UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

VAGNER CARLOS MARCOLINO LIMA

PROGRAMAÇÃO EM PAR: INVESTIGANDO SUA EFICÁCIA PERANTE TAREFAS DE MODELAGEM E CONSTRUÇÃO DE SOFTWARE

DISSERTAÇÃO

CURITIBA
2013

VAGNER CARLOS MARCOLINO LIMA

PROGRAMAÇÃO EM PAR: INVESTIGANDO SUA EFICÁCIA PERANTE TAREFAS DE MODELAGEM E CONSTRUÇÃO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de “Mestre em Computação Aplicada”. Área: Ciência da Computação. Linha de Pesquisa: Engenharia de Software.

Orientador: Dr. Adolfo G. S. Seca Neto
Co-orientadora: Dra. Maria Claudia F. P. Emer

CURITIBA

2013

Dados Internacionais de Catalogação na Publicação

L732 Lima, Vagner Carlos Marcolino

Programação em par : investigando sua eficácia perante tarefas de modelagem e construção de software / Vagner Carlos Marcolino Lima. — 2013.

122 f. : il. ; 30 cm

Orientador: Adolfo Gustavo Serra Seca Neto.

Coorientadora: Maria Claudia Figueiredo Pereira Emer.

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada, Curitiba, 2013.

Bibliografia: f. 65-71.

1. Software – Desenvolvimento – Controle de qualidade. 2. Programação eXtreme. 3. Pesquisa experimental. 4. Desenvolvimento ágil de software. 5. Modelagem (Computação). 6. Programação orientada a objetos (Computação). 7. Engenharia de software. 8. Computação – Dissertações. I. Seca Neto, Adolfo Gustavo Serra, orient. II. Emer, Maria Claudia Figueiredo Pereira, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. IV. Título.

CDD (22. ed.) 004

Biblioteca Central da UTFPR, Campus Curitiba

Título da Dissertação

"PROGRAMAÇÃO EM PAR: INVESTIGANDO SUA EFICÁCIA PERANTE TAREFAS DE MODELAGEM E CONSTRUÇÃO DE SOFTWARE"

por

Vagner Carlos Marcolino Lima

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM COMPUTAÇÃO APLICADA - Área de Concentração: Ciência da Computação pelo PPGCA - Programa de Pós-Graduação em Computação Aplicada - Mestrado Profissional – da Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba às 09:00 horas do dia 27 de agosto de 2013. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Prof. **Adolfo Gustavo Serra Seca Neto, Dr.**
presidente - (UTFPR - CT)

Profa. **Sheila dos Santos Reinehr, Dra.**
(PUC-PR)

Profa. **Maria Cláudia Figueiredo Pereira Emer, Dra.**
(UTFPR - CT)

Prof **Robinson Vida Noronha, Dr.**
(UTFPR - CT)

AGRADECIMENTOS

A toda a minha família pelo incentivo e compreensão.

Ao meu orientador professor Dr. Adolfo G. S. Seca Neto e a minha co-orientadora professora Dra. Maria Claudia F. P. Emer pela oportunidade, orientação e paciência.

Aos professores Dr. João A. Fabro, Dr. Robinson V. Noronha, Dr. Laudelino C. Bastos pelos direcionamentos durante todo o mestrado e a professora Dra. Sheila dos S. Reinehr por suas considerações e contribuições quanto à dissertação e à apresentação de defesa do mestrado.

Ao professor Me. Nelson Suga pela oportunidade e orientação na elaboração da primeira versão do projeto de pesquisa.

Ao professor Dr. Paulo Ricardo B. Guimarães (e alunos) do Laboratório de Estatística Aplicada da Universidade Federal do Paraná pelos direcionamentos quanto ao uso da Estatística durante o projeto e execução de toda a pesquisa.

As minhas gestoras na Celepar Cristina Angela F. Machado e Eleni L. Hiratsuka pelo incentivo e apoio.

Aos colegas de trabalho e amigos Marcos A. Chiarello, Danielle T. de C. Mayer, Daniel A. Nunes, Felipe H. Moreno, Anderson Graciano, Clara A. Milanez, Camila R. Furlan, Renata C. Furlan, Leandro de Araujo, João Carlos G. Arias, Fábio Suga pela grande e importantíssima colaboração.

Aos professores e alunos voluntários que viabilizaram a realização deste trabalho.

RESUMO

LIMA, Vagner Carlos Marcolino. Programação em Par: investigando sua eficácia perante tarefas de modelagem e construção de software. 122 f. Dissertação – Programa de Pós-Graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Dentre as práticas da Programação Extrema, ou eXtreme Programming (XP), destaca-se a Programação em Par, ou Pair Programming (PP). Nesta prática duas pessoas trabalham de forma colaborativa na mesma tarefa – projeto, algoritmo, código ou teste – e em um único computador. O objetivo geral deste trabalho é investigar a eficácia da Programação em Par versus Programação individual perante tarefas de modelagem e construção de software orientado a objetos. A eficácia da prática é avaliada por meio de atributos relacionados à qualidade de software, são eles: (i) tamanho dos métodos, (ii) complexidade estrutural dos métodos, (iii) acoplamento/dependência entre pacotes e, por fim, (iv) falta de coesão dos métodos por classe. Para isso, foi realizada uma pesquisa experimental envolvendo atividades práticas e aplicação de questionários junto a alunos voluntários de três instituições de ensino superior da cidade de Curitiba. A partir dessa pesquisa, conclui-se que Programação em Par mostrou-se mais eficaz perante tarefas de modelagem e construção de software que a programação individual, isso considerando tamanho e complexidade dos métodos. E mais, os alunos perceberam mais benefícios do que desafios (ou desvantagens) quando se adota a PP para realizar tais tarefas.

Palavras chaves: Programação em Par, Pesquisa experimental, Qualidade de software, Métodos ágeis, Práticas ágeis, Engenharia de Software.

ABSTRACT

LIMA, Vagner Carlos Marcolino. Pair programming: investigating its efficacy toward modeling tasks and software development. 122 f. Dissertação – Programa de Pós-Graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Among eXtreme Programming (XP) practices, Pair Programming(PP) stands out from the rest. It consists of two individuals cooperating, working in the same task - design, algorithm, code or test - in the same computer. The general objective of this study is to scrutinize the efficacy of Pair Programming versus individual programming relating to modeling tasks and object oriented software development. The efficacy of the process is evaluated through the following software quality related attributes: (i) method size, (ii) structural complexity of the methods, (iii) linkage/dependency among packages and finally (iv) lack of cohesion of the methods by class. Therefore, an experimental research was performed, involving practical activities and surveys answered by volunteer students from three different higher education institutions in Curitiba. This research showed that pair programming is more efficient when working with modeling tasks and software development than individual programming, taking into account size and complexity of methods. Furthermore, more pros than cons were found by students when pair programming was chosen to accomplish such tasks.

