desejado, as

interfaces oferecidas, as restrições do projeto e os critérios de validação. Isto é feito não importando o modelo de desenvolvimento adotado e independente da técnica utilizada para fazê-lo. Esta fase é caracterizada pela realização de três atividades:

- Análise (ou Definição) do sistema. Permite determinar o papel de cada elemento (hardware, software, equipamentos, pessoas), cujo objetivo é determinar, como resultado principal, as funções atribuídas ao software;
- Planejamento do projeto de software. A partir da definição do escopo do software, são feitas uma análise de riscos e a definição dos recursos, dos custos e da programação do processo de desenvolvimento;
- Análise de requisitos. Permite determinar o conjunto das funções realizadas e as principais estruturas de informação processadas;

- Desenvolvimento. Nesta fase, é determinado como realizar as funções do software. Aspectos como a arquitetura do software, as estruturas de dados, os procedimentos implementados, a forma como o projeto será transformado utilizando linguagens de programação, a geração de código e os procedimentos de teste. Normalmente, esta fase é organizada em três principais atividades:
    - Projeto de software. Os requisitos do software definidos na fase anterior são traduzidos em um conjunto de representações gráficas, tabulares ou textuais. Estas representações (diversas técnicas de representação podem ser adotadas em um mesmo projeto) permitem definir, com alto grau de abstração, aspectos do software como arquitetura, dados, lógica de comportamento (algoritmos) e características da interface;
    - Codificação. O mapeamento das representações realizadas na etapa de projeto é feito em uma ou em várias linguagens de programação, a qual será caracterizada por um conjunto de instruções executáveis no computador; considera-se a geração de código-fonte obtido a partir do uso de ferramentas (compiladores, linkers, etc.) e executado pelo hardware do sistema;
    - Testes de software. Uma bateria de testes no software é realizada para verificar e corrigir defeitos relativos às funções, à lógica de execução, às interfaces, etc.;
- Manutenção. Esta fase, que se inicia a partir da entrega do software, é caracterizada pela realização de alterações das mais diversas naturezas: i) para corrigir erros residuais da fase anterior; ii) para incluir novas funções exigidas pelo cliente; e iii) para adaptar

o software a novas configurações de hardware. Sendo assim, esta fase é categorizada em três principais atividades:

- Correção. Erros observados são corrigidos durante a operação do software;
- Adaptação. Alterações no software são realizadas para que ele seja executado em um novo ambiente (arquitetura, novos dispositivos de hardware, novo sistema operacional, etc.);
- Melhoramento funcional. Alterações são realizadas para melhorar aspectos do software, por exemplo, desempenho, interface, introdução de funções, etc.

Por outro lado, a norma NBR ISO/IEC 12207 – Processos do Ciclo de Vida do Software – estabelece um ciclo de vida padrão composto pelos seguintes processos fundamentais [NBR ISO/IEC 12207, 1998]: i) aquisição; ii) fornecimento; iii) desenvolvimento; iv) operação; e v) manutenção. Dessa forma, em quaisquer das duas definições de etapas de processo de desenvolvimento de software, a fase de manutenção é ativada quando o software é submetido a modificações no código-fonte e na documentação associada, em decorrência a um problema ou à necessidade de melhoria ou adaptação. O objetivo é modificar o software existente, preservando a sua integridade [BRUSAMOLIN, 2004].

Em Costa (2005), são apresentadas análises sobre os tipos de manutenção de software: corretiva, adaptativa, perfectiva e preventiva. A Tabela 2 -2 apresenta a descrição de cada tipo de manutenção e a Figura 2 -1 mostra o percentual de ocorrências de cada tipo.

Tabela 2-2 – Tipos de Manutenção (Fonte: Pressman (2006))

Tipo de Manutenção	Motivo da Manutenção
Adaptativa	Realizada quando o software precisa ser adaptado às novas tecnologias (hardware e software) implantadas no ambiente operacional.
Corretiva	Realizada quando são corrigidos erros não identificados durante o teste.
Evolutiva	Realizada quando o software deve englobar novos requisitos ou melhorias decorrentes da evolução na tecnologia de implementação empregada.
Preventiva	Realizada quando o software é alterado para aumentar sua manutenibilidade e/ou confiabilidade. Este tipo de manutenção é relativamente raro em ambientes de desenvolvimento. Modificações realizadas neste tipo de manutenção não afetam o comportamento funcional do software.

Uma vez que o processo de manutenção é definido como uma parte fundamental do ciclo de vida do software e, a partir dele, é possível realizar correções, adaptações, aperfeiçoamentos e prevenções de erro em um software garantindo sua continuidade. Dessa forma, deve-se ter a constante preocupação com a manutenibilidade de software ao longo do seu processo de desenvolvimento de modo a garantir a qualidade do processo de manutenção.

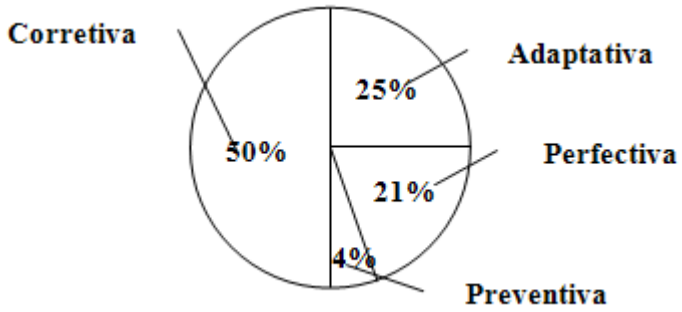

Figura 2-1 – Percentual dos Tipos de Manutenção (Fonte: Pfleeger (2001))

2.3.2. Qualidade de Software

Em Costa (2005), os modelos de qualidade de software foram propostos por causa da chamada Crise do Software, pois os gastos demandados pela manutenção de software eram e continuam sendo bem maiores que os gastos para o desenvolvimento de um software.

Segundo Vasconcelos; Maciel (2003), os avanços tecnológicos, a crescente preocupação na eliminação de defeitos, o aumento da produtividade e a redução de custos motivaram o surgimento de modelos de qualidade para o processo de manufatura. A partir da década de 60, começaram a surgir critérios, modelos e técnicas para a garantia da qualidade no processo de produção. De acordo com estes mesmos autores, diversos modelos de qualidade de software vêm sendo propostos ao longo dos últimos anos. Esses modelos têm sido fortemente adotados por organizações em todo o mundo.

Conforme Costa (2005), a literatura oferece várias definições de qualidade. A seguir, são apresentadas algumas destas definições:

- Crosby (1979) diz que qualidade significa conformidade do produto com os seus requisitos, que devem estar bem definidos, a fim de não serem mal-interpretados;
- Deming (1982) diz que qualidade é um grau previsível de uniformidade e dependência, baixo custo e satisfação no mercado, ou seja, qualidade é sempre aquilo que o cliente

necessita e quer. Controle estatístico da qualidade, participação do trabalhador no processo de decisão e limitação das fontes de fornecimento são os passos para a sua abordagem [CÔRTES; CHIOSSI, 2001];

- Feigenbaum (1994) diz que qualidade é uma estratégia que requer percepção de todos na empresa. Além disso, ele diz que é um compromisso para com a excelência, um modo de vida corporativa, um modo de gerenciamento. A liderança para a qualidade, a tecnologia moderna da qualidade e o compromisso organizacional são os passos da sua abordagem [CÔRTES; CHIOSSI, 2001];
- ISO Std. 8402 (1990) caracteriza o termo qualidade como a capacidade de satisfazer as necessidades implícitas e explícitas dos seus usuários;
- Juran; Gryna Jr. (1970) dizem que qualidade significa conveniência para o uso, ou seja, os requisitos e as expectativas dos clientes devem ser considerados, uma vez que cada cliente pode usar o mesmo produto de maneiras diferentes;
- Rothery (1993) diz que qualidade significa adequação ao uso conforme as exigências, ou seja, o produto em questão deve realizar de maneira adequada a sua função.

Segundo Rocha (2002), uma organização de bom desempenho gasta 80% de seu esforço na prevenção de problemas, enquanto uma organização de baixo desempenho gasta 90% de seu tempo corrigindo sintomas em vez de causas de problemas. Dessa forma, sua definição de qualidade refere-se a um conjunto de características a serem satisfeitas em um determinado grau, de modo que o produto de software satisfaça às necessidades de seus usuários.

Considerando o contexto de software, o termo qualidade recebeu uma definição especial na norma ISO/IEC 9126 [ISO Std. 9126, 1991; ISO Std. 9126 , 2001; ABNT NBR13596, 1996], na qual a qualidade é a totalidade das características de um software, que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas. As necessidades explícitas são requisitos para o desenvolvimento de software explicitamente sugerido pelo cliente ou obtidos na modelagem do problema. Por outro lado, as necessidades implícitas podem não ser explicitadas nem documentadas, mas devem estar presentes quando o software é usado em condições particulares; elas são necessidades subjetivas dos usuários, inclusive operadores, destinatários dos resultados do software e os mantenedores. Estas necessidades, também chamadas de fatores externos, podem ser percebidas pelos desenvolvedores e usuários. Outra denominação utilizada trata da

qualidade em uso e deve permitir a usuários atingir metas com efetividade, produtividade, segurança e satisfação em um contexto de uso especificado [GOMES, 2001].

No entanto, os problemas relacionados com a Engenharia de Software e com o gerenciamento de qualidade decorrem da situação do termo qualidade possuir diferentes definições, conduzindo a mal-entendidos [TOLEDO, 1987]. Conforme Kan (1995), isto pode ocorrer, pois:

- o termo qualidade possui um único sentido, mas com conceitos multidimensionais – a quem se interessa, os seus pontos de vista e os atributos de qualidade relevantes para cada um deles;
- para cada conceito, há níveis de abstrações diferentes, ou seja, um grupo de pessoas pode referir-se à qualidade com sentido geral e outro grupo referir-se com sentido específico;
- o termo qualidade está presente na linguagem diária das pessoas. Assim, a visão popular pode entrar em conflito com a visão profissional – visão da Engenharia de Software e visão do gerenciamento da qualidade. Na visão popular, o termo qualidade relaciona-se à classe e à elegância; enquanto, na visão profissional, o termo qualidade caracteriza funcionamento simples e barato.
- No escopo deste trabalho, a definição de qualidade de software contida na norma ISO/IEC 9126 [ISO Std. 9126, 1991; ISO Std. 9126, 2001; ABNT NBR13596, 1996] foi usada como referência, pois apresenta caráter geral e altamente conceitual, habilitando ser aplicada a uma grande variedade de ambientes e condições.

No escopo deste trabalho, a definição de qualidade de software contida na norma ISO/IEC 9126 [ISO Std. 9126 (1991); ISO Std. 9126 (2001) e ABNT NBR13596 (1996)] será utilizada como referência, pois ela tem caráter geral e altamente conceitual, o que a habilita a ser aplicada a uma grande variedade de ambientes e condições.

2.4. Processo de Manutenção

O processo de manutenção de software é diferente do processo de desenvolvimento em muitos aspectos. A atividade de manutenção é baseada em um software existente e ocorre para aprimorá-lo. As duas tarefas principais são entender o que está errado e adicionar funções ou corrigir anomalias. A atividade de manutenção é considerada como

não interessante por desenvolvedores que preferem realizar tarefas de desenvolver software [ROBILLARD *et al*., 2007].

Entender as atividades de manutenção de software permite identificar pontos críticos e tomar ações apropriadas para aumentar a eficiência do processo de manutenção. Este processo pode ser simples, no qual os processos que o compõem compartilham as mesmas atividades conceituais [ROBILLARD *et al*. 2007]: i) identificar anomalias; ii) encontrar a causa; iii) implementar as modificações; iv) testar o módulo modificado; e v) integrar o módulo modificado no software existente.

A Figura 2 -2 descreve as etapas básicas de um processo de manutenção e os principais atores envolvidos. O Gerente de Requisitos (*Request Manger*) (1) está relacionado à atividade de detectar anomalias e é responsável por estimar custos e priorizar atividades. Este trabalho requer conhecimento prévio do domínio da aplicação, representado pelas setas acima da regra (1) vinda da nuvem do domínio, e interage com o Cliente (*Client*) (C). Essa interação é representada por uma seta de duplo sentido ligando o Cliente (C) ao Gerente de Requisitos (1).

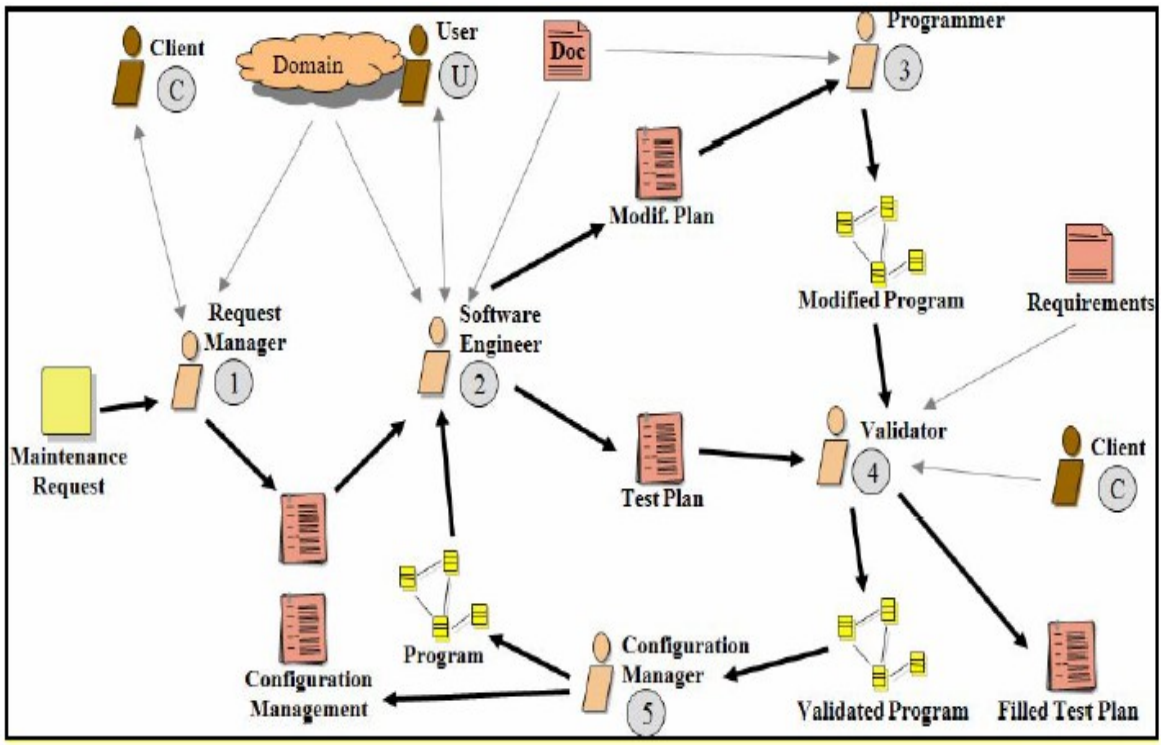

Figura 2-2 – Processo de Manutenção (Fonte: Robillard *et al*. (2007))

O Engenheiro de Software (*Software Enginner*) (2) recebe a lista de manutenção, analisa as alterações requeridas e projeta os módulos aprimorados. Para realizar essa tarefa,

ele precisa analisar a documentação existente (Doc) para entender o domínio da aplicação e, algumas vezes, falar com os usuários. Seu trabalho resulta em um plano de modificação e um plano de testes para o módulo modificado. O Programador (*Programmer*) (3), baseado no plano de modificação e na documentação, realiza as alterações e produz um módulo modificado. O Validador (*Validator*) (4), normalmente chamado de testador, testa o programa baseado no plano de testes e nos requisitos. O Validador produz um programa de validação e preenche o plano de testes. O Gerente de Configurações (*Configuration Manager*) (5) é responsável por integrar o programa validado no software e realizar os registros das novas configurações do software.

2.5. Considerações Finais

Neste capítulo, uma descrição de processos, enfatizando os processos de manutenção de software, e uma visão sobre manutenção na engenharia de software e sobre qualidade de software foram apresentadas.

Analisando a situação dos modelos de processo, mesmo que não exista um processo de software ideal, existe um escopo para aprimorar o processo de produção em muitas organizações. Processos podem adotar técnicas antigas ou utilizar as mais avançadas práticas da indústria de software. Mesmo assim, muitas empresas ainda não adotam processos de engenharia de software em sua produção [SOMMERVILLE, 2007].

Em uma visão mais geral, o processo serve como um guia para o desenvolvimento de produtos. No mesmo sentido, os processos de software servem para equipes da área computacional como guias para desenvolver software. Vale ressaltar que, além de um guia, os processos contribuem para a garantia da qualidade do produto e permite a transmissão de conhecimentos e experiências para os próximos produtos desenvolvidos a partir dele.

Além disso, como parte das referências sobre manutenção, Canning (1972) compara a manutenção de software a um iceberg, pois se espera que o que é imediatamente visível seja tudo que existe. Entretanto, enorme massa de possíveis problemas e custos fica sob a superfície, isto é, a tarefa de manutenção não se resume apenas à manutenção por si só, mas ao aparato que envolve esta tarefa.

Dessa forma, a tarefa de manutenção tem caráter de equipe e poderia ser mais fácil e efetiva se conseguisse melhor conjeturar as possíveis fontes de erros (falhas). Conforme

Pfleeger (2001), os pesquisadores estão em busca de novos modelos de qualidade para mostrar as ligações entre software e seus processos. Dessa forma, estes modelos ajudarão a identificar quando o esforço deve ser usado para manter um software e quando é apropriado descartá-lo.

