Uma Experiência na Adaptação do RUP em Pequenas Equipes de Desenvolvimento Distribuído

Rodrigo Rocha, Daniel Arcoverde, Rebeka Brito, Bruno Arôxa, Catarina Costa, Fabio Q. B. da Silva, Jones Albuquerque, Silvio Romero de Lemos Meira

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851, Cidade Universitária – 50.732-970 – Recife – PE – Brasil
{rgcr, dfa, rsb2, bma2, csc, fabio, joa, srlm}@cin.ufpe.br

Abstract. *Distributed Software Development is becoming a common practice in the business environment. However, there is still the need for practical experiences related to the adoption and adaptation of software processes to support distributed development, illustrating problems and solutions found. This article presents one such experience of adapting RUP software development process for a small distributed software development project and reports on problems identified during the development and how they have been solved.*

Resumo. *O Desenvolvimento Distribuído de Software está em ascensão e grandes empresas já estão utilizando este ambiente. No entanto, ainda faltam experiências práticas capazes de apresentar possíveis soluções para a adoção/adaptação de um Processo de Software e o seu real aproveitamento, ilustrando os problemas encontrados por cada grupo desenvolvedor e ter conhecimento das soluções utilizadas. Este artigo apresenta a experiência de adaptar o Processo de Desenvolvimento RUP para um Pequeno Time utilizando o Desenvolvimento Distribuído de Software. Relata os principais problemas identificados no decorrer do Projeto e como os mesmos foram resolvidos em cada fase do Processo.*

1. Introdução

É notável que a evolução do software tenha ocorrido de forma acelerada, de maneira que diversas áreas do conhecimento têm reconhecido a sua importância como posição estratégica perante o mercado. Carmel [1998] cita que nas últimas décadas a globalização dos negócios impactou também a indústria de Tecnologia da Informação. As forças econômicas transformaram mercados nacionais em mercados globais. Estas transformações não alteraram apenas o marketing da distribuição, mas também, a forma como os produtos são concebidos, construídos, testados e entregues aos clientes finais. Desta maneira, o software tem se tornado um componente estratégico para diversas áreas de negócio, mais especificamente na área de Engenharia de Software (ES), criando novas formas de cooperação e competição que vão além das fronteiras dos países [Herbsleb, 2001].

No final da década passada, visando diminuir os custos e buscando recursos mais qualificados, muitas organizações começaram a experimentar o Desenvolvimento Distribuído de Software (DDS) [Herbsleb, 2001]. Com a ascensão da utilização deste

conceito de desenvolvimento de software na indústria, o assunto passou a ser abordado em Congressos Internacionais como o IEEE *International Conference on Global Software Engineering* (ICGSE 2006-2008) e o IEEE/ACM *International Conference on Software Engineering* (ICSE). Esta transformação na forma como as empresas produzirão software tende a ser crescente. Segundo Kiel [2003], há toda razão para acreditar que as empresas serão pressionadas para adotar alguma abordagem global de desenvolvimento de software. Essas formas de DDS poderão variar desde uma simples distribuição em um mesmo país até a distribuição em países ou continentes diferentes (desenvolvimento global de software) [Prikladnicki e Audy, 2004] [Carmel, 1998].

Apesar da crescente demanda das empresas por métodos e técnicas que contribuam com o desenvolvimento de software além dos limites físicos da organização, a pesquisa na área ainda não é consolidada, sendo ainda escassa no Brasil [Prikladnicki e Audy, 2004]. Entre os diversos trabalhos existentes na área, parte está preocupada com a proposição de modelos de referência para o DDS [Karolak, 1998] [Carmel, 1999]. Todavia, tanto modelos quanto teorias necessitam ser confrontados em relação aos problemas que ocorrem em projetos reais. Neste sentido, abordagens práticas e experimentais funcionam como um campo interessante para empresas que estão avaliando as melhores formas de trabalhar neste novo modelo de desenvolvimento.

Este artigo endereça o problema apresentado acima ao relatar a experiência prática de uma pequena fábrica de software em uma disciplina de Engenharia de Software de um curso de pós-graduação em Ciência da Computação. O objetivo do artigo é apresentar e discutir os resultados e lições aprendidas na adequação do RUP na fábrica de software, considerando os principais problemas encontrados e a forma como foram resolvidos, desde a distribuição até a comunicação com o cliente. A fábrica de software relatada no artigo insere-se num contexto de DDS onde grupos de desenvolvimento se encontram fisicamente separados, mas necessitam trabalhar em conjunto para a realização de um projeto.

