Revisão

Programação em Duplas: Estado da Arte

Pair Programming: A Survey

Eustáquio São José de Faria
Faculdade de Engenharia Elétrica
Universidade Federal de Uberlândia – UFU – Uberlândia, MG
eustaquio@pucminas.br

Keiji Yamanaka
Faculdade de Engenharia Elétrica
Universidade Federal de Uberlândia – UFU – Uberlândia, MG
keiji@ufu.br

Resumo: Programação em Duplas (*Pair Programming* – PP) é uma prática colaborativa de desenvolvimento de *software* em que dois programadores trabalham, ao mesmo tempo, em um único computador e na mesma tarefa de programação. Foi relatado na literatura que o conhecimento sobre PP encontra-se disperso e desorganizado. Com o intuito de organizar esse caos, o presente estudo realizou uma busca exaustiva de pesquisas sobre PP em algumas bibliotecas digitais de Ciência da Computação (ACM, IEEE Explore, Springer, CiteSeer e ScienceDirect, entre outras) e no Google/Scholar. A partir da completa leitura dos trabalhos encontrados, procurou-se definir temas-chave dentro da área descrevendo todos os estudos que se relacionam com cada tema. Os achados são interessantes e extensos – eles podem ser encontrados durante toda a leitura do presente artigo.

Palavras-chave: aprendizado colaborativo; ambiente de aprendizado distribuído; programação em duplas.

Abstract: Pair Programming (PP) is a collaborative software development practice in which two programmers work at the same time in the same programming task, either in a single computer or in two computers (interlinked through a net). It

Recebido em 19/03/2010 - Aceito em 28/07/2010

has been reported in the literature that knowledge about PP (Pair Programming) is currently dispersed and disorganized. An exhaustive research about PP has been done in order to organize such chaos. There was also an effortful search for researches about PP in some digital libraries of Computer Science (i.e., ACM, IEEE Explore, Springer, CiteSeer and ScienceDirect) and in Google/Scholar as well. It has been intended to define related key-topics and to describe all the related studies. The findings are both interesting and extensive – they can be found during the reading of present paper.

Key words: collaborative learning; distributed learning environments; pair programming.

1 Introdução

Programação em Duplas (*Pair Programming* – PP) é uma prática colaborativa de desenvolvimento de *software* – que foi adicionada à Programação Extrema (*eXtreme Programming* [1] – XP) como uma de suas doze práticas-chave – em que dois programadores trabalham ao mesmo tempo em um único computador e na mesma tarefa de programação. Um dos programadores é comumente chamado de piloto (*driver*). O piloto controla o teclado e o *mouse* e ativamente executa a tarefa de programação. O outro, conhecido como navegador (*navigator*), assiste o trabalho do piloto e age como um sócio em busca de geração de idéias. O navegador, constantemente, revisa os dados digitados para identificar deficiências táticas e estratégicas, enquanto procura erros de sintaxe e de lógica e implementações que desrespeitem às normas por eles impostas durante o projeto. Depois de um período de tempo pré-determinado, os sócios invertem seus papéis.

[1] Desenvolvida durante mais de quinze anos por Kent Beck e seus companheiros, Ron Jeffries e Ward Cunningham, XP é uma abordagem de desenvolvimento ágil de *software* que credita parte do seu sucesso ao uso de Programação em Duplas por todos os programadores. A dimensão de PP em XP requer que dois programadores trabalhem simultaneamente não somente na codificação, mas também no projeto, algoritmos, e testes [19].

É importante ressaltar que PP é ideal para alunos com postura ativa, que aprendem por interação social [1], e para profissionais que possuem habilidades para o trabalho colaborativo e facilidades em comunicação. Um aspecto interessante da PP é que seu uso torna mais dinâmicos o processo de desenvolvimento de *software* e o processo de aprendizado de programação – isso ocorre principalmente devido ao conflito sócio-cognitivo e ao comportamento de pressão dos pares (pressão das duplas), descrito na seção 3.9.

PP não é uma prática nova. Ainda em 1995, Constantine [2] fez um dos primeiros relatos em que se observou o uso de emparelhamento de programadores no desenvolvimento de *software* [3]. Naquele mesmo ano, Coplien [4] publicou seu livro sobre processo de produção de *software* sugerindo um padrão organizacional de desenvolvimento emparelhado.

Esses relatos, em conjunto com a introdução de PP no mercado pela metodologia *eXtreme Programming*, suscitaram a curiosidade dos especialistas em desenvolvimento de *software*. Motivado por esta curiosidade, Nosek [5] publicou um experimento sobre PP e concluiu que tal prática aumentou em 100% o desempenho dos programadores, além de tornar o processo de solução de problemas mais prazeroso. Não obstante, Williams [6], após ter aplicado um experimento estruturado em uma classe de estudantes de Engenharia de Software, na Universidade de Utah, confirmou os relatos de [5] e mostrou que o desenvolvimento de *software* com o uso de PP resulta em um produto mais confiável devido à quantidade reduzida de erros (*bugs*). Desde então, diversos estudos têm procurado verificar os benefícios de PP no processo de desenvolvimento de software e no ensino-aprendizado de técnicas de programação em cursos relacionados à computação.

Procurou-se, neste artigo, explorar os temas chave relacionados à PP. O objetivo, portanto, é prover aos estudiosos, interessados no assunto, um referencial teórico onde poderão ser encontrados os mais importantes estudos sobre PP no mundo. O trabalho se justifica na afirmação de Mendes, Al-Fakhri e Luxton-Reilly [7] – "O conhecimento sobre PP encontra-se disperso e desorganizado".

A exemplo de [8, 9], foi realizada uma busca exaustiva em algumas bibliotecas digitais conceituadas (por exemplo, IEEE – Explore, ACM, Springer, CiteSeer e ScienceDirect) usando as palavras chave *Pair Programming* e *Collaborative Programming* em *titles* e *abstracts*. Os documentos encontrados foram coletados, avaliados quanto a sua relevância e citados no decorrer do presente artigo. Foram também realizadas buscas em outras bibliotecas digitais e no Google/Scholar.

A próxima seção descreve os estudos empíricos encontrados nas buscas. A seção 3 explora os temas-chave a respeito de PP enquanto a seção 4 apresenta as conclusões finais e propostas de trabalhos futuros.

2 Estudos empíricos em PP

Diversos foram os estudos em que os autores aplicaram PP em ambientes de desenvolvimento de *software*. A maioria dos trabalhos foi realizada na academia. Foram encontrados 101 estudos empíricos – correspondendo a 65% do total de estudos. Desses, 76 (aproximadamente 75%) foram administrados em instituições acadêmicas [5-7, 9-81] com estudantes iniciantes, medianos e formandos. Somente 25 (aproximadamente 25%) estudos empíricos [2, 8, 82-104] foram relatados na indústria de *software*. Esses números sugerem e reforçam a necessidade de estudos empíricos em ambientes reais de desenvolvimento.

A seguir são descritos os diversos benefícios do uso de PP reivindicados nos estudos supra citados.

2.1 Benefícios unânimes do uso de PP

Enquanto alguns benefícios foram mais questionados que outros na literatura, um conjunto deles ocorreu de forma unânime entre os pesquisadores, dentre eles:

1. Melhora a qualidade global do *software*;
2. promove a transferência de informação e conhecimento entre os membros de equipes;

3. aumenta o moral e a confiança dos programadores na própria solução;
4. aumenta a satisfação dos desenvolvedores durante as atividades de programação;
5. reduz a taxa de erros encontrados nos testes de unidade devido às revisões e testes contínuos durante o desenvolvimento;
6. aumenta a responsabilidade dos programadores devido à pressão dos pares;
7. aprimora as habilidades colaborativas dos programadores;
8. aprimora as habilidades de comunicação dos programadores;
9. diminui a carga de trabalho dos instrutores quando usado academicamente;
10. diminui a taxa de evasão de alunos nos cursos;
11. conduz os alunos para uma atitude mais positiva em relação ao curso e à informática;
12. aumenta o interesse dos alunos em se especializar em computação;
13. provê aos programadores, ao observarem seus sócios, o conhecimento de diferentes abordagens para a solução de problemas;
14. resulta em código menor.

2.2 Benefícios questionados do uso de PP

Os benefícios questionados por alguns e relatados por outros como ocorrentes podem ser vistos a seguir:

15. Reduz o tempo de desenvolvimento;
16. reduz o custo de desenvolvimento;
17. produz código mais eficiente;
18. aumenta o sucesso dos alunos em exames avaliativos;
19. aumenta o sucesso dos alunos em disciplinas de programação futuras;
20. reduz a taxa de erros encontrados nos testes de integração devido às revisões e testes contínuos durante o desenvolvimento.

Nos estudos em que os autores questionaram os benefícios 18 e 19, relatou-se que PP não aumenta o sucesso dos alunos em exames avaliativos e em disciplinas de programação futuras. Por outro lado, os mesmos autores concluíram que as diferenças de pontuação entre alunos emparelhados e não emparelhados não são estatisticamente significativas. Os demais benefícios encontrados e não descritos anteriormente são oriundos ou consequentes dos citados acima. O grande consenso entre os estudos envolve o uso de PP como prática-chave no ensino de técnicas de programação.