3. TIPOS DE PROCESSOS DE MANUTENÇÃO DE SOFTWARE

3.1. Considerações Iniciais

Neste capitulo, são descritos processos de manutenção de software, enfatizando o seu fluxo de atividades e a descrição de seu funcionamento. Segundo Audy (2007), o processo de desenvolvimento de software é representado por um modelo e esse modelo é operacionalizado por uma metodologia. Existem diversos modelos e metodologias sendo que a metodologia atua estabelecendo a seqüência das atividades, como elas se relacionam entre si e identifica o momento em que os métodos e as ferramentas serão utilizados.

As metodologias atuam como mecanismos que permitem o controle de atividades. Elas podem atuar em diversas áreas do desenvolvimento do software. Em se tratando da área manutenção de software, estes mecanismos são denominados processos de manutenção de software e contém as atividades e as tarefas do mantenedor. Estes processos são ativados quando o software é submetido a modificações no código e na documentação associada por causa da necessidade de melhoria, da adaptação ou do aperfeiçoamento. O objetivo é modificar o software existente, preservando a sua integridade [NBR ISO/IEC 12207, 1998].

A seção 3.2 descreve sucintamente o processo de manutenção da norma NBR ISO/IEC 12207. A seção 3.3 apresenta brevemente o processo de manutenção da IEEE. A seção 3.4 aborda resumidamente o processo massivo adaptativo de manutenção.

3.2. Processo de Manutenção da Norma NBR ISO/IEC 12207

A Figura 3 -3 mostra o fluxo de atividades do processo de manutenção da norma NBR ISO/IEC 12207 [NBR ISO/IEC 12207, 1998], apresentando uma síntese de suas atividades. Para cada atividade, estão associados os artefatos de entradas (gravura branca – lado esquer-do), o nome (retângulo azulado), os atores (boneco à esquerda) e os artefatos de saídas (gravuras cinza – lado direito). O processo de manutenção da norma NBR ISO/IEC 12207 contém as seguintes atividades:

- Implementação do processo. O objetivo é preparar, documentar e executar um plano de manutenção. Esta atividade possui as seguintes tarefas:

- O mantenedor deve desenvolver, documentar e executar planos e procedimentos para a condução das atividades e tarefas do processo de manutenção;

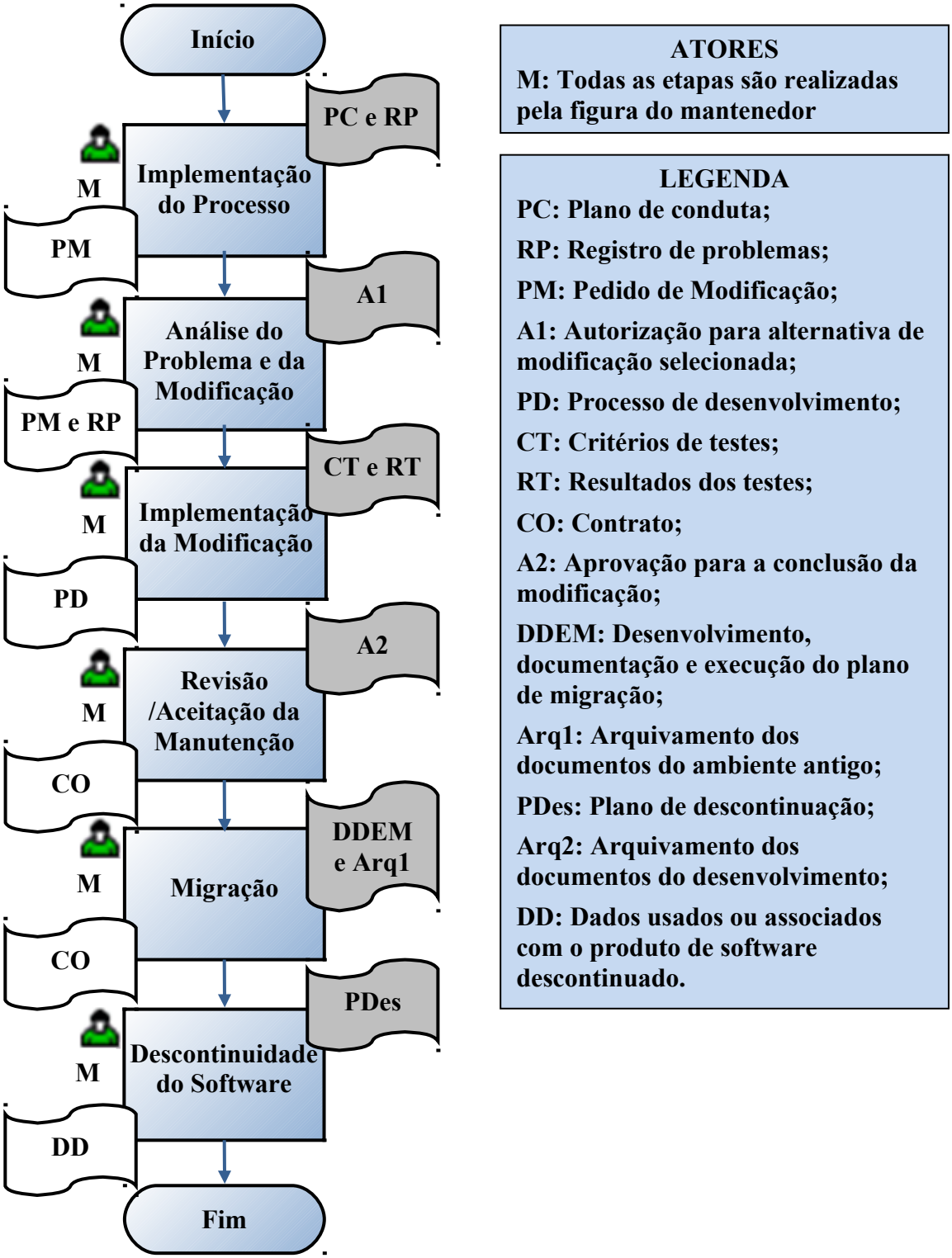

Figura 3-3 – Fluxo de Atividades do Processo de Manutenção da Norma NBR ISO/IEC 12207

    - O mantenedor deve estabelecer procedimentos para receber, registrar e rastrear relatórios de problemas e pedidos de modificação dos usuários e prover realimentação (*feedback*) aos usuários. Sempre que problemas forem encontrados, eles devem ser registrados e incluídos no processo de resolução de problemas;
    - O mantenedor deve implementar (ou estabelecer interface organizacional com) o processo de gerência de configuração, para gerenciar modificações no software existentes;
- Análise do problema e da modificação. O objetivo é analisar o Pedido de Manutenção e o Relatório do Problema, verificando o impacto da manutenção na empresa, no sistema e nos sistemas com os quais interage. Além disso, deve-se determinar o tipo de manutenção, e o alcance e as conseqüências das alterações e propor soluções para implementar a modificação. Esta atividade possui as seguintes tarefas:
    - O mantenedor deve analisar o relatório de problemas ou pedido de modificação segundo o seu impacto na organização, no sistema existente e nos sistemas com os quais interage, com relação ao seguinte: i) tipo: corretivo, melhoria, preventivo ou adaptativo; ii) escopo: tamanho da modificação, custo envolvido, prazo para modificá-la; e iii) criticidade: impacto no desempenho, proteção ou segurança;
    - O mantenedor deve produzir ou verificar o problema;
    - Baseado na analise, o mantenedor deve desenvolver alternativas para a implementação da modificação;
    - O mantenedor deve documentar o problema/pedido de modificação, resultados da análise e as alternativas de implementação;
    - O mantenedor deve obter aprovação para a alternativa de modificação selecionada, conforme especificado no contrato;
- Implementação da modificação. O objetivo é realizar alterações conforme recomendações da engenharia de software aplicáveis durante o desenvolvimento. Esta atividade possui as seguintes tarefas:
    - O mantenedor deve conduzir a análise e determinar que documentação, unidade de software e versões destas necessitam ser modificadas. Estas devem ser documentadas;

    - O mantenedor deve utilizar o processo de desenvolvimento para implementar as modificações. Os requisitos do processo de desenvolvimento devem ser complementados, como se segue: i) devem ser definidos e documentados critérios testes e de avaliação para testar e avaliar as partes modificadas do sistema (unidades de software, componentes e itens de configuração); ii) deve ser garantida a implementação completa e correta dos requisitos novos e dos modificados; e iii) deve ser garantido que os requisitos originais não modificados não afetados. Os resultados dos testes devem ser documentados;
- Revisão/Aceitação da manutenção. O objetivo é garantir que as alterações sejam realizadas satisfatoriamente de acordo com o que foi especificado no contrato. Esta atividade possui as seguintes tarefas:
    - O mantenedor deve conduzir revisão(ões) com a organização que autorizou a modificação para determinar a integridade do sistema modificado;
    - O mantenedor deve obter aprovação para a conclusão satisfatória da modificação, conforme especificado no contrato;
- Migração. O objetivo é preparar e realizar um plano de migração caso seja necessário migrar o software para outra plataforma. Esta atividade possui as seguintes tarefas:
    - Se um sistema ou software (incluindo dados) é migrado de um ambiente de operação antigo para um novo, deve ser assegurado que qualquer software ou dados produzidos ou modificados durante a migração estejam de acordo com esta norma;
    - Um plano de migração deve ser desenvolvido, documentado e executado. As atividades de planejamento devem incluir os usuários. Os itens incluídos no plano devem conter: i) análise e definição de requisitos de migração; ii) desenvolvimento de ferramentas de migração; iii) conversão de software e dados; iv) execução da migração; v) verificação da migração; e vi) suporte ao ambiente antigo;
    - Usuários devem receber notificações dos planos e atividades de migração, ou seja, eles devem receber: i) explicação do porquê o ambiente antigo não será mais atualizado; ii) descrição do novo ambiente com sua data de disponibilização; e iii) descrição de outras opções de opções disponíveis, se existirem, uma vez que o apoio ao ambiente antigo seja descontinuado;

- Operações paralelas dos ambientes antigo e novo podem ser conduzidas para a transição gradual ao novo ambiente. Durante este período, deve ser fornecido treinamento necessário, conforme especificado no contrato;
- Quando a migração programada ocorrer, devem ser enviadas notificações aos interessados. A documentação, os históricos (*logs*) e o código-fonte associados ao ambiente antigo devem ser arquivados;
- Após a migração, uma revisão deve ser executada para avaliar o impacto da mudança para o novo ambiente. Os resultados da revisão devem ser enviados às autoridades apropriadas para informação, orientação e providências;
- Dados utilizados ou associados ao ambiente antigo devem estar acessíveis, de acordo com os requisitos do contrato, para preservação e autoria dos dados.

- Descontinuidade do software. O objetivo é preparar e realizar um plano de descontinuação[1] do sistema que envolve, entre outras, a interrupção total ou parcial do apoio após um determinado período de tempo, o arquivamento do produto e da documentação associada e a transição para um novo produto, se aplicável. Esta atividade possui as seguintes tarefas:
    - Um plano de descontinuação, para remover o apoio ativo pelas organizações responsáveis pela operação e manutenção, deve ser desenvolvido e documentado. As atividades de planejamento devem incluir os usuários. O plano deve conter os seguintes itens: i) cessação total ou parcial de apoio após certo período de tempo; ii) arquivamento do software e sua documentação associada; iii) responsabilidade por quaisquer questões futuras de suporte residual; iv) transição para o novo software, se aplicável; e v) disponibilidade de cópias de arquivos de dados;
    - Os usuários devem receber notificações dos planos e das atividades de descontinuação. Notificações devem incluir: i) descrição da substituição ou atualização com sua data de disponibilidade; ii) explicação do porquê o software não receberá mais atualizações; e iii) descrição de outras opções de apoio disponíveis, uma vez que não terá mais atualizações;
    - Operações paralelas do software em descontinuação e do novo deveriam ser conduzidas para transição gradual ao novo software. Durante este período, deve ser provido treinamento de usuário, conforme especificado no contrato;

[1] O software deverá ser descontinuado a pedido do proprietário.

- Quando a descontinuação programada ocorrer, devem ser enviadas notificações aos interessados. A documentação, os históricos (*logs*) e o código-fonte associados ao desenvolvimento deve ser arquivados, quando apropriado;
- Dados utilizados ou associados ao software descontinuado devem ser acessíveis, de acordo com os requisitos do contrato, para preservação e auditoria dos dados.

3.3. Padrão de Processo IEEE

O processo de manutenção de IEEE [IEEE, 2002] se divide em seis atividades principais: i) Identificação, Classificação e Priorização do Problema/Modificação; ii) Análise; iii) Projeto (*Design*); iv) Implementação; V) Regressão/Teste de Sistema; vi) Entrega. A Figura 3 -4 apresenta a convenção adotada pela IEEE para representar as atividades que descrevem entradas, controle, processos associados e saída.

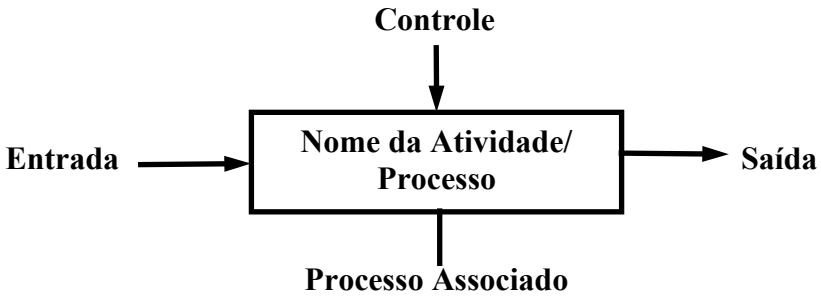

Figura 3-4 – Representação das Atividades do Processo de Manutenção da IEEE

Segundo Martins (1998), o processo da IEEE é um processo orientado a negócios (*business*) e suas atividades e seus fluxos de dados são apresentados na Figura 3 -5.

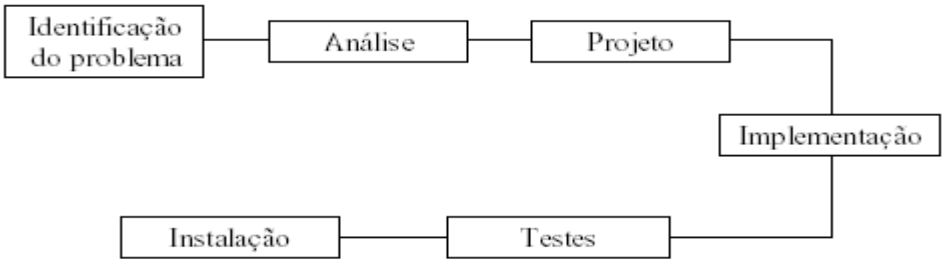

Figura 3-5 – Fluxo de Atividades do Processo da IEEE

O processo de manutenção da IEEE é composto pelas seguintes atividades [IEEE, 2002]:

- Identificação do Problema. O artefato de entrada é Pedido de Manutenção. Os processamentos são: i) Atribuir id ao Pedido; ii) Classificar Quanto ao Tipo de Manutenção; iii) Determinar Prioridade; e iv) Encaminhar Pedido ao Grupo de Controle de Configuração. O controle é Cadastrar Pedido no Banco de Dados. Os artefatos de saída são: i) Relatório de Rejeição da Alteração; ii) Aviso de Incorporação ao Banco de Dados; e iii) Pedido de Manutenção Validado;
- Análise. Os artefatos de entradas são: i) Pedido de Manutenção Validado; e ii) Registro do Banco de Dados. Os processamentos são: i) Analisar Impacto da Alteração; ii) Estudar Alternativas de Solução; iii) Avaliar Custos, Recursos e Prazos; e iv) Decidir se Aceita ou Não o Pedido. O controle é Analisar Relatório de Avaliação do Pedido. Os artefatos de saídas são: i) Relatório de Avaliação do Pedido; ii) Relatório de Rejeição; e iii) Proposta de Alteração;
- Projeto. Os artefatos de entradas são: i) Relatório de Avaliação do Pedido; ii) Proposta de Alteração; e iii) Itens da Linha Básica a serem Alterados (Modelos e Documentos). Os processamentos são: i) Alterar Modelos e Documentos; ii) Preparar Casos de Teste para as Alterações; iii) Identificar Testes de Regressão Aplicáveis; e iv) Atualizar Proposta de Alteração. O controle é Revisar Proposta de Alteração e Itens Alterados. Os artefatos de saídas são: i) Proposta de Alteração Revisada; ii) Descrição dos Testes; e iii) Itens Alterados;
- Implementação. Os artefatos de entradas são: i) Itens Alterados (Modelos e Documentos); ii) Proposta de Alteração Revisada; e iii) Código dos Itens a Serem Alterados. Os processamentos são: i) Alterar o Código; ii) Realizar Testes de Unidade; iii) Realizar Testes de Integração; e iv) Atualizar Modelos e Documentos. O controle é Revisar Itens Alterados. Os artefatos de saídas são: i) Código dos Itens Alterados; ii) Itens Alterados (Modelos e Documentos); iii) Relatório dos Testes Realizados; e iv) Versão do Produto (Preliminar);
- Testes: Os artefatos de entradas são: i) Itens a Serem Alterados (Código); ii) Itens Alterados (Modelos e Documentos); e iii) Descrição dos Testes. Os processamentos são: i) Realizar Testes de Regressão; ii) Realizar Testes de Sistemas; e iii) Realizar Testes de Aceitação. Os controles são: i) Revisar Itens Alterados; e ii) Realizar Auditoria de Configuração. Os artefatos de saída são: i) Nova Versão do Produto; ii) Cópias dos Itens a Serem (Re)Instalados junto ao Cliente; e iii) Aviso de Incorporação da Alteração à Linha Básica;

- Instalação: Os artefatos de entradas são: i) Cópias dos Itens a Serem (Re)Instalados; e ii) Aviso de Incorporação da Alteração à Linha Básica Instalação. O processamento é Instalar Cópias. O controle é Aceitar Alterações (Usuário). O artefato de saída é Aviso de Aceitação.