O trabalho está dividido da seguinte forma: a Seção 2 discute brevemente os aspectos que envolvem o Desenvolvimento Distribuído de Software, algumas de suas características e seus principais problemas; a Seção 3 apresenta o projeto que serviu como estudo de caso para este trabalho; a Seção 4 apresenta uma breve discussão sobre a definição e modelagem do processo; a Seção 5 aborda os problemas encontrados no decorrer do projeto e a evolução do processo; na Seção 6 são apresentadas algumas considerações finais referentes ao aprendizado da fábrica.

2. Desenvolvimento Distribuído de Software

Desenvolver software no mesmo espaço físico, na mesma organização ou até no mesmo país, tem se tornado cada vez mais custoso e menos competitivo [Audy e Prikladnicki, 2008]. O avanço da economia, a sofisticação dos meios de comunicação e a pressão por custos têm incentivado o investimento maciço no DDS.

Segundo Brooks [1978], o software possui muitas características que lhe fazem unicamente apropriado para esta abordagem e o diferencia de outros setores, pois pode ser replicado, transmitido, corrigido e usado sobre distâncias ilimitadas com custo virtual zero. Contudo, também é sujeito a mudanças, é intangível e pode tornar-se potencialmente complexo. Desta forma, o Desenvolvimento Distribuído tem se

apresentado nos últimos anos como uma alternativa para o desenvolvimento de software. É um fenômeno que vem crescendo desde a última década e que tem sido caracterizado pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, localizados em cidades ou países diferentes [Meyer, 2006].

Estes, porém, não são os únicos fatores que contribuem para que os processos de desenvolvimento das empresas sejam distribuídos, como cita Prikladnicki [2003]:

- A necessidade de aumentar os recursos quando a disponibilidade de mão de obra local estiver escassa;
- Vantagem de estar perto do mercado local, incluindo o conhecimento dos clientes e as condições locais;
- A rápida formação de corporações e times virtuais visando explorar as oportunidades de mercado;
- Pressões relativas ao *time-to-market*, com o uso de diferentes fusos-horários permitindo inclusive o desenvolvimento 24x7 (vinte e quatro horas, sete dias por semana).

Nesse contexto, uma fábrica de software que trabalhe em um ambiente de desenvolvimento distribuído está sujeita a diversos problemas técnicos, humanos e organizacionais, como problemas de comunicação, gestão, relacionamento, entre outros. Audy [2008] explica que o desenvolvimento de software sempre se apresentou de forma complexa, e o Desenvolvimento Distribuído de Software acrescentou outros desafios ao processo ao adicionar fatores como dispersão física, distância temporal e diferenças culturais.

3. O Projeto CitIX Mobile

O presente estudo de caso é parte de uma disciplina de Engenharia de Software [IN953, 2008] com foco na criação de fábricas de software que façam uso de desenvolvimento distribuído para a realização de projetos reais. Com a definição dos clientes e projetos, os alunos têm quatro meses para executar um ambiente de fábrica de software e entregar os produtos definidos em comum acordo com o cliente.

A fábrica relatada neste trabalho, denominada *TechnoSapiens*, foi composta por nove estudantes, todos trabalhando de forma distribuída, com parte da equipe trabalhando na mesma cidade e alguns membros em outras cidades, totalizando 3 cidades. Além disso, nenhum dos membros havia trabalhado anteriormente com outro membro da equipe. Logo, todos estavam juntos pela primeira vez para o desenvolvimento do projeto.

A equipe se reunia duas vezes durante a semana, um encontro no início da semana e outro no final da semana. No primeiro encontro eram definidas as atividades e no segundo as mesmas eram avaliadas. Essas atividades e todos os trabalhos da fábrica eram gerenciados e discutidos durante a semana através de um grupo de discussão e uma ferramenta *web* de planejamento e acompanhamento. Além destas, outras

ferramentas foram fundamentais para a comunicação entre os membros, tais como comunicadores instantâneos ou *messengers* e o próprio *website* da fábrica[1].