3 Temas-chave em programação em duplas

Durante o levantamento bibliográfico sobre PP e, após leitura e análise cuidadosa do mesmo, procurou-se identificar os temas chave de maior relevância sobre o assunto. Esses temas são explorados a seguir.

3.1 Custo/benefício de PP

Apenas dois estudos sobre análise de custo/benefício de PP foram encontrados. O primeiro, realizado por Cockburn e Williams [105], relatou dois custos iminentes: (1) Custo de desenvolvimento; e (2) Custo de treinamento dos membros das equipes em habilidades colaborativas e em PP. Segundo os autores, 8 (oito) fatores importantes justificam o uso de PP na indústria e na academia:

1. Economia – Apesar de exigir uma quantidade dobrada de recursos humanos em um período de tempo, na maioria das vezes, superior a 50% do tempo gasto por programadores individuais, o código resultante de emparelhamentos contém uma menor quantidade de erros. A economia com a remoção de erros supera em valor o aumento do custo de desenvolvimento.
2. Satisfação – Membros de equipes emparelhadas defendem PP como uma experiência mais agradável que o trabalho individual.
3. Qualidade de projeto – Pares produzem programas mais curtos, culminando, geralmente, em projetos de maior qualidade.

4. Revisões contínuas – Emparelhamento provê projeto e revisão ininterruptos de código, enquanto conduz, na maioria dos casos, a taxas de remoção de erros eficientes.
5. Resolução de problemas – Devido às habilidades colaborativas, pares tendem a resolver problemas complexos mais rapidamente.
6. Aprendizado – Emparelhamento permite que programadores aprendam uns com os outros.
7. Construção de senso de equipe e melhoria nas habilidades de comunicação – Os participantes de equipes emparelhadas aprendem a discutir e trabalhar juntos. Isto melhora e torna mais efetiva a comunicação entre os parceiros.
8. Melhoria na administração dos projetos e redução de riscos – Considerando a familiaridade dos participantes com as diversas partes do código, Programação em Duplas reduz o risco e o impacto da perda de pessoal.

Baseando-se em questionários aplicados aos participantes de experimentos empíricos prévios e em dados encontrados na literatura sobre os oito fatores, os autores se apresentaram muito positivos e defenderam que os benefícios superam os seus custos de maneira significativa.

O segundo estudo foi realizado por Padberg e Müller [106]. De acordo com os autores, as vantagens da velocidade de programação e redução de defeitos são o grande potencial da PP. A pergunta é se o custo extra de PP é equilibrado pelos benefícios potenciais. Os autores mostraram como combinar diferentes métricas para avaliar o custo e benefício de PP. Foram integradas métricas de processos, métricas de produtos e métricas de contexto de processos dentro de um modelo de valor de negócio de um projeto – o modelo é baseado em uma concepção de valor líquido atual, o qual permite considerar o impacto do tempo no valor de mercado de um projeto. O modelo considera ainda que o número de pares, a vantagem da velocidade de desenvolvimento e a redução de defeitos proporcionados pelo emparelhamento, têm um forte impacto no valor de projetos que usam PP. Os resultados do estudo proporcionam um guia esclarecendo quando PP é mais indicado e quando não é aconselhável.

Os achados importantes do estudo de Padberg e Müller [106] são os que se seguem: (1) gerentes devem usar PP quando a pressão pela disponibilização do programa no mercado for muito forte, pois seus programadores são mais rápidos e produzem software de maior qualidade quando trabalham em pares, em comparação a quando trabalham individualmente – nestes casos, um curto tempo de negócio é decisivo para o sucesso do projeto; (2) por outro lado, se a quantidade de mão-de-obra disponível não permitir trabalhar com pares suficientes para explorar o grau de paralelismo possível no projeto, o gerente deve considerar a adição de mão-de-obra de programadores individuais, em vez de usar a PP.

3.2 Ferramentas para suportar PP

Ferramentas *groupware* disponíveis no mercado podem ser usadas como apoio a PP. A exemplo, foram encontrados alguns estudos [12, 13, 26, 62, 107-109] em que os autores avaliaram a efetividade de aplicativos comerciais com tal finalidade. Os aplicativos envolvem Skype, Microsoft NetMeeting, PC Anywhere, VNC, Yahoo Messenger, PalTalk, AOL Messenger, LMS (*Learning Mannager System*), entre outros. Ao todo, 34 estudos [9, 12-14, 16, 22, 24, 26, 28-31,35, 52, 62, 107-125] discutem o uso de PP através de aplicativos específicos e não específicos para prover PP.

Em uma discussão do modelo de solução de problemas e tecnologias de grupo, DeFranco-Tommarello e Deek [112] argumentam que as ferramentas *groupware* dão mais ênfase na tecnologia da colaboração do que na metodologia e coordenação de atividades. Segundo os autores, modelos atuais precisam considerar a psicologia e sociologia associadas à resolução colaborativa de problemas. Por exemplo, nenhuma ferramenta *groupware* de desenvolvimento de *software* citada pelos autores levou em consideração a cognição de grupo no processo de colaboração. Acredita-se que isto ocorre porque os desenvolvedores destas ferramentas são bons em desenvolver hardwares e softwares, mas pouco experientes em metodologias de comunicação.

Ferramentas específicas para prover PP foram desenvolvidas: Langton, Hickey e Alterman [31, 35] desenvolveram e fizeram experiências com uma ferramenta inicialmente denominada GHT e renomeada como GREWPtool; Atsuta e Matsuura [110] produziram um ambiente distribuído de PP e propuseram melhorias

necessárias para que os gerentes de projetos possam utilizá-la também em suas funções de gerência; um sistema denominado SANGAM foi descrito em [124] – o sistema provê PP, mas não suporta projeto emparelhado; um ambiente de apoio à colaboração chamado GILD foi construído em [22, 111] – o sistema permite PP, porém, não incorpora suporte à comunicação, realizada através de ferramentas comerciais como Instant Messenger, Skype, e outras; DeFranco-Tommarello e Deek [113] desenvolveram um modelo cognitivo de colaboração chamado CCM, em cujo trabalho, os autores provêem um tutorial para ensinar como resolver problemas em programação colaborativa e descrevem uma ferramenta de colaboração construída sob os conceitos do modelo; Mendes et al. [9] produziram o COLLEGE, um ambiente colaborativo de programação que suporta PP experimentando o sistema com estudantes da Universidade de Coimbra, mas não descreveram os resultados do experimento; Natsu et. al. [52] produziram um editor compartilhado de código fonte síncrono chamado COPPER (*COllaborative Pair Programming EditoR*), que permite dois engenheiros de software distribuídos codificarem programas emparelhadamente – cujo sistema implementa características de ambientes *groupware* como mecanismos de comunicação, consciência de colaboração, controle de concorrência, entre outros; Hanks [30] desenvolveu uma ferramenta de suporte à Programação em Duplas e aplicou uma pesquisa empírica com alunos universitários que aprovaram a ferramenta e defenderam o segundo cursor do mouse (cursor de gesticulação do navegador) como um recurso muito valioso durante a discussão e realização das tarefas.

É importante ressaltar que nenhuma das ferramentas específicas para prover PP encontradas na literatura dispõe de técnicas de inteligência artificial no intuito de mediar a colaboração. Adicionalmente, não foram encontrados relatos de aplicativos que suportem projetos, testes e revisões emparelhados (muito embora as buscas também não tenham sido específicas para tal propósito).

3.3 PP no mercado x PP pedagogicamente

PP na academia se distingue de PP no mercado por diversos motivos. A tabela 1 distingue PP baseando-se nos Benefícios atribuídos a PP na seção 2.

Tabela 1. Distinguindo PP no Mercado de PP na Academia

Benefícios	PP no Mercado	PP na Academia
Qualidade global do *software*	É um dos pilares que sustentam a viabilização de PP no mercado	Apresenta-se como consequência do aprendizado promovido pelo conflito e pela colaboração
Transferência de informação e conhecimento	Muito importante para tornar todos os membros de equipes especialistas nas diversas nuances de um sistema	Permite aos membros emparelhados o conhecimento de novas habilidades lógicas e de novas instruções em linguagens de programação
Moral e confiança dos programadores na própria solução	Gera *software* de maior qualidade	Diminui a evasão dos alunos nos cursos de computação
Satisfação dos desenvolvedores durante as atividades de programação	Permite um nível maior de união entre os membros de equipes de desenvolvimento	Diminui a evasão dos alunos nos cursos de computação
Redução da taxa de erros encontrados nos testes de unidade devido às revisões e testes contínuos durante o desenvolvimento	Provê o desenvolvimento ágil de software de qualidade	Torna menos necessário o processo de depuração, muitas vezes dispendioso para alunos pouco experientes e impacientes
Responsabilidade dos programadores devido à pressão dos pares	Diminui a possibilidade dos membros da dupla se dispersarem, o que, certamente, reduziria a produtividade	Diminui a possibilidade de plágio durante a realização de trabalhos

Continua...

Continuação...