3.4. Processo Massivo Adaptativo de Manutenção

A Figura 3 -6 mostra o fluxo de atividades do Processo de Massivo Adaptativo de Manutenção adaptado de Lucia *et al*. (2002). O fluxo apresenta síntese das atividades, mostrando para cada atividade os artefatos de entradas (gravura branca – lado esquerdo), o nome (retângulo azulado), os atores (boneco) e os artefatos de saídas (gravuras cinza – lado direito). O processo de Massivo Adaptativo de Manutenção contém as seguintes atividades [Lucia *et al*. (2002)]:

- Portfólio e Avaliação. A primeira fase do processo e o portfólio e a avaliação: a aplicação do portfólio é analisada e decomposta em aplicações e *work-packets*. Tipicamente essa fase é conduzida por analistas de software juntamente com diferentes proprietários do sistema, i.e., pessoal técnico responsável pela customização da organização e que tenham conhecimento da aplicação a ser alvo da manutenção. Questionários e entrevistas são usados em conjunto com análise da documentação disponível para a identificação dos componentes pelas diferentes aplicações. Coesão e métricas de divisão de pares (*coupling metrics*) são também usadas para decomporem a aplicação em *work-packets*. Nessa fase, os critérios para estimar os impactos causados e as estratégias de solução são definidos. Certamente, a maioria dos impactos de uma manutenção adaptativa pode ser resolvida usando um grupo de soluções pré-definidas. A decomposição das aplicações em *work-packets* abre o caminho para o Processo Massivo Adaptativo de Manutenção: um subprojeto específico com uma pequena equipe pode ser instanciado para cada *work-packet*. Essa aproximação reduz a zona de congelamento de um componente de software em modificação, como todos os componentes de um *work-packet* podem ser colocados em operação uma vez que o Processo Massivo Adaptativo de Manutenção para o *work-packet* está completo. Isto também permite estimar mais facilmente o tempo total requerido pelo projeto todo, dependendo do cronograma adotado. Um teste de consistência baseado na analise de dependência é requerido para identificar se todos os componentes dos *work-packets* foram corretamente incluídos e se algum componente não é relevante para o *work-*

packet ou mesmo relevante para outros *work-packets*. Componentes relevantes para mais do que um *work-packet* necessitam que os *work-packets* relacionados sejam adaptados e integrados juntamente. Para grandes projetos, a identificação dos *work-packets* pode ser feita de maneira incremental. As fases remanescentes do PMAM são conduzidas para os diferentes *work-packets* em caminhos diferentes;

- Fase de Análise. A fase de Análise consiste em identificar os impactos estimados baseado no critério definido na fase de inventário. Tipicamente, o ponto de impacto inicial é primeiramente identificado, consistindo nos pontos do programa que confrontam diretamente como a requisição de manutenção; então, o processo continua com os novos pontos de impacto pelo efeito de encrespamento. Nessa fase algumas, *guildelines* pa stratégia de modificação são também definidas.
- Fase de Design. O ponto pri de is causados j ç os os AS e PS los e classifica os im s. Para cada im é se nenhuma das sponíveis pode ser aplicada no impacto, uma solução pr e par m o custo d normalmente e al dos
- Fas emen os com a apli e par
- Fase de es é implemen le uni em cada unid o enta para modificações ser e enc a mais efici e código p e testados u ados. Essa estratégia é chamada de teste de isolamento, e é n caso de impactos previstos onde os drives podem ser con viamente;
- Entrega e AM é a entrega e instalação. Se uma versão estava em u uma manutenção corretiva, a nova versão e os *work-* os antes da entrega.

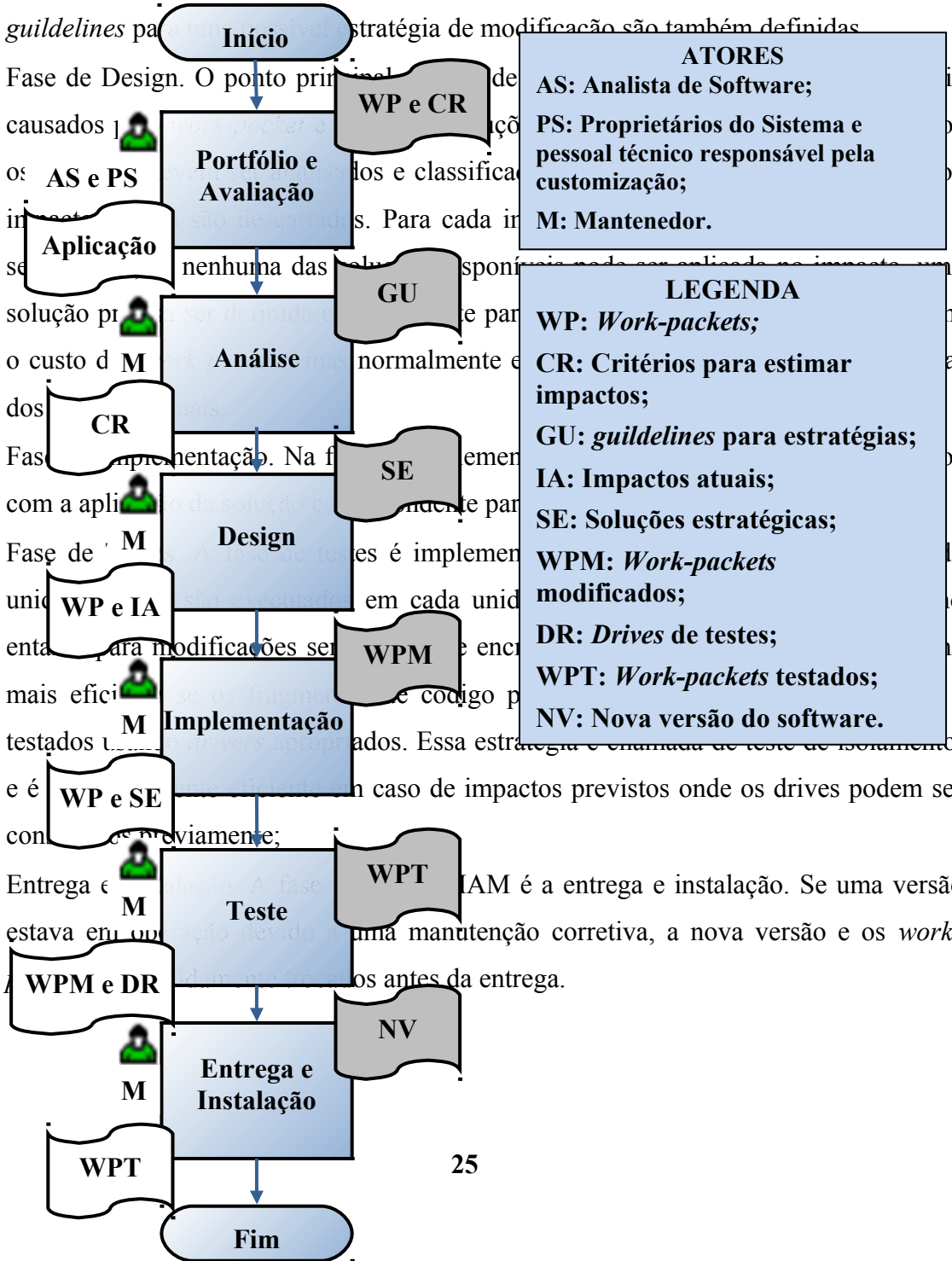

Figura 3-6 – Fluxo de Atividades do Processo Massivo Adaptativo de Manutenção

3.5. Considerações Finais

Este capítulo tratou de processos de manutenção de forma a criar um entendimento sobre as principais atividades e fluxos das atividades dos processos de manutenção. Em todos os casos as atividades começam com a identificação do problema, e seguem para a fase de análise, posteriormente realizando as alterações e finalizam com a entrega do produto.

Um importante aprendizado está relacionado com o fato de que tanto os pedidos de alteração aceitos quanto os recusados são registrados, para evitar futuros desperdícios de analise.

Os processos de manutenção de software são de vital importância para a continuidade dos sistemas legados, pois através deles é possível realizar manutenções adaptativas, corretivas, evolutivas ou preventivas garantindo rumo aos atores envolvidos e contribuindo para a garantia da qualidade de software.

4. DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

4.1. Considerações Iniciais

Este capítulo apresenta uma visão geral sobre o DDS, ressaltando as razões que levam ao DDS e seus níveis de dispersão, e uma unificação dos conceitos envolvidos na sua operacionalização no âmbito organizacional.

A crescente busca por maior competitividade tem levado as empresas a adotarem um modelo distribuído de desenvolvimento de software, onde diferentes partes do software são desenvolvidas em localidades distintas. Tentando realizar desenvolvimento a baixo custo, empresas têm atravessado fronteiras, formando um mercado global. Essa mudança de paradigma tem causado impacto no *marketing*, na distribuição e na forma de concepção, de produção, de projeto, de teste e de entrega de software aos clientes [HERBSLEB; MOITRA, 2001].

Além do desenvolvimento de baixo custo, as empresas buscam investir em DDS, buscando possibilidades de maior qualidade no processo de desenvolvimento, de obtenção de recursos em âmbito global, de aumento da produtividade e de diminuição dos riscos [AUDY, 2007]. Alguns fatores contribuíram significativamente para acelerar o seu surgimento e a sua aceitação, entre eles [CARMEL, 2005 *apud* AUDY, 2007]: i) a necessidade de recursos globais para uso a qualquer hora; ii) a proximidade com o mercado local; iii) os incentivos fiscais; iv) as soluções globais; v) o *Time-to-market*[2]; e vi) o *Follow-the-sun*[3].

Quando relacionado ao desenvolvimento tradicional de software, o DDS apresenta alguns pontos diferentes, sendo eles focados em três características principais: i) dispersão geográfica (distância física); ii) dispersão temporal (diferença de fuso horário); e iii) diferenças sócio-culturais (idiomas, tradições, costumes, normas e comportamentos). Estas diferenças afetam diretamente fatores como: i) questões estratégicas (decisão de desenvolver ou não um projeto de forma distribuída, tendo por base análises de riscos e de custo-benefício); ii) questões culturais (valores, princípios, etc. entre as equipes distribuídas); iii) questões técnicas (infra-estrutura tecnológica e comunicação, tais como:

[2] Pressões para reduzir o tempo necessário para colocar um produto no mercado.
[3] Equipes distribuídas ao redor do mundo, explorando os diferentes fusos horários de forma que sempre há uma equipe disponível para realizar o trabalho.

redes de comunicação e plataformas de hardware); iv) questões de gestão de conhecimento (criação, armazenamento, processamento e compartilhamento de informações) [AUDY, 2007].

A seção 4.2 descreve as razões que levam ao DDS. A seção 4.3 mostra os níveis de dispersão em relação ao DDS. A seção 4.4 esclarece alguns conceitos relacionados ao DDS e apresentados os aspectos organizacionais envolvidos no DDS. A seção 4.5 apresenta e categoriza os desafios do DDS. A seção 4.6 apresenta o Modelo MuNNDos.

4.2. Razões que Levam ao DDS

Não existe uma única razão envolvida ao DDS, mas um conjunto de razões. Segundo Audy (2007), essas razões ou subconjunto delas motivam um crescente número de organizações a desenvolverem software de forma distribuída. Os principais razões que motivam o uso de DDS são (Figura 4 -7):

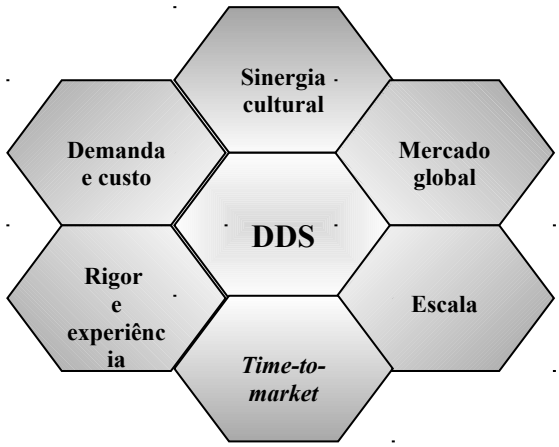

Figura 4-7 – Principais Razões Envolvidas no DDS

- Demanda e custo. Segundo Karolak (1998), a demanda por serviços de software tem superado historicamente a disponibilidade de pessoas que os realizam (Figura 4 -8). Como conseqüência, o custo do profissional aumentou. Outro fator agravante está relacionado à mão de obra vinda de outros países, pois em alguns deles o visto de entrada termina antes do final do ano fiscal, obrigando as empresas a realizarem novas contratações. Além disso, pode ocorrer disponibilidade de recurso a custo mais baixo em outras localidades, favorecendo a implantação da produção naquela localidade;
- *Time-to-market*. Rapidez de resposta ao mercado. Muitas empresas vêem vantagens do DDS como uma forma de realizar o desenvolvimento *follow-the-sun* e, assim, reduzirem o tempo de entrega de seus produtos;

- Mercado global. Para melhor satisfazer o mercado consumidor, a presença das corporações se torna necessária para venda, projeto e manutenção do software. Dessa forma, muitas empresas optam pelo DDS para atingir o mercado global e ficar próxima a seus consumidores [AUDY, 2007];

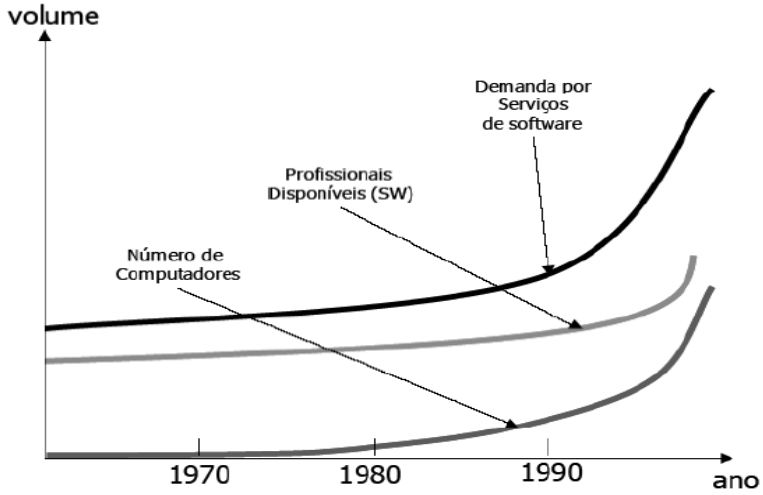

Figura 4-8 – Demanda por Software em Relação ao Número de Computadores e Profissionais Disponíveis (Fonte: Karolak (1998))

- Rigor e experiência. Em equipes de desenvolvimento co-localizadas, existe uma tendência do uso de mecanismos informais para a comunicação. No caso das equipes de DDS, existe tendência para melhorar a documentação, o uso de ferramentas de comunicação e de colaboração. Cada equipe co-localizada adquire experiência específica propiciando diferencial para o desenvolvimento ser realizado nela;
- Sinergia cultural. Diversidade de culturas é objetivada, pois amplia a criatividade e inspira a organização. Novas formas de resolver um problema são facilmente encontradas mediante as diferenças do modo de pensar e projetar. Além disso, a sinergia cultural, somada à demanda e aos desafios envolvidos, amplia a capacidade de aprendizado da organização;
- Escala. Centros de desenvolvimento de software quando ficam muito grandes tornam-se difíceis de gerenciar. Com isso, é necessário distribuir o desenvolvimento para atender a demanda necessária.

4.3. Níveis de Dispersão em DDS

Segundo Herbsled *et al*. (2001), quando a distância entre colaboradores distribuídos atinge 30 metros ou mais, a freqüência de comunicação diminui para um nível idêntico ao de colaboradores distribuídos a milhares de metros.

Em relação ao nível de dispersão, este diz respeito à distância física entre os atores envolvidos em um determinado projeto ou fase. No estudo realizado por Prikladnicki *et al*. (2003), alguns critérios foram definidos para classificar os níveis de dispersão dos *stakeholders*. Estes critérios foram utilizados em estudos de caso considerando clientes, usuários e equipe de projeto como atores. São definidas quatro situações que verificam o tipo de distância física e suas características principais:

- Mesma localização física: os *stakeholders* estão em um mesmo local. Nesta situação, reuniões ocorrem sem dificuldades e a equipe pode interagir estando fisicamente presente. Não existe diferença de fuso horário e as diferenças culturais raramente envolvem a dimensão nacional. Os obstáculos são os existentes no desenvolvimento centralizado de software (Figura 4 -9);

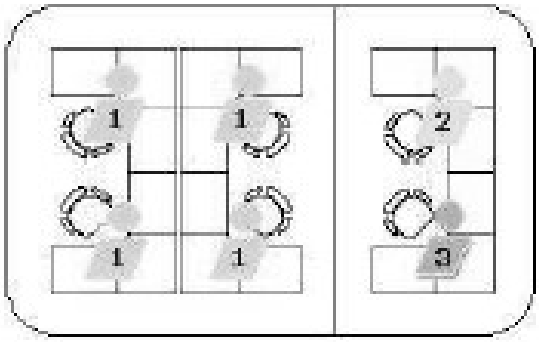

Figura 4-9 – Mesmas Localização Física (Fonte: Prikladnicki *et al*. (2004))

- Distância nacional: os *stakeholders* estão localizados dentro de um mesmo país, podendo se reunir em curtos intervalos de tempo. Dependendo do país, pode haver diferenças em relação ao fuso-horário e as diferenças culturais podem ocorrer em maior escala do que na situação anterior (Figura 4 -10);

Figura 4-10 – Distância Nacional (Fonte: Prikladnicki *et al*. (2004))

- Distância continental: os *stakeholders* estão localizados em países diferentes, necessariamente dentro do mesmo continente. Nesta situação, as reuniões ficam um pouco mais difíceis de serem realizadas face a face por causa da distância física. O fuso-horário exerce papel importante na equipe, podendo dificultar algumas interações (Figura 4 -11);

Figura 4-11 – Distância Continental (Fonte: Prikladnicki *et al*. (2004))

- Distância global: os *stakeholders* estão localizados em países diferentes e em continentes diferentes, formando muitas vezes uma distribuição global. Nesta situação, reuniões face a face ocorrem geralmente no início dos projetos e, entre outros fatores, a comunicação e as diferenças culturais podem ser barreiras para o trabalho. O fuso-horário exerce papel fundamental, podendo impedir interações entre as equipes (Figura 4 -12).

