Quando a programação em pares deve ser adotada em um ambiente empresarial

When the pair programming should be adopted in a business environment

José Augusto Fabri
Universidade Tecnológica Federal do Paraná
Programa de Pós-Graduação em Informática (PPGI)
Cornélio Procópio – Paraná - Brasil
fabri@utfpr.edu.br

Alexandre L´Erario
Universidade Tecnológica Federal do Paraná
Programa de Pós-Graduação em Informática (PPGI)
Cornélio Procópio – Paraná - Brasil
alerario@utfpr.edu.br

Resumo — **Atualmente existem várias questões sobre a adoção da programação em pares. Este tipo de arranjo é mais produtivo se comparado ao arranjo individual? É possível estebelecer momentos em que uma empresa deve adotar o arranjo emparelhado no ambiente de programação? Para responder estas questões os autores deste trabalho desenvolveram 7 experimentos controlados, 4 deles no ambiente acadêmico e 3 no ambiente empresarial. O realização dos experimentos mostrou que para problemas (programas) de complexidade alta a produção em pares é mais produtiva (tempo de produção) e provê código de maior qualidade. Com base nos resultados gerados com os experimentos, os autores deste trabalho concluem que o arranjo emparelhado deve ser adotado em um ambiente empresarial somente na solução de problemas (programas) complexos. Para detectar problemas de complexidade alta a empresa deve possuir um processo de software definido e institucionalizado.**

Palavras Chave – Programação em Pares, Programação Individual, Ambiente empresarial

Abstract — **Currently there are several questions about the pair programming adopt. Is the pair programming more productive when compared to individual programming? When a company must adopt pair programming? To answer these questions the authors of this paper developed 7 controlled experiments, 4 of them in the academic environment and 3 in a software company. The experiments show that the pair programming should be used to production of code highly complex. Based on the results generated by the experiments, the authors of this study conclude that the pair programming should be adopted only in a business environment to solve problems (programs) complex. To detect the complexity problems the company has a defined and institutionalized software process.**

Keywords – Pair Programming; Individual Programming, Business Environment.

I. INTRODUÇÃO

O desenvolvimento de um produto caracterizado como software é norteado por um processo de produção. Este processo, por sua vez, é dividido em fases/atividades, tais como: levantamento requisitos, modelagem de negócio, projeto do software, implementação, teste e a implantação. A programação em par é uma das "boas práticas" propostas para atividade de implementação. Mas o que vem a ser tal tipo de programação? Quando ela surgiu? Existem benefícios de se trabalhar com a programação em par? Ela melhora a capacidade produtiva de uma empresa? Se sim, existem resultados quantitativos que comprovam este fato? Quando adotar este arranjo produtivo em uma empresa?

O primeiro relato da programação em par é datado da década de 1990 ([1], [2] [12]), posteriormente, esta prática foi incorporada pela proposta do eXtreme Programming.

A programação em par é uma técnica na qual dois desenvolvedores trabalham em um mesmo problema, ao mesmo tempo e em uma mesma estação de trabalho. Enquanto uma pessoa assume o teclado (o PILOTO) e digita os comandos que farão parte do programa, a outra (o NAVEGADOR) realiza o trabalho de estrategista.

Os benefícios da programação em par são: a revisão constante do código, pois alguns erros são corrigidos no ato; a modelagem do programa pode ficar mais otimizada, pois existem dois programadores trabalhando para solucionar um mesmo problema e a modelagem do programa é, geralmente, mais simples.

Porém existem algumas questões que pairam sobre a prática aqui apresentada: Por que colocar dois programadores para fazer o trabalho de um? Não estamos desperdiçando um programador? Quando a programação em pares deve ser adotada?

Para responder as questões apresentadas nesta introdução os autores deste trabalho desenvolveram vários experimentos, em ambientes empresarial e acadêmico, com o objetivo de buscar indícios para verificar quando a programação em pares deve ser adotada.

