Uma Proposta de Processo de Manutenção Distribuída

André Fonseca Amâncio[1], Heitor Augustus Xavier Costa[2], Antônio Maria Pereira de Resende[3], Fábio Fagundes Silveira[4]

[1, 2, 3] PqES – Grupo de Pesquisa em Engenharia de Software – Departamento de Ciência da Computação (DCC) – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – CEP 37.200-000 – Lavras – MG – Brasil

[4] Departamento de Ciência e Tecnologia (DCT) – Universidade Federal de São Paulo (UNIFESP) – CEP 04020-041 – São José dos Campos – SP – Brasil

[1]afancio@comp.ufla.br, [2]heitor@ufla.br, [3]tonio@dcc.ufla.br, [4]fsilveira@unifesp.br

Abstract*. The area of software engineering have been required new demands due changes elapsed by internalization and 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 paper intends to contribute with the growth of software process technology, with the development of distributed maintenance process. To the development of the work techniques and methods of software maintenance existent in the literature were analyzed to construct the knowledge about actors, activities and artifacts of a software maintenance process.*

Resumo*. A área de Engenharia de Software tem sido alvo de novas demandas em função de mudanças decorrentes da internacionalização e 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 direção. Este trabalho visa contribuir com o crescimento da Tecnologia de Processos de Software, com a proposta de um processo de manutenção distribuída. Para isso, foram analisados técnicas e métodos que compõem os processos de manutenção de software existentes na literatura, que contribuíram para o entendimento das principais atividades, atores e artefatos existentes nestes processos.*

1. Introdução

Estudos em Engenharia de Software, acerca de mecanismos para gerenciar processos de software, constituem uma subárea denominada Tecnologia de Processos de Software [Freitas, 2005]. Esta subárea envolve um conjunto de linguagens, métodos, arquiteturas e ferramentas para apoiar a gerência de processos de software [Totland; Conradi, 1995].

Desde o início dos anos 90, uma tendência no desenvolvimento de software tem despertado atenção dos pesquisadores: distribuição do desenvolvimento. Este fenômeno é reflexo de mudanças sócio-econômicas, que têm levado organizações a distribuírem geograficamente recursos e investimentos, visando aumentar produtividade, melhorar qualidade e reduzir custos no desenvolvimento de software [Audy; Prikladnicki, 2007].

A distribuição física das equipes agrava problemas existentes e inerentes a gerência do desenvolvimento. Diferenças culturais, de linguagem, de fuso-horário entre outros aspectos aumentam a complexidade na comunicação, na coordenação e no controle durante o desenvolvimento [Lanubile *et al*., 2003; Pilatti *et al*., 2006]. O uso de recursos tecnológicos pode contribuir para solucionar vários problemas causados por razões sócio-culturais encontrados no desenvolvimento distribuído de software (DDS) [Maidantchick; Rocha, 2002].

O objetivo deste artigo é apresentar uma proposta de processo de manutenção distribuída. Para isso, foram estudados técnicas e métodos de DDS, compilando resultados presentes na literatura, bem como os processos de manutenção tradicionais.

O artigo está organizado da seguinte forma: a seção 2 discorre sobre processo de manutenção em âmbito geral; a seção 3 apresenta descrição resumida dos processos de manutenção: Processo de Manutenção da Norma ISO/IEC 12207, Processo de Manutenção da IEEE e Processo Massivo Adaptativo de Manutenção; a seção 4 contextualiza e descreve sucintamente o DDS; a seção 5 apresenta a proposta de processo de manutenção distribuída; e a Seção 6 os resultados e discussões.

2. Processo de Manutenção de Software

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. Esta atividade é considerada não interessante pelos desenvolvedores que preferem construir um software [Robillard *et al.*, 2007].

Entender as atividades de manutenção permite identificar pontos críticos e tomar ações apropriadas para aumentar a eficiência do processo de manutenção. Os processos de manutenção podem ser simples, no qual eles compartilham as mesmas atividades conceituais: identificar anomalias, encontrar causa, implementar modificações, testar o módulo modificado e integrar o módulo no software existente [Robillard *et al.*, 2007].

