PEP – Uma Ferramenta de Apoio à Programação em Par Distribuída

Leandro P. de Aguiar, Giorgio S. C. Merize, Frank Siqueira

Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

{leandrop,giorgio,frank}@inf.ufsc.br

Abstract. *In Distributed Pair Programming (DPP), instead of developing software side-by-side, as in Pair Programming (PP), programmers can interact through the network. This is a valuable alternative in particular for global organizations, where the practice of cooperative distributed work is lucrative. The only limitation presented by DPP is the lack of good tools that allow the interaction between programmers. PEP – Pair Eclipse Programming – is a open source plugin for the Eclipse IDE which makes possible to practice DPP with productivity and reliability, solving the problems presented by the few existing tools for distributed programming and adding interesting resources to this practice.*

Resumo. *Na Programação em Par Distribuída (PPD), ao invés de haver o desenvolvimento de software lado-a-lado, como na Programação em Par (PP), os programadores podem interagir através da rede. Esta é uma alternativa valiosa principalmente para organizações globais, onde a prática do trabalho cooperativo distribuído é lucrativa. A única limitação apresentada para PPD é a carência de boas ferramentas que permitam a interação entre os programadores. O PEP – Pair Eclipse Programming – é um plugin de código livre para o IDE Eclipse que permite a prática do PPD com produtividade e confiabilidade, solucionando os problemas apresentados pelas poucas ferramentas existentes para programação distribuída e adicionando recursos interessantes para esta prática.*

1. Introdução

A Programação em Par (do inglês *pair programming*) é a prática onde dois programadores desenvolvem software lado a lado em um computador (COCKBURN 2000). Tal conceito, embora não seja muito novo, ganhou popularidade apenas recentemente, quando foi introduzido com a metodologia de desenvolvimento ágil *Extreme Programming* (WILLIAMS et. Al., 2000). Na Programação em Par (PP) os programadores trabalham em conjunto para apresentar a melhor solução para um problema proposto. Um membro é responsável por comandar (pilotar) o computador e digitar o código, enquanto o outro integrante revisa o código e auxilia na análise mais detalhada do problema em busca da melhor solução a ser adotada. Embora, por muito tempo, a técnica tenha sido vista pelos gerentes de projeto menos críticos como desvantajosa, experimentos práticos (WILLIAMS; KESSLER, 1999) mostraram que,

além de apresentar um custo de desenvolvimento apenas 15% superior à programação individual, contrariando as expectativas que afirmavam que o custo seria dobrado, o desenvolvimento em pares produziu um código de melhor qualidade, com uma menor quantidade de linhas e com um percentual de erros em média 15% menor. Alistair Cockburn e Laurie Williams (2000) afirmam que "com um custo de até 15% a mais, a programação em par melhora a qualidade da arquitetura do software, reduz defeitos, reduz risco de perda de membro da equipe, melhora as habilidades técnicas, melhora a comunicação no time e é considerada mais 'agradável' em níveis estatísticos significativos".

Embora a adoção da programação em par já tenha sido claramente apresentada como benéfica, em comparação com a atividade realizada individualmente, várias pessoas questionam se tais benefícios se manteriam caso praticada a chamada Programação em Par Distribuída (PPD), ou seja, quando os membros do par não se encontram fisicamente próximos um do outro. Para testar estes questionamentos, um estudo conduzido na Universidade da Carolina do Norte em 2002 (BAHETI; WILLIAMS; et. Al., 2002), aplicou um experimento onde foram comparados a qualidade do código gerado, a produtividade e o desempenho na comunicação dentro de times locais e distribuídos. Como resultado, os autores apresentaram as seguintes conclusões:

- O desenvolvimento de software utilizando PPD parece ser comparável ao desenvolvimento local em termos de duas métricas: produtividade (em termos de linhas de código por hora) e qualidade;
- Pares locais não produzem, estatisticamente e significativamente, melhores resultados do que pares distribuídos;
- O respaldo dado pelos estudantes que participaram do experimento indica que a PPD promove trabalho em equipe e comunicação dentro do time virtual.

Extreme Programming (XP) foi a primeira metodologia de desenvolvimento popular a implementar a programação em par. Embora esta prática, quando proposta, não tenha necessariamente sido associada a uma metodologia de desenvolvimento ágil, foi a partir de XP que a programação em par efetivamente ganhou os olhos da comunidade de desenvolvimento de software.