Os experimentos mostraram que a programação em par, com equipes experimentes, pode-se mostrar produtiva na resolução de problemas complexos. Na resolução de problemas simples a programação em par não se caracteriza como um arranjo vantajoso.

Para atingir o objetivo apresentado nesta introdução os autores estruturaram este trabalho em seções. A seção 2 apresenta os trabalhos relacionados a técnica de programação em pares. Os métodos e procedimentos utilizados durante a execução dos experimentos é apresentado na seção 3, a seção 4 apresenta os resultados obtidos com a execução dos experimentos, por fim, a seção 5 apresenta indícios para responder algumas questões apresentadas nsta introdução.

II. TRABALHOS REALACIONADOS

Existem diversos trabalhos na litetura que tratam a questão da programação em pares, alguns indicam que este tipo de programação é mais produtiva se comparada a programação individial. A maioria dos trabalhos é realizando em um ambiente acadêmicos, poucos autores transpõe seus experimentos em um ambiente empresarial. Este trabalho apresenta um compilidado de alguns trabalhos.

Wilson et. al. [1] realiza um experiência com 34 alunos do curso graduação da *Temple Universtiy* na *Philadelphia* – USA. O trabalho detecta que a programação em pares e ligeiramente melhor qua a programação individual.

Já Nosek [2] desenvolveu um estudo com 15 programadores em um ambiente empresarial, o autor detectou que a programação se mostra mais eficaz na legibilidade do programa, na qualidade das funcionalidades produzidas e no tempo de produção. O autor detectou também que os programadores quando emparelhados possuem uma maior confiança na realização de seu trabalho.

Na *Utah University*, Williams et. al. [3] desenvolveram uma experiência com 41 estudantes de graduação. Os autores detectam que os programas quando produzidos em pares passam em um número maior de casos de testes quando comparados com a produção dos programadores individuais. Os pares gastam 15% mais tempo para produzir o código e o número médio de defeitos é 15% menor.

Já Nawrocki. e Wojciechowski [4] desenvolveram na *Poznan University Technology* na Polônia uma experiência com 21 alunos. Os autores detectaram que a programação em pares é mais eficiente que a programação individual.

Um estudo utilizando a programação em pares com 600 alunos das disciplinas introdutórias de um curso de computação foi desenvolvido, na *California University*, por McDowell et. al. [5]. Os autores detectaram que a programação em pares melhora a qualidade das funcionalidades e contribui diretamente no aprendizado dos estudantes.

Rostaher e Hericko [6] desenvolveram na Eslovênia uma experiência com 16 programadores. Os autores concluíram que não existe diferença de produtividade e qualidade entre a programação em par e a programação individual.

Na *Helsinki University*, Vanhanen Lasseniusem [7] em 2005, desenvolveu de um experimento com 10 estudantes dos cursos de ciência de computação. De posse dos dados gerados com os experimentos os autores concluíram que: o tempo de produção em par é 29 menor se comparado a programação individual; os pares geram 8% menos de defeitos, e; que a transferência de conhecimento na programação em pares ocorre de forma efetiva.

Hulkko e Abrahamsson [8] desenvolveram no VTT *Technical Research Centre* da *Finland* dois estudos de casos com a programação em pares, o primeiro somente com estudantes de mestados e o segundo com estudantes de mestrados e cientistas. Os estudos comprovaram que não existem diferenças entre o tempo de produtividade dos programadores nos arranjos em pares ou individual. A programação em pares é indicada para a aprendizagem de tarefas mais complexas e possui uma melhor legibilidade quando comparada a programação individual.

Na *Wayne University* em *Detroit* USA, Xu e Rajlich [9] realizam um experimento com um intuito de comprar a programação em pares com a programação individual. Participaram 12 alunos e os autores concluíram que a produtividade temporal em pares é muito elevada se comparada a individual, os códigos são melhores concebidos e possuem uma melhor qualidade.