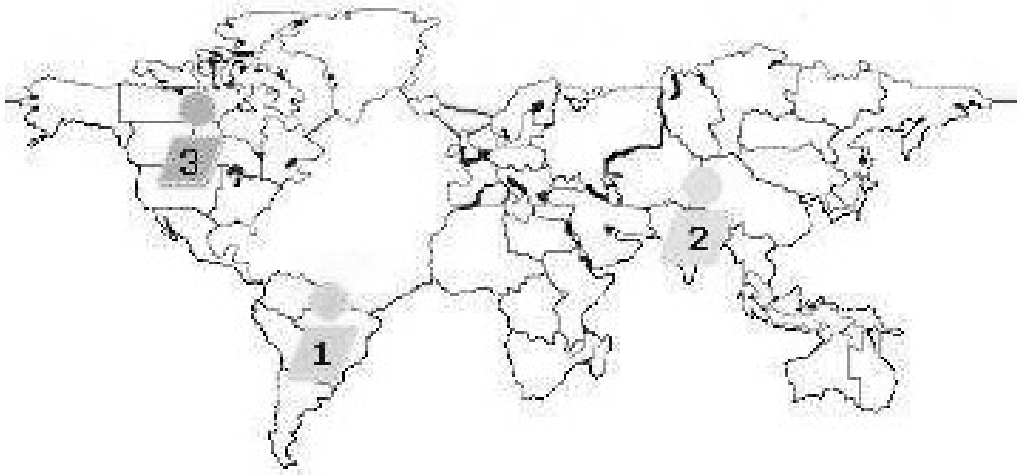

Figura 4-12 – Distância Global (Fonte: Prikladnicki *et al*. (2004))

4.4. Unificação dos Conceitos de DDS e seus Aspectos Organizacionais

Alguns conceitos relacionados ao DDS comumente encontrado na literatura:

- GSD (*Global Software Development*): caracterizado quando a distância física entre os elementos de um DDS envolve mais de um país;
- BPO (*Business Process Outsourcing*): transferência do processo de negócio;
- *Outsourcing* (terceirização de atividades ou serviços): transferência de uma função organizacional para terceiros dentro da mesma empresa;
- *Offshore outsourcing*: quando terceiros estão localizados em outro país. Pode envolver terceirização e desenvolvimento *in-house*, mas em outro país;
- *Offshoring*: transferência de uma função organizacional para outro país, independente de ser executada dentro ou fora da organização. Inclui empresas que montam seus *dedicated captive centers* (centros de desenvolvimento dedicado) em um local de baixo custo;
- *Nearshore sourcing*: caracteriza a existência de atividades em outros países geograficamente não tão distantes;
- *Bodyshopping*: uso de recursos *offshore* sem a transferência do processo de negócio.

A Figura 4 -13 apresenta a delegação do controle das atividades pela empresa ou por outra empresa contratada ou subsidiada e relaciona estes fatores com a localidade onde cada atividade deverá ser realizada.

Controle e Relação entre as Pessoas	Terceirizar *Outsorce* "Comprar"	*Onshore Outsourcing* ou *Outsourcing*	*Offshore Outsoucing* ou *Offshoring*
	Departamento ou Subsidiária *Insource* "Desenvolver"	*Onshore Insourcing* ou Demanda Doméstica Interna	*Offshore Insourcing* ou *Captive*/*internal Offshoring*
		Onshore/Mesmo Pais	*Offshore*/Outro Pais
		Localização Geográfica	

Figura 4-13 – Modelo de Negócios de DDS (Fonte: Adaptado de Robinson (2004))

Em termos organizacionais, as características alcançadas com o DDS estão relacionadas com: i) aumento da produtividade; ii) diminuição nos riscos; e iii) aumento da capacidade de concorrência. Em termos de estratégia, o *offsourcing* é uma das mais conhecidas para a operacionalização do DDS, sendo que os candidatos para a execução em uma operação de *offshoring* devem ser atividades que não necessitem de contato próximo com o cliente, não tenham interações complexas e não precisem de um conhecimento

existente somente na matriz da empresa. Para isso, adota-se o fluxo de atividades da Figura 4 -14, que expressa uma possível maneira de definir o cenário onde a atividade deverá ser realizada.

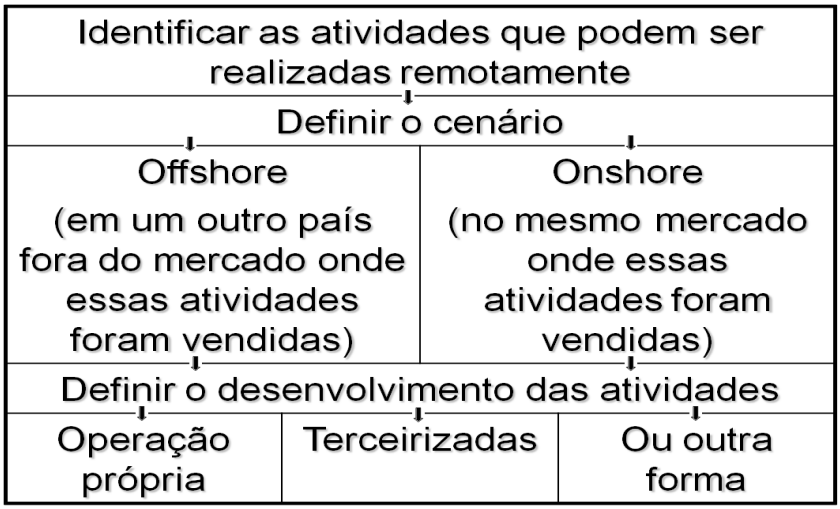

Figura 4-14 – Fluxo de Tarefas para a Definição do Cenário de Realização das Atividades Envolvidas no DDS (Fonte: Adaptado de Audy (2007))

Do ponto de vista da relação entre empresas, Hydler *et al*. (2006) destacam pelo menos seis possibilidades de relacionamentos (Figura 4 -15):

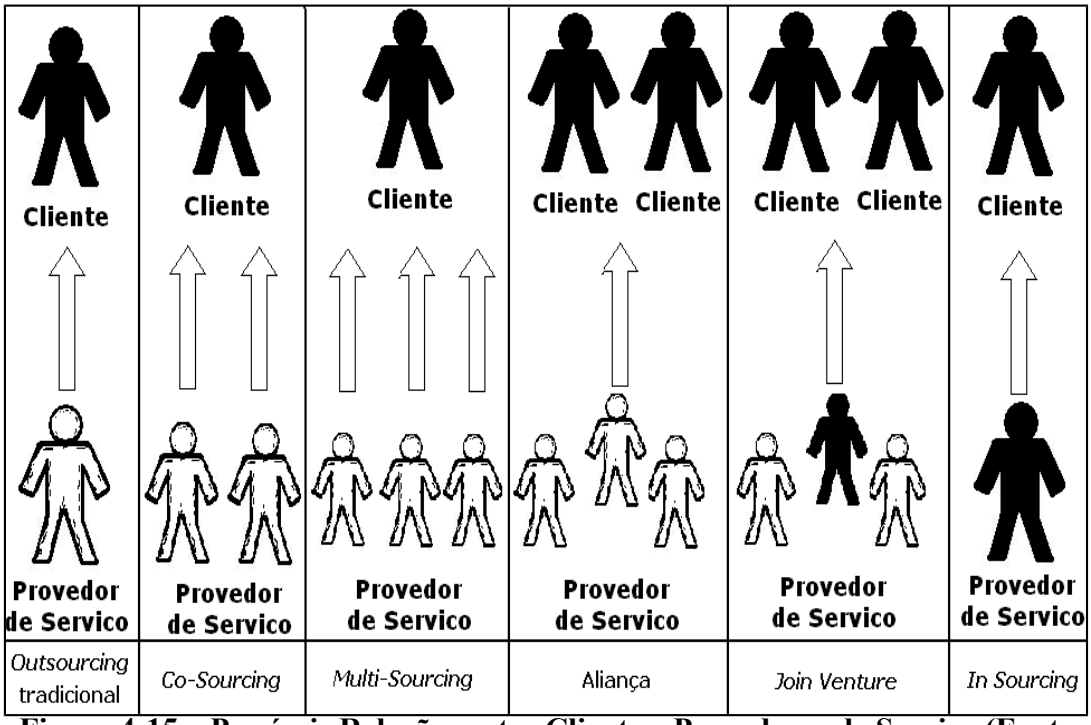

Figura 4-15 – Possíveis Relações entre Cliente e Provedores de Serviço (Fonte: Adaptado de Hyder *et al.* (2006))

- *Outsourcing* tradicional. Um provedor de serviço entrega um projeto para um cliente;
- *Co-sourcing*. Quando dois provedores trabalham em conjunto;

- **Multi-sourcing*. Há múltiplos provedores de serviço trabalhando para um cliente (o cliente gerencia e entrega os serviços);
- Aliança. Colaboração de múltiplos provedores (um deles como responsável) trabalhando para um ou mais clientes;
- *Join venture*. Provedores de serviço estabelecem um acordo, através da união de seus recursos, para executar projetos para clientes;
- *In Sourcing*. Uma entidade independente é estabelecida e é comum uma empresa *offshore* estabelecer parceria com empresa local, bem como o primeiro cliente ser parceiro na *join venture*.

4.5. Desafios do DDS

Os desafios do DDS podem ser categorizados em pessoa, processo, tecnologia, gestão e comunicação.

4.5.1. Desafios do DDS Relacionados a Pessoas

Estes desafios têm como principal foco:

- Confiança: Uma equipe cujos membros não confiam uns nos outros não funciona de maneira efetiva, podendo falhar completamente [Audy, 2007]. A interação efetiva tem como base a confiança e, se bem empregada, traz melhoras nos custos, no desempenho e para a sociabilidade. Existem diversas formas de construir a confiança: i) encontros de *kick-off*[4]; ii) marcos do projeto; iii) socialização do início do desenvolvimento; e iv) reuniões com recursos tecnológicos de comunicação. O principal objetivo da criação e da manutenção da confiança é prevenir conflitos desgastantes ao longo do tempo;
- Conflitos. Conflitos em projetos são quase inevitáveis, devendo utilizar uma autoridade central de modo a assegurar a resolução do conflito sem comprometer o progresso das atividades;
- Diferenças culturais. A cultura é tratada como a formadora de valores. As equipes de DDS tendem a aumentar a produtividade, a efetividade e a satisfação quando envolvem em uma cultura comum; existe o papel denominado como *Liaison*, uma pessoa que desempenha um papel de ponte entre duas ou mais culturas;
- Ensino. Para o mercado de trabalho e para executivos, a formação de recursos humanos enfrenta alguns problemas, pois o DDS, por ser uma prática nova, sendo pouco

[4] Marco no projeto no qual a principal idéia é reunir os membros de uma equipe para alguns dias de trabalho e socialização no início do ciclo de desenvolvimento do trabalho [AUDY, 2007].

abordado em universidades, faz com que as empresas acabem sendo responsáveis pela formação complementar do profissional;

- Espírito de equipe. Fatores que afetam diretamente: i) distância e diferenças culturais e de fuso-horário; ii) esforço e capacidade dos colaboradores; iii) tamanho e complexidade da equipe e do problema abordado; e iv) fator de algumas culturas apresentarem pouca experiência com o trabalho em equipe e em outra a palavra (equipe) nem mesmo existe;
- Formação de equipes e grupos. Há duas maneiras de realizá-la: i) valorizando a independência e os direitos individuais; e ii) valorizando a coletividade e subordinando os interesses do indivíduo ao bem do grupo. Para isso, a criação de gráficos com a estrutura da equipe ou um espaço específico na intranet são usados para auxiliar a equipe;
- Liderança. Destaca-se a existência de duas formas de lideranças baseadas em aspectos culturais: i) participativa e democrática (são encorajadas manifestações e opiniões); e ii) hierárquica (baseada em um *status* como família, idade, sexo ou títulos);
- Tamanho da equipe. A quantidade de pessoas na equipe influencia diretamente a coordenação e o controle, tornando-as mais difíceis. Esse fator se agrava, pois, normalmente, em uma equipe distribuída, há mais membros que em uma equipe co-localizada.

4.5.2. Desafios Relacionados ao Processo

Estes desafios têm como principal foco:

- Arquitetura do software. A arquitetura é um fator determinante da efetividade e redução das dificuldades, é focada na divisão do esforço entre as equipes do DDS e deve-se focar na modularidade. Com a modularização, um projeto alcança redução da complexidade, possibilidade de desenvolvimento em paralelo, diminuição da interdependência entre os locais de desenvolvimento e contribui para ampliar o paralelismo entre os locais de desenvolvimento;
- Engenharia de requisitos. O DDS apresenta algumas características que o tornam fundamentalmente diferente do desenvolvimento de software co-localizado. A engenharia de requisitos contém diversas tarefas que necessitam de alto nível de comunicação e de coordenação. Isso tende a aprofundar os problemas apresentados quando em um contexto de DDS;

- Gerência de configuração. Sua abordagem auxilia no controle da documentação do software e, em ambientes distribuídos, aborda: i) gerência de modificações simultâneas; ii) aplicações consistentes de padrões; e iii) coordenação de modificações de forma efetiva em tempo. A maioria das empresas adota uma solução caseira com controles manuais ou espelhamento em cada local distribuído, pois poucas ferramentas CASE oferecem apoio ao DDS;
- Processo de desenvolvimento. O ponto crítico está na sincronização das atividades. Para isso, adota-se uma metodologia de desenvolvimento onde: i) termos e marcos possam ser entendidos corretamente; ii) desenvolvedores saibam em que ponto se encontra o processo; e iii) equipe tenha rigor. Além disso, há crescente uso de modelos de governança e de soluções, cada vez mais integrados nos níveis de decisão.

4.5.3. Desafios do DDS Relacionados à Tecnologia

Estes desafios têm como principal foco:

- Tecnologia de colaboração (*groupware*). Objetiva-se ampliar a comunicação informal e encontrar novas formas de comunicação formal entre diversos locais de desenvolvimento. A tecnologia de colaboração se divide em dois tipos: i) genérica de colaboração (correio eletrônico, vídeo conferência, correio de voz, etc.); e ii) para apoio às atividades de engenharia. Além disso, a tecnologia pode ser caracterizada: i) quanto ao tempo: síncrona e assíncrona; e ii) quanto à localização: co-localizada e dispersa. Para obter bom resultado, é indicada a combinação de ferramentas CASE e de gerencia de projetos;
- Telecomunicação. As conexões entre os locais devem ser feitas de forma segura, sendo comumente as conexões dedicadas via satélite a redes privadas virtuais, pois apresentam baixo custo, alta disponibilidade e confiabilidade. É importante ressaltar que, mesmo com excelente comunicação, viagens não deixam de ser necessárias em DDS.