A criação de um projeto segue uma seqüência de passos para que ele seja concluído e são 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 produz uma saída esperada. Além disso, um processo envolve um conjunto de ferramentas e técnicas, possuindo as seguintes características [Pfleeger, 2001]: i) prescreve as suas atividades; ii) usa recursos, se sujeita a prazos e produz produtos intermediários e finais; iii) pode ser composto por sub-processos ligados ao processo de alguma forma, ou seja, precisa ser definido como hierarquia de processos, organizado de forma que cada sub-processo tenha seu próprio modelo; iv) possui atividades com critérios de entrada e saída, sendo possível saber quando uma atividade começa e termina; v) possui atividades organizadas em seqüência, de modo que fique claro quando uma atividade está relacionada com outra; vi) possui guia de princípios que explica os objetivos de cada atividade; e vii) possui prazos e controles a serem aplicados às atividades, aos recursos ou ao produto.

Para Costa (2005), o desenvolvimento de software está completo quando ele foi entregue e é utilizado 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 a idéia equivocada, segundo Costa (2005), que a manutenção de

software é semelhante à manutenção de hardware que consiste em substituir peças gastas e/ou usar técnicas para prolongar a sua vida útil. No entanto, o software não degrada com o tempo e a manutenção está associada com alteração de software.

A manutenção de software é considera a fase mais dispendiosa do ciclo de vida do software sendo que muitas vezes a qualidade dos códigos reparados e atualizados é baixa e pode comprometer o seu desempenho. A demanda de esforço e de tempo decorre da falta de disciplina e do controle nas atividades iniciais do processo de desenvolvimento de software. As decisões não adequadas de projeto podem influenciar negativamente o software e aumentar a complexidade de sua manutenção [Araújo, 1993].

De modo geral, pode-se organizar o processo de desenvolvimento de software em três grandes fases [Sommerville, 2007; Pressman, 2006; Pfleeger, 2001]: i) Definição; ii) Desenvolvimento; e iii) Manutenção. A fase Manutenção, foco deste trabalho, é caracterizada pela realização de alterações para corrigir erros residuais, incluir novas funções exigidas pelo cliente e/ou adaptar o software a novas configurações do ambiente onde ele está implantando.

O processo de manutenção está definido como uma parte fundamental do ciclo de vida do software e, a partir dele, é possível realizar correções, aperfeiçoamentos, adaptações 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 desenvolvimento.

3. Tipos de Processo de Manutenção de Software

3.1. Processo de Manutenção da Norma ISO/IEC 12207

A Figura 1 mostra o fluxo de atividades do Processo de Manutenção da norma 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 esquerdo), o nome (retângulo azulado), os atores (boneco à esquerda) e os artefatos de saídas (gravuras cinza – lado direito). O processo de manutenção da norma ISO/IEC 12207 contém as seguintes atividades:

- **Implementação do Processo**. Preparar, documentar e executar um plano de manutenção;
- **Análise do Problema e da Modificação**. Analisar o Pedido de Manutenção e o Relatório do Problema, verificando o impacto da manutenção na empresa, no software e nos sistemas com os quais interage. Além disso, é preciso determinar o tipo de manutenção, o alcance e as conseqüências das alterações e propor soluções para implementar a modificação;
- **Implementação da Modificação**. Realizar alterações conforme recomendações da Engenharia de Software aplicáveis durante o desenvolvimento;
- **Revisão/Aceitação da Manutenção**. Garantir que as alterações foram realizadas satisfatoriamente de acordo com o especificado no contrato;
- **Migração**. Preparar e realizar um plano de migração, caso seja necessário migrar o software para outra plataforma;
- **Descontinuidade do Software**. Preparar e realizar um plano de descontinuidade do software que envolve 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.