Habilidades colaborativas dos programadores	Importantes para profissionais que pretendem trabalhar em grandes projetos de *software*	Prepara melhor o aluno para exercer carreiras em informática
Habilidades de comunicação dos programadores	Importantes para tornar os profissionais mais respeitados, aumentando a empregabilidade	Prepara melhor o aluno para exercer carreiras em informática
Carga de trabalho dos instrutores quando usado academicamente	Não se aplica	Permite que os instrutores ocupem seu tempo com questões e dúvidas mais complexas
Redução da taxa de evasão de alunos no curso	Não se aplica	Aumenta a viabilidade de manutenção dos cursos de computação
Atitude mais positiva em relação ao curso e à informática	Não se aplica	Diminui a evasão dos alunos nos cursos de computação
Interesse dos alunos em se especializar em computação	Não se aplica	Permite a expansão dos cursos de pós-graduação em computação
Conhecimento de distintas abordagens para a solução de problemas	Torna os profissionais imprescindíveis em suas equipes	Aprimora as habilidades de resolução de problemas, necessárias aos alunos
Código menor	Pode influenciar na eficiência e na eficácia dos *softwares*	Não se aplica

Conitnua...

		Continuação...
Tempo de desenvolvimento reduzido	Influencia diretamente no custo e na viabilidade dos *softwares*	Não se aplica
Custo de desenvolvimento reduzido	Influencia diretamente na viabilidade dos *softwares*	Não se aplica
Código mais eficiente	Potencializa a comercialização dos *softwares*	Não se aplica
Sucesso dos alunos em exames avaliativos	Não se aplica	Diminui a evasão dos alunos nos cursos de computação
Sucesso dos alunos em disciplinas de programação futuras	Não se aplica	Diminui a evasão dos alunos nos cursos de computação
Redução da taxa de erros encontrados nos testes de integração devido às revisões e testes contínuos durante o desenvolvimento	Provê o desenvolvimento ágil de *software* de qualidade	Permite aos alunos a sensação de sucesso nos seus trabalhos

Conclusão

Compreender as diferenças entre PP na academia e no mercado é primordial para a escolha de pesquisadores em trabalhos futuros que envolvem Programação em Duplas e aprendizado colaborativo de programação. Adicionalmente, acredita-se que os benefícios de PP na educação em cursos de computação são inquestionáveis, potencializando seu uso no ensino de programação. Por outro lado, conforme dito anteriormente, muitos estudos ainda se fazem necessários para sua validação em ambientes reais de desenvolvimento de software.

3.4 Metodologias para o uso de PP no ensino de programação e na indústria de *software*

De acordo com o protocolo de PP proposto por Kent Beck, Ron Jeffries e Ward Cunningham, em um emparelhamento de programadores, dois desenvolvedores de *software* compartilham um único monitor e teclado, exercendo papéis de piloto e navegador; os programadores trocam seus papéis regularmente, cuja troca de papéis é um processo informal e cujo intervalo típico compreende vinte minutos; o rodízio dos pares é incentivado para promover a transmissão de informação e conhecimento entre os membros de equipes após execução de tarefas modulares [26, 27, 108].

Embora o protocolo direcione para um bom funcionamento de PP, diversas facetas podem tornar a metodologia de implantação de Programação em Duplas mais efetiva ou não, seja para fins acadêmicos, seja para fins profissionais. Por exemplo, duplas podem ser constituídas de maneira aleatória ou pré-definida; rodízio dos pares pode ser incentivado ou restringido; troca de papéis pode ser feita periodicamente ou quando os participantes assim desejarem.

Os estudos que fizeram referência a metodologias de implantação de PP são os que se seguem: [7, 20, 23, 28, 29, 34, 40, 41, 43, 44, 51, 61, 63, 65, 71, 72, 78, 85, 99, 102, 113, 126-135]. Não foram encontrados estudos definindo ou sugerindo metodologias perfeitas. Ao contrário, a maioria procurou descrever os riscos e benefícios do uso de rodízio dos pares, troca de papéis e escolha dos pares.

3.4.1 Rodízio dos pares

Os estudos [7, 20, 23, 41, 51, 61, 65, 72, 113, 133, 135] trabalharam com rodízio dos pares. Enquanto alguns sugeriram o rodízio como solução para os problemas de compatibilidade da dupla, outros propuseram seus benefícios em relação à troca de conhecimentos e às habilidades de comunicação.

Fato é, o rodízio dos pares é importante para que alguns benefícios de PP sejam alcançados. Esse parece ser o consenso entre os especialistas visto que, apenas dois trabalhos [40, 78] declararam não trabalhar com tal prática. É importante

também ressaltar: quando o objetivo do emparelhamento estiver relacionado ao compartilhamento de informação e conhecimento, o rodízio dos pares é eficiente e ideal [41].

3.4.2 Troca de papéis

De acordo com a literatura, a troca de papéis é importante para manter o equilíbrio técnico e emocional entre os membros da dupla. Seis estudos [7, 71, 85, 99, 113, 127] que abordaram a troca de papéis foram encontrados. Segundo os autores, parece ser intuitivo que a troca de papéis impede os sócios de permanecerem em atitudes passivas – o simples ato de digitar traz o sócio para a realidade ativa do desenvolvimento emparelhado.

Uma visão surpreendente e interessante foi encontrada no estudo [85]. Segundo os autores, além da tarefa de digitar, parece não haver uma divisão consistente de trabalho entre o piloto e o navegador. Ao invés disso, os dois programadores movem-se de tarefa a tarefa juntos, considerando e discutindo problemas no mesmo nível em termos de estratégia ou abstração. Assim, de acordo com Bryant, Romero e Boulay [136], a passividade não estaria diretamente relacionada ao simples ato de dominar ou não o teclado, mas às atitudes dos sócios quando se comunicam e quando há conflito.

3.4.3 Escolha dos pares

Os estudos [7, 29, 41, 51, 72, 102, 113, 134] investigaram formação aleatória das duplas. Em sua grande maioria, ficou clara a intenção de se beneficiar do rodízio dos pares, justificando o emparelhamento aleatório, pois é defendido que o rodízio com composição aleatória permite aos participantes uma troca maior de experiências emocionais e habilidades lógicas.

Os estudos [20, 34, 38, 40, 43, 63, 78, 85, 125, 130, 133-135] abordaram composição pré-definida dos pares. Enquanto alguns permitiram emparelhamento escolhido pelos participantes, outros propuseram emparelhamento por nível de habilidade, semelhança e diferença de personalidade, autoconfiança semelhante, etnicidade semelhante, gênero semelhante, amor-próprio semelhante e por afinidade.

As observações abaixo foram consensuais entre os estudos referenciados na presente seção:

- É importante incentivar a comunicação entre o navegador e o piloto para que ambos consigam acompanhar o raciocínio do parceiro.
- O aculturamento dos participantes em atividades colaborativas é primordial para que metodologias de implantação de PP sejam bem sucedidas – as pessoas não estão acostumadas a colaborar, em vez disso, são altamente competitivas, motivo pelo qual PP poderia falhar.
- Os participantes precisam ser treinados em PP de forma que seus benefícios sejam plenamente percebidos.

3.5 Diretrizes para o bom funcionamento de PP

À medida que as pesquisas evoluíram no sentido de tornar a PP e a programação colaborativa mais explorada e aprimorada, alguns estudiosos estabeleceram diretrizes de como aplicá-las de maneira mais efetiva. Os estudos [6, 8, 10, 43, 62, 69, 72, 73, 76, 126, 127, 137-148] contribuíram para o estabelecimento de algumas destas diretrizes, dentre elas:

- Descrever e detalhar todos os passos do processo de PP, todos os objetivos e todas as regras de funcionamento. É fundamental que os participantes não tenham dúvidas sobre como devem agir para diminuir o risco de enganos e atitudes prejudiciais ao processo como, por exemplo, a divisão de tarefas e sua solução individual. Corresponde a dizer que os sócios precisam ser treinados em habilidades colaborativas [43, 69, 72, 73, 76, 126, 139-143]. É importante ressaltar que treinamento gera custos. Um estudo interessante sobre o impacto do custo com treinamento em PP, no valor comercial de um projeto de *software*, pode ser encontrado em [149].
- Incentivar que os sócios relatem casos de incompatibilidade da dupla em que estejam, de alguma maneira, degradando o processo de colaboração [126].