4.5.4. Desafios Relacionados à Gestão

Estes desafios têm como principal foco:

- Coordenação, Controle e Interdependência. Coordenação como integração das tarefas e das unidades de forma que o esforço da equipe contribua para o objetivo geral. Controle como processo de adesão às metas, políticas e padrões sendo que o

gerenciamento não pode ser feito por observações devido à distância. Isso acarreta um fator de atraso e cria situações nas quais: i) aumento na necessidade de coordenação ocasiona aumento na necessidade de comunicação; e ii) quando as tarefas são novas ou incertas, aumenta o custo para a coordenação. Segundo Audy (2007), o custo para coordenar o trabalho aumenta quando as tarefas são novas ou incertas ou quando as unidades de trabalho tornam-se mais interdependentes. Isso significa que, conforme a interdependência entre as equipes aumenta, a necessidade de coordenação aumenta (Figura 4 -16);

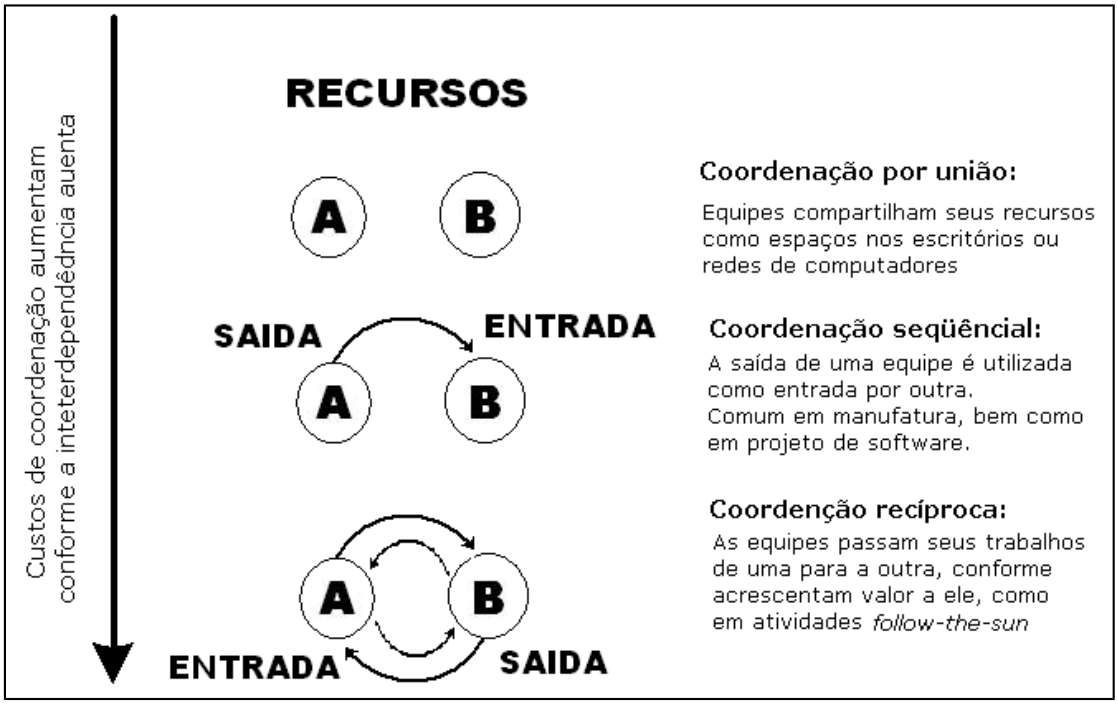

Figura 4-16 – Custos de Comunicação e Interdependência (Fonte: Audy (2007))

- Gestão de portfólio de projeto. O objetivo é realizar os projetos corretos e não o projeto corretamente. O responsável pelo portfólio de projeto deve administrar os projetos distribuídos e determinar, pelas análises qualitativas e quantitativas, como e onde devem ser desenvolvidos. Quando a administração é distribuída, ela gera uma participação mais efetiva dos profissionais em cada unidade de trabalho;
- Gerenciamento de projetos. O objetivo é adaptar a equipe co-localizada para atender e reduzir as dificuldades impostar pela distância. Para isso, o uso de métricas atua como um componente essencial para monitor de forma efetiva o processo. Como exemplos de métricas: i) tempo usando tecnologias de colaboração; ii) quantidade de sessões; iii) índice de construção e equipes; iv) dias de viagem; e v) taxa de resolução de problemas. Com as diferenças culturais, torna-se difícil o reconhecimento e as

bonificações, pois o foco está no aumento da produtividade. Para isso, um bom líder para o DDS dever possuir fácil adaptação às culturas e à comunicação eletrônica, ter conhecimento internacional, saber promover a equipe e gostar de viajar;

- Gerência de riscos. Categorizar os riscos: i) organizacional; ii) técnico; e iii) de comunicação. O foco da gerência de riscos é o nível organizacional, pois tem a função de analisar vantagens de um desenvolvimento de forma distribuída, identificar riscos envolvidos e repassar informações para o nível de projeto;
- Legislação (incentivos fiscais e tributários). Há políticas de incentivo para atração de *offshore* de grandes empresas internacionais, cujo foco está em políticas fiscais e tributárias, envolvendo leis de fomento e redução de impostos e encargos trabalhistas para as novas operações nesses países;
- Legislação (propriedade intelectual). O foco é a propriedade intelectual de projetos desenvolvidos de forma segmentada, o registro e a comercialização de patentes, a titularidade e os direitos sobre os *royalties* e o registro, o licenciamento e o direito de uso do software;
- Modelo de negócio. O foco é nas reais necessidades do projeto e os objetivos são em quais os tipos de projetos são distribuídos geograficamente e no bom planejamento juntamente com o estudo de locais tem determinado o sucesso no modelo de negócio escolhido;
- Seleção e alocação de projeto. Destacam-se as atividades: i) análise de viabilidade; ii) confidencialidade ou restrições legais; e iii) análise das melhores unidades para desenvolver o projeto. Nestes casos, os aspectos relevantes são o fuso horário a experiência, o conhecimento e o *overhead* previsto de gerenciamento e conhecimento.

4.5.5. Desafios do DDS relacionados à comunicação

Estes desafios têm como principal foco:

- *Awareness*. Representa a consciência, a percepção e o conhecimento das atividades desenvolvidas, focando: i) o compartilhamento de resultados; ii) as novas informações para chegar ao destino; iii) o estudo de coordenação de equipes de projeto; iv) a análise de redes sociais; e v) o desenvolvimento de mecanismos *awareness* automatizados;
- Contexto de desenvolvimento. Há diversas formas de realizar o compartilhamento de contexto: i) intranet com mensagens de feriados; ii) informativos de ausência; iii) protocolos contra mal entendidos; e iv) demora de respostas;

- Dispersão geográfica e temporal. A distância física e psicológica é considerada como a segunda maior dificuldade em equipes distribuídas, pois: i) diminui o contato face a face; ii) cria demora de aprovações; iii) diminui o tempo de interação síncrona; e iv) cria um sentimento de isolamento;
- Estilo de comunicação. Há duas classificações: i) pessoal; e ii) impessoal. O estilo de comunicação pessoal pode variar de: i) expressivo (preocupa-se com o estabelecimento e a manutenção de conexões pessoais e sociais, deixando a precisão em segundo plano); ii) instrumental (mais impessoal, centrada em problemas); e iii) orientada a objetivos (precisão é mais importante que o formato);
- Formas de comunicação. Alguns fatores são agravantes: i) baixa comunicação informal; ii) grande necessidade de reuniões; iii) complexidade para coordenação das reuniões; e iv) comunicação torna-se indireta por causa da distância temporal. Quando as formas de comunicação são relacionadas, é importante citar: i) a cultura de alto contexto (na qual existe o compartilhamento de experiências, mensagens devem ser inteligíveis e idéia de que o sentido de uma frase não está somente nela); e ii) a cultura de baixo contexto (na qual há trocas de fatos na comunicação e na idéia de que a mensagem é mais importante que a forma). Por fim, é importante adotar meios de interação ricos onde existe interação nos dois sentidos com mais de um canal sensorial;
- Fusos horários. Devem ser estabelecidos horários das localidades envolvidas e, principalmente, horários para a comunicação síncrona ante os membros.

Para realizar a comunicação de forma eficiente, métodos e tecnologias devem ser usados para viabilizar a comunicação ao longo do projeto. Essas tecnologias são denominadas tecnologias de comunicação colaborativas e, segundo Carmel (1999), são classificadas em dois grupos: i) tecnologias genéricas; e ii) tecnologias de apoio à engenharia de software.

Segundo Dantas (2003), as tecnologias genéricas servem para a comunicação e para a troca de informações entre as equipes. Elas podem ser síncronas, como vídeo-conferência, áudio-conferência, *e-chat* e *e-whiteboard*, ou assíncronas, como *e-mail*, *voice-mail*, v*ideo-mail* e listas de discussão. Por causa da diferença de fusos-horários existente em projetos globais, as tecnologias síncronas são menos usadas, apesar de bastante eficientes para permitir aos participantes do projeto se conhecerem, aproximando as equipes. As tecnologias de apoio à engenharia de software são usadas especificamente para

ajudar os desenvolvedores no desempenho de suas atividades. Alguns exemplos: gerenciadores de configuração de software, as ferramentas CASE e de programação e ferramentas para registro de defeitos e acompanhamento de mudanças no software.

4.6. O Modelo MuNNDos

O modelo de referência MuNNDoS foi elaborado para atuar como facilitador nos projetos de DDS. Além disso, a forma como este modelo foi concebido permite a identificação de fraquezas e de oportunidades de melhorias nos projetos. Para isso, o modelo sugere a existência de duas dimensões: organizacional e de projetos.

Ampliando a visão relativa ao processo de desenvolvimento de software e buscando adotar uma visão estratégica com relação ao processo, pode-se identificar a etapa de planejamento como a primeira a ocorrer. Esta etapa envolve a definição das estratégias que deverão conduzir o processo de desenvolvimento como um todo, ao longo do tempo (dimensão organizacional). Pode-se identificar a etapa de planejamento como preliminar a um conjunto de ciclos de projetos de desenvolvimento de software (dimensão de projetos) derivados do processo de planejamento.

A Figura 4 -17 apresenta dois ciclos de planejamento necessários para a gestão de projetos de DDS; o primeiro ciclo envolve o planejamento estratégico e o segundo ciclo envolve o planejamento tático-operacional. A transição entre estes dois ciclos ocorre exatamente na alocação dos projetos.

O ciclo planejamento estratégico é conduzido pela matriz e diz respeito à identificação e à priorização de novos projetos a serem desenvolvidos, sejam eles projetos internos (de departamentos internos à organização) ou externos (projetos requisitados por clientes externos). Cabe aos participantes deste nível de planejamento buscar o alinhamento estratégico entre os objetivos de cada unidade distribuída e da matriz. Para a adoção de novos projetos são estipulados critérios de exclusão direta para definir quais deles podem ou não ser desenvolvidos de maneira distribuída. Segundo Audy (2007), os critérios variam de organização para organização, mas alguns pontos devem ser considerados: i) restrições de exportação; ii) privacidade dos dados; iii) propriedade intelectual; iv) disponibilidade em ambiente físico; v) restrições de segurança; e vi) tipo de *engagement*. Além dos critérios de exclusão direta, são realizadas duas análises: i) de risco

da distribuição; e ii) de custo/beneficio da distribuição. A análise de risco da distribuição considera os seguintes critérios: i) nível de documentação existente; ii) clareza e estabilidade dos requisitos; iii) riscos da tecnologia; iv) experiência dos atores em projetos distribuídos; v) capacidade de controle; vi) complexidade e duração do *engagement*; e vii) tamanho do projeto. A análise de custo/benefício considera os seguintes critérios: i) percentual de esforço necessário nas unidades fisicamente dispersas; ii) necessidade de haver recursos humanos junto do cliente e/ou usuário; e iii) custo do percentual de *overhead* gerencial[5] que a distribuição do projeto poderá causar. Para alocar projetos, Audy (2007) sugere quantificar um fator de riscos para cada unidade envolvida, considerando: i) capacidade e experiência da unidade em projetos similares; ii) existência de algum centro de competência na tecnologia requerida no projeto; iii) disponibilidade de recursos humanos; iv) tempo necessário para o treinamento de novos colaboradores; v) espaço físico disponível; vi) fator de *turn-over*[6]; vii) barreiras de idiomas; viii) barreira de fuso horário; e ix) custo de *overhead* gerencial de cada unidade de manutenção distribuída.

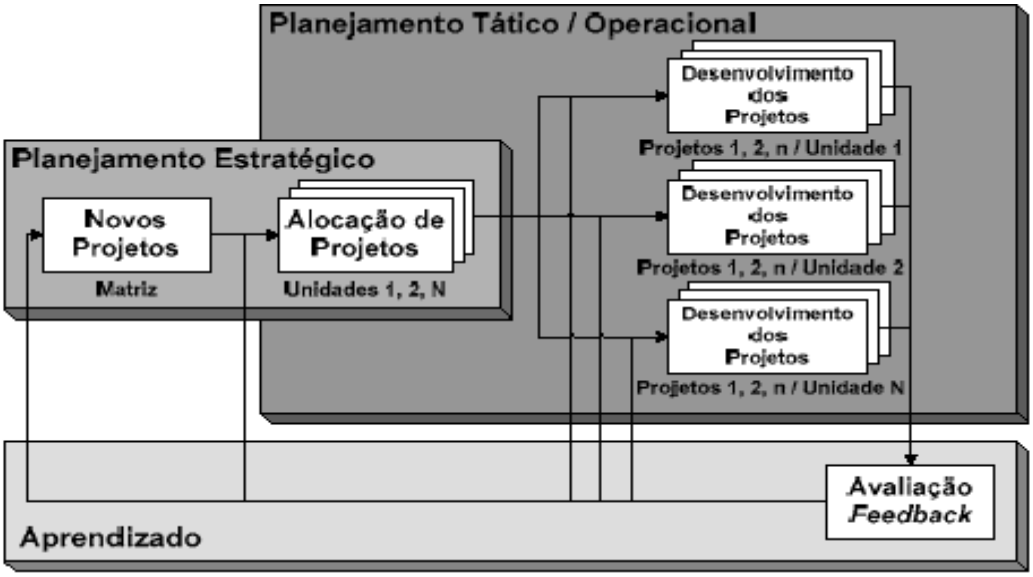

Figura 4-17 – Estágios de Capacidade do Modelo MuNNDos (Fonte: Audy (2007))

O ciclo planejamento tático-operacional envolve atividades de planejamento e de definição dos projetos desenvolvidos em cada unidade distribuída, de acordo com políticas de alocação previamente definidas e análise de risco e de custo-benefício. O planejamento tático é de responsabilidade final dos responsáveis por cada unidade de desenvolvimento,

[5] Despesas adicionais para o gerenciamento de unidades distribuídas.
[6] Riscos de colaboradores saírem no meio do projeto.

enquanto que o planejamento operacional envolve a gestão do projeto (dimensão de projetos), sob responsabilidade do gerente de projeto.

A dimensão de projetos envolve especificamente a gerência do projeto de desenvolvimento de software, centrada na coordenação geral do trabalho entre os colaboradores, interfaces entre as equipes, comunicação, contato com os clientes e resolução de conflitos. Apesar de ser caracterizada pelo desenvolvimento dos projetos, esta dimensão não está focada na definição de um processo, mas na identificação dos fatores e na relação entre eles.

Assim, mapas conceituais (Figura 4 -18) podem prover uma representação gráfica simples dos principais elementos identificados, tornando claro quais os conceitos envolvidos e os fatores relacionados que podem ser diretamente afetados. Além disso, pode-se avaliar a necessidade de ajustes no desenvolvimento dos projetos, em decorrência da existência ou não de algum fator.

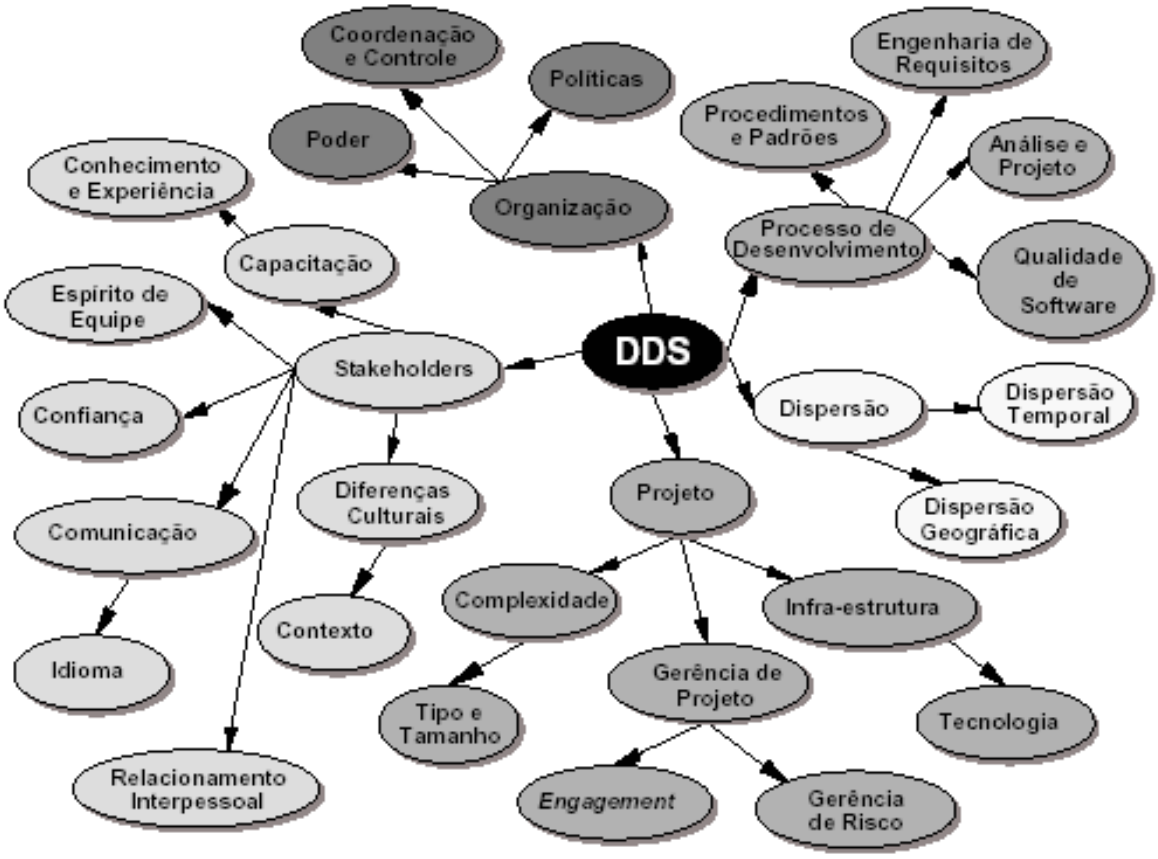

Figura 4-18 – Diagrama de Influência das Categorias (Fonte: Audy (2007))

O último ciclo proposto no modelo de referência é o de aprendizado, relativo à avaliação das atividades realizadas, e estratégias adotadas. O modelo sugere a existência de um processo para suportar a coleta de dados, envolvendo a avaliação dos trabalhos realizados, lições aprendidas, etc. Desta forma, o modelo realimenta os ciclos de planejamento.