Keywords: Pair programming, Experimental research, Software quality, Agile Methods, Agile practices, Software engineering.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

BSI	Bacharelado em Sistemas de Informação
CSV	*Comma-separared values*
EC	Engenharia da Computação
IDE	*Integrated Development Environment*
UML	*Unified Modeling Language*
XML	*eXtensible Markup Language*
XP	*eXtreme Programming*

SUMÁRIO

1. INTRODUÇÃO

1.1 APRESENTAÇÃO

Para Kon (2009, texto de Fabio Kon extraído do capítulo Apresentação do livro "O Mítico Homem-Mês" de Frederick P. Brooks Jr.):

> Desenvolvimento de Software é uma arte e demanda muita criatividade. Mas essa criatividade só traz bons frutos se for embasada em muita técnica, conhecimento científico, raciocínio lógico-matemático e boas práticas de engenharia. Além disso, quem escreve software são seres humanos, quase sempre trabalhando em grupos heterogêneos em torno de um projeto comum; com isso, as interações sociais entre as pessoas envolvidas no projeto e as suas características individuais têm um papel fundamental no sucesso da empreitada.

Os processos de desenvolvimento de software podem ser divididos em dois grandes grupos: os tradicionais e os ágeis. Ressalta-se que os grupos utilizam abordagens distintas, mas possuem a mesma finalidade – desenvolver software de alta qualidade que atendam as expectativas do cliente (BASSI FILHO, 2008; DYBÅ e DINGSØYR, 2009).

O *Scrum*[1] e um híbrido de *Scrum* e Programação Extrema, ou *eXtreme Programming* (XP), são os processos ágeis existentes mais adotados pela indústria de *software* (VERSIONONE, 2012), inclusive nacional (MELO et al., 2012). Para este trabalho destaca-se a Programação Extrema. De acordo com Beck e Andres (2004, p. 3), ela é um processo ágil que endereça as dificuldades encontradas no desenvolvimento de software, e seus três elementos essenciais são valores, princípios e práticas.

Dentre as práticas XP, há a Programação em Par, ou *Pair Programming*. Ela é uma das práticas primárias (chave) desse processo (BECK e ANDRES, 2004, p.42). Nela dois desenvolvedores trabalham lado a lado em um computador no mesmo projeto, algoritmo, código ou teste (WILLIAMS et al., 2000, p.1). A prática em questão

[1] De acordo com seus idealizadores, o Scrum é um *framework* para desenvolver e manter produtos complexos, que podem ser software. Entretanto, não é foco deste trabalho discutir tal diferença. Maiores detalhes consultar, por exemplo, http://www.scrum.org/Scrum-Guides.

foi adicionada à primeira edição do processo XP e por ele foi introduzida no mercado. Mas, na realidade, ela apareceu no meio científico em 1995 com o relato realizado por Constantine, onde se observou o uso de “pareamento” de programadores no desenvolvimento de software, e com a publicação do livro de Coplien sobre processo de produção de software, no qual foi sugerido um padrão organizacional de desenvolvimento “pareado” (FARIA e YAMANAKA, 2010).

Alguns anos depois ela ganhou outra vez destaque na publicação dos estudos de Nosek (1998) e Williams e Kessler (2000b), pois nesses dois estudos os autores relataram que PP conduzia a um produto com menos erros e a um aumento do desempenho e da satisfação dos programadores quando comparada à maneira tradicional de se programar – um desenvolvedor por computador (NOSEK, 1998; WILLIAMS e KESSLER, 2000b). Desde então, inúmeros estudos têm buscado investigar os benefícios do uso da Programação em Par e os desafios para realizá-la de maneira eficaz, tanto na indústria de software, quanto na academia (FARIA e YAMANAKA, 2010; SALLEH et al., 2011).

Além dos benefícios citados anteriormente da Programação em Par versus a Programação individual – ou no contexto deste trabalho, oportunamente identificadas respectivamente apenas pelas siglas PP e PT –, podem-se mencionar outros, tais como: 1) os pares ficam mais confiantes do que os desenvolvedores que programam sozinhos quando realizam tarefas de manutenção de software (BALIJEPALLY et al., 2009); 2) PP melhora a qualidade do projeto e do código, o trabalho em equipe, a comunicação e a disseminação do conhecimento (SFETSOS e STAMELOS, 2010).

Estudos têm mostrado que alunos dos cursos da área de Computação estão se beneficiando da Programação em Par como uma ferramenta pedagógica. Por exemplo, ela: 1) traz divertimento às aulas e motivação para o aprendizado (SALLEH et al., 2009); 2) melhora a aquisição de habilidades individuais de programação dos alunos (BRAUGHT et al., 2011); 3) auxilia no aprimoramento das habilidades colaborativas dos alunos (PRESTON, 2006; WILLIAMS e KESSLER, 2000b); 4) mantém os alunos focados na atividade em desenvolvimento devido à “pressão positiva” da dupla (HO et al., 2004; WILLIAMS e KESSLER, 2000a). Se comparada à Programação em Par Distribuída (PPD), segundo Hayward (2009), alguns dos benefícios da PP podem ser

percebidos com o uso da PPD, mas mesmo assim os alunos preferem a Programação em Par em vez da PPD.

Porém, os desafios inerentes ao uso da Programação em Par, seja no ambiente acadêmico ou empresarial, são, por exemplo, necessidade de: (1) gerenciar os níveis de experiência de cada integrante e da dupla; (2) infraestrutura especifica, tal como, estações de trabalho com espaço suficiente para que duas pessoas possam trabalhar uma ao lado da outra de maneira confortável; (3) gerenciar as diferentes características inerentes ao relacionamento humano, que vai desde higiene e saúde pessoal até formação de pares de desenvolvedores de gêneros distintos (BECK e ANDRES, 2004; SHORE e WARDEN, 2008; TELES, 2004).

Mesmo com esses inúmeros estudos, de acordo com Salleh et al. (2011) e Balijepally et al. (2009), há a escassez de pesquisas sobre o uso da PP envolvendo atividades de desenvolvimento de software – e não manutenção –, preferencialmente tarefas de modelagem/projeto de software em conjunto com tarefas de programação.

Há dois estudos[2] nacionais que abordam o tema “Programação em Par”, são eles: Faria (2010) e Sawicki (2002). O primeiro estudo é uma tese de doutorado onde o autor usou a PP como meio para aplicação de um novo método de ensino de programação. O segundo estudo é uma dissertação de mestrado na qual o autor usou a prática em questão como base para elaboração de um processo colaborativo de desenvolvimento de circuitos integrados.