- Incentivar a honestidade, a colaboração e o senso de responsabilidade. Sócios, na maioria das vezes alunos, motivados pela natureza corporativista de suas atitudes, tendem a proteger os colegas que não participam idealmente dos emparelhamentos. Essa atitude deve ser desencorajada [43, 126, 127, 140-144, 148].
- Compor duplas compatíveis, seja por habilidade, personalidade, afinidade, ou por outros aspectos. A incompatibilidade pode tornar os alunos menos receptivos à pratica de PP [10, 62, 72, 76, 126, 127, 139, 148].
- Prover tratamento especial aos membros que se mostrarem não receptivos às diretrizes de PP. Reforçar a importância e os benefícios de PP pode ser um caminho promissor nestes casos [126, 127, 148].
- Promover sessões obrigatórias de emparelhamentos em laboratórios quando usado pedagogicamente, pois assegura que uma quantia razoável de tempo será despendida em dupla na tarefa [126, 148].
- Adaptar o cumprimento esperado das tarefas tal que uma porcentagem razoável da classe possa terminá-las no período definido para uma sessão de emparelhamento em laboratório [126, 148].
- Instituir um padrão de codificação, já que programadores possuem estilos distintos de desenvolver suas atividades. Esse diferencial pode ser prejudicial ao emparelhamento [126]. É interessante ressaltar a contraditoriedade pedagógica desta diretriz. Quase todos os estudos onde PP foi avaliada defendem que esse diferencial é primordial para que os alunos compreendam estilos diferentes e aprimorem seus próprios estilos. Porém, padrão de codificação é fundamental quando se pretende instaurar PP na indústria.
- Construir uma cultura orientada para a Programação em Duplas e para a colaboração. As culturas da dominância-complacência e da competitividade devem ser combatidas, se possível, extintas [8, 72, 73, 126, 140-144, 148].
- Incentivar a comunicação ativa entre os sócios durante o emparelhamento: O ideal é que o piloto, enquanto codifica, explique detalhadamente

e discuta com o navegador que resultados pretende obter com cada instrução ou conjunto de instruções adicionadas ao código. O navegador, por outro lado, deve inquirir o piloto sempre que se sentir confuso ou que não concordar com seu parceiro [6, 62, 69, 72, 127, 139, 142, 144].

- Incentivar a troca de conhecimento entre os sócios. Se um sócio conhece melhor um processo ou instrução, é perfeitamente saudável que ele use parte do tempo para ensiná-lo ao seu parceiro [127, 144].
- Promover sucessivas sessões de emparelhamento aleatório até que os participantes escolham o parceiro mais compatível com eles pois isso aumenta a possibilidade de compatibilidade e permite aos sócios conhecerem melhor seus colegas de turma ou de trabalho [127, 148].
- Permitir que os sócios definam o melhor momento de trocar seus papéis. [46] Uma ressalva deve ser feita: o uso desta prática aumenta o risco de os participantes permanecerem constantemente no mesmo papel, o que não seria saudável ao processo de PP.
- Motivar os sócios pois, sócios motivados tendem a emparelhar melhor e resolver suas tarefas com maior satisfação [127, 144].
- Promover *feedback* aos sócios em relação ao comportamento e efetividade da dupla no processo de PP [43].
- Promover rodízio dos pares se o objetivo envolver a troca de informação e conhecimento sobre o sistema ou a transmissão de habilidades [139].
- Conscientizar os participantes sobre a importância em se manter pensamentos sempre positivos, por exemplo: somos capazes, não há nada impossível, entre outros [144].
- Envidar esforços para promover extinção do egocentrismo porque sócios egocêntricos são resistentes à colaboração [6, 144].
- Incentivar a humildade. Os sócios precisam compreender que não são infalíveis e precisam aprender a valorizar a opinião de seus parceiros [73,144].

Quase todas as diretrizes estão relacionadas a algum aspecto do comportamento humano, por esse motivo PP, às vezes, é tão difícil de ser implantada com sucesso.

3.6 Heterogeneidade X homogeneidade em PP

Estudos sobre composição de pares homogêneos e heterogêneos são encontrados em [34, 38, 43, 45, 63, 65, 66, 126,130, 135]. A maioria empregou pesquisa empírica no intuito de verificar a efetividade de emparelhamentos pré-definidos heterogêneos e homogêneos por nível de habilidade, personalidade, autoconfiança, etnicidade, gênero e amor-próprio como meio de encontrar as variáveis que influenciam a compatibilidade dos pares.

Os autores crédulos nos benefícios provindos de emparelhamento homogêneo defendem a compatibilidade dos pares como primordial para o sucesso de práticas de PP, principalmente na indústria de *software*. Porém, isto é fácil de ser justificado. Intuitivamente, duplas heterogêneas aumentam o risco de sócios menos experientes se tornarem passivos, diminuindo sensivelmente os benefícios que PP agregaria ao projeto.

Os estudos de Mujeeb-U-Rehmann [45], especificamente sobre heterogeneidade e homogeneidade das duplas, chegou a conclusões aparentemente contraditórias.

> Na Programação em Duplas, os alunos interagem com suas diferenças de personalidade. A variação na performance é possível se a dupla for homogênea ou heterogênea. Em nossa análise empírica para demonstrar tais variações em termos de eficiência, precisão e qualidade, quantidade de defeitos, visualização de idéias e o potencial criativo entre as duplas durante a atividade de programação, percebeu-se que as duplas heterogêneas provaram lidar melhor com a resolução dos problemas que inicialmente aparecem, e assim tiveram maior êxito na visualização de idéias bem como um potencial criativo mais apurado. As análises empíricas comparativas demonstraram que as duplas heterogêneas se saíram melhor do que as homogêneas. Os dados demonstram que as duplas femininas homogêneas ficaram em segundo lugar durante o processo geral de programação [45].

As conclusões dos autores parecem contraditórias à crença de que a heterogeneidade pode levar a um índice de passividade prejudicial ao processo de PP. Porém, nos estudos supracitados, não fica claro se foram resolvidos e como foram resolvidos os problemas de dominância/complacência e passividade de sócios menos experientes. Por outro lado, nada foi relatado a respeito de como os pares foram moderados, deixando, portanto, uma dúvida latente: "Duplas heterogêneas normalmente são compostas de membros muito experientes ou, pelo menos, existe uma grande possibilidade de que isto ocorra. Neste sentido, a causa do sucesso das duplas heterogêneas não poderia ser atribuída aos alunos mais experientes?" Esta, certamente, é uma pergunta que ainda precisa ser respondida.

Em termos pedagógicos, ao utilizar PP como prática no apoio ao ensino-aprendizado de programação de computadores, espera-se que o conhecimento seja disseminado através da colaboração e do conflito, em geral muito ocorrentes em duplas homogêneas. Por outro lado, em duplas heterogêneas, os sócios experientes, quando treinados para tal, podem aproveitar o processo de emparelhamento no intuito de transmitir seus conhecimentos e habilidades aos menos experientes. Neste sentido, talvez seja interessante que os instrutores pratiquem uma certa alternância de heterogeneidade e homogeneidade em emparelhamentos sucessivos, de modo que os aprendizes se beneficiem de ambas as experiências.

3.7 PP distribuída x PP co-situada

Estudos sobre PP distribuída podem ser encontrados em [9, 12, 13, 16, 22, 26, 28-31, 35, 52, 62, 81, 108-110, 113, 122, 124, 150-153]. Enquanto alguns exploraram ferramentas comerciais para prover PP, outros se preocuparam em desenvolver aplicativos específicos. As conclusões dos trabalhos em que PP distribuída foi comparada à PP co-situada são destacadas abaixo:

- Pares co-situados não alcançaram resultados estatísticos significativamente melhores que os pares distribuídos [12, 13].
- PP distribuída pode resultar em código de igual qualidade se comparada à PP co-situada [62].

- Pares distribuídos mantêm muitos dos benefícios (pressão dos pares, transmissão de conhecimento, entre outros) relatados em pares co-situados [62].
- Embora não seja estatisticamente significativo ou conclusivo, pares de estudantes distribuídos apresentaram rendimento superior aos pares co-situados nos exames finais [26].
- Ferramentas de PP distribuída provêem muitas informações importantes em relação à comunicação, histórico do trabalho, situação atual, tempo decorrido, etc, aos gerentes de projetos [110].
- A comunicação é fator preponderante para o sucesso de PP. O processo de PP falha quando uma ferramenta de PP distribuída não provê comunicação e colaboração efetivamente [16. 153]. Canfora et al. [16] descreveram as características exigidas para que ambientes de PP distribuídos sejam efetivos. Segundo os autores, esses ambientes devem permitir comunicação por áudio, videoconferências e compartilhamento de código.

Em um estudo empírico usando PP distribuída, Hanks [29] levantou duas dúvidas importantes: (1) Professores podem influenciar as atitudes dos alunos perante PP? e (2) Professores podem influenciar o aprendizado geral em ambientes de PP? Os autores sugeriram a necessidade de novas pesquisas para estudar as atitudes dos alunos e instrutores diante de técnicas pedagógicas no intuito de torná-las mais efetivas.

Algumas dicas importantes para o bom funcionamento de PP distribuído podem ser encontradas em [62]. Os autores discutiram diretrizes básicas e concluíram que o maior problema em ambientes de PP distribuída está relacionado à comunicação entre os participantes. Sugeriu-se também que esses ambientes sejam providos de recursos de áudio que permitam comunicação do tipo voz sobre IP.

3.8 O impacto de PP aplicado à grupos minoritários e às mulheres

PP parece ser muito efetivo quando grupos minoritários são o público alvo. Os estudos [11, 14, 29, 32, 34, 40, 43, 60, 69, 70, 79, 154] procuraram medir

a eficiência da prática de PP com grupos minoritários e com mulheres. Alguns resultados podem ser encontrados a seguir:

- Indivíduos de grupos minoritários trabalham melhor com parceiros também membros de grupos minoritários [34].
- Ao programar com um sócio, mulheres acreditam que solucionaram problemas que não seriam capazes de resolver sozinhas [70].
- PP influencia positivamente na diminuição da evasão de sócios do sexo feminino [40, 69, 70].
- Existem indícios de que PP é especialmente benéfica a estudantes não brancos [14, 79].
- Existem indícios de que PP é especialmente benéfica a estudantes do sexo feminino [14, 29, 79].