4.7. Considerações Finais

A área de DDS é tratada como recente, pois surgiu para o desenvolvimento de software a partir de 1990, mas somente nos últimos 10 anos seu crescimento acelerou e, conseqüentemente, gerou diversas dificuldades aos projetos distribuídos. Dessa forma, a diversidade de experiências na indústria tem sido utilizada para elabora guias de boas práticas e estratégias, nos mais diversos modelos de DDS.

O desenvolvimento de software sempre se apresentou de forma complexa. Existem diversos problemas e desafios inerentes ao processo. Assim como o processo de desenvolvimento de software tem se tornado mais complexo, a distribuição das equipes no tempo e no espaço tem tornado os projetos distribuídos cada vez mais comuns.

O software é cada dia mais indispensável para a sociedade moderna, onde a globalização é uma característica fundamental. Atualmente, diversas empresas distribuem seus processos de desenvolvimento de software ao redor do mundo, visando ganhos de produtividade, redução de custos e melhorias na qualidade. Neste contexto, o ambiente de DDS surge como um grande desafio para a área de engenharia de software.

Em linhas gerais, observa-se que em DDS o nível de formalização, de planejamento e de gerência da pesquisa como um todo é considerado maior por causa dos níveis de dispersão. Além disso, a fase de coleta de dados na maioria das vezes é responsável pela maior parte das dificuldades ao longo do processo de pesquisa. Ainda, é importante destacar que os desafios apresentados estão baseados em experiências vivenciadas por alguns pesquisadores na área. Desta forma, eles podem não ser validos em todas as situações, organizações e países.

5. PROCESSO DE MANUTENÇÃO DISTRIBUÍDA DE SOFTWARE

5.1. Considerações iniciais

Esse capítulo descreve o processo de manutenção distribuída, partindo de uma abordagem geral sobre o processo e, posteriormente, descrevendo o processo de maneira mais detalhada através da descrição completa de cada atividade.

A manutenção é considerada uma importante fase do processo de manutenção de software que, como descrito nos capítulos anteriores, consome quantidade significante dos recursos de um projeto. Conciliando a idéia de manutenção com a área inovadora DDS foi definido um processo de manutenção que aborda a manutenção de software de maneira distribuída.

A seção 5.2 apresenta uma visão geral do processo de modo a facilitar o entendimento global do processo de manutenção distribuída de software. A seção 5.3 apresenta um detalhamento de cada atividade do processo de manutenção distribuída.

5.2. Visão Geral

A Figura 5 -19 apresenta um fluxograma de atividades do processo de manutenção distribuída de software proposto, ressaltando a idéia das diversas formas de dispersão das equipes envolvidas no DDS. Além disso, esta figura mostra onde as atividades devem ser realizadas:

- Co-localizada significa a equipe responsável geral pelo projeto;
- Distribuída significa as equipes colaboradoras no processo de manutenção distribuída.

A decisão de classificar cada atividade como Co-localizada ou Distribuída é embasada na existência ou não de dispersão o grupo de tarefas realizadas pela atividade em questão.

A atividade Avaliação Primária permite realizar uma ponderação da solicitação de manutenção que precisa ser realizada. Em seguida, a atividade Análise do Problema e/ou da Modificação permite realizar melhor compreensão da solicitação da alteração. Nesta atividade, foi adotado um método de quebra do Problema/Solicitação de Manutenção em pacotes, módulos do Problema/Solicitação, que serão desenvolvidos de forma distribuída e trarão mais velocidade ao desenvolvimento do projeto.

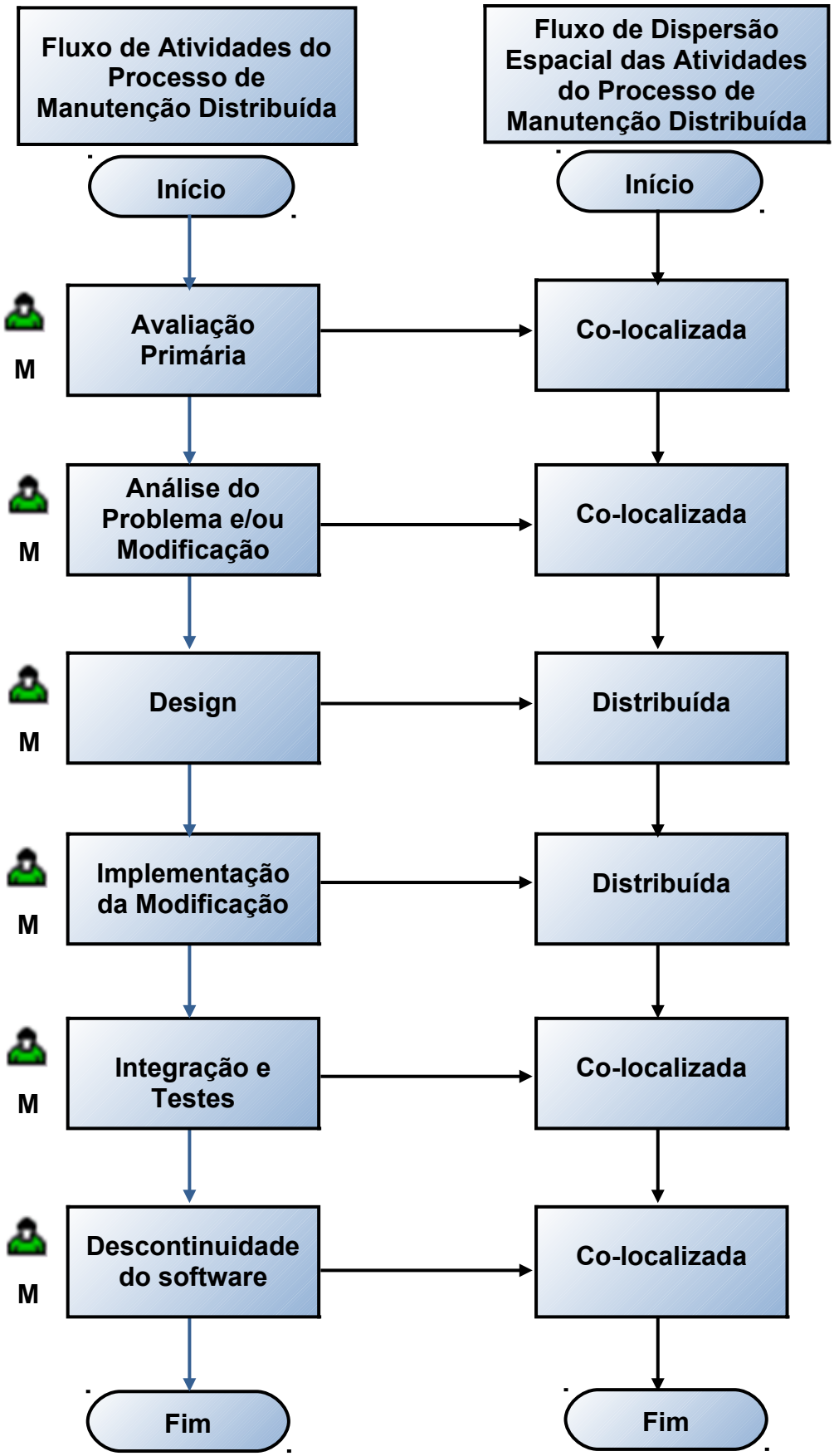
Fluxo de Atividades do Processo de Manutenção Distribuída
Fluxo de Dispersão Espacial das Atividades do Processo de Manutenção Distribuída
Início
Início
M
Avaliação Primária
Co-localizada
M
Análise do Problema e/ou Modificação
Co-localizada
M
Design
Distribuída
M
Implementação da Modificação
Distribuída
M
Integração e Testes
Co-localizada
M
Descontinuidade do software
Co-localizada
Fim
Fim

Figura 5-19 – Fluxo de Atividades do Processo de Manutenção Distribuída de Software

Feito isso, a atividade Projeto permite realizar a distribuição da tarefa de manutenção, considerando as equipes participantes no momento do desenvolvimento do software, bem como as novas equipes agregadas ao projeto. Nessa atividade, cada equipe envolvida recebe um pacote e suas tarefas são realizadas no mesmo. A atividade Implementação da Modificação é realizada nos pacotes pelas equipes distribuídas e responsáveis por realizar a tarefa de manutenção. A atividade Integração e Teste é realizada após o término das atividades das equipes distribuídas e é de responsabilidade da equipe responsável geral pelo projeto. Nessa atividade é feita a integração dos pacotes das unidades envolvidas no projeto, no entanto, mesmo recebendo os pacotes das outras unidades, o tratamento desses pacotes é realizado localmente de modo a tornar o problema novamente atômico. Por fim a atividade Descontinuidade do Software é realizada quando se tem a decisão de encerar quaisquer alterações no software para ele ainda ser útil. Essa atividade pode ser realizada pela equipe responsável geral pelo projeto ou por uma equipe mais capacidade para realizar a descontinuidade.

5.3. Detalhamento das Atividades do Processo de Manutenção Distribuída

Nessa seção, são descritas de forma detalhada as entradas, as tarefas e as saídas das seis atividades do processo de manutenção distribuída de software proposto. Cada subseção é referente a uma atividade e está dividida em três partes, sendo:

- A primeira parte é referente a entradas: descreve os artefatos de software de entrada das atividades;
- A segunda parte é referente a tarefas: descreve as tarefas e a ordem em que as tarefas devem ser realizadas;
- A terceira parte é referente a saídas: descreve os artefatos de saída das atividades.

5.3.1. Atividade Avaliação Primária

O objetivo é filtrar clientes e projetos previamente identificados. Os filtros são baseados em experiências anteriores e na viabilidade da tarefa ser realizada de maneira distribuída. A Figura 5 -20 apresenta um resumo desta atividade.

5.3.1.1. Artefatos de Entrada

Os artefatos de software de entrada são: i) Lista de Propostas de Manutenção; e i) Histórico de Experiências Anteriores de Manutenções Distribuídas.

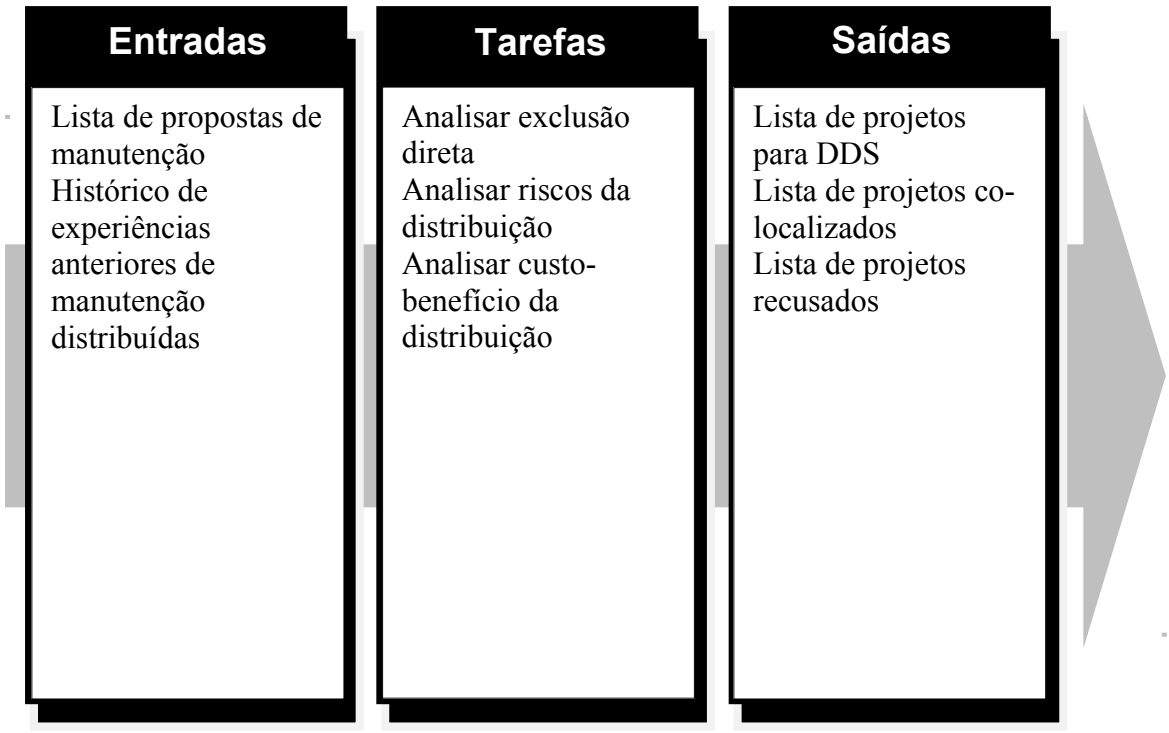

Figura 5-20 – Quadro de Entradas, Tarefas e Saídas da Atividade Avaliação Primária

5.3.1.2. Tarefas

A Lista de Propostas de Manutenção deve ser percorrida de modo a identificar os projetos que se enquadram na categoria em que a manutenção pode de ser distribuída. Para isso, o primeiro passo é Analisar Exclusão Direta, na qual é verificada a existência de critérios que rejeitam a realização da manutenção de forma distribuída. Esses critérios foram detalhados na seção 4.6. Ao final desse passo, são identificadas três listas: i) lista de projetos cuja manutenção pode ser realizada de forma distribuída; ii) lista de projetos cuja manutenção não pode ser realizada de forma distribuída; iii) lista contendo os projetos recusados para a manutenção. Os projetos cuja manutenção não pode ser realizada de forma distribuída representam possíveis projetos a serem realizados de maneira co-localizada.

Uma vez realizada a tarefa Analisar Exclusão Direta, a lista de projetos cuja manutenção pode ser realizada de forma distribuída é submetida às tarefas Analisar Riscos da Distribuição e Analisar Custo-Benefício da Distribuição. Analisar Riscos da

Distribuição consiste em quantificar um fator de riscos e são adotados os critérios definidos na seção 4.6. Por outro lado, a atividade Analisar Custo-Benefício da Distribuição está centrada no custo de desenvolver o projeto de forma distribuída e os critérios adotados para sua realização estão na seção 4.6. Com isso, deve-se calcular o custo-benefício da distribuição, comparando o custo de desenvolver de forma co-localizada e de forma distribuída. Ao final desse passo, deve ser identificada uma lista de projetos que estão aptos (possuem riscos e benefícios considerados razoáveis) a terem manutenção realizada de forma distribuída.

5.3.1.3. Artefatos de Saída

Os artefatos de software de saída são: i) lista de projetos aptos a serem desenvolvidos de forma distribuída; ii) lista de projetos a serem realizados de maneira co-localizada; e iii) lista de projetos recusados.

5.3.2. Atividade Análise do Problema e/ou Modificação

O objetivo é analisar o pedido de manutenção mais detalhadamente, de forma a decompô-lo e alocar as partes decompostas às demais unidades envolvidas no processo de manutenção distribuída. A Figura 5 -21 apresenta um resumo desta atividade.

5.3.2.1. Artefatos de Entrada

Os artefatos de software de entrada são: i) lista de projetos a terem sua manutenção realizada de forma distribuída; ii) documentação do projeto (caso exista); iii) tabela de capacidades; e iv) histórico de experiências de manutenções distribuídas anteriores.

5.3.2.2. Tarefas

A tarefa Decompor em Pacotes está centrada em dividir a proposta de manutenção, que se encontra em sua forma atômica, em um grupo de componentes coesos denominados pacotes. Os pacotes são módulos independentes entre si que compõem a proposta de manutenção distribuída.

A tarefa Alocar Pacotes está centrada na escolha da melhor unidade para receber cada pacote originado da tarefa Decompor em Pacotes. Nesse sentido, deve-se selecionar a melhor forma de distribuí-los. Para isso, devem ser realizadas as tarefas Analisar Riscos das Unidades e Analisar Custo-Benefício das Unidades. Os critérios usados na tarefa

Analisar Riscos das Unidades são os mesmos adotados na tarefa Analisar Riscos da Distribuição na atividade Avaliação Preliminar, com a exceção de serem mais específicos no sentido de levar em consideração critérios e características das unidades envolvidas. No entanto, para a tarefa Analisar Custo-Benefício das Unidades, na qual se deve quantificar os fatores de risco para cada unidade envolvida na atividade de manutenção distribuída. A tarefa Analisar Custo-Benefício das Unidades consiste em cálculos mais específicos do real custo de desenvolver cada pacote em cada unidade envolvida, identificando as unidades que apresentem os menores valores.