Arisholm et. al. [10] desenvolveram um pesquisa com 295 programadores na Noruega, na Suécia e no Reino Unido. A pesquisa teve como meta comparar se a programação em pares traz benefícios quando comparada a programação individual. Os autores fracionaram a pesquisa em 2 fases, a primeira ocorreu em 2001 com 99 programadores trabalhando individualmente, a segunda envolveu 98 pares (196 programadores). Os autores concluíram que não houve diferença no tempo de produção e na qualidade do código. Os pares tinham um esforço maior para adicionar novas funcionalidades ao código.

Por fim, existem vários estudos na literatura que colaboram com as questões de produtividade e qualidade do código gerado com a programação em pares quando comparada com a individual. Neste contexto, alguns estudos são apresentados nos trabalhos de. Swamidurai e Umphress [11]; Padberg e Muller [12]. O primeiro analisa os impactos estáticos e dinâmicos na programação em pares. Já o segundo analisa os custos e os benefícios deste tipo de programação.

III. MÉTODOS E PROCEDIMENTOS

Para apontar o momento que a programação em pares deve ser adotada em um ambiente empresarial os autores deste trabalho vêm desenvolvendo, desde 2007, uma série de experimentos nos ambientes acadêmicos (universidades) e empresariais (setor produtivo de software). A Tabela I aponta algumas informações dos experimentos realizados.

TABELA I - INFORMASÇÕES DOS EXPERIMENTOS REALIZADOS

Ambiente/Ano	Qtd. Alunos/ Programadores	Nível de conhecimento programadores	Complexidade dos programas
Acadêmico/2007	21	4	4
Acadêmico/2008	15	4	4
Acadêmico/2009	33	3	3
Acadêmico/2014	25	2	2
Empresa A/2013	18 de 54	2	2
Empresa A/2013	18 de 54	4	4
Empresa A/2014	18 de 54	5	5
O nível de conhecimento dos programadores e a complexidade dos programas utilizados no experimento são mapeados pela escala Likert: 1 muito baixo, 2 baixo, 3 médio, 4 alto, 5 muito alto			

Ao analisar a Tabela I é possível perceber a presença de 4 experimentos acadêmicos realizados em 4 ambientes diferentes:

- Fundação Educacional do Município de Assis-SP. Curso de Ciência da Computação – Disciplina de Engenharia de Software II ministrada no décimo semestre – ano de 2007.
- Fundação Educacional do Município de Assis-SP. Curso de Ciência da Computação – Disciplina de Engenharia de Software II ministrada no décimo semestre – ano de 2008.
- Faculdade de Tecnologia de Ourinhos – Curso de Análise de Sistemas – Disciplina de Engenharia de Software II ministrada no sexto semestre.
- Universidade Tecnológica Federal do Paraná (Campus Cornélio Procópio) – Curso de Engenharia de Software – Disciplina de Introdução a Engenharia de Software ministrada no primeiro semestre do curso.

Além dos 4 ambientes acadêmicos é possível perceber a realização de 3 experimentos na Empresa A[1] em 3 momentos distintos – outubro de 2013, novembro de 2013 e janeiro de 2014. A empresa possui uma equipe de 54 pessoas. Estas foram classificadas:

- Em programadores iniciantes – programadores recém-contratados e programadores com menos 1 ano de casa.
- Em programadores - nível de conhecimento médio – programadores com 1 a 3 anos de caso
- Em programadores plenos – nível de conhecimento muito auto – programadores com mais de 3 anos de casa.

É importante salientar que o julgamento do nível dos programadores foi efetuado pela própria empresa.

Já nos experimentos realizados nos ambientes acadêmicos, o nível de conhecimento dos programadores foi delineado de acordo com semestre letivo cursado. Alunos pertencentes aos semestres de final de curso foram classificados com nível de conhecimento alto, caso dos experimentos realizados em 2007 e 2008. Alunos pertencentes aos semestres iniciais foram classificados com nível de conhecimento baixo, caso do experimento realizado em 2014.