O *Distributed Extreme Programming* (DXP) foi conceituado recentemente como sendo *"Extreme Programming* com certos relaxamentos em requisitos de proximidade física dos membros do time*"*, permitindo a aplicação dos princípios de XP em um ambiente distribuído e móvel (KIRCHIER et al., 2001). Os autores afirmam que, para que a DXP seja possível, alguns requisitos devem estar presentes: deve haver alguma forma de conexão entre os praticantes; o e-mail deve ser usado como forma de trocar informações e agendamentos; é necessária alguma ferramenta de gerência de configuração; é importante a familiaridade entre os membros do time; alguma forma de vídeo e áudio conferência deve estar disponível; uma aplicação ou software de compartilhamento deve ser usado.

Com o intuito de fornecer o suporte computacional necessário para a prática de PPD e DXP, foi desenvolvido o PEP – *Pair Eclipse Programming*, um plugin para o ambiente de desenvolvimento Eclipse que provê suporte para o desenvolvimento

distribuído de software. Este artigo descreve em sua segunda sessão alguns trabalhos relacionados à prática de PPD, enfatizando suas limitações; em seguida, na sessão 3, descreve o suporte desenvolvido para programação em par distribuída; e por fim, na sessão 4, são apresentadas as conclusões dos autores acerca do trabalho desenvolvido e as perspectivas em relação à continuidade deste trabalho.

2. Trabalhos Relacionados

Os autores de PPD e DXP afirmam que a principal dificuldade para a prática destas técnicas é justamente a carência de ferramentas de suporte ao trabalho distribuído. Podemos citar duas ferramentas que provêem suporte para a programação em par distribuída desenvolvidas previamente a este trabalho:

- **Collab**: foi desenvolvido para o IDE NetBeans, da *Sun Microsystems*, para "permitir aos times distribuídos ou grupos de trabalho interagirem e trabalharem dinamicamente juntos de uma maneira altamente produtiva" (DEVELOPER, 2005). Com a sua distribuição com o NetBeans, o *plugin* ganhou mais popularidade sendo, por várias vezes, referenciado por revistas da área como a única solução para o desenvolvimento colaborativo em grupo. Todas as suas funcionalidades somente estão disponíveis após autenticação em um servidor da própria Sun Microsystems.
- **Sangam**: é um plugin desenvolvido para o IDE Eclipse com o propósito específico de possibilitar a programação em par distribuída, apesar de não restringir as suas funcionalidades às práticas do PP. Apesar de haver sido criado como um software de código livre, o projeto teve sua última atualização no ano de 2002, podendo ser considerado, portanto, um projeto abandonado.

Embora os autores destas ferramentas defendam suas soluções como algo que permite agilidade no uso do trabalho de equipe, alguns recursos ausentes tornam a tarefa de programar nestes ambientes um trabalho pouco confiável e ineficiente por várias razões:

- Não há um mecanismo de detecção de inconsistências, o que implica na necessidade de verificação manual constante através de mecanismos como chat;
- Não há um mecanismo para resincronismo pós-falha, ou seja, uma vez detectada a inconsistência manualmente, os membros devem procurar outras alternativas fora do software para reestabelecer o sincronismo;
- Há um relaxamento nas ferramentas no que tange às práticas do PP, como a possibilidade de edição por mais de dois membros e a possibilidade de edição simultânea em classes ou trechos de código diferentes, o que, além de tornar o processo mais vulnerável para o surgimento de inconsistências, impede que os principais benefícios do PP, como a revisão de código e a troca de experiências, ocorram com a intensidade necessária para tornar-lhe uma prática vantajosa;
- A configuração que antecede a utilização da ferramenta é complexa, exigindo dos programadores vários passos antes de iniciar a programação.

Devido à dificuldade encontrada para obter ferramentas adequadas para a prática de PPD e DXP, outras ferramentas começaram a ser desenvolvidas paralelamente a este

trabalho. DPP, XEclip e XPairtise são projetos em andamento, registrados no *SourceForge*[1], que possuem propósitos similares aos deste trabalho, mas cujas funcionalidades ainda não puderam ser avaliadas por estarem em desenvolvimento.

3. PEP – Pair Eclipse Programming

O PEP[2] – *Pair Eclipse Programming* – consiste em uma ferramenta de suporte para o desenvolvimento distribuído de software através da programação em par (PP). De modo a evitar a necessidade de desenvolver um suporte completo para criação, gerenciamento e execução de projetos de software, optou-se por estender um ambiente de desenvolvimento já existente, gratuito e de código aberto. Tal estratégia favorece também a adoção da ferramenta por parte dos programadores, que evitam a necessidade de ter que se adaptar a um novo ambiente de desenvolvimento para praticar a programação em par distribuída.