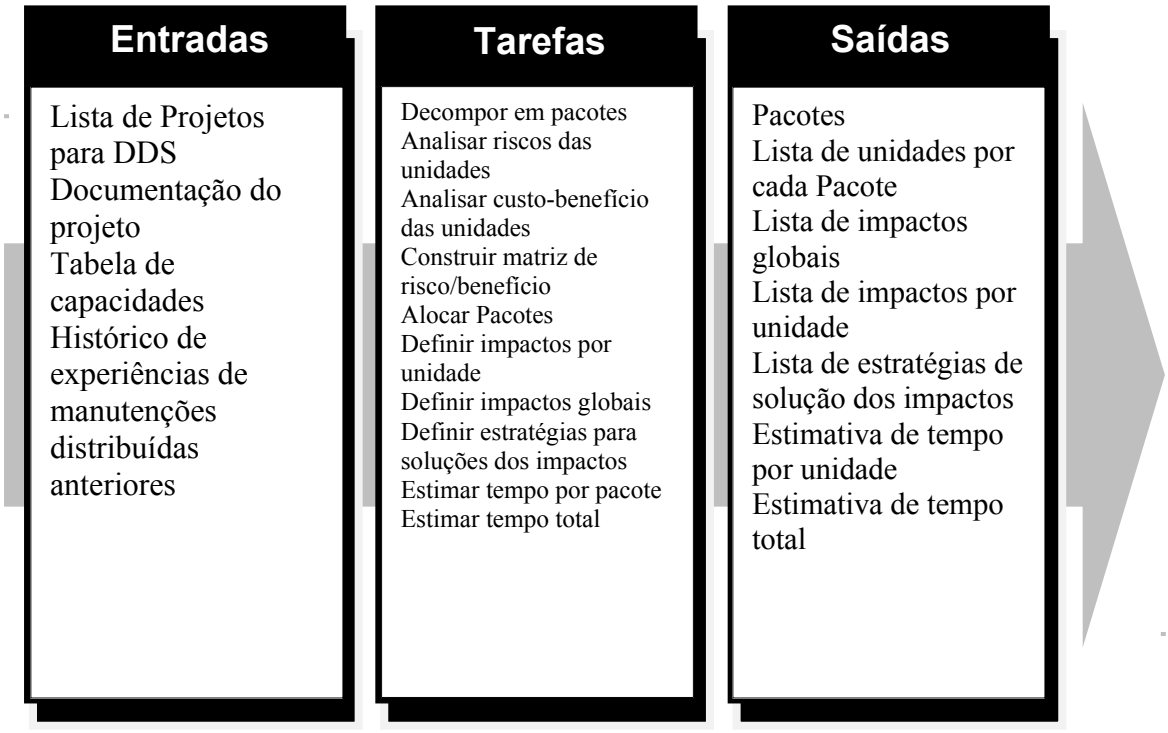

Figura 5-21 – Quadro de Entradas, Tarefas e Saídas da Atividade Análise do Problema e/ou Modificação

Como tarefa opcional, pode-se construir uma matriz que relacione o risco e o benefício de cada unidade sobre cada pacote, de modo a entender graficamente os benefícios que cada unidade possui para desenvolver cada pacote. Nesse trabalho, esta matriz é construída na tarefa Construir Matriz de Riscos e Benefícios. Ao final da realização das análises e da elaboração da matriz de riscos e benefícios, a unidade que melhor se identifica com os objetivos de cada pacote deve ser selecionada e deve-se elaborar a lista de unidades que desenvolverão cada pacote.

Após a definição desta lista, deve-se: i) definir critérios para estimar impactos do projeto para cada unidade e para a abordagem global, envolvendo as unidades (tarefas

Definir Impactos por Unidade e Definir Impactos Globais, respectivamente); ii) definir estratégias para soluções dos impactos (tarefa Definir Estratégias para Soluções dos Impactos); iii) estimar tempo de desenvolvimento de cada unidade por pacote (tarefa Estimar Tempo por Pacote); e iv) estimar o tempo total levando em conta a diferenciação de fuso-horário e a utilização de estratégias como o *follow-the-sun* (tarefa Estimar tempo total).

5.3.2.3. Artefatos de Saída

Os artefatos de software de saída são: i) pacotes; ii) lista de unidades que desenvolverão cada pacote; iii) lista de impactos globais; iv) lista de impactos por unidade; v) lista de estratégias de solução dos impactos; vi) estimativa de tempo por unidade; e vii) estimativa de tempo total.

5.3.3. Atividade Design

O objetivo é atualizar a documentação existente e desenvolver alternativas para a implementação da modificação. A Figura 5 -22 apresenta um resumo desta atividade.

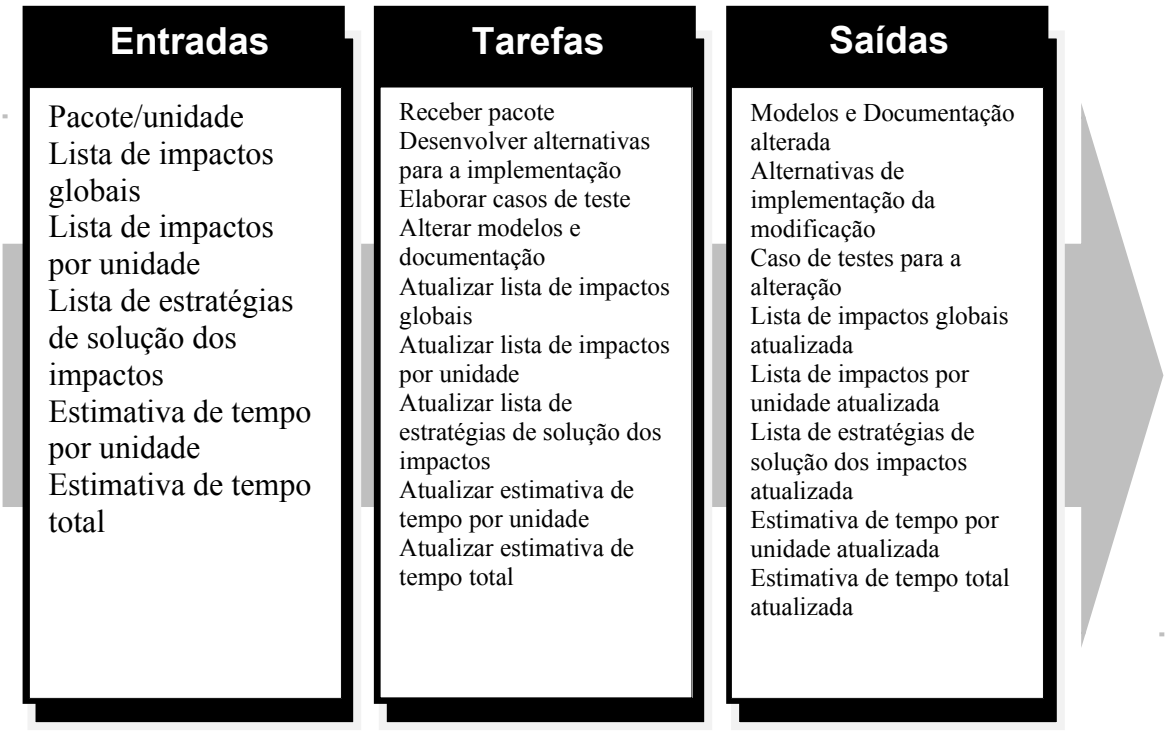

Figura 5-22 – Quadro de Entradas, Tarefas e Saídas da Atividade Design

5.3.3.1. Artefatos de Entrada

Os artefatos de entradas para esta atividade são: i) pacote/unidade; ii) lista de impactos globais; iii) lista de impactos por unidade; iv) lista de estratégias de solução dos impactos; v) estimativa de tempo por unidade; e vi) estimativa de tempo total.

5.3.3.2. Tarefa

A atividade Design trata da primeira fase onde as unidades recebem seus pacotes. Nesta atividade, cada unidade recebe o pacote da unidade responsável pelo gerenciamento das atividades globais.

Após o recebimento dos pacotes, a unidade responsável pelo pacote apresenta alternativas para a implementação da modificação a ser realizada no pacote. A alternativas devem respeitar os requisitos do software definidos na atividade Avaliação Primária.

A tarefa Elaborar Casos de Teste consiste em elaborar casos de teste[7]. Mesmo não sendo viável desenvolver casos de teste para todas as possibilidades, é preciso escolher um conjunto de condições que simule os comportamentos mais prováveis dos usuários, principalmente os erros mais comuns, para garantir a qualidade da aplicação.

As tarefas subseqüentes são: i) Alterar Modelos e Documentação (respeitando a documentação previamente existente); ii) Atualizar Lista de Impactos Globais; iii) Atualizar Lista de Impactos por Unidade; iv) Atualizar Lista de Estratégias de Solução dos Impactos; v) Atualizar Estimativa de Tempo por Unidade; e vi) Atualizar Estimativa de Tempo Total.

5.3.3.3. Artefatos de Saída

Os artefatos de saídas são: i) modelos e documentação alterada; ii) alternativas de implementação da modificação; iii) caso de testes para a alteração; iv) lista de impactos globais atualizada; v) lista de impactos por unidade atualizada; vi) lista de estratégias de solução dos impactos atualizada; vii) estimativa de tempo por unidade atualizada; e viii) estimativa de tempo total atualizada.

[7] Cada caso de teste é composto por uma condição, uma descrição dos passos a serem seguidos para realizar a verificação, a indicação do resultado esperado e as informações sobre a aprovação ou a reprovação no teste e os erros encontrados [DANTAS, 2003].

5.3.4. Atividade Implementação da Modificação

O objetivo é realizar alterações conforme estipulado na atividade Design. A Figura 5 -23 apresenta um resumo desta atividade.

5.3.4.1. Artefatos de Entrada

Os artefatos de entrada são: i) pacotes; ii) documentação alterada; iii) alternativas de implementação da modificação; iv) caso de testes para a alteração; v) estimativa de tempo atualizada por unidade; e vi) estimativa de tempo total atualizada.

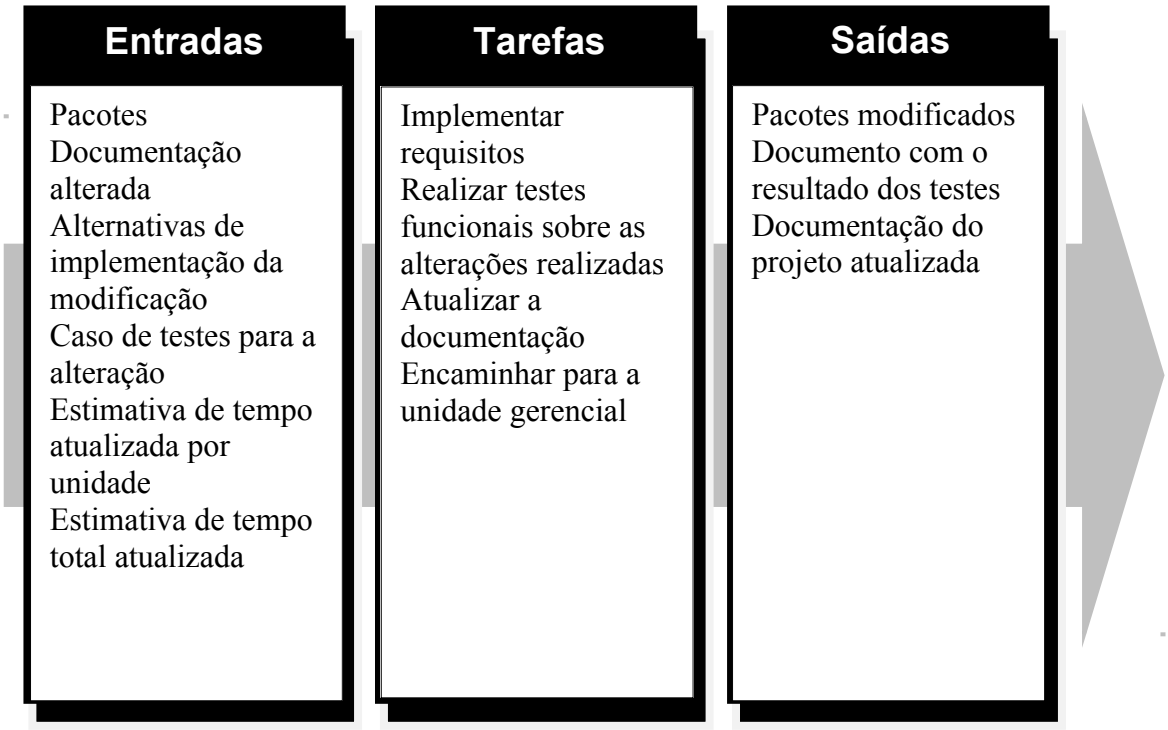

Figura 5-23 – Quadro de Entradas, Tarefas e Saídas da Atividade Implementação da Modificação

5.3.4.2. Tarefas

As tarefas devem garantir a implementação completa e correta dos novos requisitos e dos requisitos modificados (tarefa Implementar Requisitos). Para isso, as unidades envolvidas utilizam tecnologias e ferramentas para codificar os requisitos. Essa codificação será realizada sobre o pacote endereçado a cada unidade.

Durante a implementação, pode ser necessário o uso de tecnologias colaborativas para a troca de resultados e esclarecimentos quanto às decisões tomadas por outras unidades. A necessidade de comunicação dependerá de como os pacotes foram

modularizados. Uma boa modularização reduz a necessidade de comunicação, assim como baixa modularização aumenta a necessidade de comunicação.

Testes funcionais devem ser realizados nas alterações efetuadas e os resultados dos testes devem ser documentados (tarefa Realizar Testes Funcionais sobre as Alterações Realizadas). Além disso, atualizações na documentação do projeto devem ser feitas de forma a manter a adequação com a documentação original (tarefa Atualizar a Documentação). Essa medida deve ser tomada em caso da empresa que trabalha com o pacote utilizar um padrão de documentação diferente da documentação sugerida na manutenção.

Ao final da Atividade Implementação da Modificação, as saídas devem ser encaminhadas das unidades que a desenvolveram para a unidade gerencial (tarefa Encaminhar para a Unidade Gerencial).

5.3.4.3. Artefatos de Saída

Os artefatos de saída são: i) pacotes modificados; ii) documento com o resultado dos testes; e iii) documentação do projeto atualizada.

5.3.5. Atividade Integração e Testes

O objetivo é garantir que as alterações em cada pacote sejam realizadas de maneira satisfatória, sem afetar outras partes do software. A Figura 5 -24 apresenta um resumo desta atividade.

Figura 5-24 – Quadro de Entradas, Tarefas e Saídas da Atividade Integração e Testes

5.3.5.1. Artefatos de Entrada

Os artefatos de entrada são: i) pacotes modificados pelas unidades distribuídas; ii) atualizações de cada unidade no na documentação do projeto; iii) documentação com os resultados dos testes; e iv) histórico de experiências de manutenções distribuídas anteriores.

5.3.5.2. Tarefas

A tarefa inicial está em receber os pacotes de cada unidade que o desenvolveu e integrá-los ao software de forma a torná-lo novamente atômico (tarefas Receber os Pacotes e Tornar o Software Atômico). Sendo assim, o código-fonte produzido pelas diversas equipes envolvidas no projeto precisa ser integrado para evitar que o software final apresente inconsistências ou falhas. Essa integração costuma ser bastante complexa, principalmente quando as equipes usam plataformas e ferramentas diferentes (tarefa Realizar Testes de Integração).

Quando o software desenvolvido na fase de implementação possui uma boa especificação, ele facilita a integração de seu código à funcionalidade geral, evitando inconsistências e redundâncias no trabalho das equipes.

A atividade de integração trata os pacotes como componentes que podem ser vistos como uma montagem da funcionalidade. Ou seja, o código-fonte gerado na construção de um componente ou pacote não precisa ser conhecido para que ele possa ser usado ou integrado à funcionalidade geral. Seu funcionamento é encapsulado e o integrador se preocupa apenas com as interfaces de comunicação. Pois, um componente funciona como uma caixa-preta, que possui interfaces definindo os serviços que oferece e os parâmetros necessários para que o código seja executado.

Posteriormente, as seguintes atividades devem ser realizadas: i) Realizar Testes de regressão, com o objetivo de testar a adição de cada funcionalidade distribuída no software como um todo; ii) Realizar Testes de Integração, cujo objetivo é verificar se a

comunicação entre os módulos ou subsistemas da aplicação ocorre corretamente; iii) Realizar Testes de Sistema; e iv) Realizar Testes de Aceitação.

A tarefa final da atividade Integração e Testes consiste em Atualizar o Histórico de Experiências de Manutenções Distribuídas Anteriores.

5.3.5.3. Artefatos de Saída

Os artefatos de saída são: i) nova versão do produto; ii) resultado dos testes; e iii) histórico de experiências anteriores atualizado.

5.3.6. Atividade Descontinuidade do Software

O objetivo é criar um plano de descontinuidade, constando a interrupção total ou parcial do apoio ao software depois de determinado período de tempo, o arquivamento do software e da documentação associada e a transição para o novo software. A Figura 5 -25 apresenta um resumo desta atividade.