A academia deve aumentar o número de estudos envolvendo métodos e práticas ágeis, pois o interesse do mercado de software em processos ágeis para o desenvolvimento de sistemas vem crescendo ano após ano (DYBÅ e DINGSØYR, 2008; DYBÅ e DINGSØYR, 2009; VERSIONONE, 2012). Na realidade, o cenário que antes era de certo ceticismo por parte dos pesquisadores na academia e desenvolvedores e gerentes na indústria, hoje em dia mudou completamente (CORBUCCI et al., 2011). Por exemplo, desde 2001 alguns desenvolvedores têm tido a oportunidade de ter contato prático com métodos e práticas ágeis durante a graduação

[2] Pesquisa realizada na Biblioteca Digital Brasileira de Teses e Dissertações em 13 de Novembro de 2011 (“Busca Avançada”, campo “Resumo” e opção “Todas as palavras”). Buscaram-se resultados para as seguintes “strings” de pesquisa: “pair programming”, “programação em par”, “programação em dupla” e, por fim, “programação pareada”. Disponível em http://bdtd.ibict.br/pt/inicio.html.

em Ciência da Computação em uma instituição pública de ensino superior em São Paulo (SILVA, 2007). Assim como há empresas envolvidas com desenvolvimento de software que reivindicam que seus colaboradores tomem como modelo algumas das recomendações do Manifesto Ágil[3] (CORBUCCI et al., 2011).

1.2 OBJETIVOS

Perante o exposto até aqui, os objetivos deste trabalho estão divididos em dois tipos: Objetivo Geral e Objetivos Específicos.

1.2.1 Objetivo geral

O objetivo desta pesquisa é *investigar a eficácia da Programação em Par versus Programação individual perante tarefas de modelagem e construção de software orientado a objetos*. A eficácia é avaliada por meio da qualidade do software implementado pelos desenvolvedores que utilizaram PP versus os que não utilizaram. Sendo assim, a hipótese e a questão examinadas neste trabalho são:

- H: A qualidade do software implementado pelos desenvolvedores que utilizaram a PP é melhor que a dos desenvolvedores que realizaram as tarefas individualmente.
- Q: Qual a percepção dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par?

[3] Documento eletrônico com os valores e princípios que norteiam praticamente todos os processos[/métodos/metodologias/técnicas/práticas] ágeis. Maiores detalhes em http://agilemanifesto.org/.

1.2.2 Objetivos específicos

Os objetivos específicos deste estudo são:

- Verificar se a qualidade do software desenvolvido pelos desenvolvedores que utilizaram a PP é melhor que a dos desenvolvedores que realizaram as tarefas individualmente.
- Examinar as percepções dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par.

Para examinar os objetivos apresentados anteriormente, foi conduzida uma pesquisa experimental envolvendo atividades práticas e aplicação de questionários junto a quatro turmas voluntárias de cursos da área de Computação de três instituições de ensino superior da cidade de Curitiba.

1.3 ESTRUTURA DO TRABALHO

Este texto está estruturado da seguinte forma: o Capitulo 2 apresenta a revisão bibliográfica. O Capítulo 3 descreve em detalhes a pesquisa experimental adotada neste trabalho. O Capítulo 4 apresenta os resultados obtidos por meio deste estudo e a discussão sobre eles e, por fim, no Capítulo 5 há as considerações finais sobre o estudo como um todo.

2. REVISÃO BIBLIOGRÁFICA

2.1 VISÃO GERAL

Nos processos tradicionais de desenvolvimento despende-se bastante tempo na modelagem gerando diversos modelos UML, *Unified Modeling Language*, antes de iniciar a construção das funcionalidades do software – especificadas detalhadamente por meio de casos de uso, por exemplo. Na XP busca-se modelar em equipe apenas o necessário para que se possa começar a construção das funcionalidades do software – "especificadas" a partir de unidades "visíveis ao cliente", ou Histórias (BASSI FILHO, 2008; BECK E ANDRES, 2004).

Em outras palavras, na XP, a modelagem serve para construir uma ideia inicial sobre a solução que será desenvolvida para uma dada funcionalidade, com isso não há uma grande "lacuna" entre a modelagem e a codificação da "ideia do mundo real". Sendo assim, na prática os desenvolvedores podem utilizar qualquer tipo de notação em suas discussões, desde que eles entendam e acordem entre si aquilo que precisa ser realizado. Logo, a (notação) UML é utilizada na XP para documentar aquilo que a equipe fez e não o que irá fazer (TELES, 2004). Ou, durante o desenvolvimento XP pode-se usar a Modelagem Ágil[4], na qual a finalidade da modelagem – rascunhos em UML, por exemplo – é principalmente entender, não documentar (LARMAN, 2007).

Enfatizar o desenvolvimento do software, e não a "geração" de modelos/documentos, não significa negligenciar princípios de [um bom] projeto de software, tais como, construir software o mais coeso e menos acoplado possível. No caso de software orientado a objetos, (mais) coesão implica em componente ou classe que encapsula somente os atributos e operações relacionados entre si e com a classe ou componente propriamente dita, e (menos) acoplamento implica em componentes ou classes mais interdependentes. Sendo assim, na XP esses princípios são aplicados

[4] Uma visão geral dessa técnica de modelagem (ágil) proposta por Scott W. Amber está disponível em http://www.agilemodeling.com/shared/AMPanfleto.pdf

durante, por exemplo, a adoção da prática Programação em Par para o desenvolvimento de uma dada funcionalidade de um determinado software (LARMAN, 2007; TELES, 2004).

2.2 PROGRAMAÇÃO EM PAR (PP)

Na segunda edição da XP há 24 práticas, sendo 13 primárias e 11 corolárias. As práticas que fazem parte do primeiro grupo são independentes, e sem o domínio dessas, o uso das práticas do segundo grupo está suscetível a certa dificuldade. Todas elas funcionam melhor juntas, porém pode-se adotar uma de cada vez. A prática Programação em Par é um exemplo de uma prática primária (BECK e ANDRES, 2004).

2.2.1 Definição e Contexto de uso

A Programação em Par é uma prática onde duas pessoas trabalham lado a lado de forma colaborativa no mesmo projeto, algoritmo, código ou teste compartilhando o mesmo computador (BECK e ANDRES, 2004, p.42; WILLIAMS et al., 2000, p.1).

Essa prática pode ser integrada a qualquer processo de software (LIMA, SECA NETO e EMER, 2013, p. 16; WILLIAMS, 2001). Por exemplo, em um “ambiente Lean”, o uso de programação em par é uma forma de evitar o acúmulo de código esperando uma revisão formal, pois revisão é realizada de forma instantânea, enquanto o código está em desenvolvimento (POPPENDIECK e POPPENDIECK, 2011, p. 204).

Porém, como inicialmente evoluiu e se destacou dentro de um processo ágil, a XP, e vem sendo utilizada por outros modelos ágeis, logo, acredita-se que o seu máximo será obtido em ambientes e equipes que adotam tais modelos (BECK e ANDRES, 2004; SHORE e WARDEN, 2008; TELES, 2004).

2.2.2 Dinâmica da prática

Ressalta-se primeiramente que programação em par não é uma sessão de tutoria, onde uma pessoa desenvolve e outra fica apenas observando (BECK, 2000, p.105). Em vez disso, como será descrito a seguir, programação em par é uma prática dinâmica e colaborativa.