O ambiente de desenvolvimento integrado Eclipse[3] é considerado "o IDE Java dominante do mercado" (GEER, 2005), permitindo também o desenvolvimento de software em outras linguagens além do Java, como por exemplo C, C++ e PHP. O Eclipse foi projetado como uma plataforma extensível por meio de plugins, que permitem estender as funcionalidades do IDE, possibilitando a automatização de uma série de funções que os programadores comumente teriam que executar manualmente ou desenvolver em outros aparatos de software. Por possuir o código fonte aberto, o que proporciona grande flexibilidade para os programadores, a plataforma Eclipse facilita a elaboração de plugins mais complexos. Devido a estas facilidades propiciadas pelo Eclipse, optou-se por estender este ambiente de desenvolvimento através da adição de um plugin para programação em par distribuída.

O plugin PEP apresenta as seguintes facilidades para programação em par distribuída:

- Permite que um projeto de software seja compartilhado entre dois participantes da programação em par;
- Gerencia a comunicação entre os participantes, de modo que todas as alterações realizadas por um programador no código do projeto compartilhado são replicadas na máquina do seu parceiro, mantendo as cópias do projeto consistentes;
- Controla o acesso aos arquivos do projeto, fazendo com que apenas um dos programadores possa alterá-lo, enquanto o outro participante acompanha as modificações realizadas;
- Fornece um mecanismo de *chat*, usado para interação entre os programadores;
- Detecta inconsistências nas cópias locais dos arquivos do projeto compartilhado mantidas por cada participante, permitindo que os programadores efetuem a resincronização dos arquivos do projeto quando necessário.

[1] http://sourceforge.net
[2] Disponível para download em http://sourceforge.net/projects/pep-pp/
[3] http://www.eclipse.org

Tais facilidades, assim como o funcionamento e a forma de utilização do plugin PEP, serão descritas em maiores detalhes ao longo desta sessão.

3.1. Funcionamento do Plugin PEP

O modo de funcionamento do plugin segue, acima de tudo, os princípios da programação em par. A aplicação é capaz de se comportar como cliente ou como servidor, possuindo, portanto, uma arquitetura idêntica nas duas pontas. O esquema da Figura 1 apresenta uma visão dinâmica do funcionamendo do PEP. Enquanto um plugin tem a responsabilidade de enviar as ações de alterações de código correspondentes ao que este membro do par executou, o outro fica com a atribuição de receber estas informações e disponibilizá-las ou aplicá-las de forma que este membro tenha a percepção de que as ações ocorreram localmente. Esta configuração, naturalmente, e como ocorre também na programação em par não distribuída, deve permitir a inversão dos papéis a qualquer momento, desde que haja demanda através de comando executado por qualquer um dos membros do par. O plugin executado no lado do membro que detém o controle (representado no lado esquerdo da Figura 1) é responsável pela detecção e captação de todas as ações produzidas localmente para então encaminhá-las ao membro remoto (representado no lado direito). Tais eventos podem ser provenientes tanto de ações praticadas diretamente através de botões disponibilizados ao usuário, quanto de eventos gerados pelos componentes da plataforma Eclipse ao serem acionados.

Cada um destes eventos, independentemente de sua origem, deve, antes de ser encaminhado ao destino, passar por uma etapa de pré-processamento. Nesta etapa, o evento de origem é analisado e suas informações relevantes para o processo futuro de reprodução no ponto remoto são adequadamente selecionadas, uma vez que nem sempre todas as informações geradas por um evento captado localmente são necessárias para sua reprodução. Depois disso, estas informações são inseridas em uma mensagem compreensível pelo plugin remoto e serializadas para envio em um objeto específico para cada tipo de ação ou evento que é processado. Tal mensagem contém todo o conjunto de informações úteis ao processo de reprodução que será executado remotamente.