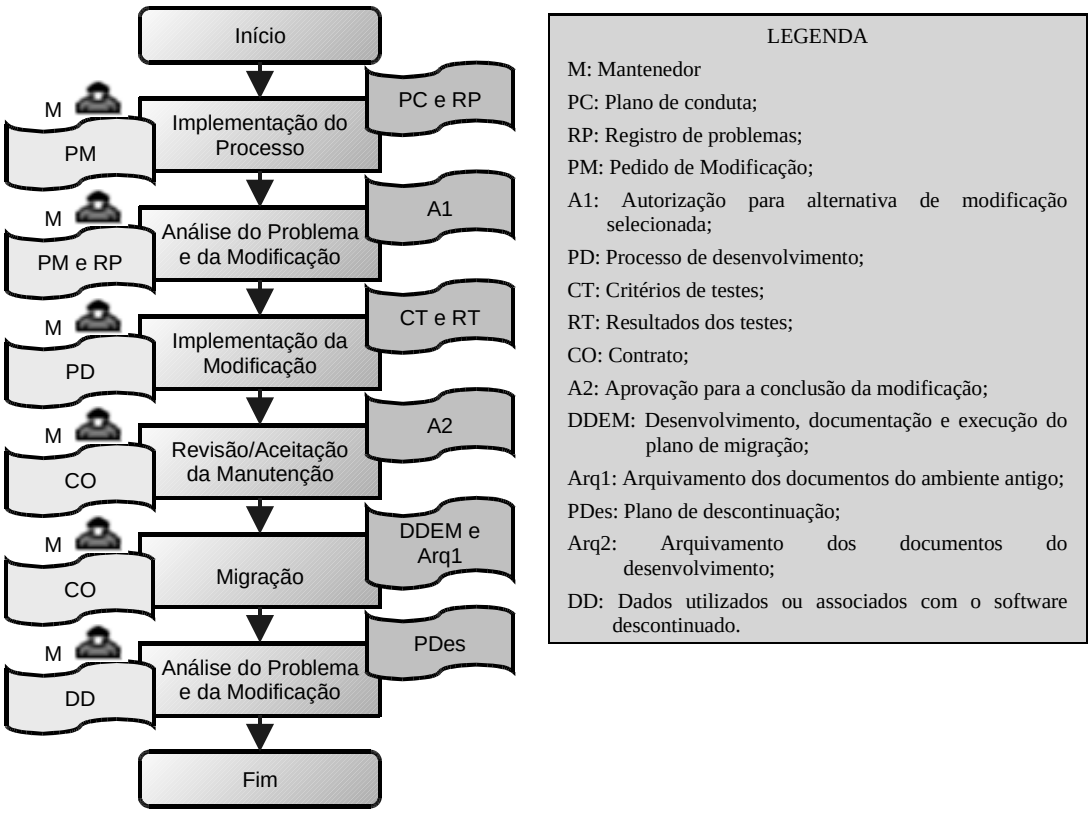

Figura 1 – Atividades do Processo de Manutenção da Norma ISO/IEC 12207

3.2. Padrão de Processo de Manutenção da IEEE

O processo de manutenção da IEEE [IEEE, 2002] é composto de seis atividades principais: i) Identificação, Classificação e Priorização do Problema/Modificação; ii) Análise; iii) Projeto; iv) Implementação; v) Teste de Sistema; e vi) Instalação. A Figura 2 apresenta a convenção adotada pela IEEE para representar as atividades que descrevem entradas, controle, processos associados e saída.

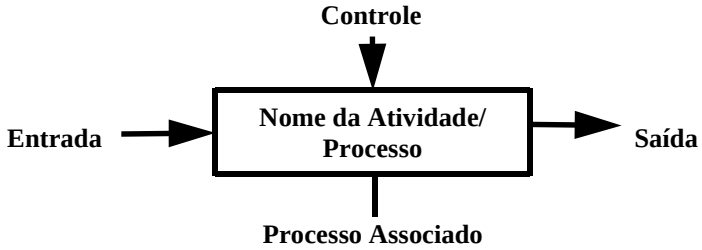

Figura 2 – Representação das Atividades do Processo de Manutenção da IEEE

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

- **Identificação do Problema.** O artefato de entrada é Pedido de Manutenção. Os processamentos são Atribuir id ao Pedido, Classificar quanto ao Tipo de Manutenção, Determinar Prioridade e Encaminhar Pedido para o Grupo de Controle de Configuração. O controle é Cadastrar Pedido no Banco de Dados. Os artefatos de

saídas são Relatório de Rejeição da Alteração ou Aviso de Incorporação ao Banco de Dados e Pedido de Manutenção Validado;