A complexidade dos programas utilizados no experimento foi delineada pelos autores deste trabalho. Ao analisar a Tabela I é possível perceber a presença de programas com complexidade baixa, média, alta e muito alta. Programas com complexidade baixa utilizam em sua estrutura comandos de decisão e repetição. Programas de complexidade muito alta utilizam estrutura de dados como fila, pilha, árvore e chama de forma remota um procedimento a um dispositivo de hardware específico.

É importante salientar que durante a realização dos experimentos os grupos (alunos ou programadores) eram divididos, aleatoriamente, em duas categorias: 1 – os programadores individuais; 2 – programadores em pares. Após a divisão, o enunciado de um programa foi apresentado aos alunos ou programadores e estes, por sua vez, desenvolviam uma solução. Durante a execução, algumas medidas foram capturadas pelos próprios alunos ou programadores:

- M1: Número de erros lógicos detectados durante o desenvolvimento do programa.
- M2: Número de testes efetuados.
- M3: Tempo de desenvolvimento (em minutos).

É importante salientar que a captura das medidas foi monitorada presencialmente pelos autores deste trabalho.

Terminada a solução, os autores deste trabalho submetiam os programas gerados a um conjunto de testes (testes utilizados: análise do valor limite, funcional sistemático). O percentual de casos de testes que foram processados corretamente pelos programas foi colecionado na variável M4 (%).

IV. RESULTADOS OBTIDOS COM A EXECUÇÃO DOS EXPERIMENTOS

Conforme apresentado na Tabela I, a execução dos experimentos ocorreram em 2007, 2008, 2009, 2013 e 2014. Durante a execução os programadores ou alunos desenvolveram um único programa, em todos estes experimentos parte dos alunos trabalhavam emparelhados e parte trabalhavam individualmente. É possível verificar os resultados obtidos com a execução dos experimentos na Tabela II.

Ao efetuar uma leitura cuidadosa na Tabela II é possível perceber que:

1. Os programadores em pares detectam mais erros lógicos durante o desenvolvimento dos programas.
2. Quanto maior a complexidade dos programas e maior o nível de conhecimento dos programadores, maior é número de erros detectados durante o desenvolvimento do programa.

[1] Os autores deste trabalho não possuem autorização formal para divulgar o nome das empresas. A empresa está localizada no Estado do Paraná, mais precisamente na região metropolitana de Londrina.

TABELA II - RESULTADOS OBTIDOS COM A EXECUÇÃO DOS EXPERIMENTOS

Experimento	Medidas	Pares	Individual
Acadêmico/2007 Pares: 6 Individual: 9 Total: 21	M1 (med)	2,3	1,3
	M2 (med)	20	22
	M3 (min)	21	45
	M4 (%)	100	55
Acadêmico/2008: Pares: 5 Individual: 5 Total: 15	M1 (med)	12	8,8
	M2 (med)	25	21
	M3 (min)	40	70
	M4 (%)	60	20
Acadêmico/2009 Pares: 15 Individual: 13 Total: 33	M1 (med)	9	8
	M2 (med)	15	16
	M3 (min)	36	44
	M4 (%)	54	45
Acadêmico/2014 Pares: 8 Individual: 9 Total: 25	M1 (med)	10	9
	M2 (med)	10	11
	M3 (min)	33	33
	M4 (%)	56	53
Empresa A/2013 Pares: 6 Individual: 6 Total: 18	M1 (med)	12	12
	M2 (med)	9	10
	M3 (min)	30	31
	M4 (%)	100	89
Empresa A/2013 Pares: 6 Individual: 6 Total: 18	M1 (med)	20	13
	M2 (med)	25	19
	M3 (min)	50	85
	M4 (%)	100	82
Empresa A /2014 Pares: 6 Individual: 6 Total: 18	M1 (med)	32	18
	M2 (med)	30	22
	M3 (min)	145	302
	M4 (%)	100	83

3. O número de erros lógicos detectados no experimento realizado com programadores com nível de conhecimento (ou experiência) com grau baixo é idêntico para o arranjo produtivo emparelhado e para o arranjo individual.