Na programação em par há dois papéis que o desenvolvedor irá exercer em momentos distintos: piloto e navegador. O primeiro controla o equipamento, por exemplo, entra com os comandos do código em desenvolvimento, e o segundo revisa e acompanha continuamente e ativamente o trabalho do primeiro, pensando e sugerindo novas ideias ou estratégias (TELES, 2004, p.89-93).

Orientações gerais sobre a dinâmica da programação em par:

- *Quando [ou para que] formar pares:* forme pares para qualquer coisa dentro do projeto que a equipe precise manter e que irá para produção (BECK e ANDRES, 2004, p. 42; SHORE e WARDEN, 2008, p.76);
- *Composição dos pares (ou escolha dos pares):* a) devem-se formar pares de forma natural, buscando trabalhar com todos da equipe quando possível, e deve-se evitar a situação “professor-estudante” (SHORE e WARDEN, 2008), essa situação em particular será explicada nas seções a seguir, ou b) devem-se trocar de pares aleatoriamente com frequência (TELES, 2004);
- *Durante o pareamento:* programação em par é sinônimo de comunicação, conversa (SHORE e WARDEN, 2008, p. 76; TELES, 2004, p.91); ou seja, conforme Shore e Warden (2010, p.76), “dê passos pequenos e [...] fale sobre suas suposições, objetivos de curto-prazo, orientação geral e qualquer cenário relevante sobre o recurso ou sobre o projeto”; nesse caso deve-se entender “recurso” como tarefa em desenvolvimento;
- *Troca de papel (ou revezamento):* troque com frequência (SHORE e WARDEN, 2008, p. 77; TELES, 2004, p.93); por exemplo, para Beck e Andres (2004), a cada 7 minutos ou, conforme Faria (2010, p. 74), 20

minutos. De acordo com Shore e Warden (2010, p.77), “se estiver navegando e se pegar dizendo ao piloto quais teclas ele deve pressionar, peça o teclado. Se está pilotando e precisa de uma pausa, passe o teclado para o seu navegador”.

- *Rodízio do par:* quando terminar uma tarefa, ou se a tarefa for muito grande pode-se trocar a cada quatro horas, ou ainda, quando precisar de uma nova perspectiva do problema/tarefa (SHORE e WARDEN, 2008, p.76);

2.2.3 Infraestrutura necessária

Conforme citado na introdução deste trabalho, as estações de trabalho devem proporcionar o trabalho em dupla. Por exemplo, os equipamentos devem ter monitores de 17 polegadas e a mesa de trabalho deve ter 1,80 m de comprimento por 1,20 de largura. Algo bem diferente das típicas mesas em L presentes na maioria das empresas (SHORE e WARDEN, 2008, p. 78).

A seguir, nas Figuras 1 e 2 são apresentados exemplos de ambientes de trabalho que suportam programação em par. Os itens indicados com as letras:

- “PP”: são locais com mesas que suportam programação em par;
- “R”: identificam espaços com estações de trabalho “não-PP” para eventuais tarefas do projeto que podem ser realizadas individualmente e para atividades particulares por parte dos integrantes do projeto em geral;
- “Q” – quadros brancos – e “Qr” – são quadros brancos com rodinhas: itens com presença certa em ambientes de desenvolvimento ágil de software, principalmente os primeiros;
- “Re”: identificam mesas de reunião;

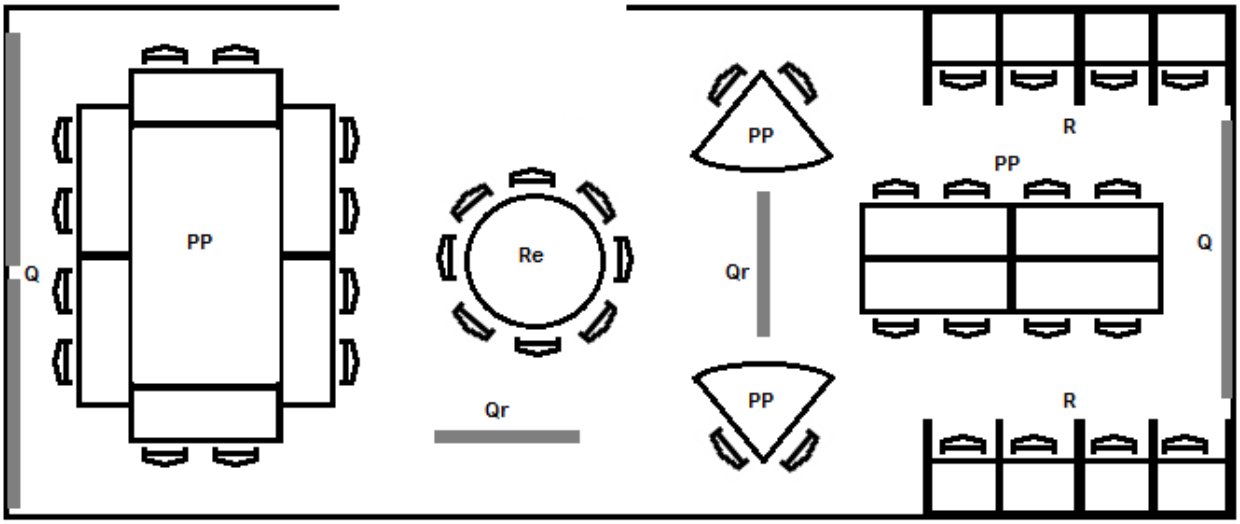


Figura 1 – Exemplo de um ambiente grande que suporta PP
Fonte: Adaptado de SHORE e WARDEN (2008, p. 120).

Podem-se ver na Figura 1 mesas com um formato distinto ao descrito no exemplo dado no inicio desta seção, mas que podem ser utilizadas para a prática da programação em par.

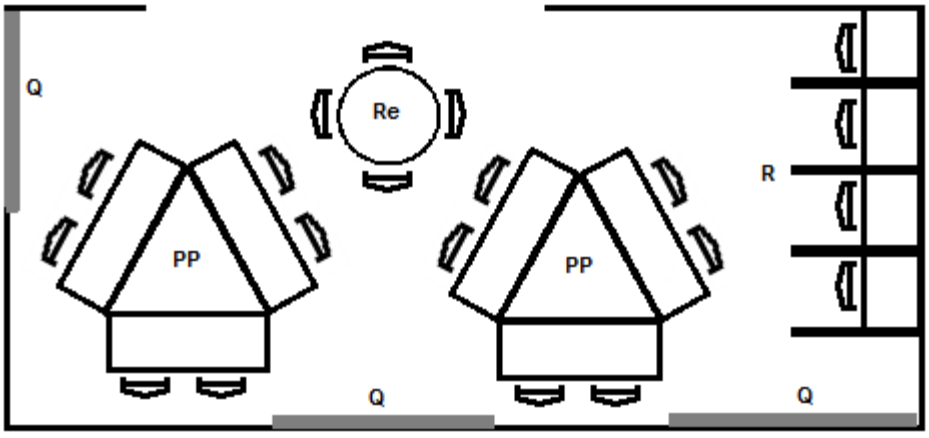


Figura 2 – Exemplo de um ambiente pequeno que suporta PP
Fonte: Adaptado de Beck e Andres (2004, p. 40).