- **Análise**. As entradas são Pedido de Manutenção Validado e Registro no Banco de Dados. Os processos associados são Analisar Impacto da Alteração, Estudar Alternativas de Solução, Avaliar Custos/Recursos/Prazos e Aceitar ou não o Pedido. O controle é Analisar Relatório de Avaliação do Pedido. As saídas são Relatório de Avaliação do Pedido e Relatório de Rejeição ou Proposta de Alteração;
- **Projeto**. As entradas são Relatório de Avaliação do Pedido, Proposta de Alteração e Itens da Linha Básica a serem Alterados. Os processos associados são Alterar Modelos e Documentos, Preparar Casos de Teste para as Alterações, Identificar Testes de Regressão Aplicáveis e Atualizar Proposta de Alteração. O controle é Revisar Proposta de Alteração e Itens Alterados. As saídas são Proposta de Alteração Revisada, Descrição dos Testes e Itens Alterados;
- **Implementação**. As entradas são Itens Alterados, Proposta de Alteração Revisada e Código dos Itens a serem Alterados. Os processos associados são Alterar o Código, Realizar Testes de Unidade, Realizar Testes de Integração e Atualizar Modelos e Documentos. O controle é Revisar Itens Alterados. As saídas são Código dos Itens Alterados, Itens Alterados, Relatório dos Testes Realizados e Versão do Produto Preliminar;
- **Testes**. As entradas são Itens a serem Alterados, Itens Alterados e Descrição dos Testes. Os processos associados são Realizar Testes de Regressão, Realizar Testes de Sistemas e Realizar Testes de Aceitação. Os controles são Revisar Itens Alterados e Realizar Auditoria de Configuração. As saídas são Nova Versão do Produto, Cópias dos Itens a serem (Re)Instalados junto ao Cliente e Aviso de Incorporação da Alteração à Linha Básica;
- **Instalação**. As entradas são Cópias dos Itens a serem (Re)Instalados e Aviso de Incorporação da Alteração à Linha Básica Instalação. O processo associado é Instalar Cópias. O controle é Aceitar Alterações. A saída é Aviso de Aceitação.

3.3. Processo Massivo Adaptativo de Manutenção

A Figura 3 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:

- **Portfólio e Avaliação.** O uso do portfólio é analisado e decomposto em aplicações e *work-packets*. Essa atividade é conduzida por analistas de software juntamente com seus diferentes proprietários. A decomposição do software em *work-packets* abre caminho para o Processo Massivo Adaptativo de Manutenção: um subprojeto específico com pequena equipe pode ser instanciado para cada *work-packet*;
- **Fase de Análise**. Consiste em identificar impactos. O ponto de impacto inicial é identificado, consistindo nos pontos do programa que confrontam diretamente como a requisição de manutenção; em seguida, o processo continua com os novos pontos de impacto;
- **Fase de Design**. Consiste em identificar impactos causados pelo *work-packet* e definir soluções estratégicas a serem adotadas. Os impactos devem ser analisados e

classificados. Para cada impacto, uma solução estratégica é selecionada; se não há soluções disponíveis a ser aplicada no impacto, uma solução precisa ser definida especialmente para ele. Impactos não previstos aumentam o custo dos *work-packets*, mas eles representam um pequeno percentual dos impactos;