Apesar de que os benefícios advindos de PP aplicada a grupos minoritários e estudantes do sexo feminino parecem ser consensuais na literatura, acredita-se que a quantidade de estudos sobre o tema não é muito expressiva.

3.9 A pressão dos pares decorrente de PP

Um fenômeno constantemente relatado em [32, 40, 57, 62, 65, 74, 75, 77, 92, 105, 134, 140, 155, 156] foi identificado como pressão dos pares. Programadores emparelhados colocam uma forma positiva de pressão nos seus respectivos parceiros. Os programadores admitem que trabalham firme e bem mais atentos porque eles não querem deixar seus parceiros falharem. Também, quando se encontram em seções de emparelhamento, ambos trabalham intensivamente porque são altamente motivados a completar sua tarefa durante a sessão. Parece que duas pessoas trabalhando de maneira emparelhada tornam seus respectivos tempos mais valiosos: eles tendem a deixar de fazer ligações curtas; eles não checam mensagens de *e-mails* ou páginas da *web*; eles não desperdiçam o tempo uns dos outros. À medida que cada um se concentra em sua atividade, ambos se mantém focados nas tarefas, e melhorias são realizadas [156].

Outro benefício da pressão dos pares é a adesão a procedimentos e padrões. Devido à natureza humana, os pares colocam uma forma positiva de pressão uns

nos outros para seguir os processos orientados. Adicionalmente, a pressão dos pares faz os estudantes seguirem o processo e praticar o que o instrutor está ensinando de fato. Até mesmo quando eles têm vontade de saltar um passo importante do processo, como documentar o projeto, eles ficam envergonhados de relatar ao sócio ou o mesmo os persuade a completar os passos [75].

4 Conclusões e trabalhos futuros

Em nossos estudos, percebemos que PP não foi aplicada na grande maioria dos países em desenvolvimento, apesar de seu potencial pedagógico. Acredita-se que existe uma grande demanda pela aplicação de suas práticas nesses países.

Estudos empíricos acadêmicos comprovaram o potencial de PP no ensino em cursos de computação e afins, principalmente em disciplinas introdutórias de programação de computadores. De acordo com a literatura, os benefícios pedagógicos de PP são inquestionáveis. Por outro lado, em face de os benefícios de PP em ambientes reais de desenvolvimento ainda serem contraditórios, novas pesquisas empíricas na indústria de software se fazem necessárias.

Diversas ferramentas foram produzidas para prover PP co-situada e distribuída. Defendemos que novos estudos empíricos sobre o uso dessas ferramentas ajudariam a tornar a PP mais presente em instituições de ensino superior em computação.

É importante compreender as diferenças entre o potencial de PP na indústria e na academia. Embora congruentes sob alguns aspectos, diversos fatores podem ajudar futuros pesquisadores a se decidirem por que caminho investigativo trilhar.

Metodologias pragmáticas podem incentivar e facilitar o uso de práticas de PP em disciplinas introdutórias de programação. A disseminação dessas práticas pode ter influência direta na qualidade dos profissionais egressos de cursos de informática. A literatura defende que os alunos precisam aprender técnicas colaborativas o mais cedo possível para terem sucesso no mercado.

Parece que a heterogeneidade e homogeneidade das duplas influenciam substancialmente na efetividade de PP. Esta influência, não obstante, diferencia-se em relação aos objetivos (acadêmicos ou mercadológicos) do seu uso.

A literatura aprova o uso de PP distribuída, porém, é importante verificar e garantir a qualidade da comunicação e coordenação de atividades para que realmente demonstre efetividade. Sistemas que suportam comunicação por voz sobre IP são ideais.

PP parece ser especialmente importante para grupos minoritários e para mulheres. Não obstante, acredita-se que novas pesquisas são necessárias no sentido de esclarecer as causas deste achado.

A pressão dos pares é fator preponderante pelo qual PP obtém sucesso. A literatura defende a pressão dos pares neste sentido. É comum perceber que os participantes de equipes emparelhadas desenvolvem um senso de responsabilidade que os torna mais ativos.

Aplicamos experimentos de Programação em Duplas com alguns alunos de um curso de Sistemas de Informação entre o primeiro semestre de 2008 e o segundo semestre de 2009 para verificar os benefícios pedagógicos do uso de PP em disciplinas de programação. Resultados destes experimentos e outros estudos em PP podem ser vistos em nossos artigos já publicados no XXII Simpósio Brasileiro de Engenharia de Software (SBES 2008), no 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008), no VI International Conference on Engineering and Computer Education (ICECE 2009) e na Revista de Iniciação Científica da SBC (REIC – publicado em março de 2010).

Acreditamos que trabalhos futuros podem estar relacionados à aplicação de PP em outros cursos que possuam a disciplina de programação em seus currículos, ainda que elas não sejam consideradas disciplinas-chave nestes cursos (preferencialmente em instituições acadêmicas públicas, no intuito de diferenciar de nossas pesquisas anteriores). É importante também ressaltar que todos os temas-chave descritos na seção 3 deste artigo são combustíveis para incentivar a produção de trabalhos futuros sobre PP.

5 Referências

[1] ZUALKERNAN, I. A. Using Soloman-Felder Learning Style Index to Evaluate Pedagogical Resources for Introductory Programming Classes. *Proc Int Conf Software,* p. 723-726, 2007.

[2] CONSTANTINE, L. L. Constantine on Peopleware. *Yourdon Press Computing Series*, ed. E. Yourdon, Englewood Cliffs, NJ: Yourdon Press, 1995.

[3] NAWROCKI, J.; WOJCIECHOWSKI, A. Experimental Evaluation of Pair Programming. Presented at European Software Control and Metrics, 2001.

[4] COPLIEN, J. O. A Development Process Generative Pattern Language. In: Pattern Languages of Program Design, J. O. Coplien and D. C. Schmidt, Ed. Reading Mass: Addison-Wesley, p.183-237, 1995.

[5] NOSEK, J. T. The Case for Collaborative Programming. *Commun ACM*, p.105-108, 1998.

[6] WILLIAMS, L.; KESSLER, R. R.; CUNNINGHAM, W.; JEFFRIES, R.; Strengthening the Case for Pair Programming. *IEEE Software*, v. 17, n. 4, p. 19-25, 2000.

[7] MENDES, E.; AL-FAKHRI, L. B.; LUXTON-REILLY, A. Investigating Pair Programming in a 2nd-year software development and design computer science course, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, *SIGCSE Bull,* 2005.

[8] AIKEN, J. Technical and Human Perspectives on Pair Programming, *ACM SIGSOFT,* v. 29, n. 5, 2004.

[9] MENDES, A. J.; GOMES, A.; ESTEVES, M.; MARCELINO, M. J.; BRAVO, C.; REDONDO, M. A. Using simulation and collaboration in CS1 and CS2. In: ITiCSE, p. 193-197, 2005.

[10] ALLY, M.; DARROCH, F.; TOLEMAN, M. A Framework for Understanding the Factors Influencing Pair Programming success. Proceedings of the XP'05 Conference, 2005.

[11] BAGLEY, C. A.; CHOU, C. C. Collaboration and the importance for novices in learning java computer programming. Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, *SIGCSE Bull*, v. 39, n. 3, p. 211-215, 2007.

[12] BAHETI, P.; WILLIANS, L.; GEHRINGER, E.; STUTTS, D. Exploring Pair Programming in Distributed Object-Oriented Team Projects. In: OOPSLA'02 Educator's Symposium, 2002.

[13] BAHETI, P.; GEHRINGER, E.; STOTTS, D. Exploring the efficacy of distributed pair programming. In: Extreme Programming and Agile Methods - XP/Agile Universe, n. 2418 in LNCS, Springer, p. 208-220, 2002.

[14] BECK, L. L.; CHIZHIK, A. W.; MCELROY, A. C. Cooperative learning techniques in CS1: design and experimental evaluation. Proceedings of the 36th SIGCSE technical symposium on Computer science education, *SIGCSE Bull*, 2005.

[15] BRERETON, P.; TURNER, M.; KAUR, R. Pair programming as a teaching tool: A student review of empirical studies. Proceedings of 22nd Conference on Software Engineering Education and Training, p. 240-247, 2009.

[16] CANFORA, G.; CIMITILE, A.; DI LUCCA, G. A.; VISAGGIO, C. A. How distribution affects the success of pair programming. *Int JSoftw Eng Know*, v. 16, n. 2, p.293-313, 2006.

[17] CHIGONA, W.; POLLOCK, M. Pair Programming for Information Systems Students New to Programming: Students Experiences and Teachers Challenges, PICMET'08 Conference, 2008.

[18] CHOI, K. S.; DEEK, F. P.; IM, I. Exploring the underlying aspects of pair programming: The impact of personality. *Inform Software Tech*, v. 50, n. 11, p.1114-1126, 2008.

[19] CIOLKOWSKI, M.; SCHLEMMER, M. Experiences with a Case Study on Pair Programming. First International Workshop on Empirical Studies in Software Engineering, 2002.