Na Figura 2 há mesas com uma configuração que, se comparado à configuração apresentada na figura anterior, facilita ainda mais a comunicação entre os integrantes de uma equipe de software e proporciona a prática da programação em par.

2.2.4 Benefícios esperados

São benefícios esperados do uso da programação em par no desenvolvimento de software (POPPENDIECK e POPPENDIECK, 20114; SHORE e WARDEN, 2008; TELES, 2004):

1. Desenvolvimento de soluções mais complexas em menos tempo e com melhor qualidade, devido, por exemplo, ao efeito “duas cabeças pensam melhor do que uma”;
2. Código com menos defeitos, devido, por exemplo, à frequente revisão do código – ora piloto, ora navegador;
3. Foco quase que total na atividade em desenvolvimento devido à pressão do par, pois faz com que os desenvolvedores se distraiam bem menos e gastem a maioria do seu tempo com a tarefa, com trabalho produtivo;
4. Maior produtividade, em geral, no médio e longo prazo, em consequência, por exemplo, do menor número de defeitos e soluções eficazes e de mais simples manutenção;
5. Conhecimento geral da equipe fica mais homogêneo, por causa da disseminação do conhecimento causada pela troca de papeis e do rodízio dos pares;
6. Um meio em potencial para o aprendizado, e, por consequência, nivelamento do conhecimento e experiência dentre os integrantes da equipe, devido à constante comunicação entre os parceiros;
7. Maior satisfação no trabalho realizado, devido, por exemplo, à maior colaboração entre os membros da equipe e maior volume de trabalho com qualidade realizado.

2.2.5 Desafios do uso de PP

Embora os benefícios citados anteriormente sirvam de grande incentivo para o uso da programação em par, tanto na indústria de software em geral, quanto em outros contextos, há alguns desafios a serem enfrentados, inclusive alguns que podem inviabilizar a adoção da prática. Alguns desses desafios são descritos a seguir.

Visão gerencial

Em geral os gerentes das empresas, devido à sua formação fortemente baseada no modelo “produção em massa”, associam o projeto a uma fábrica e os desenvolvedores a máquinas, logo, é inconcebível ter duas “máquinas” fazendo o que apenas uma faria (TELES, 2004, p.97). Segundo Shore e Warden (2010, p.79), “...duas pessoas não estão, realmente, fazendo o trabalho de uma. Embora haja apenas um teclado sendo utilizado, a programação envolve muito mais coisas”. E mais, os pares, de acordo com Poppendieck e Poppendieck (2011, p. 204), “...aumentam a produtividade devido à qualidade e à robustez do código serem melhoradas quando vistas por dois conjuntos de olhos”.

Infraestrutura, Ferramentas e Padrões de Codificação

Conforme citado anteriormente, em diversas empresas as mesas são posicionadas nos cantos das baias, normalmente nesse caso a mobília tem um formato em “L”, algo que pode inviabilizar o uso da PP (SHORE e WARDEN, 2008, p.78; TELES, 2004, p.95-96).

Em geral, cada desenvolvedor possui uma ferramenta e um padrão de codificação de sua preferência. Logo, para evitar uma “guerra” de ferramentas e padrões de codificação deve-se definir antes do inicio do desenvolvimento um padrão tanto para ferramentas quanto para codificação (SHORE e WARDEN, 2008, p.79).

Higiene pessoal, saúde e limpeza

Antes de iniciar o trabalho certifique-se que a mesa está limpa e lembre-se que há alimentos – por exemplo, alho – que podem causar o mau-hálito (SHORE e WARDEN, 2008, p.78). Deve-se evitar ir ao trabalho quando estiver doente, deve-se cobrir a boca quando tossir, e também não se deve usar perfume forte (BECK e ANDRES, 2004, p.43).

Qualificação do desenvolvedor

Devem-se evitar pares do tipo “professor-estudante”, um desenvolvedor sênior faz par com um júnior. Antes, deve-se restabelecer o equilíbrio. Para isso pode-se solicitar ao menos experiente que ele pesquise um assunto que é de interesse da equipe e que ninguém domina, então, a partir desse momento ele não será “júnior” (SHORE e WARDEN, 2008, p.78-79).

Relacionamento Humano

Segundo Teles (2004, p.97-98), o relacionamento humano é um dos grandes desafios da programação em par, pois pressupõe: a) conciliar o sentimento de insegurança gerado por ter que mostrar a solução para o parceiro; b) ter humildade para aceitar facilmente as críticas; c) paciência para esperar o parceiro terminar o raciocínio para depois discutir uma solução que é a soma das ideias, isso principalmente quando se conhece outra solução potencialmente melhor; e por fim, d)

não pode haver competição entre os membros da dupla, isso não favorece o trabalho colaborativo.

Podem-se ter episódios de desenvolvimento em par, em que se fala pouco ou muito, e em ambas as situações, devido a diferentes características pessoais, pode-se falar de forma muito incisiva, por exemplo, “Este método está muito grande!”. Independente da situação deve-se buscar uma atitude que favoreça a solução colaborativa, por exemplo, “Poderíamos diminuir este método, o que acha?”. Essa seria a forma mais colaborativa de se realizar o questionamento anteriormente realizado (SHORE e WARDEN, 2008, p.79).

Para alguns, trabalhar lado-a-lado pode ser algo novo e bom. Porém, deve-se ater a determinadas situações. Por exemplo, quando se necessita separar um desenvolvedor imaturo emocionalmente de seu parceiro de gênero oposto porque ele não soube discernir o que é trabalho do que é pessoal, logo se entende que determinadas duplas podem não ter um bom desempenho, com isso afetando a equipe como um todo (BECK e ANDRES, 2004, p.43).

2.3 HISTÓRIAS

A prática Histórias é outro exemplo de prática primária dentro da XP. Com ela o software é desenvolvido a partir de unidades de funcionalidade "visíveis ao cliente". Para isso, as funcionalidades são brevemente descritas pelo cliente idealmente em cartões de papel, que podem ser manuseados facilmente por ele mesmo e pelos desenvolvedores (BASSI FILHO, 2008, p.58; BECK E ANDRES, 2004, p.44). Nelas pode haver definição de regras de negócio e/ou testes de aceitação (SHORE e WARDEN, 2008, p.42; WELLS, 1999).

O cliente (ou usuário) verifica se a funcionalidade (ou software) está fazendo aquilo que ele espera por meio dos testes de aceitação. Eles são criados a partir das histórias de usuários. Uma história pode ter um ou mais testes de aceitação, e eles são testes do tipo “caixa-preta” (WELLS, 1999; BASSI FILHO, 2008, p.58). Esses tipos de testes são conduzidos na interface do software. Um teste caixa-preta examina algum

aspecto fundamental do sistema, pouco se preocupando com a estrutura lógica interna do software (DELAMARO, JINO e MALDONADO, 2007).