As constatações apontadas nos itens 1, 2 e 3 podem ser verificadas, graficamente, na Figura 1.

4. Programadores emparelhados testam mais se comparados aos programadores individuais nos programas classificados com complexidades alta e muito alta (vide Figura 2).

5. Programadores emparelhados tendem a ser mais rápidos quando a complexidade dos programas é maior (níveis 4 e 5 na escala Likert) – vide Figura 3.

6. Programas desenvolvidos por programadores emparelhados possuem um maior percentual de aprovação nos casos de testes quando comparados aos programas desenvolvidos por programadores individuais (vide Figura 4).

Apresentados os resultados obtidos com a execução dos experimentos, este trabalho irá apresentar na próxima seção as conclusões e trabalhos futuros.

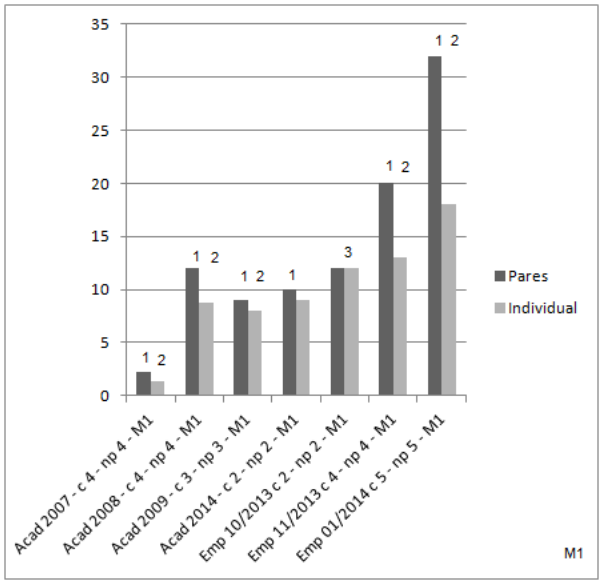

Figura 1. Resultados dos experimentos analisando a medida (M1) – média de erros lógicos detactados. Os números 1, 2 e 3 acimas das barras se relacinam diretamente com os itens 1, 2 e 3 apresentados nesta seção

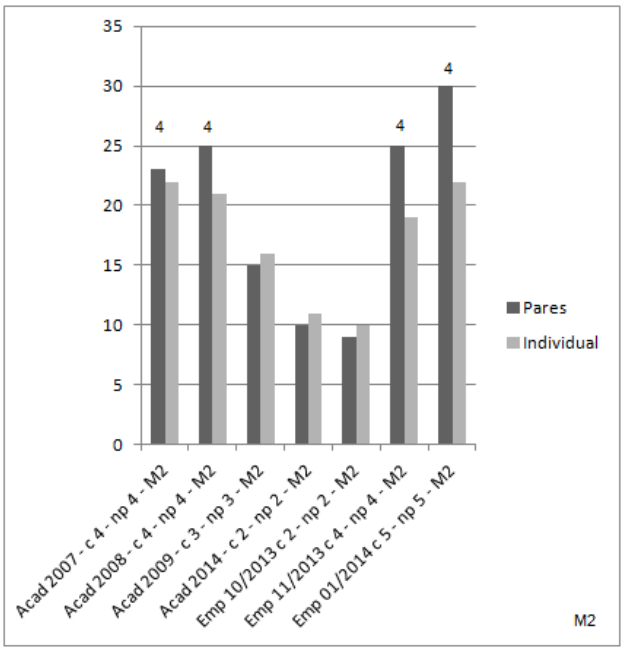

Figura 2. Resultados dos experimentos analisando a medida (M2) – média de testes efetuados. O número 4 acima das barras se relacina diretamente com os item 4 apresentados nesta seção