[20] CLIBURN, D. C. Experiences with pair programming at a small college. *J Comput Sci Coll*, v. 19, n. 1, p. 20-29, 2003.

[21] COSTA, R. H. P.; FARIA, E. S. J.; YAMANAKA, K. Programação em Duplas no Aprendizado de Programação de Computadores em um Curso de Engenharia de Produção: Um Estudo Empírico. *REIC – Revista Eletrônica de Iniciação Científica* (Online), v. 10, n. 1, 2010.

[22] CUBRANIC, D.; STOREY, M. A. D.; RYALL, J. A Comparison of Communication Technologies to Support Novice Team Programming. ICSE´06, 2006.

[23] DECLUE, T. H. Pair programming and pair trading: effects on learning an motivation in a CS2 course. *J Comput Sci Coll*, v. 18, n. 5, p.49-56, 2003.

[24] DOU, W; HE, W. Compatibility and Requirements Analysis of Distributed Pair Programming. In: Second International Workshop on Education Technology and Computer Science, p. 467-470, 2010.

[25] GEHRINGER, E. F. A pair-programming experiment in a non-programming course. In: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, p. 187-190, 2003.

[26] HANKS, B. F. Distributed Pair Programming: An Empirical Study. Extreme Programming and Agile Methods - XP/Agile, Universe, 2004.

[27] HANKS, B. F.; MCDOWELL, C.; DRAPER, D.; KRNJAJIC, M. Program Quality with Pair Programming in CS1. In: SIGCSE Conference on Innovation and Technology in Computer Science Education, p. 176-180, 2004.

[28] HANKS, B. Student performance in CS1 with distributed pair programming. Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education, p. 316–320, 2005.

[29] _____. Student attitudes toward pair programming. In: SIGCSE Conference on Innovation and Technology in Computer Science Education, p. 113-117, 2006.

[30] _____. Empirical evaluation of distributed pair programming. *Int J Hum-Comput St,* v. 66, n. 7, p. 530-544, 2008.

[31] HICKEY, T. J.; LANGHTON, J; ALTERMAN, R. Enhancing CS programming lab courses using collaborative editors. *J Comput Sci Coll*, v. 20, n. 3, p. 157-167, 2005.

[32] HO, C.; SLATEN, K.; WILLIAMS, L.; BERENSON, S. Work in progress-unexpected student outcome from collaborative agile software development practices and paired programming in a software engineering course. In: Frontiers in Education, Session F2C, p. 15-16, 2004.

[33] JAMI, S.; IMRAN, S.; ZUBAIR A. Teaching Computer Science Courses Using Extreme Programming (XP) Methodology. In: 9th International Multitopic Conference, p.1-6, 2005.

[34] KATIRA, N.; OSBORNE, J.; WILLIAMS, L. Towards increasing the compatibility of student pair programmers, International Conference on Software Engineering, p. 625-626, 2005.

[35] LANGHTON, J. T.; HICKEY, T. J.; ALTERMAN, R. Integrating tools and resources: a case study in building educational groupware for collaborative programming. *J Comput Sci Coll*, v. 19, n. 5, p. 140-153, 2004.

[36] LUI, K. M.; CHAN, K. C. C. A Cognitive Model for Solo Programming and Pair Programming. Proceedings of the Third IEEE International Conference on Cognitive Informatics, p. 94-102, 2004.

[37] _____. Pair programming productivity: Novice–novice vs. expert–expert. *Int J Hum-Comput St*, v. 64, n. 9, p. 915-925, 2006.

[38] MATZKO, S.; DAVIS, T. Pair design in undergraduate labs. *J Comput Sci Coll*, v. 22, n. 2, p. 123–130, 2006.

[39] MCDOWELL, C.; WERNER, R.; BULLOCK, H. E.; FERNALD, J. The effects of Pair Programming on performance in an introductory programming course. Proceedings of the 33rd SIGCSE technical symposium on Computer science education, *SIGCSE Bull,* 2002.

[40] _____|. The impact of pair programming on student performance, perception and persistence. *Proc Int Conf Softw,* 2003.

[41] MCDOWELL, C.; HANKS, B.; WERNER, L. Experimenting with pair programming in the classroom. *SIGCSE Bull,* p. 60-64, 2003.

[42] MCDOWELL, C.; WERNER, R.; BULLOCK, H. E.; FERNALD, J. Pair programming improves student retention, confidence, and program quality. *Commun ACM*, v. 49, n. 8, p. 90-95, 2006.

[43] MCKINNEY, D.; DENTON, L. F. Developing collaborative skills early in the CS curriculum in a laboratory environment. Proceedings of the 37th SIGCSE technical symposium on Computer science education, *SIGCSE Bull,* 2006.

[44] MENDES, E.; AL-FAKHRI, L.; LUXTON-REILLY, A. A replicated experiment of Pair Programming in a 2nd-year software development and design computer science course. *ACM SIGCSE Bull*, v. 38, n. 3, 2006.

[45] MUJEEB-U-REHMAN, M.; YANG, X.; DONG, J.; ABDUL GHAFOOR, M. Heterogeneous and homogenous pairs in Pair Programming: an empirical analysis. In: Conference on Electrical and Computer Engineering, p. 1116-1119, 2005.

[46] MÜLLER, M. M. Are Reviews an Alternative to Pair Programming?. In: 7th International Conference on Empirical Assessment in Software Engineering, 2003.

[47] _____. Two controlled experiments concerning the comparison of pair programming to peer review. *J Syst Software*, v. 78, n. 2, p. 166-179, 2005.

[48] _____. A preliminary study on the impact of a pair design phase on pair programming and solo programming. *Inform Software Tech*, v. 48, n. 5, p. 335-344, 2006.

[49] _____. Do programmer pairs make different mistakes than solo programmers?. *J Syst Software*, v. 80, n. 9, p. 1460-1471, 2007.

[50] NAGAPPAN, N.; WILLIAMS, L.; WIEBE, E.; MILLER, C.; BALIK, S.; FERZLI, M.; PETLICK, M. Pair Learning: With an Eye Toward Future Success. Extreme Programming/Agile Universe, p.185-198, 2003.

[51] NAGAPPAN, N.; WILLIAMS, L.; FERZLI, M.; WIEBE, E.; YANG, K.; MILLER, C.; BALIK, S. Improving the CS1 Experience with Pair Programming. In: ACM Technical Symposium on Computer Science Education, p.359-362, 2003.

[52] NATSU, H.; FAVELA, J.; MORÁN, A. L.; DECOUCHANT, D.; MARTINEZ-ENRIQUEZ, A. M. Distributed Pair Programming on the Web. Proceedings of the Fourth Mexican International Conference on Computer Science, 2003.

[53] RAMLI, N.; FAUZI, S. S. M. The effects of pair programming in programming language subject. International Symposium on Information Technology, v. 1, p. 1-4, 2008.

[54] SALLEH, N.; MENDES, E.; GRUNDY, J. Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE T Software Eng*, 2010.

[55] SALLEH, N.; MENDES, E.; GRUNDY, J.; BURCH, G. S. An empirical study of the effects of conscientiousness in pair programming using the five-factor personality model. *Proc ACM/IEEE Int Conf Softw*, v. 1, p. 577-586, 2010.

[56] SANDERS, D. Student Perceptions of the Suitability of Extreme and Pair Programming. In: Extreme Programming Examined, M. Marchesi, G. Succi, D. Wells, and L. Williams, Eds. Boston, MA: Addison Wesley, p. 261-271, 2002.

[57] SHERRELL, L. B.; ROBERTSON, J. J. Pair programming and agile software development: experiences in a college setting. *J Comput Sci Coll*, v. 22, n. 2, p. 145-153, 2006.

[58] SISON, R. Investigating Pair Programming in a Software Engineering Course in an Asian Setting. *Asia Pac Softwr Eng*, p. 325-331, 2008.

[59] _____. Investigating the Effect of Pair Programming and Software Size on Software Quality and Programmer Productivity. *Asia Pac Softwr Eng*, p. 187-193, 2009.

[60] SLATEN, K.M.; DROUJKOVE, M.; DERENSON, S. B.; WILLIAMS, L.; LAYMAN, L. Undergraduate student perceptions of pair programming and agile software methodologies: verifying a model of social interaction. Agile '05, p. 323-330, 2005.

[61] SRIKANTH, H.; WILLIAMS, L.; WIEBE, E.; MILLER, C.; BALIK, S. On Pair Rotation in the Computer Science Course. Conference on Software Engineering Education and Training, p. 144-149, 2004.

[62] STOTTS, D.; WILLIAMS, L.; NAGAPPAN, N.; BAHETI, P.; JEN, D.; JACKSON, A. Virtual Teaming: Experiments and Experiences with Distributed Pair Programming. Extreme Programming/Agile Universe, p.129-141, 2003.

[63] THOMAS, L.; RATCLIFFE, M.; ROBERTSON, A. Code warrior and code-a-phobes: a study in attitude and pair programming. In: ACM Technical Symposium on Computer Science Education, p. 363-367, 2003.

[64] TOMAYKO, J. E. A Comparison of Pair Programming to Inspections for Software Defect Reduction. *J Comput Scie Educ*, v. 12, p. 213–222, 2002.