A fim de permitir adequações e melhorias na fábrica, um projeto piloto foi executado no primeiro mês da disciplina. O objetivo foi validar a estrutura organizacional da fábrica e do processo de desenvolvimento adotado pela mesma. Após o primeiro projeto, um segundo projeto de dimensões maiores foi realizado num período de três meses. Na disciplina, o projeto real pode ser uma continuação do projeto piloto, o que permite uma maior adequação quando da execução do projeto piloto em relação ao projeto real.

O projeto assumido pela fábrica foi o desenvolvimento, com funcionalidades reduzidas, de uma versão para dispositivos móveis do serviço citIX. O citIX utiliza um sistema de informações geográficas que possibilita que os usuários do serviço construam uma rede de conhecimento sobre as características (segurança pública, entretenimento, infra-estrutura, serviços públicos, etc.) de uma determinada região.

4. Modelagem do Processo: o Caso citIX

Em um ambiente de desenvolvimento distribuído, um processo de desenvolvimento comum à equipe é fundamental, tendo em vista que uma metodologia auxilia diretamente na sincronização, fornecendo a todos os membros da equipe uma nomenclatura comum de tarefas e atividades, e um conjunto comum de expectativas aos elementos envolvidos no processo [Audy e Prikladnicki, 2008].

Após a definição do projeto a ser desenvolvido e do modelo organizacional da *TechnoSapiens*, passou-se à definição do processo de desenvolvimento a ser adotado. Para esta decisão, quatro principais fatores foram considerados:

- A pouca experiência da equipe com outros processos e metodologias em oposição ao maior conhecimento de processos baseados no RUP;
- A necessidade de um processo leve que contemplasse uma quantidade menor de artefatos - apenas os artefatos considerados essenciais para o seu progresso;
- A hipótese de que um processo mais adequado para o desenvolvimento distribuído deveria ser prescritivo em relação às fases, atividades e papéis, desta forma, o desenvolvimento distribuído seria facilitado em momentos em que a equipe possuísse dificuldade de encontros físicos e o entendimento por parte da equipe no decorrer do processo seria melhorado;
- O desconhecimento ou pouca experiência no uso da tecnologia JavaME era uma realidade para cerca de 65% da equipe.

Estes fatores levaram à definição de uma instância de processo baseada no *Rational Unified Process* (RUP). O *TechnoProcess*, como foi denominado o processo criado pela *TechnoSapiens*, foi elaborado de forma a viabilizar e atender aos fatores acima citados. A busca por um processo leve, flexível, porém prescritivo, levou a reuniões para definir como o processo atenderia a estas necessidades. As principais fases consideradas para o processo foram: Pré-Venda, Requisitos, Análise e Projeto, Revisão

[1] Website TechnoSapiens: http://tsapiens.cin.ufpe.br/

e Aprovação[2], Implementação, Testes e Implantação[3]. Fases de suporte às fases anteriores também foram definidas: Planejamento e Gerenciamento do Projeto, Gerência de Configuração e Garantia da Qualidade.

Com uma equipe onde poucos conheciam da tecnologia do projeto, alguma medida precisava ser tomada para mitigar o risco de atrasos no projeto. Contraditoriamente, a equipe deveria trabalhar em muitos casos de forma distribuída, quer pela exigência da disciplina, quer pelo fato de muitos dos membros não poderem estar presentes no mesmo espaço físico. Assim, verificou-se a possibilidade de uma melhor adequação do processo pela adição de algumas práticas de metodologias ágeis. Entre as práticas podemos destacar a comunicação freqüente entre o grupo e o cliente e entre os membros do grupo. A fim de esclarecer questões relativas ao escopo do projeto e outras questões sobre tecnologia, por exemplo; a programação em par que, mesmo à distância, serviu para que membros com menos experiência pudessem, virtualmente ou fisicamente, ser auxiliados por outros com maior conhecimento; e *refactoring*, realizado pelo membro mais experiente do grupo na tecnologia com o intuito de organizar a estrutura, melhorar a legibilidade e padronizar o código do sistema.