2.4 QUALIDADE DE SOFTWARE

A ideia de qualidade, segundo Koscianski e Soares (2007, p. 17), "...é aparentemente intuitiva; contudo, quando examinado mais longamente, o conceito se revela complexo. Definir qualidade para estabelecer objetivos é, assim, uma tarefa menos trivial do que aparenta a princípio".

No contexto deste trabalho, adota-se a definição de qualidade dos produtos de software de Rocha, Maldonado e Weber (2001, p.111), a saber:

> Qualidade de software pode ser vista como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários. É por meio desse conjunto de características que a qualidade de um produto de software pode ser descrita e avaliada. Cada uma das características pode ser detalhada em vários níveis de subcaracterísticas, chegando-se a um amplo conjunto de atributos que descrevem a qualidade de um produto de software. Torna-se, então, necessário escolher um modelo que organize os atributos e permita avaliar a qualidade do software.

De acordo com Koscianski e Soares (2007), podem-se seguir os seguintes passos (diretrizes) para se definir uma forma de avaliar a qualidade de um software: (1) definição do modelo/característica para a avaliação do software em questão, (2) identificação dos atributos do software que serão medidos considerando modelo/característica adotado e tipo de software a ser desenvolvido, (3) definição da[s] métrica[s], (4) definição da forma de medição e, por fim, (5) identificação da forma de análise das medidas.

Mas antes de aplicar tais diretrizes, uma vez que o foco deste trabalho é software orientado a objetos, segundo Rocha, Maldonado e Weber (2001, p.140-141), há critérios a serem considerados quando se avalia a qualidade desses tipos de software, alguns são: (i) minimizar o acoplamento entre as operações que pertencem a objetos diferentes – por consequência, a pacotes diferentes, (ii) maximizar a coesão

entre as operações que pertencem ao mesmo objeto e entre objetos que pertencem a mesma estrutura hierárquica, (iii) aplicar restrições de tamanho aos objetos e, por fim, (iv) deve-se tratar a complexidade, "...uma vez que as consequências da alta complexidade de um sistema de software se refletem na facilidade de desenvolvimento, teste e manutenção, influenciando a qualidade do produto".

2.5 MÉTRICAS DE QUALIDADE DE SOFTWARE

Conforme Meirelles (2008, p. 4), "quando se tem acesso ao código-fonte, é possível realizar medições sobre a sua estrutura e organização interna de forma a avaliar sua qualidade". Em linhas gerais, segundo Koscianski e Soares (2007), métrica é o método de medição. A medição é o ato de determinar uma medida, que por sua vez define o que será medido – quantidade, tamanho ou capacidade de algum atributo de um produto ou processo.

Há diferentes tipos de métricas: a) quanto ao objeto: métricas de produto e de processo; b) quanto aos critérios: métricas objetivas e subjetivas; e por fim, c) quanto ao método de obtenção: primitivas ou compostas. Complexidade e tamanho final do programa são exemplos de métricas de produto, e tipo de metodologia usada é exemplo de métrica de processo. Número de linhas de código é um exemplo de métrica objetiva. Modelo de estimativa de custo é um exemplo de métrica subjetiva. Erros indicados em um teste de unidade e número de linhas de código é um exemplo de métrica primitiva. Por outro lado, número de erros encontrados a cada mil linhas de código é um exemplo de métrica composta (MEIRELLES, 2008, p. 4).

A seguir, a descrição das métricas adotadas neste trabalho – para outras métricas, ver Meirelles (2008).

1) *Linhas de código dos métodos (ou Method Lines of Code – MLOC):* é uma métrica de produto, objetiva e primitiva, ela conta e soma o número de linhas de código dentro do corpo do método, não contando linhas em branco e comentários (ECLIPSE METRICS PLUGIN, 2012). Para Koscianski e

Soares (2007, p. 229), a medida de tamanho de software (linhas de código), considerando produtos maiores versus menores, pode ser utilizada para indicar maior tempo de desenvolvimento – custo maior – e trabalho mais complexo.

2) *Complexidade ciclomática de McCabe dos métodos (ou McCabe Cyclomatic Complexity – VG):* é uma métrica de produto, objetiva e primitiva, ela conta o número de caminhos em um trecho de código – calculado apenas para os métodos. Por exemplo, cada vez que um "ramo" ocorre (if, for, while, do, case, catch, operador ternário[5], bem como "&&" e "||" operadores lógicos em expressões condicionais) o valor dessa medida é incrementado de um (ECLIPSE METRICS PLUGIN, 2012). A premissa básica é que o aninhamento de comandos de decisão e laços está relacionado com a complexidade de execução e esforço necessário para compreender o método, neste caso (MCCABE, 1976). Há uma forte correlação entre número de erros encontrados em software e tamanho e complexidade interna (KOSCIANSKI e SOARES, 2007, p. 233-234). Para Watson e McCabe (1996 apud KOSCIANSKI e SOARES, 2007, p. 236), "a complexidade ciclomática deve ser mantida em um valor máximo de 10, além disso, deve-se particionar o código [...]". Porém, conforme Koscianski e Soares (2007, p. 236), há estudos mais recentes que apontam que 15 é um valor máximo aceitável.
3) *Falta de coesão dos métodos (ou Lack of Cohesion of Methods – LCOM):* é uma métrica de produto, objetiva e primitiva, ela é uma medida para a coesão de uma classe. Calculada com o método de Henderson-Sellers, de 1996 (LCOM*). Se m(A) é o número de métodos que acessam um atributo A, calcula-se a média de m(A) para todos os atributos, subtrai-se o número de métodos m e dividi o resultado por (1-m). Um valor baixo indica um classe coesa e um valor próximo de 1 indica falta de coesão. Por padrão o *plugin* não considera atributos e métodos estáticos no cálculo dessa métrica,

[5] Sintaxe e exemplo disponíveis em http://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/section11.html

mas há uma opção para tal (ECLIPSE METRICS PLUGIN, 2012). De acordo com Koscianski e Soares (2007, p. 241), a falta de coesão indica oportunidade para dividir a classe, com essa divisão trata-se a complexidade, pois classes menores são mais fáceis de entender e de realizar manutenção. Há outro método, o de Chidamber e Kemerer, de 1994, mais detalhes em Cosín e Anz (2000, p. 89).

4) Instabilidade por pacote – ou acoplamento/dependência entre pacotes – (ou apenas *Instability* – RMI): é uma métrica de produto, objetiva e composta, ela é a relação entre acoplamento eferente (Ce) e acoplamento total (Ce+Ca). O acoplamento aferente (ou *afferent coupling* – Ca) é o número de classes de fora de um pacote que dependem de classes de dentro do pacote. O acoplamento eferente (ou *efferent coupling* – Ce) é o número de classes de dentro de um pacote que dependem de classes fora do pacote (ECLIPSE METRICS PLUGIN, 2012). Conforme Madeyski (2006, p. 3-4), essa métrica é um indicador de capacidade do pacote para mudar e varia de 0 a 1, onde 0 indica uma estabilidade máxima do pacote e 1 uma instabilidade máxima do pacote, em outras palavras, o acoplamento de um pacote pode variar de fracamente acoplado – 0 – a fortemente acoplado – 1.