[65] VAN TOLL III, T.; LEE, R.; AHLSWEDE, T. Evaluating the Usefulness of Pair Programming in a Classroom Setting. In: 6th International Conference on Computer and Information Science, p. 302–308, 2007.

[66] VANDEGRIFT, T. Coupling pair programming and writing: learning about students' perceptions and processes. In: ACM Technical Symposium on Computer Science Education, p. 2-6, 2004.

[67] XU, S.; CHEN, X. Pair programming in software evolution. In: Conference on Electrical and Computer Engineering, 2005, p.1846–1849.

[68]XU, S.; RAJLICH, V. Pair Programming in Graduate Software Engineering Course Projects. *Proc Front Educ Conf*, 2005.

[69] WERNER, L.; DENNER, J.; BEAN, S. Pair Programming Strategies for Middle School Girls. In: International Conference on Computers and Advanced Technology in Education, p. 161-166, 2004.

[70] WERNER, L.; HANKS, B.; MCDOWELL, C. Pair programming helps female computer science students persist. *ACM Journal of Educational Resources in Computing*, v. 4, n. 1, 2005.

[71] WHITTINGTON, K. J. Infusing active learning into introductory programming courses. *JComput Sci Coll*, v. 19, n. 5, p. 249-259, 2004.

[72] WIEBE, E.; WILLIAMS, L.; PETLICK, J.; NAGAPPAN, N.; BALIK, S.; MILLER, C.; FERZLI, M. Pair Programming in Introductory Programming Labs. In: American Society for Engineering Education Annual Conference & Exposition, 2003.

[73] WILLIAMS, L. But, isn't that cheating?. In: Frontiers in Education, 1999.

[74] WILLIAMS, L.; KESSLER, R.R. The Effects of Pair-Pressure and Pair-Learning on Software Engineering Education. Conference of Software Engineering Education and Training, 2000.

[75] WILLIAMS, L.; KESSLER, R.R. Experimenting with Industry's Pair Programming Model in the Computer Science Classroom. *Computer Science Education*, v. 11, n. 1, p. 7-20, 2001.

[76] WILLIAMS, L.; YANG, K.; WIEBE, E.; FERZLI, M.; MILLER, C. Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations. In: OOPSLA Educator's Symposium, 2002.

[77] WILLIAMS, L.; WIEBE, E.; YANG, K.; FERZLI, M.; MILLER, C. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*, v. 12, n. 3, p. 197-212, 2002.

[78] WILLIAMS, L.; MCDOWELL, C.; FERNALD, J.; WERNER, L.; NAGAPPAN, N. Building Pair Programming Knowledge Through a Family of Experiments. In: IEEE International Symposium on Empirical Software Engineering, p. 143-152, 2003.

[79] WILLIAMS, L.; LAYMAN, L.; SLATEN, K. M.; BERENSON, S. B.; SEAMAN, C. On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering. In: 29th International Conference on Software Engineering, p. 677–687, 2007.

[80] YANG, Y. Application of software engineering approaches to interdisciplinary education. In: 4th International Conference on Computer Science & Education, p. 1678-1682, 2009.

[81] ZACHARIS, N. Z. Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course. IEEE T Educ. v. PP, n. 99, p. 1-1, 2010.

[82] ARISHOLM, E.; GALLIS, H.; DYBÅ, T.; SJØBERG, D. I. K. Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE T Software Eng*, v. 33, n. 2, 2007.

[83] BELSHEE, A. Promiscuous Pairing and Beginner's Mind: Embrace Inexperience, Proceedings of the Agile Development Conference, 2005.

[84] CANFORA, G.; CIMITILE, A.; GARCIA, F.; PIATTINI, M. VISSAGIO, C. A. Evaluating performances of pair designing in industry. *J Syst Software*, v. 80, n. 8, p.1317-1327, 2007.

[85] CHONG, J.; HURLBUTT, T. The Social Dynamics of Pair Programming. *Proc Int Conf Softw*, p.354-363, 2007.

[86] COHAN, S. Successful Customer Collaboration Resulting in the Right Product for the End User. Proceedings of the Agile'08, p. 284-288, 2008.

[87] DOMINO, M. A.; COLLINS, R. W; HEVNER, A. R. Controlled experimentation on adaptations of pair programming. *Inform Technol Manag*, Springer Netherlands, v. 8, n. 4, p. 297-312, 2007.

[88] FRONZA, I.; SILLITTI, A.; SUCCI, G. An interpretation of the results of the analysis of pair programming during novices integration in a team. Proceedings of the 3rd international Symposium on Empirical Software Engineering and Measurement, p. 225-235, 2009.

[89] GALLIS, H.; ARISHOLM, E.; DYBÅ, T. A Transition from Partner Programming to Pair Programming - an Industrial Case Study. Pair Programming Work Shop in 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'02), 2002.

[90] HANNAY, J. E.; ARISHOLM, E.; ENGUIK, K.; SJØBERG, D. I. K. Effects of Personality on Pair Programming, *IEEE T Software Eng*, v. 36, n. 1, p. 61-80, 2010.

[91] HAUNGS, J. Pair programming on the C3 project. *Computer*, v. 34, p. 118-119, 2001.

[92] HULKKO, H.; ABRAHAMSSON, P. A multiple case study on the impact of pair programming on product quality. *Proc Int Conf Softw*, 2005.

[93] JENSEN, R. W. A Pair Programming Experience. *CrossTalk, The Journal of Defense Software Engineering*, v. 16, p. 22–24, 2003.

[94] KOLCHIER, K. Exploring Synergistic Impact through Adventures in Group Pairing. Agile'09, p. 265-270, 2009.

[95] LUI, K. M.; CHAN, K. C. C. When Does a Pair Outperform Two Individuals? In: XP'03, 2003.

[96] LUI, K. M.; CHAN, K. C. C. Software process fusion by combining pair and solo programming. *IET Software*, v. 2, n. 4, p. 379–390, 2008.

[97] LUI, K. M.; CHAN, K. C.C.; NOSEK, J. T. The Effect of Pairs in Program Design Tasks[J]. *IEEE T Software Eng*, v. 34, n. 2, p. 197-211, 2008.

[98] PATEL, C.; RAMACHANDRAN, M. Bridging Best Traditional SWD Practices with XP to Improve the Quality of XP Projects. In: International Symposium on Computer Science and its Applications, Australia, 2008.

[99] ROSTAHER, M.; HERICKO, M. Tracking Test First Pair Programming – An Experiment, Extreme Programming and Agile Methods - XP/Agile Universe, p. 174-184, 2002.

[100] SATO, D. T.; CORBUCCI, H.; BRAVO, M. V. Coding Dojo: An environment for Learning and Sharing Agile Practice. Proceedings of Agile'08, 2008.

[101] SCHINDLER, C. Agile Software Development Methods and Practices in Austrian IT-Industry: Results of an Empirical Study. International Conferences on Computational Intelligence for Modelling, Control and Automation, Intelligent Agents, Web Technologies and Internet Commerce, and Innovation in Software Engineering, p. 321-326, 2008.

[102] VANHANEN, J.; LASSENIUS, C. Effects of pair programming at the development team level an experiment. International Symposium on Empirical Software Engineering, p. 336-345, 2005.

[103] VANHANEN, J.; LASSENIUS, C.; MÄNTYLÄ, M. V. Issues and Tactics when Adopting Pair Programming: A Longitudinal Case Study. In: International Conference on Software Engineering Advances, p. 70, 2007.

[104] WALLE, T.; HANNAY, J. E. Personality and the nature of collaboration in pair programming. Proceedings of the 2009 3rd international Symposium on Empirical Software Engineering and Measurement, p. 203-213, 2009.

[105] COCKBURN, A.; WILLIAMS, L. The Costs and Benefits of Pair Programming. In: Extreme Programming Examined, Succi, G.; Marchesi, M. eds.; Boston, MA: Addison Wesley, p. 223.248, 2001.

[106] PADBERG, F.; MÜLLER, M. M. Analyzing the Cost and Benefit of Pair Programming. Proceedings of the 9th International Symposium on Software Metrics, p. 166, 2003.

[107] BRYANT, S.; ROMERO, P.; BOULAY, B. Pair programming and the re-appropriation of individual tools for collaborative programming, Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, 2005.

[108] HANKS, B. F. Tool Support for Distributed Pair Programming. Workshop on Distributed Pair Programming. Extreme Programming and Agile Methods - XP/Agile Universe, 2002.

[109] ZIN, A. M.; IDRIS, S.; SUBRAMANIAM, N. K. Improving Learning of Programming Through E-Learning by Using Asynchronous Virtual Pair Programming. *The Turkish Online Journal of Distance Education*, v. 7, n. 3, p. 162-173, 2006.

[110] ATSUTA, S.; MATSUURA, S. Extreme Programming support tool in distributed environment. *P Int Comp Softw App*, v. 2, p. 32-33, 2004.

[111] CUBRANIC, D.; STOREY, M. A. D. Collaboration support for novice team programming. Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting group work, 2005.

[112] DEFRANCO-TOMMARELLO, J.; DEEK, F. P. Collaborative software development: a discussion of problem solving models and groupware technologies. *P Ann H I C S S*, p. 568-577, 2002.