Com o intuito de garantir que os Requisitos fossem atendidos, as Gerências de Configuração, de Projeto e a Garantia de Qualidade ficaram responsáveis por um conjunto de artefatos que deveriam ser planejados e elaborados na fase preliminar do projeto de modo que fossem seguidos por todos os integrantes da equipe. Os principais artefatos obrigatórios foram: Proposta Comercial, Plano de Projeto, Documento de Especificação do Processo, Plano de Qualidade, Documento de Especificação de Requisitos, Plano de Configuração, Plano e Projeto de Testes.

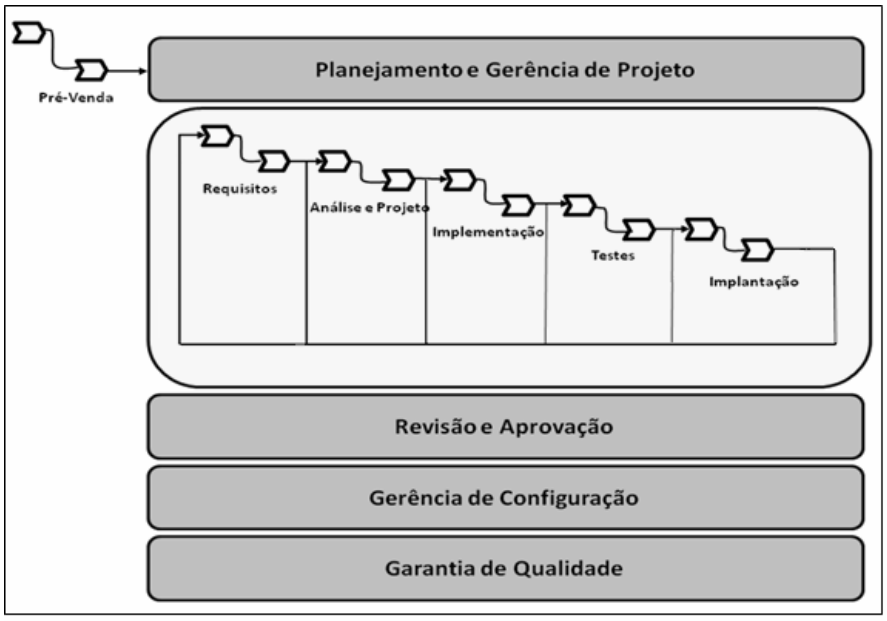

Figura 1. Processo de software.

[2] A fase de Revisão e Aprovação foi adicionada ao processo com o objetivo de realizar a prevenção e correção de erros técnicos, garantindo assim a sua consistência e corretude.

[3] Apesar de definido no processo, a fase de Implantação não foi necessária ao projeto visto que a aplicação foi distribuída diretamente ao cliente para uma utilização futura, ainda sem previsão.

5. Evolução do Processo

Estabelecer um processo de software eficiente e que se adéqüe ao contexto da fábrica é algo de fundamental importância para o sucesso de um projeto, por isso, é necessário que o mesmo seja constantemente avaliado a fim de alcançar um estado de adequação e maturidade cada vez maior.

As seções seguintes apresentam, em linhas gerais, os passos da evolução do processo. A seção 5.1 apresenta uma descrição mais detalhada dos problemas enfrentados em relação à execução do processo para cada área; a seção 5.2 mostra como foram resolvidos os problemas e quais as mudanças ocorridas.

5.1 Problemas

O projeto foi desenvolvido em duas etapas: o projeto piloto, utilizado para fins de verificação da estrutura da fábrica (infra-estrutura, processo, entre outros) e o projeto real, que teve como objetivo utilizar as melhores práticas adotadas no projeto piloto e buscar melhorias.

Com o escopo reduzido, no projeto piloto foi possível verificar a viabilidade do processo proposto, bem como identificar, na prática, os problemas mais comuns do desenvolvimento distribuído. O projeto real encontrou uma estrutura mais estável e com um processo melhor adaptado à realidade da fábrica. Essa etapa abrangeu todos os requisitos citados no início desta seção, a saber: autenticação, visualização de mapas e relatos, adição e edição de relatos. Neste ponto, a arquitetura básica do sistema estava pronta e alguns serviços já estavam implementados. Apesar da mitigação dos riscos levantados no projeto piloto, alguns problemas continuaram impactando o andamento das atividades e novos riscos foram levantados. De forma geral, estão listados abaixo todos os problemas encontrados durante o projeto como um todo:

- **Análise de viabilidade*:*** não houve esse estudo, apenas uma reunião informal para a escolha, o que no futuro impactou, já que nem todos tinham conhecimento da tecnologia que envolvia o desenvolvimento do projeto. Nesse momento houve pouco contato com o cliente para esclarecimentos, o que gerou no início indefinição.
- **Falta de domínio e experiência nas tecnologias utilizadas***:* um dos maiores risco levantados no projeto piloto (e que impactaria todo o projeto) era a falta de conhecimento técnico da maioria da equipe.
- **Definição de estimativas:** inicialmente as estimativas para a conclusão dos artefatos eram feitas pelo Gerente. Logo nas primeiras semanas percebeu-se a fragilidade desta abordagem, pois os membros não estavam realmente comprometidos com as datas que eram estabelecidas.
- **Atraso devido a dependências externas:** devido à característica do Projeto, a fábrica teve algumas dependências externas com a disponibilização dos serviços relativos ao escopo da versão para dispositivos móveis. O serviço planejado para o projeto piloto não foi disponibilizado a tempo.
- ***Feedbacks* demorados ou inexistentes:** apesar da fábrica ter uma pessoa fisicamente próxima do cliente, os *feedbacks* sobre as sugestões e dificuldades

levantadas pela equipe nem sempre eram retornados ou demoravam ao ponto de não poderem ser levados em consideração.

- **Gestão de pessoas:** fatores imprevistos, como a redução do número de pessoas que integravam a equipe (equipe inicialmente possuía 12 integrantes e depois teve este número reduzido para 9), causaram bastantes mudanças no planejamento inicial e das tarefas.
- **Gestão de Atividades:** Houve uma estimativa de esforço menor do que o necessário, dessa forma, com o Desenvolvimento Distribuído, tornaram-se difícil acompanhar, avaliar e monitorar as atividades de cada membro da equipe.
- **Ambiente de Configuração:** Como a equipe em geral desconhecia o ambiente de configuração, houve problemas nas máquinas de membros da equipe e como o desenvolvimento foi distribuído tornou-se significativamente mais difícil a resolução deste problema.
- **Realização de Testes:** Com o atraso no desenvolvimento, os testes foram afetados diretamente, tendo em vista o cronograma da disciplina, e a entrega do produto. Além disso, devido à falta de infra-estrutura (aparelho alvo, SDK – *Software Development Kit*, cabos, etc.) para a execução dos testes no período planejado, apenas foram realizados testes unitários nos emuladores. A abordagem foi acordada com o cliente como uma forma de mitigação do risco de não existir infra-estrutura disponível para testes (a disponibilização dessa infra-estrutura estava sob responsabilidade do cliente).

5.2 Adequações

Durante o desenvolvimento do projeto citIX *mobile* algumas dificuldades na execução do processo foram percebidas porque inicialmente, na sua definição, possuía um grande número de artefatos a serem produzidos, além de uma grande quantidade de atividades que tornavam o desenvolvimento bastante burocrático e inviável para uma equipe de trabalho de pequeno porte. Em meio a esses pequenos impasses, o *TechnoProcess* precisou ser adequado no decorrer do projeto para realmente atender às necessidades da fábrica. Na fase de adequações do processo, foram definidos ajustes nas atividades, adequando também alguns *templates* dos artefatos propostos. A seguir são mais detalhadas as principais alterações no processo, por área:

- **Pré-Venda:** Devido à exigência de uma decisão rápida sobre qual projeto a fábrica iria selecionar e os problemas advindos dessa escolha, a fábrica decidiu por algumas ações que ajudassem no melhor entendimento do projeto. Para isto, algumas medidas foram tomadas: elaborou-se uma Proposta Comercial com uma definição clara dos objetivos, escopo do projeto, responsabilidades da fábrica e do cliente e escopo negativo do projeto; intensificou-se a comunicação com o cliente através de um membro da equipe mais próximo dele; e procurou-se identificar quais integrantes da equipe eram mais experientes com projetos para dispositivos móveis. Estas medidas trouxeram benefícios para as fases seguintes.
- **Planejamento e Gerenciamento do Projeto:** Com as dificuldades naturalmente impostas pelo DDS e os atrasos nas entregas de atividades, outro conjunto de

mudanças foram propostas. Percebeu-se que os atrasos ocorriam, em sua maioria, devido à ausência de um autogerenciamento e comprometimento dos membros com sua atividade e não por falhas nas estimativas. Para aumentar o grau de comprometimento com a atividade, a responsabilidade pelas estimativas das atividades foi atribuída aos respectivos responsáveis pelas atividades.