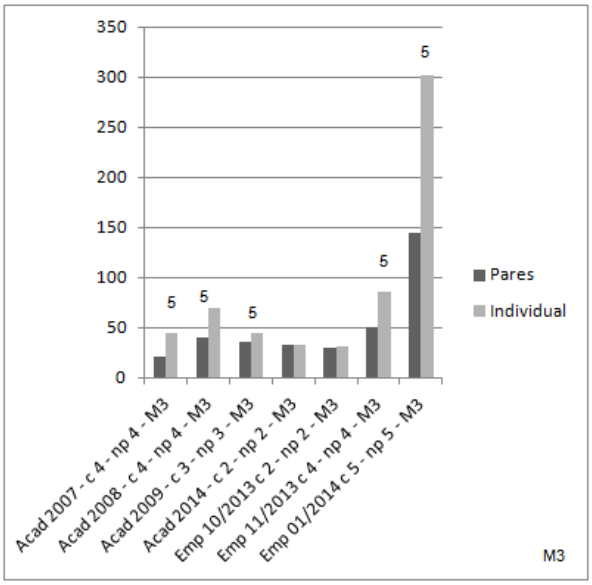

Figura 3. Resultados dos experimentos analisando a média do tempo de desenvolvimento (M3) – média de testes efetuados. O número 5 acima das barras se relacina diretamente com os item 5 apresentados nesta seção

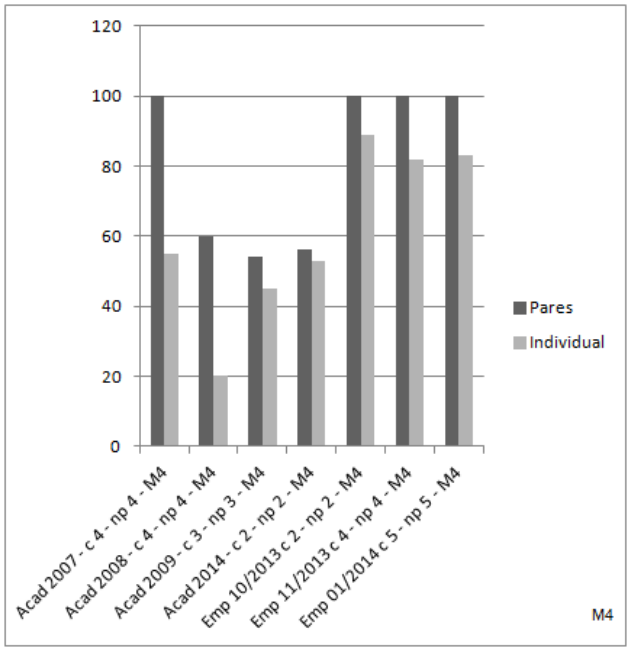

Figura 4. Resultados dos experimentos analisando a percentual de programas aprovados nos casos de teste (M4)

V. CONCLUSÕES E TRABALHOS FUTUROS

A programação em par é uma técnica na qual dois desenvolvedores trabalham em um mesmo problema, ao mesmo tempo e em uma mesma estação de trabalho. Enquanto uma pessoa assume o teclado (o PILOTO) e digita os comandos que farão parte do programa, a outra (o NAVEGADOR) realiza o trabalho de estrategista.

Existem algumas questões que pairam sobre a prática aqui apresentada: Por que colocar dois programadores para fazer o trabalho de um? Não estamos desperdiçando um recurso? **Quando a programação em pares deve ser adotada?**

Para resonder estas questões os autores deste trabalho vêm desenvolvendo, desde 2007, uma série de experimentos nos ambientes acadêmicos (universidades) e empresariais (setor produtivo de software). A Tabela I aponta algumas informações dos experimentos realizados.