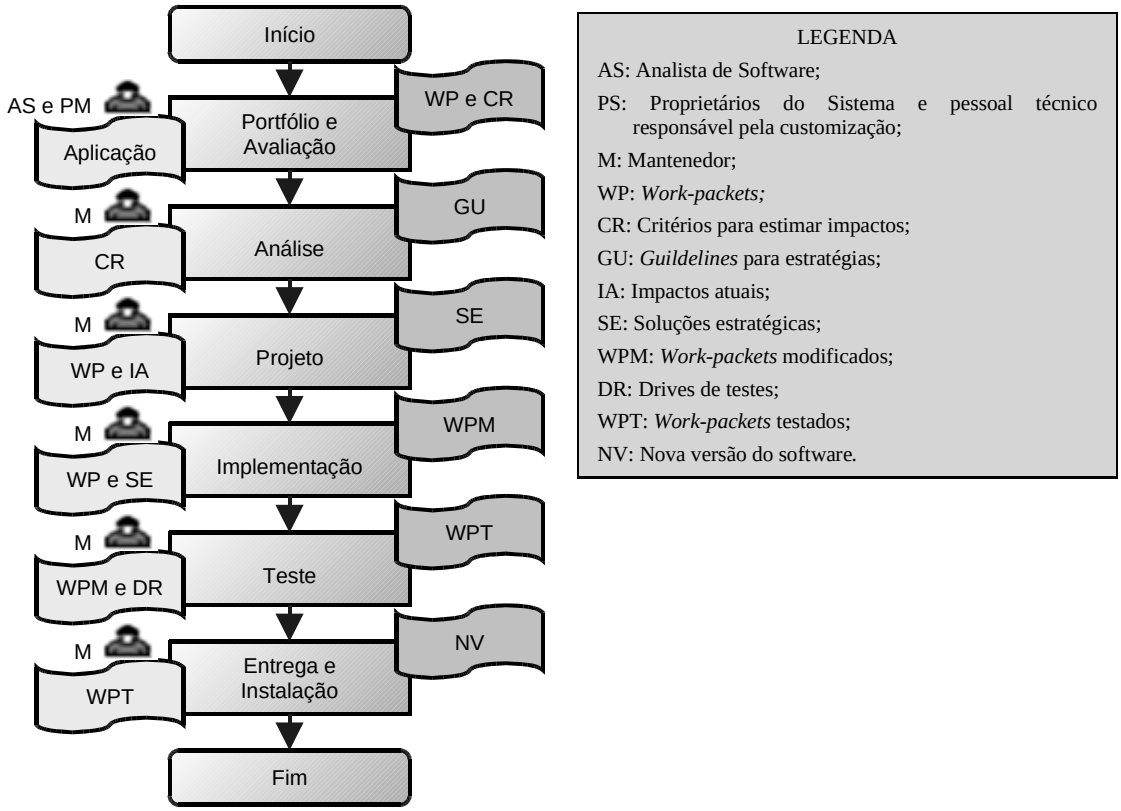

Figura 3 – Atividades do Processo Massivo Adaptativo de Manutenção

- **Fase de Implementação**. Consiste em modificar os *work-packets* com o uso da solução correspondente para cada impacto;
- **Fase de Testes**. Consiste em realizar testes em cada unidade de software; no entanto, isso pode ser feito de forma mais eficiente se os fragmentos de código próximos do impacto forem extraídos e testados usando *drivers* apropriados. Essa estratégia é chamada de teste de isolamento, sendo eficiente em caso de impactos previstos onde os *drivers* podem ser construídos previamente;
- **Entrega e Instalação**. Consiste em entregar e instalar a nova versão do software.

4. Desenvolvimento Distribuído de Software

A crescente busca por maior competitividade tem levado as empresas a adotarem o DDS, 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 do software aos clientes [Herbsleb (2001)].

Além do desenvolvimento de baixo custo, as empresas buscam investir em DDS na possibilidade de obter maior qualidade no processo de desenvolvimento, recursos em âmbito global, aumento da produtividade e diminuição dos riscos [Audy; Prikladnicki, 2007]. Alguns fatores contribuíram significativamente para acelerar o seu surgimento e

a sua aceitação. Dentre estes fatores, podem ser citado: i) necessidade de recursos globais para uso a qualquer hora; ii) proximidade com o mercado local; iii) incentivos fiscais; iv) soluções globais; v) *Time-to-market*; e vi) *Follow-the-sun* [Carmel (2005) *apud* Audy; Prikladnicki, 2007].

Quando relacionado ao desenvolvimento tradicional de software, de acordo com Audy; Prikladnicki (2007), 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 custo-benefício); ii) questões culturais (valores, princípios, etc. entre equipes distribuídas); iii) questões técnicas (infra-estrutura tecnológica e comunicação); e iv) questões de gestão de conhecimento (criação, armazenamento, processamento e compartilhamento de informações).

Alguns autores [Audy; Prikladnicki, 2007; Karolak, 1998] apontam um conjunto de razões que motivam o crescente número de organizações a desenvolverem software de forma distribuída. As principais são:

- **Demanda e custo.** A demanda por serviços de software tem superado a disponibilidade de pessoas que os realizam. Como conseqüência, o custo do profissional aumentou. Outro fator agravante está relacionado à mão de obra vinda de outros países, pois o visto de entrada acaba antes do final do ano fiscal, obrigando as empresas a realizarem novas contratações. Além disso, pode ocorrer disponibilidade de recurso a um custo mais baixo em outras localidades, favorecendo a implantação da produção naquela localidade;
- **Rapidez de resposta ao mercado.** Devido ao *Time-to-market*, muitas empresas vêem vantagens do DDS em equipes distribuídas como forma de realizar o desenvolvimento *follow-the-sun* e reduzem o tempo de entrega;
- **Mercado e presença 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 atingirem o mercado global e ficarem próximas de seus consumidores;
- **Rigor e experiência no desenvolvimento.** Em equipes de desenvolvimento co-localizadas, há tendência do uso de mecanismos informais para a comunicação. No caso das equipes de DDS, há tendência para melhorar a documentação e o uso de ferramentas de comunicação e de colaboração. Cada equipe co-localizada adquire experiência específica, propiciando diferencial para que o desenvolvimento seja 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 devido às diferenças do modo de pensar e de projetar. Além disso, a sinergia cultural somada à demanda e aos desafios envolvidos amplia as capacidades de aprendizado da organização;
- **Escala.** Centros de desenvolvimento de software quando ficam grandes se tornam difíceis de gerenciar. Com isso, é necessário distribuir o desenvolvimento para atender a demanda necessária.

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 desafio para a área de Engenharia de Software.

5. Proposta de Manutenção Distribuída

A Figura 4 apresenta um fluxograma de atividades como resultados iniciais/intermediários do processo de manutenção distribuída proposto, ressaltando a idéia das diversas formas de dispersão das equipes envolvidas no DDS.

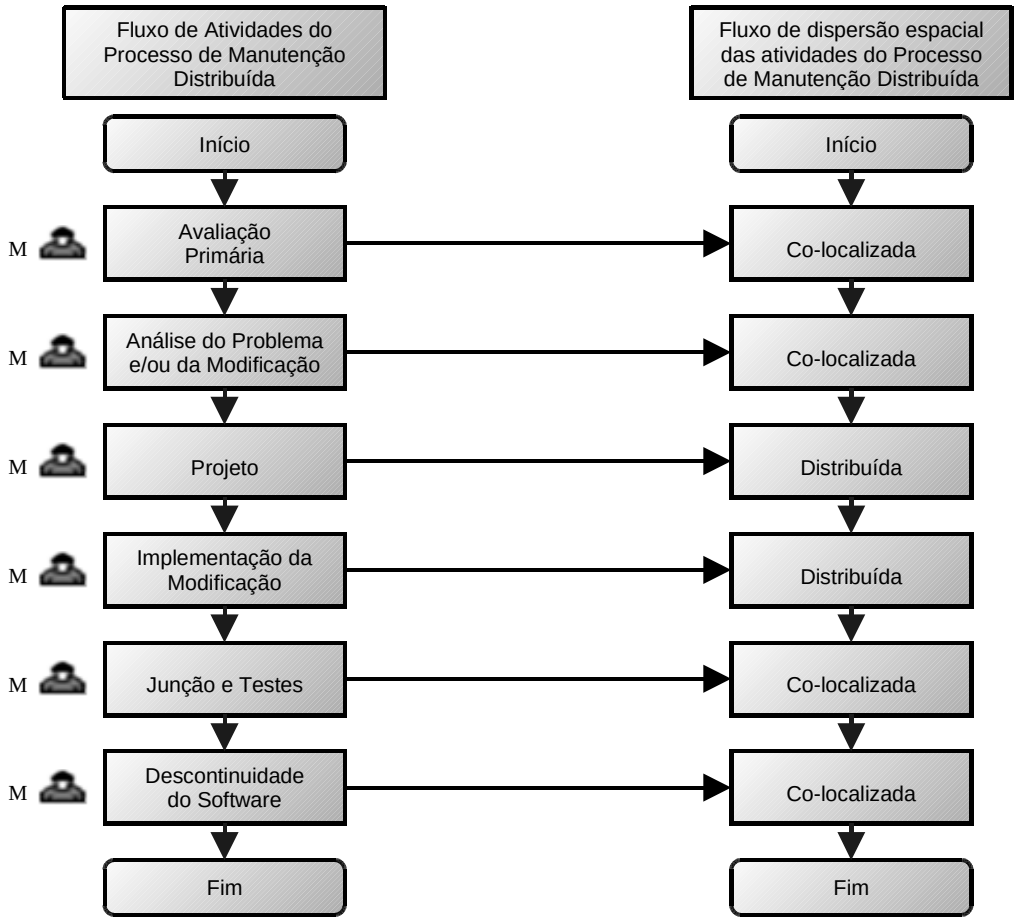

Figura 4 – Atividades do Processo de Manutenção Distribuída Proposto

Além disso, esta figura mostra onde as atividades devem ser realizadas, ou seja, Co-localizada significa a equipe responsável geral pelo projeto e Distribuída significa as