- **Requisitos:** O escopo foi definido no início devido à dificuldade no entendimento do projeto e da tecnologia envolvida. Tendo em vista a impossibilidade de uma constante comunicação com o cliente, o Documento de Especificação de Requisitos foi elaborado segundo os requisitos e funcionalidades elicitados pelo cliente nas primeiras discussões. Após a definição do documento, qualquer funcionalidade adicionada ao escopo foi sendo desenvolvida na iteração seguinte de acordo com a disponibilidade da equipe. Cada iteração resultava em incrementos de funcionalidades do sistema, acomodando solicitações de mudanças que pudessem vir a surgir. O membro da equipe que possuía um melhor contato com o cliente passou a auxiliar parte da equipe responsável pela especificação dos requisitos.
- **Análise e Projeto:** O Plano de Projeto visou estabelecer uma visão geral do projeto e o escopo de forma a facilitar o entendimento geral, definindo quando as tarefas envolvidas deveriam ser realizadas. As atividades foram atribuídas a cada integrante segundo o seu perfil e a experiência. O *TechnoProcess*, que previa a escrita do Documento de Arquitetura, foi adaptado para tornar a existência deste artefato opcional, visto que o principal objetivo deste artefato era esclarecer para os desenvolvedores a arquitetura definida.
- **Implementação:** Com a redução da equipe, as tarefas foram redistribuídas entre os demais integrantes da fábrica para garantir a evolução do projeto. Os problemas com atrasos de serviços que seriam disponibilizados pelo cliente necessitaram ser resolvidos com antecipação de outras funcionalidades. Assim, enquanto umas eram antecipadas outras eram adiadas em acordo com o cliente.
- **Testes:** Em acordo com o cliente, como medida de mitigação dos riscos pela não disponibilização da infra-estrutura necessária, os testes foram realizados com base nos resultados obtidos no emulador. Todos os requisitos funcionais foram testados, exceto aqueles cujos serviços disponibilizados não estavam funcionando.
- **Gerência de Configuração:** A fim de fornecer aos desenvolvedores as informações necessárias para configurar o ambiente de desenvolvimento do Projeto, o Plano de Configuração foi elaborado. Conforme o andamento do projeto e o conhecimento dos problemas de Configuração, medidas foram tomadas para eliminar ou minimizar os problemas. Algumas foram apenas recomendações sobre como proceder se problemas ocorressem nas etapas de configuração.
- **Garantia de Qualidade:** O Plano de Garantia de Qualidade foi elaborado a fim de atender os requisitos de Qualidade adotados pela *TechnoSapiens*, como: Qualidade do processo de desenvolvimento de software e dos produtos gerados durante o desenvolvimento do projeto. O acompanhamento das versões dos

artefatos foi realizado via e-mail, onde cada autor elaborava e enviava o artefato pelo qual era responsável para o e-mail do grupo. O SQA realizava a revisão e as correções necessárias. Depois da avaliação, o artefato era publicado no site da fábrica.

6. Considerações Finais

Quando o ambiente é Desenvolvimento Distribuído, o cenário muda com relação ao desenvolvimento de software tradicional, pois as variáveis e os riscos aumentam, então, se não houver uma boa metodologia para o processo de desenvolvimento, o projeto terá boas chances de não corresponder ao planejamento inicial.