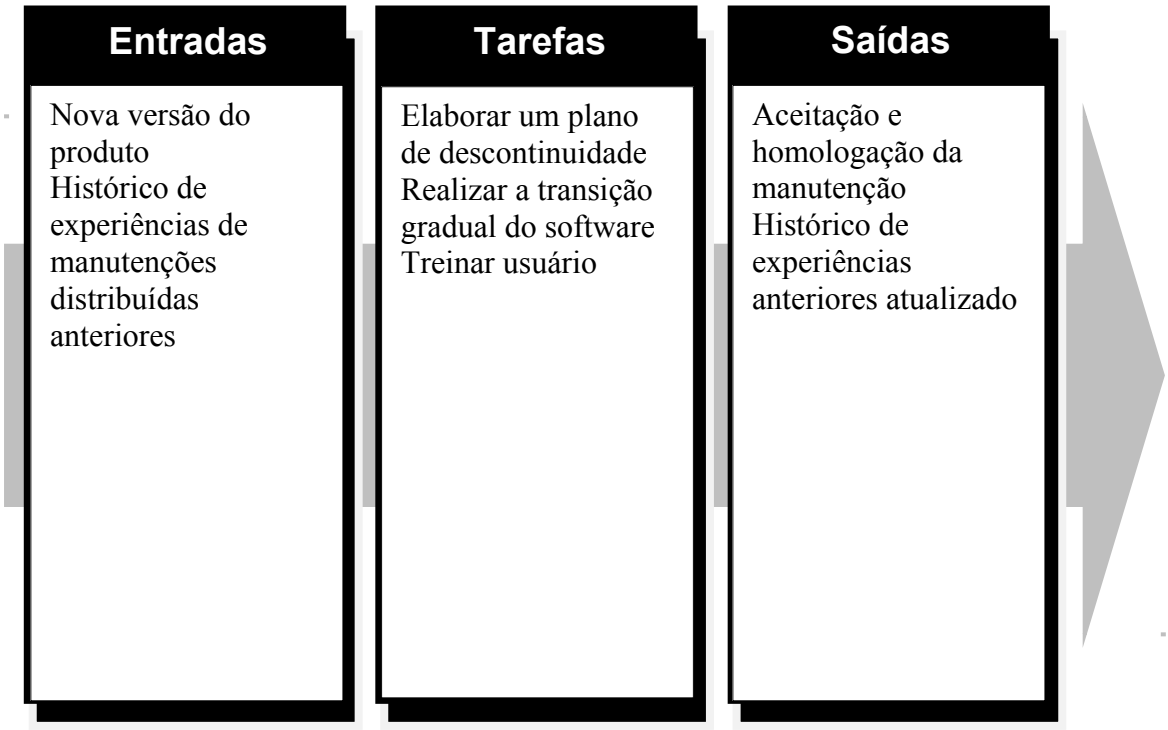

Figura 5-25 – Quadro de Entradas, Tarefas e Saídas da Atividade Descontinuidade do Software

5.3.6.1. Artefatos de Entrada

Os artefatos de entradas são: i) nova versão do produto; e ii) histórico de experiências de manutenções distribuídas anteriores.

5.3.6.2. Tarefas

A tarefa Elaborar um Plano de Descontinuidade consiste em cessar total ou parcial o suporte depois de certo período de tempo, arquivar o software e a sua documentação, parar qualquer suporte residual, migrar para o novo software, se aplicável; e disponibilizar cópias de arquivos de dados.

Após a elaboração do plano de descontinuidade, deve-se seguir para transição gradual ao novo software (tarefa Realizar a Transição Gradual do Software). Durante este período, deve ser provido treinamento de usuário, conforme especificado no contrato (tarefa Treinar usuário).

Para obter qualidade no software, é desejável que o software seja submetido à avaliação de usuários do mercado-alvo antes de ser disponibilizado ao usuário final.

5.3.6.3. Artefatos de Saída

Os artefatos de saídas são: i) aceitação e homologação da manutenção; ii) histórico de experiências anteriores atualizado.

5.4. Considerações Finais

Este capítulo apresentou uma descrição detalhada de cada atividade do processo de manutenção distribuída de software proposto. A descrição foi focada nos artefatos de software de entrada, nas tarefas e nos artefatos de software de saídas de cada atividade. Esse formato de apresentação de dados foi baseado no formato da IEEE, por causa da clareza e da facilidade do entendimento do fluxo das atividades gerado por ele.

6. CONSIDERAÇÕES FINAIS

A manutenção tornou-se inevitável por vários motivos ora discutidos neste trabalho. Dessa forma, abordou-se a manutenção de software, a qual é um fator relevante na indústria de software. Por isso, recomenda-se construir software com qualidade e com informações que possam apoiar a sua manutenção.

A seção 6.1 apresenta uma breve conclusão da relevância do trabalho. A seção 6.2 apresenta as contribuições do trabalho. A seção 6.3 faz uma síntese dos trabalhos que podem ser desenvolvidos com base nesta pesquisa, objetivando a sua continuidade.

6.1. Conclusões

Não existe um método aprovado que sistematize o processo de manutenção de software nem um grupo de regras que auxilie o controle das mudanças dentro de um sistema metodológico. Isto é, mesmo sendo uma área de extrema importância no ciclo de vida de um software, a característica de manutenibilidade ainda não apresenta um padrão que possa ser usado e aplicado. A manutenção é considerada a fase mais dispendiosa do ciclo de vida de um software e, muitas vezes, a qualidade dos códigos-fonte reparados e atualizados é baixa, podendo comprometer o seu desempenho. Enquanto as novas metodologias de desenvolvimento de software reduzem a necessidade da manutenibilidade corretiva, elas apresentam pequena influência na manutenibilidade perfectiva e evolutiva, que se trata da área mais dispendiosa.

Esta pesquisa é relevante no sentido de procurar analisar os principais problemas existentes no DDS, considerando as dimensões técnicas e não-técnicas. No entanto, o processo não se aprofundou na análise das razões que levam uma organização a adotar estratégias de distribuição do desenvolvimento de software tampouco em níveis maturidade de DDS.

Com o aumento do número de empresas que distribuem seus processos de desenvolvimento de software ao redor do mundo, visando ganhos de produtividade, redução de custos e melhorias na qualidade, o desenvolvimento distribuído tem atraído grande número de pesquisas na área de engenharia de software nos últimos anos. Os engenheiros de software têm reconhecido a influência desta nova forma de trabalho e estão

em busca de modelos que facilitem o desenvolvimento de software com equipes geograficamente distantes. Engenheiros, gerentes e executivos têm enfrentado diversos desafios e dificuldades em diferentes níveis, desde os fatores técnicos até os não-técnicos. Este trabalho atua como um repositório de dados para o desenvolvimento distribuído de software e colabora com o DDS de modo a acrescentar uma visão comparativa das atividades de manutenção co-localizada e distribuída de software e indicar em quais pontos de processos da manutenção co-localizada há vantagens na adoção de medidas distribuídas.

O nível de detalhamento do fluxo de atividades do processo de manutenção distribuída de software proposto serve para nortear a realização da manutenção de forma distribuída, sendo que o processo atua sugerindo atividades e artefatos considerados importantes no processo de manutenção distribuída. Ainda assim, sendo desenvolvido com um forte embasamento em processos de manutenção consagrados, o fato de não ser testado em casos reais pode acarretar inconformidades ainda não identificadas durante as atividades do processo. Essas inconformidades podem ser desde falta de artefatos vitais para o processo à não adaptação das realidades exigidas pelas empresas atuantes na área.

6.2. Contribuições

A elaboração do processo de manutenção distribuída de software seguiu os padrões dos principais processos existentes na literatura além de ser embasado nos principias conceitos existentes sobre o DDS. A manutenção é necessária para a continuidade do software, logo realizá-la de forma distribuída seria uma forma de aperfeiçoar sua realização.

Esse trabalho contribuiu para área de engenharia de software através da definição de um processo na área de manutenção e, além disso, traz uma nova maneira de abordar a manutenção de software, que está sendo considerada e abordada no ambiente distribuído de produção de software.

Esse trabalho contribuiu para a sociedade científica através da publicação de um artigo denominado “Uma Proposta de Manutenção Distribuída” no II Workshop de Desenvolvimento Distribuído de Software. O artigo apresentou o estado inicial/intermediário da monografia.

6.3. Trabalhos Futuros

Alguns trabalhos podem ser desenvolvidos como continuidade:

- Implantação do processo em uma organização real que usa o DDS para medir e avaliar a sua eficiência. A implantação na organização buscaria:
    - Adequar o processo ao sistema de trabalho da empresa;
    - Adequar o processo às reais necessidades do mercado;
    - Realizar medições comparativas nos tempos de produção entre o processo de manutenção distribuída e o co-localizado;
- Refinar o processo após implantações e comparações realizadas durante as medições de sua aplicação;
- Construir templates dos documentos usados no processo.
- Definir critérios eficientes para a tarefa de quebra de pacotes;
- Estudar a viabilidade da construção de um software para o gerenciamento do processo de forma distribuída.
- Construir de um software para o gerenciamento do processo de forma distribuída visando ganhos no tempo de implantação do processo e maior controle das atividades.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT NBR13596. **Tecnologia de Informação – Avaliação de Produto de Software – Características de Qualidade e Diretrizes para o seu Uso**. ABNT, abr. 1996.

ARAÚJO, R. M. de. **Um Sistema de Suporte à Decisão em Grupos para o Desenvolvimento de Software**; II Workshop de Pesquisas de Tese em Engenharia de Software. 1993.

AUDY, J.; PRIKLADNICKI, R**. Desenvolvimento Distribuído de Software – Desenvolvimento de Software com Equipes Distribuídas**. **2007.**

BECK, K. **Extreme Programming Explained-Embrance Change**. Addison-Wesley, 2000.

BRERETON, P., Budgen, D. ***Component-based Systems: A Classification of Issues,*** IEEE Computer, Vol. 33, No. 11, November 2000.

BRUSAMOLIN, V. **Manutenibilidade de Software**. In: Revista Digital Online, v. 2, jan. 2004.

CANNING, R. **The Maintenance Iceberg.** EDP Analyzer, v. 10, n. 10, Oct. 1972.

CARMEL, E. ***Global Software Teams: Collaborating Across Borders and Time Zones,*** Prentice Hall, Upper Saddle River, NJ 1999.

CARMEL, E.; TIJA, P. **Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce.** Cambridge University Express. 2005.

CÔRTES, M. L.; CHIOSSI, T. C. S. **Modelos de Qualidade de Software**. Editora da UNICAMP, 2001, 148p.

COSTA, H. A. X. **Critérios e Diretrizes de Manutenibilidade para a Construção do Modelo de Projeto Orientado a Objetos**. Tese (Doutorado). Escola Politécnica – Universidade de São Paulo, São Paulo, SP, 2005, 199p.

CROSBY, P. B. **Quality Is Free – The Art of Making Quality Certain**. McGraw-Hill, 1979, 352p.

DANTAS, V.F.. **WideWorkWeb – Uma Metodologia para o Desenvolvimento de Aplicações Web num Cenário Global.** Dissertação (Mestrado). Universidade Federal de Campina Grande, Campina Grande, PB. 2003, 168p.

DEMING, W. E. **Out of Crisis: Quality, Productivity and Competitive Position**. MIT Press, 1982, 507p.

EISENMANN, A. L. K.; ZACCONI, L. T.; MARTINS, R. A. G. C. **Preparação POSCOMP**. 2ª ed. Central de Ensino para Graduados, 2005.

FEIGENBAUM, A. V. **Controle da Qualidade Total**. Makron Books, 1994. 210p.

FREITAS, A. V. P. (2005) **APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software**. Faculdade de Informática – UFRS – RS – Brasil. Dissertação.

GOMES, N. S. **Qualidade de Software: uma Necessidade**. Ministério da Fazenda, 2001.

HERBSLEB, J. D.; MOITRA, D. **Global Software Development**. IEEE Software, 2001.

HYDER, E. B.; HESTON, K. M.; PAULK, M. C. **TbesCM-SP v2.01: The Sourcing Capability Model for Service Providers (SCM-SP) v2.01 – Model Overview.** Disponivel em: <http://itsqc.cs.cmu.edu/>. Carnegie Mellon University, Pittsburgh, 2006.

IEEE Standard for Software Maintenance, Sponsor, Software Engineering Standards Committee of the IEEE Computer Society Approved 25 June 1998.Empirical analysis of massive maintenance processes De Lucia, A.; Pannella, A.; Pompella, E.; Stefanucci, S.;Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on11-13 March 2002 Page(s):5 – 14.Digital Object Identifier 10.1109/CSMR.2002.995785.

ISO Std. 8402. International Standard ISO/CD 8402-1. **Quality Concepts and Terminology, Part One: Generic Terms and Definitions**. International Organization for Standardization, dez. 1990.

ISO Std. 9126. **Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their use**. International Organization for Standardization, dez. 1991.

ISO Std. 9126. **Software Enginnering – Product Quality Part 1: Quality Model**. International Organization for Standardization, jun. 2001.

JUNG, C. F. **Metodologia Para Pesquisa & Desenvolvimento: Aplicada a Novas Tecnologias, Produtos e Processos**. Rio de Janeiro. Axcel Books. 2004.

JURAN, J. M.; GRYNA JR., F. M. **Quality Planning an Analysis From Product Development Through Use**. McGraw-Hill, 1970. 684p.

KAN, S. H. **Metrics and Models in Software Quality Engineering**. Addison-Wesley, 1995, 344p.

KAROLAK, D. W. **Global Software Development – Managing Virtual Teams and Environments**. Los Alamitos, IEEE Computer Society, USA, 1998, 159p.

KRUCHTEN, P. **The Rational Unified Process: An Introduction.** Addison-Wesley, 2001.

LANUBILE, F.; DAMIAN, D.; OPPENHEIMER, H. **Global Software Development: Technical, Organizational, and Social Challenges.** ACM SIGSOFT Software Engineering Notes, New York, v.28, n.6, Nov. 2003.

LIEBMAM, A. **Melhoria no Processo de Software**: Implantação do MPS.BR Nível G em uma Empresa de Pequeno Porte. Monografia (Graduação em Ciência da Computação) – Universidade Federal de Lavras, Lavras. 2006.

LUCIA, A. de; Pannella, A.; Pompella, E.; Stefanucci, S.(2002) **Empirical Analysis of Massive Maintenance Processes. Software Maintenance and Reengineering. Proceedings**. Sixth European Conference on 11-13 Page(s): 5 – 14.

MAIDANTCHICK, C.; ROCHA, A. da. **Managing a Worldwide Software Process. In: International Workshop on Global Software Development**, ICSE, 2002, Orlando, Florida, 2002.

MARTINS E. **Manutenção e Ferramentas CASE**. notas de curso, 1998.

MONTALVÃO A. **Biblioteca do líder**, 6 ed. São Paulo, Novo Brasil Editora Ltda, 1979.

MSF. Microsoft Solutions Framework, www.mocrosoft.com/msf, 2007.

NBR ISO/IEC 12207 (1998) **Tecnologia de informação – Processos de ciclo de vida de software**.

OLIVEIRA, J. F. de. **Diagnóstico de uma Fábrica de Software Visando a Implantação do Processo Garantia da Qualidade do MPS.BR Nível F** / Janaina Faria de Oliveira – Minas Gerais, 2007, 90p. Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação

OPEN Consortium. **The open website,** 2003. Disponível em http://www.open.org.au.

PESSOA, M. S. P. **Introdução ao CMM - Modelo de Maturidade da Capacidade de Processo de Software.** Lavras: UFLA/FAEPE, 2003. Curso de Pós-graduação "Lato Sensu" à Distância: Melhoria de Processo de Software. 77 p.

PFLEEGER. S. L. **Software Engineering Theory and Practice**. Prentice Hall, 2001.

PILATTI, L.; AUDY, J. L. N.; PRIKLADNICKI, R. **Software Configuration Management over a Global Software Development Environment: Lessons Learned from a Case Study. In: First International Workshop on Global Software Development for the Practitioner**, 2006, Shangai. Global Software Development for the Practitioner. USA: ACM Press, 2006. p. 45-50.

PMBOK **A Guide to the Project Management Body of Knowledge.** Project Management Institute, 2000 Edition.

PRESSMAN, R. S. **Engenharia de Software**.McGraw-Hill, 2006.

PRIKLADNICKI, R.; AUDY, J. L. N. **Um Modelo de Referência para Desenvolvimento Distribuído de Software.** Dissertação de mestrado. Pontifícia Universidade Católica do Rio Grande do Sul, Faculdade de Informática Programa de Pós-Graduação em Ciência da Computação. WTES 2003.

PRIKLADNICKI, R.; AUDY, J. L. N.; EVARISTO, R.. **Distributed Software Development: Toward an understanding of the relationship between project team, users and customers.** In: ICEIS, 2003, Angers. Proceedings… França, p. 417-423, Abr. 2003.

PRIKLADNICKI, R.; LOPES, L.; AUDY, J. L. N.; EVARISTO, R. **Desenvolvimento Distribuído de Software: um Modelo de Classificação dos Níveis de Dispersão dos Stakeholders.** SBSI 2004.

PRINCE2. **Office Government Commerce, Managing Successful Projects with PRINCE2.** The Stationery Office, 2005.

ROCHA, A. R. **Qualidade de *Softwtare*: Processo ou Produto**. In: Encontro da Qualidade e Produtividade em *Software* (EQPS'2002). Petrópolis, RJ, ago. 2002.

ROTHERY, B. **ISO 9000**. Makron Books, 1993.

ROBILLARD, P. N.; KERZAZI, N.; TAPP, M.; HMIMA, H.**; Outsourcing Software Maintenance: Processes, Standards & Critical Practices.** Electrical and Computer

Engineering, 2007. CCECE 2007. Canadian Conference on 22-26 April 2007 Page(s):682 – 685 Digital Object Identifier 10.1109/CCECE.2007.175

ROBINSON, M.; KALAKOTA, R. **Offshore outsourcing: business models, ROI and best practices.** Mirvar Press, 2004.S

CHWABER, K. **Agile project management with SCRUM.** Microsoft Press, 2004.

SOMMERVILLE, I. Engenharia de Software. São Paulo: Addison Wesley, 2003.

TOLEDO, J. C. **Qualidade Industrial – Conceitos, Sistemas e Estratégias**. Atlas, 1987, 184p.

TOTLAND, T.; CONRADI, R. (1995) **A Survey and Comparison of Some Research Areas Relevant to Software Process Modeling**. Workshop on Software Process Technology. 65-69p.

VASCONCELOS, A. M. L de; MACIEL, T. M. de M. **Introdução à engenharia de software e aos princípios de qualidade.** Lavras: UFLA/FAEPE, 2003. Curso de Pós-graduação "Lato Sensu" à Distância: Melhoria de Processo de Software. 104 p.