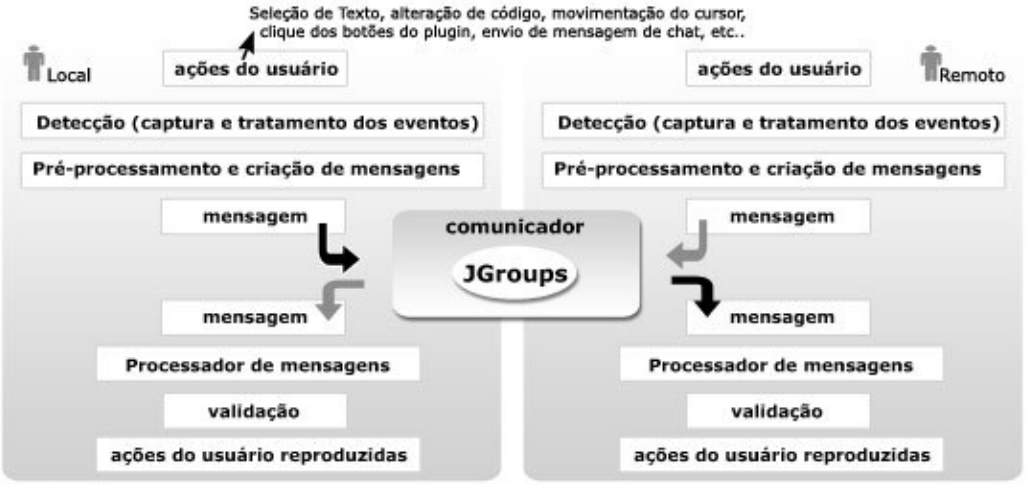

Figura 1. Esquema de Funcionamento do Plugin PEP

3.2. Infra-estrutura de Comunicação

O elemento "Comunicador" é responsável pelo controle, envio e recepção das mensagens geradas ao utilizar o ambiente de desenvolvimento. De maneira geral, para o membro que detém o controle, suas atribuições se restringem a propagar mensagens indicando as ações executadas no projeto compartilhado. Para o terminal remoto, de maneira análoga, o elemento comunicador fica com a responsabilidade de receber mensagens através da rede, identificar o tipo de mensagem, validar o conteúdo recebido com base em um conjunto de mensagens conhecidas e interpretáveis, e fazer a chamada ao processador de mensagens correspondente.

Cada processador de mensagens detém o conhecimento necessário para validar o conteúdo da mensagem e para reproduzi-la, caso a validação ocorra com sucesso. Para tanto, o processador interage com os objetos da plataforma responsáveis pelo controle das ações relacionadas, ou seja, os objetos capazes de reproduzir o evento a partir das informações recebidas. Com a reprodução transparente, estes eventos geram, para o membro remoto, a percepção de que houve uma intervenção.

O processo de comunicação em si é executado em mais baixo nível pela API JGroups[4], que é um *framework* para comunicação confiável em grupo escrito inteiramente em Java e com licença LGPL. JGroups é um projeto de código aberto utilizado como infra-estrutura de comunicação para outros projetos livres, como o Jakarta, JBoss e Tomcat. Seu princípio básico de funcionamento estabelece a confiança na comunicação entre membros de um grupo, mesmo quando a comunicação é realizada através do protocolo UDP, que não oferece garantias de entrega por si próprio.

3.3. Utilização do Plugin PEP

Após a instalação do plugin PEP, o desenvolvedor pode ativar a perspectiva PEP, por meio da qual podem ser acessadas as facilidades para programação em par distribuída. Nesta perspectiva, um projeto aberto no navegador do Eclipse pode ser compartilhado com outro programador. O programador que compartilha o projeto deve iniciar uma conexão no papel de servidor. Já o seu parceiro na programação em par deve conectar-se ao servidor, sendo, portanto, cliente. O processo de conexão é gerenciado através do botão PEP da barra de ferramentas do Plugin, exibida quando a perspectiva é ativada. A Figura 2 mostra as opções oferecida antes (a) e depois da conexão (b).

A partir do estabelecimento da conexão entre dois programadores, o plugin conectado como servidor passa a ter o controle das alterações, cabendo ao cliente o papel de observador. O controle pode ser trocado a qualquer momento por solicitação dos programadores, utilizando o terceiro botão (botão com ícone circular que aparece pressionado nas figuras 2.b e 2.c) da barra de ferramentas PEP.

(a) (b) (c)

Figura 2. Barra de Ferramentas do Plugin PEP

[4] http://www.jgroups.org

Figura 3. Janela de Chat do Plugin PEP

O processo de verificação de sincronismo do código sendo editado ocorre a cada alteração efetuada e é baseado na combinação de análise do tamanho do arquivo e análise do seu resumo criptográfico (*hash*) correspondente. Sempre que uma diferença de tamanho é detectada, este indício dispara uma verificação baseada no *hash* do arquivo usando o algoritmo MD5 de forma a comprovar a existência de inconsistências. Este recurso de verificação também pode ser acionado diretamente pelos programadores através da barra de ferramentas PEP, como mostra a Figura 2(c).