As experiências vivenciadas neste projeto foram abrangentes no sentido de trabalhar com uma equipe pequena e distribuída, onde boa parte do desenvolvimento foi, de fato, realizado à distância. A experiência obtida pela equipe, especialmente em relação ao uso do RUP em ambientes de desenvolvimento distribuído, traz algumas lições aprendidas:

- **Processos em DDS:** Uma primeira percepção sobre o uso e estudo de DDS, é que essa forma de desenvolvimento ainda necessita de processos mais adequados, especialmente no caso de pequenos times de desenvolvimento, pois, com base nas pesquisas realizadas pelo grupo, não foi fácil a identificação de modelos de referência de processos para o ambiente DDS.
- **Programação em par à distância:** Inicialmente, o grupo em geral acreditou que esse método não fosse ser eficiente comparado com a programação em par com presença física, mas, no decorrer da fase de Implementação, este meio de colaboração se mostrou bastante eficaz, sendo suportado por ferramentas que contribuíssem para isso, como por exemplo, *e-mails* e *messengers* com ou sem áudio.
- **Ferramentas de comunicação:** A equipe utilizou várias ferramentas para comunicação e essa experiência mostrou que as ferramentas comuns precisariam evoluir para poder suportar o trabalho distribuído. Um problema encontrado com os *messengers*, por exemplo, está na sua capacidade de suportar vários usuários ao mesmo tempo em uma conferência. Alguns *messengers* utilizados demonstraram não suportar mais de quatro pessoas nas conferências de áudio para realizar reuniões. Outro ponto percebido foi a falta de uma ferramenta que pudesse agregar as funções de uma IDE (*Integrated Development Environment*) e compartilhar o código fonte em tempo real, de maneira que os desenvolvedores pudessem exibir exatamente o que estivessem codificando e os parceiros distribuídos fisicamente pudessem acompanhar a edição de linha por linha de código;
- **Equipes autogerenciáveis:** Ao se adicionar o fator dispersão física no desenvolvimento distribuído, pode-se intensificar problemas que impactam negativamente no projeto, tais como atrasos, descomprometimento, entre outros. A experiência da equipe com DDS mostrou existir uma relação positiva entre equipes autogerenciáveis e o progresso do projeto. Equipes autogerenciáveis

tendem a agir proativamente em ambientes distribuídos por manterem uma busca por resultados para o projeto.

Com o crescimento do número de empresas utilizando DDS, a evolução no número de estudos na área e a consolidação de modelos de referência em DDS, é possível que esta forma de trabalhar seja, em breve, a mais utilizada por grandes empresas do setor de software e outros setores, pois, como visto, existem vários benefícios em desenvolver distribuidamente. É importante que abordagens práticas e experimentais, como a apresentada neste trabalho, sejam colocadas por outras pessoas à disposição da comunidade de software, e que mais fábricas de software, universidades, grupos de pesquisas e empresas possam difundir suas experiências na utilização de diferentes processos em ambientes distribuídos buscando uma maior colaboração, objetivando resultados mais relevantes.

Referências

Audy, J.; Prikladnicki, R. (2008) Desenvolvimento Distribuído de Software: Desenvolvimento de software com equipes distribuídas. Rio de Janeiro: Elsevier.

Brooks, F. P. (1978) The Mythical Man-Month: Essays on Softw. 1st. Addison- Wesley Longman Publishing Co., Inc.

Carmel, E. (1998) Global Software Teams: Collaborating Across Borders and Time Zones. Prentice Hall.

Carmel, E. (1999) Global Software Teams: Collaboration Across Borders and Time Zones. Prentice-Hall, EUA.

IN953 (2008). Software Engineering: Building Open Source Software Factories. Disponível em http://www.cin.ufpe.br/~in953/.

Herbsleb, J. D.; Moitra, *D.* (2001). *Guest editors' introduction: global software development*. IEEE Software.

Karolak, D. W. (1998) *Global Software Development – Managing Virtual Teams and Environments*. IEEE Computer Society, EUA.

Kiel, L. (2003) *Experiences in Distributed Development: A Case Study*. Workshop on Global Software Development at ICSE 2003, Oregon, EUA.

Meyer, B. (2006). *The unspoken revolution in software engineering*. IEEE Computer.

Prikladnicki, R. (2003). Munddos - Um Modelo de Referência para Desenvolvimento Distribuído de Software. Dissertação de Mestrado, Pontifícia Universidade Católica Do Rio Grande Do Sul.

Prikladnicki, R., Audy, J. (2004) MuNDDoS - Um Modelo de Referência para Desenvolvimento Distribuído de Software. XVIII SBES - Simpósio Brasileiro de Engenharia de Software.