3. MÉTODO DE PESQUISA

3.1 VISÃO GERAL

A investigação proposta neste trabalho é realizada por meio da pesquisa experimental, sendo assim, dados quantitativos são coletados e então métodos estatísticos são aplicados. Entretanto, conforme sugerido em Wohlin et al. (2012, p. 12), dados quantitativos e qualitativos são coletados por meio de questionários visando auxiliar e/ou complementar o processo experimental.

A investigação por experimentação no contexto da Engenharia de Software não é algo recente. A pesquisa experimental caracteriza-se pela manipulação de algumas variáveis e a observação de outras em um ambiente controlado. Em geral, essa forma traz um maior controle sobre a execução, medição, investigação e facilidade de repetição. Porém, o custo e o risco são altos (MAFRA e TRAVASSOS, 2006; TRAVASSOS, GUROV e AMARAL, 2002).

3.2 PESQUISA EXPERIMENTAL

O método de investigação adotado neste estudo foi elaborado com base nos modelos de pesquisa experimental definidos em Wohlin et al. (2012) e Mafra e Travassos (2006), as etapas são:

- Delimitação
- Planejamento
- Avaliação
- Execução
- Análise, Apresentação e Interpretação dos dados

Nesta seção descrevem-se as três primeiras etapas Delimitação, Planejamento e Avaliação, as outras etapas são abordadas no próximo capítulo.

3.2.1 Delimitação

Os objetos de estudo são Programação em Par (PP) versus Programação individual. O propósito é investigar a eficácia da PP versus PT perante tarefas de modelagem e construção de software orientado a objetos e também as percepções dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par.

A eficácia da PP é comparada à da PT por meio da avaliação da qualidade do código implementado pelos desenvolvedores. Os seguintes atributos do software são medidos e avaliados separadamente: 1) tamanho; 2) complexidade estrutural; 3) coesão; 4) acoplamento. No quadro a seguir, apresenta-se em detalhes o modelo de avaliação da qualidade de software adotado neste trabalho. O modelo em questão foi definido conforme diretrizes presentes na seção Qualidade de Software, do capítulo Revisão Bibliográfica.

Passo 1: Característica.
A característica avaliada é a Manutenibilidade. Essa característica está relacionada à facilidade de modificação de um produto de software e é de interesse especialmente dos desenvolvedores (KOSCIANSKI e SOARES, 2007, p.208-212).
Passo 2: Atributos
Com base nas definições presentes na Revisão Bibliográfica, em Salleh et al. (2011, p. 14) e Aniche (2012, p. 18), os atributos do software que serão avaliados são: 1) tamanho; 2) complexidade estrutural; 3) coesão; por fim, 4) acoplamento.
Passo 3: Métricas
Ainda com base em Salleh et al. (2011, p.14), Aniche (2012, p.18) e nas definições anteriores, e mais, em Cosín e Anz (2000, p. 89) e Madeyski (2006, p. 3-4), as métricas para cada atributos são: 1) Tamanho: média do *número de linhas de código dos métodos* do software desenvolvido; 2) Complexidade estrutural: média da *complexidade ciclomática de McCabe dos métodos* do software desenvolvido; 3) Coesão: média da *falta de coesão dos métodos por classe* do software; 4) Acoplamento: média da *instabilidade por pacote – ou acoplamento/dependência entre pacotes –* do software desenvolvido;
Passo 4: Forma de coleta dos dados
Utiliza-se de um recurso complementar à IDE Eclipse, o *plugin* Metrics - ele não está disponível por padrão no ambiente de desenvolvimento em questão. Uma vez importado o projeto para dentro da IDE com o *plugin*, ele calcula as medidas de várias métricas a partir do código-fonte deste determinado projeto – ou pacote, classe, método – Java, se o projeto estiver sem erros de compilação. Além disso, o *plugin* permite a exportação de tais dados por meio de um arquivo XML (ECLIPSE METRICS PLUGIN, 2012). Ver Apêndice M para exemplo (figura) da interface do *plugin*. Abaixo as medidas citadas no passo anterior com suas correspondentes métricas no *plugin*. 1) Média do *número de linhas de código dos métodos* do software desenvolvido: coluna *Mean* da métrica *Method Lines of Code*, ou MLOC; 2) Média da *complexidade ciclomática de McCabe dos métodos* do software desenvolvido: coluna *Mean* da métrica McCabe *Cyclomatic Complexity*, ou VG; 3) Média da *falta de coesão dos métodos por classe* do software: coluna *Mean* da métrica *Lack of Cohesion of Methods*, ou LCOM; 4) Média da *instabilidade por pacote – ou acoplamento/dependência entre pacotes –* do software desenvolvido: coluna *Mean* da métrica *Instability*, ou RMI;
Passo 5: Forma de análise das medidas
Análise quantitativa/estatística dos dados coletados.

Quadro 1 – Modelo de avaliação da qualidade do software
Fonte: Autoria própria.

Por outro lado, a percepção dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par é examinada por meio de aspectos como: 1) pressão do par, 2) aprovação dos desenvolvedores quanto ao uso da PP e Histórias e, por fim, 3) benefícios e desafios (ou desvantagens) no uso da Programação em Par.

A pesquisa foi realizada com alunos voluntários de cursos da área de Computação de instituições de ensino superior da cidade de Curitiba – mais detalhes sobre a quantidade e tamanho das amostras, ver seção 4.1 a seguir. Foi dada preferência a alunos dos últimos períodos – ou que no mínimo passaram pelas disciplinas de Lógica para Computação, Fundamentos de Programação (ou Algoritmos e Estrutura de Dados), Banco de Dados, Análise e Projeto de Sistemas (ou Desenvolvimento de Sistemas), Engenharia de Software, e pelo menos uma disciplina de programação utilizou a linguagem/tecnologia Java.

Abaixo uma figura que demonstra o modelo conceitual da pesquisa experimental.