Com o intuito de propiciar uma melhor comunicação entre os pares, o plugin PEP incorpora uma ferramenta de *chat*, mostrada na Figura 3, que permite a troca de mensagens de texto entre os participantes e a apresentação de outras mensagens informativas sobre o processo de sincronismo de código. Ferramentas externas para interação por voz ou vídeo também podem ser utilizadas como forma de interação entre os programadores.

4. Conclusões

Comparando o PEP com outras duas ferramentas disponíveis na Internet projetadas para o suporte da atividade de programação em par distribuída, nota-se que o nível de maturidade de tais ferramentas, principalmente em termos de robustez e confiabilidade, torna tais aplicativos pouco usuais para os adeptos da prática do Pair Programming. Assim, além das atividades básicas de sincronismo de código e chat, uma das grandes metas planejadas para o PEP foi a de criar uma alternativa mais confiável como ambiente de desenvolvimento colaborativo e distribuído. A consequência direta desta decisão, a priori, foi a concepção dos novos requisitos de verificação de consistência e resincronismo, embora outras melhorias como o compartilhamento automatizado do projeto e o uso de uma boa API de comunicação tenham sido aplicadas. Como resultado, embora ainda não tenha sido possível realizar um estudo empírico comprovando ou atestando sua aceitação, vários dos problemas antes disponíveis foram eliminados, o que cria uma perspectiva favorável neste sentido.

4.1. Contribuições

O projeto PEP trouxe à comunidade de desenvolvimento distribuído algumas contribuições significativas. Em primeira instância, o plugin foi planejado com um propósito verdadeiro de contribuir com um problema ainda não resolvido, surgido com a criação da metodologia DXP (KIRCHIER et. Al., 2001), que é a carência de ferramentas que adequadamente suportem a atividade. Em segundo lugar, podemos afirmar que, com os resultados obtidos com o PEP, foi criada uma alternativa onde efetivamente é possível se estabelecer a prática do PPD com maior produtividade e robustez, dada sua capacidade de detectar e recuperar inconsistências, indisponível em ferramentas semelhantes. Além de ser uma ferramenta desenvolvida sobre uma API de comunicação

de comprovado desempenho e reconhecimento da comunidade de software livre - o JGroups - e sobre uma versão recente da plataforma Eclipse (3.2), novas funcionalidades foram adicionadas, criando uma forma verdadeiramente útil de se programar em pares de modo distribuído. Como última contribuição, podemos afirmar que a experiência obtida no desenvolvimento do plugin PEP com a aplicação prática do PPD resultou em um software funcional em um período curto de tempo e em um excelente conhecimento das minúcias do sistema por todos os integrantes da equipe, o que são evidências empíricas dos benefícios da programação em par. Tal informação pode e deve ser considerada como referência positiva aos que avaliam a viabilidade de sua utilização em ambientes mais exigentes em termos de prazos e produtividade.

4.2. Trabalhos Futuros

Embora possamos julgar o PEP como um software para PPD com avanços significativos em relação às ferramentas comparáveis, há ainda uma série de melhorias aplicáveis para que o plugin se torne completo. São algumas delas: a sofisticação do mecanismo de resincronismo, substituindo o método de "simples troca" por um sistema que permita atualizações incrementais ("*merge*"), a internacionalização do sistema, que atualmente se encontra em inglês, a extensão do plugin para permitir a interação entre programadores por meio de voz e imagem, e o suporte a outras linguagens de programação além do Java.

Referências

Cockburn, A. and Williams, L. (2000) "The Cost and Benefits of Pair of Programming", In: IEEE Software, University of Utah Computer Science.

Williams, L., Kessler, R., Cunningham, W. and Jeffries, R. (2000) "Strengthening the Case for Pair-Programming", IEEE Software.

Williams, L. and Kessler, R. (1999) "Experimenting with Industry's Pair-Programming Model in the Computer Science Classroom", IEEE Software.

Baheti, P., Williams, L., Gehringer, E., Sttots, D., and Smith J. (2002) "Distributed Pair Programming: Empirical Studies and Supporting Environments", Technical Report, Departament of Computer Science University of Noth Carolina at Chapel Hill.

Kirchier, M., Prashant, J., Corsaro, A., and Levine D. (2001) "Distributed Extreme Programming", Simens AG, Dept. of Computer Science Washington University.

Geer, D. (2005) "Eclipse Becomes The Dominant Java IDE", In: IEEE Computer Society, Edited by L. Gaber, Industry Trends.

Developer.com – Revista Eletrônica (2005) "Sun Presents Java Studio Enterprise Tool to Further Code-Aware Team Collaboration", Disponível no endereço http://www.developer.com/java/ent/article.php/3434771. Acessado em Julho 2006.