Ao analisar os resultados encontrados com a execução dos experimentos (vide seção IV) detectamos:

- Os programadores em pares detectam erros, fato estes que os levam a testar mais.
- O arranjo produtivo dos pares provê uma melhoria sensível no tempo de produção para os problemas mais complexos.
- O arranjo produtivo emparelhado, quando aplicado na resolução de problemas mais complexos tendem produzir melhor qualidade do código desenvolvido - o percentual de programas aprovados em casos de teste e maior neste tipo de arranjo.

Com base nos dados gerados com a execução dos experimentos é possível concluir que:

- As experências realizadas no ambiente acadêmico não provêem deferenças significativas se comparadas as realizadas no ambiente empresarial. Neste caso os relatos gerados por [1], [2], [3], [4], [5], [6], [7] e [9] podem subsidiar a escolha pelo arranjo produtivo em pares nas empresas.
- Não é necessário configurar uma produção de código emparelhada para solução de problemas de complexidade muito baixa, baixa e média.
- A produção em pares deve ser adotada para solucionar problemas de complexidade alta e muito alta. Neste caso não estaremos disperdiçando recursos.
- Para detectar os problemas de complexidade alta e muito alta a empresa de produção de software deve possuir um processo bem definido e institucionalizado, capaz de promover a comunicação entre os membros da equipe. Tal comunicação irá garantir que os problemas complexos sejam detectados com certa antecedência, possibilitando a configuração da produção em pares.

REFERÊNCIAS BIBLIOGRÁFICA

[1] Wilson, J., Hoskin, N., Nosek, J. “The benefits of collaboration for student programmers”. In: Proceedings 24th SIGCSE Technical Symposium on Computer Science Education, pp. 160–164. 1993.

[2] John T. Nosek, “The Case for Collaborative Programming”, Communications of the ACM March 1998/Vol. 41, No. 3.

[3] Laurie Williams, Robert R. Kessler, Ward Cunningham, Ron Jeffries, “Strengthening the Case for Pair Programming”, July/August 2000 IEEE SOFTWARE.

[4] Nawrocki, J. and Wojciechowski, A. “Experimental Evaluation of pair programming”. In: Proceedings of the European Software Control and Metrics Conference (ESCOM 2001). ESCOM Press, 2001, pp. 269-276.

[5] McDowell, C., Werner, L., Bullock, H., Fernald, J. “The effects of pair-programming on performance in an introductory programming course”. In: Proceedings of the 33rd SIGCSE Technical Symposium on. 2002

[6] Matevz Rostaher and Marjan Hericko, “Tracking Test First Programming – An Experiment, XP/Agile” Universe 2002, LNCS 2418, pp. 174-184, 2002.

[7] Jari Vanhanen and Casper Lassenius. “Effects of Pair Programming at the Development Team Level: An Experiment” in Empirical Software Engineering, 2005.

[8] Hanna Hulkko and Pekka Abrahamsson, “A Multiple Case Study on the Impact of Pair Programming on Product Quality”, ICSE’05, May 15-21, 2005, St. Louis, Missouri, USA.

[9] Shaochun Xu, Vaclav Rajlich. “Empirical Validation of Test-Driven Pair Programming in Game Development”, Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on ComponentBased Software Engineering, Software Architecture and Reuse (ICISCOMSAR’06). 2006.

[10] R. Swamidurai; D. Umphress. “The Impact of Static and Dynamic Pairs on Pair Programming”. In: Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on. 2014.

[11] F. Padberg; M.M Muller. “Analyzing the cost and benefit of pair programming”. In: Software Metrics Symposium, 2003. Proceedings. Ninth International – 2003.

[12] L. L. Constantine. “Constantine on Peopleware”. Englewood Cliffs, NJ: Yourdon Press, 1995.