equipes colaboradoras no processo de manutenção distribuída. A decisão de classificar cada atividade como Co-localizada ou Distribuída é embasada no grupo de tarefas realizadas pela atividade em questão.

A atividade Avaliação Primária permite realizar uma ponderação da solicitação de manutenção e precisa ser realizada enquanto o problema/solicitação de manutenção se encontra em sua forma atômica. Em seguida, a atividade Análise do Problema e/ou da Modificação permite melhor compreensão do problema a ser atacado mediante a solicitação da manutenção. 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. 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. A atividade Implementação da Modificação é realizada nos pacotes pelas equipes distribuídas e responsáveis por realizar a tarefa de manutenção. A atividade Junção 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 junção 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. 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.

6. Resultados e Discussões

Há poucos métodos aprovados que sistematizem o processo de manutenção de software e poucos grupos de regras que auxiliem o controle das mudanças, dentro de um sistema metodológico. Isto é, mesmo sendo uma área de extrema importância, a manutenibilidade ainda não apresenta um padrão, que possa ser usado e aplicado. Ela é considerada como a fase mais dispendiosa do ciclo de vida de um software e a qualidade dos códigos reparados e atualizados apresenta-se 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.

Com o aumento do número de empresas distribuindo seus processos de desenvolvimento de software ao redor do mundo, visando ganhos de produtividade, redução de custos e melhorias na qualidade, o DDS 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. Além dos engenheiros, gerentes e executivos têm enfrentado diversos desafios e dificuldades em diferentes níveis (fatores técnicos e não-técnicos). Este trabalho apresenta uma contribuição para esta nova forma de trabalho e para a fase mais árdua e custosa do ciclo de vida do software: manutenção.

Como continuidade deste trabalho será detalhado e descrito o processo proposto, assim como serão definidos os artefatos utilizados e produzidos de cada atividade do

processo de manutenção distribuída. Além disso, vislumbra a possibilidade de implantar o processo em uma das organizações que usam o DDS para medir e avaliação a sua eficiência.

Referências

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

Audy, J. L. N.; Prikladnicki, R. (2007) Desenvolvimento Distribuído de Software – Desenvolvimento de Software com Equipes Distribuídas. Campus.

Carmel, E.; Tija, P. (2005) Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce. Cambridge University Express.

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

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.

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

Herbsleb, J. D.; Mockus, A.; Finholt, T. A.; Grinter, R. E. (2001) An Empirical Study of Global Software Development: Distance and Speed. Proceedings of the 23rd International Conference on Software Engineering. 81-91p.

IEEE Standard for Software Maintenance (1998), Sponsor, Software Engineering Standards Committee of the IEEE Computer Society.

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

Lanubile, F.; Damian, D.; Oppenheimer, H.(2003) Global Software Development: Technical, Organizational, and Social Challenges. ACM SIGSOFT Software Engineering Notes, New York, v.28, n.6, Nov.

Maidantchick, C.; Rocha, A da. (2002) Managing a Worldwide Software Process. In: International Workshop on Global Software Development, International Conference on Software Engineering. Orlando, Florida.

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

Pfleeger, S. L. (2001) Software Engineering Theory and Practice. 2ª ed. Prentice Hall.

Pilatti, L.; Audy, J. L. N.; Prikladnicki, R. (2006) 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, Shangai. Global Software Development for the Practitioner. ACM Press. p. 45-50.

Pressman, R. S. (2006) Engenharia de Software. 6ª ed. McGraw-Hill.

Robillard, P. N.; Kerzazi, N.; Tapp, M.; Hmima, H. (2007) Outsourcing Software Maintenance: Processes, Standards & Critical Practices. Electrical and Computer Engineering. Page(s): 682 – 685

Sommerville, I. (2007) Software Engineering. 8ª ed. Addison-Wesley.

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.