[113] _____. An on-line tutorial for collaborative problem solving and software development. Proceedings of the 6th Conference on Information Technology Education, p. 349–352, 2005.

[114] DOU, W.; HONG, K.; ZHANG, X. A Framework of Distributed Pair Programming System. In: International Conference on Computational Intelligence and Software Engineering, p. 1-4, 2009.

[115] DOU, W.; HONG, K.; HE, W. A conversation model of collaborative pair programming based on language/action theory. In: International Conference on Computer Supported Cooperative Work in Design, p. 7-12, 2010.

[116] DOU, W.; HE, W. A Preliminary Design of Distributed Pair Programming System. In: Second International Workshop on Education Technology and Computer Science, p. 256-259, 2010.

[117] DUQUE, R.; BRAVO, C. Analyzing Work Productivity and Program Quality in Collaborative Programming. *Proc Int Conf Softw*, p. 270-276, 2008.

[118] _____. Supporting Distributed Pair Programming with the COLLECE Groupware System: An Empirical Study. Agile Processes in Software Engineering and Extreme Programming (XP'08), *Lect Notes Comput Sc*, p. 232-233, 2008.

[119] FARIA, E. S. J. YAMANAKA, K.; TAVARES, J. A.; PINTO, G. H. L.; MELO, L. H. S. AIDDES - Distributed Intelligent Pair-Software Development Environment. In: 32nd Annual IEEE International Computer Software and Applications Conference*, P Int Comp Softw App*, 2008.

[120] _____. Intelligent Software Agents Mediating the Pair Participation in a Distributed Intelligent Pair-Software Development Environment. In: 3rd IEEE International Workshop on Engineering Semantic Agent Systems in conjunction with 32nd Annual IEEE International Computer Software and Applications Conference, 2008.

[121] FARIA, E. S. J. YAMANAKA, K.; TAVARES, J. A.; PINTO, G. H. L.; MELO, L. H. S. Distributed Intelligent Pair-Software Development Tool. In: Workshop em Desenvolvimento Distribuído de Software co-localizado com o Simpósio Brasileiro de Engenharia de Software, 2008.

[122] FLOR, N. V. Globally distributed software development and pair programming. *Commun ACM*, v. 49, n. 10, 2006.

[123] HAN, K.; LEE, E.; LEE, Y. The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course. *IEEE T Educ*, v. 53, n. 2, p. 318-327, 2010.

[124] HO, C.; RAHA, S.; GEHRINGER, E.; WILLIAMS, S. Sangam: a distributed pair programming plug-in for Eclipse. In: OOPSLA workshop on Eclipse Technology Exchange (Eclipse '04), p. 73-77, 2004.

[125] SHAW, A. C. Extending the Pair Programming Pedagogy to Support Remote Collaborations in CS Education. *Proc Int Conf Inf Tech:* New Generations, p. 1095-1099, 2009.

[126] BEVAN, J.; WERNER, L.; MCDOWELL, C. Guidelines for the Use of Pair Programming in a Freshman Programming Class. Proceedings of the 15th Conference on Software Engineering Education and Training, p. 100, 2002.

[127] FERZLI, M.; WIEBE, E.; WILLIAMS, L. Paired Programming Project: Focus Groups with Teaching Assistants and Students. *NCSU Technical Report* (TR-2002-16), 2002.

[128] FREUDENBERG, S.; ROMERO, P.; BOULAY, B. 'Talking the talk': Is Intermediate-level conversation the key to the pair programming success story? In: Agile'07, p. 84-91, 2007.

[129] GEHRINGER, E. F.; DEIBEL, K.; HAMER, J.; WHITTINGTON, K. J. Cooperative learning: beyond pair programming and team projects. Proceedings of the 37th SIGCSE technical symposium on Computer science education, *SIGCSE Bull*, v. 38, n. 1, p. 458–459, 2006.

[130] KATIRA, N.; WILLIAMS, L.; WIEBE, E.; MILLER, C.; BALIK, S.; GEHRINGER, E. On Understanding Compatibility of Student Pair Programmers, In: ACM Technical Symposium on Computer Science Education, p. 7-11, 2004.

[131] SEGUNDO, V. P. M.; MOURA, A. M. Introdução de Metodologias Ágeis de Desenvolvimento de Software nos Currículos de Referência do Ensino Universitário. Disponível em: <http://www.frb.br/ciente/Impressa/Info/2004.2/Introducao%20de%20metodologias.pdf.> Acesso em: 10/12/2007.

[132] SHARIFLOO, A. A.; SAFFARIAN, A. S.; SHAMS, F. Embedding Architectural Practices into Extreme Programming. In: 19th Australian Conference on Software Engineering, p. 310-319, 2008.

[133] VANHANEN, L. K. Experiences of Using Pair Programming in an Agile Project. *P Ann H I CS S,* p. 274b, 2007.

[134] WILLIAMS, L.; LAYMAN, L.; OSBORNE, J.; KATIRA, N. Examining the Compatibility of Student Pair Programmers. AGILE'06, p. 411-420, 2006.

[135] YUKSEL, A. Pair programming: matching pairs in a sequence of learning sessions. In: 8th Human Centred Technology Postgraduate Workshop (Advancing the Potential for Communication, Learning and Interaction), 2005.

[136] BRYANT, S.; ROMERO, P.; BOULAY, B. Pair programming and the mysterious role of the navigator. *Int J Hum-Comput St*, v. 66, n. 7, p. 519-529, 2008.

[137] DYBÅ, T.; ARISHOLM, E.; SJØBERG, D.; HANNAY, J.; SHULL, F. Are Two Heads Better Than One? On the Effectiveness of Pair Programming. *IEEE Software*, v. 24, n. 6, p. 12-15, 2007.

[138] FARIA, E. S. J.; YAMANAKA, K. How Should The Participants Of The Pair-Programming Process Act? (Guidelines For The Success In Pair Programming). In: VI International Conference on Engineering and Computer Education, 2009.

[139] GALLIS, H.; ARISHOLM, E.; DYBÅ, T. An Initial Framework for Research on Pair Programming. *Proc Int Sym Empirical Softw Engn*, p. 132, 2003.

[140] LEJEUNE, N. Critical Components for Successful Collaborative Learning in CS1. *J Comput Sci Coll*, v. 19, n. 1, p. 275-285, 2003.

[141] PRESTON, D. Pair programming as a model of collaborative learning: a review of the research. *J Comput Sci Coll*, v. 20, n. 4, p. 39-45, 2005.

[142] _____. Adapting Pair Programming Pedagogy for use in Computer Literacy Courses. *J Comput Sci Coll*, v. 21, n. 5, p. 84-93, 2006.

[143] _____. Using collaborative learning research to enhance pair programming pedagogy. *ACM SIGITE Newsletter*, v. 3, n. 1, p. 16–21, 2006.

[144] WILLIAMS, L.; KESSLER, R. R. All I Really Need to Know about Pair Programming I Learned In Kindergarten. *Commun ACM*, May, 2000.

[145] WILLIAMS, L.; MCCRICKARD, D.S.; LAYMAN, L.; HUSSEIN, K. Eleven Guidelines for Implementing Pair Programming in the Classroom. Proceedings of the Agile'08, p. 445-452, 2008.

[146] WRAY, S. How does pair programming work?. *IEEE Software*. v. PP, n. 99, p. 1-1, 2009.

[147] _____. How Pair Programming Really Works. *IEEE Software*. v. 27, n. 1, p. 50-55, 2010.

[148] YERION, K. A.; RINEHART, J. A. Guidelines for collaborative learning in computer science. *ACM SIGCSE Bull*, v. 27, n. 4, p. 29-34, 1995.

[149] PADBERG, F.; MÜLLER, M. M. Modeling the Impact of a Learning Phase on the Business Value of a Pair Programming Project. *Asia Pac Softwr Eng*, 2004.

[150] BRYANT, S.; BOULAY, B.; ROMERO, P. XP and Pair Programming practices. *PPIG Newsletter* (Psychology of Programming Interest Group), v. 30, n. 5, p. 17-20, 2006.

[151] DEWAN, P.; AGRAWAL, P.; SHROFF, G.; HEGDE, R. Distributed side-by-side programming. In: Workshop on Cooperative and Human Aspects on Software Engineering, p. 48-55, 2009.

[152] _____. Experiments in Distributed Side-By-Side Software Development. In: 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2009.[153] Visaggio, C. A. Empirical Validation of Pair Programming. *Proc Int Conf Softw,* p. 654-654, 2005.

[153] VISAGGIO, C. A. Empirical Validation of Pair Programming. *Proc Int Conf Softw,* p. 654-654, 2005.

[154] WILLIAMS, L. Debunking the Nerd Stereotype with Pair Programming (Broadening Participation in Computing Series), *IEEE Computer*, v. 39, n. 5, p. 83-85, 2006.

[155] WILLIAMS, L.; UPCHURCH, R. In Support of Student Pair Programming. ACM Technical Symposium on Computer Science Education, p. 327-331, 2001.

[156] WILLIAMS, L. Integrating Pair Programming into a Software Development Process. In: 14th Conference on Software Engineering Education and Training, p. 27, 2001.