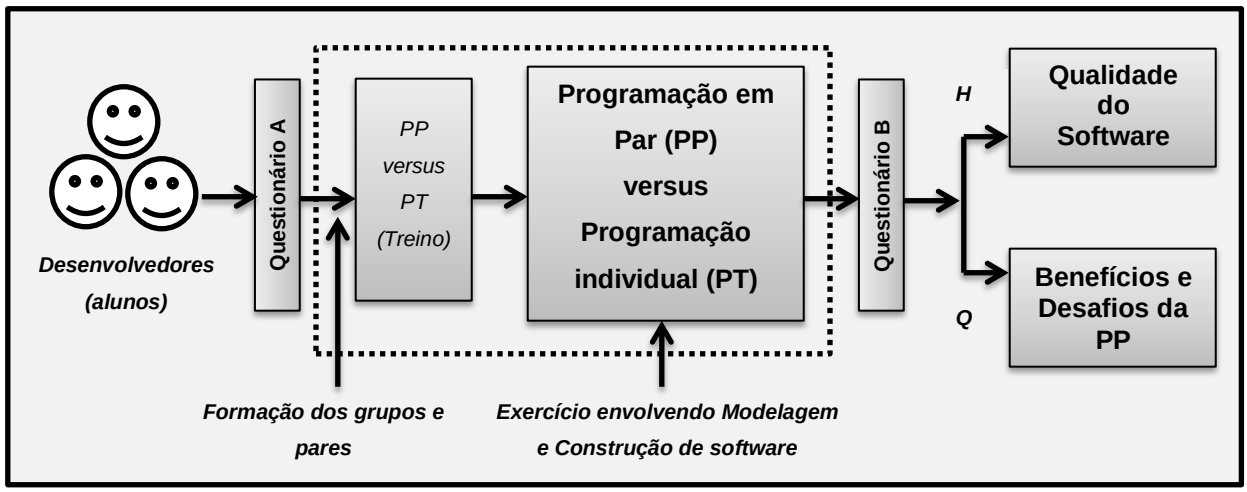


Figura 3 – Modelo conceitual desta pesquisa experimental
Fonte: Autoria própria.

No dia da execução do estudo os alunos foram convidados a preencher um questionário (Questionário A: Levantamento inicial) e a realizar uma atividade treino e uma atividade de desenvolvimento de software. Logo após o preenchimento do questionário, há a formação dos grupos, experimental (PP) e controle (PT), e, dentro do primeiro, das duplas. Em ambos os momentos, os alunos foram direcionados para determinados grupos ou formaram determinadas duplas de acordo com seus respectivos níveis de qualificação/habilidade, sendo esse definido antes da execução do estudo com base no coeficiente de rendimento acadêmico.

A atividade treino e de desenvolvimento constitui o experimento propriamente dito (ver linha tracejada na Figura 3 apresentada anteriormente). A seguir uma breve descrição da finalidade de cada uma dessas atividades.

- Atividade treino: recordar o funcionamento da IDE e algumas peculiaridades da tecnologia Java, e também exercitar as regras e diretrizes da PP. Ela é norteada por um exercício de manutenção de uma aplicação (inclusão de uma nova funcionalidade em projeto Java já configurado dentro da IDE).
- Atividade de desenvolvimento: realizar a modelagem e construção de um software. Nesta atividade há uma descrição de uma aplicação hipotética (exercício) envolvendo modelagem e construção de software – com restrições e diretrizes gerais para o desenvolvimento da aplicação e as funcionalidades esperadas pelo cliente/usuário. As funcionalidades foram descritas com base na prática de Histórias e complementadas por testes de aceitação e exemplos de telas/layouts para cada teste.

As atividades de treino e de desenvolvimento foram realizadas nos laboratórios dessas instituições em horário de aula, sendo que a duração é de duas aulas (aproximadamente 1 hora e 40 minutos) e era esperado um número não muito grande de alunos (participantes), entre 8 e 18 participantes.

Após o experimento, foi enviado por e-mail para cada aluno um convite de preenchimento do segundo questionário (Questionário B: Levantamento pós-atividades de desenvolvimento).

Os instrumentos (questionários, por exemplo), artefatos (exercício de manutenção, por exemplo), roteiros/guias e afins necessários para realização da pesquisa são descritos e avaliados nas subseções a seguir.

3.2.2 Planejamento

Detalhamento da Hipótese e Questão de Pesquisa

A hipótese de pesquisa definida no início deste documento, *“H: A qualidade do software implementado pelos desenvolvedores que utilizaram a PP é melhor que a dos*

desenvolvedores que realizaram as tarefas individualmente", requer desdobramentos. E, de acordo com os atributos descritos na subseção anterior, as hipóteses e quais as medidas de código que são necessárias para avalia-las estão descritas a seguir.

a) Tamanho dos métodos (HP.tm)
   - Hipótese nula ($HP.tm_0$): o tamanho médio (em linhas de código) dos métodos desenvolvidos pelos pares é igual ao dos desenvolvedores individuais.
     - $HP.tm_0$: média do número de linhas de código dos métodos do software desenvolvido pela PP é = à da PT.
   - Hipótese alternativa ($HP.tm_a$): o tamanho médio (em linhas de código) dos métodos desenvolvidos pelos pares é menor que o dos desenvolvedores individuais;
     - $HP.tm_a$: média do número de linhas de código dos métodos do software desenvolvido pela PP é < que a da PT.
   - Medidas necessárias: média do número de linhas de código dos métodos do software desenvolvido.

b) Complexidade estrutural dos métodos (HP.cem)
   - Hipótese nula ($HP.cem_0$): a complexidade ciclomática média dos métodos desenvolvidos pelos pares é igual a dos desenvolvedores individuais.
     - $HP.cem_0$: média da complexidade ciclomática de McCabe dos métodos do software desenvolvido pela PP é = à da PT.
   - Hipótese alternativa ($HP.cem_a$): a complexidade ciclomática média dos métodos desenvolvidos pelos pares é menor que a dos desenvolvedores individuais.
     - $HP.cem_a$: média da complexidade ciclomática de McCabe dos métodos do software desenvolvido pela PP é < que a da PT.
   - Medidas necessárias: média da complexidade ciclomática de McCabe dos métodos do software desenvolvido.

c) Falta de coesão dos métodos por classe (HP.fcc)

- Hipótese nula ($HP.fcc_0$): a média da falta de coesão dos métodos por classe desenvolvidos pelos pares é igual a dos desenvolvedores individuais.
    - $HP.fcc_{0:}$ média da falta de coesão dos métodos por classe do software desenvolvido pela PP é = à da PT.
- Hipótese alternativa ($HP.fcc_a$): a média da falta de coesão dos métodos por classe desenvolvidos pelos pares é menor que a dos desenvolvedores individuais.
    - $HP.fcc_a$: média da falta de coesão dos métodos por classe do software desenvolvido pela PP é < que a da PT.
- Medidas necessárias: média da falta de coesão dos métodos por classe do software.

d) Acoplamento/dependência entre pacotes (HP.ap)

- Hipótese nula ($HP.ap_0$): a média da instabilidade por pacote do software desenvolvido pelos pares é igual a dos desenvolvedores individuais.
    - $HP.ap_{0:}$ média da instabilidade por pacote do software desenvolvido pela PP é = à da PT.
- Hipótese alternativa ($HP.ap_a$): a média da instabilidade por pacote do software desenvolvido pelos pares é menor que a dos desenvolvedores individuais.
    - $HP.ap_a$: média da instabilidade por pacote do software desenvolvido pela PP é < que a da PT.
- Medidas necessárias: média da instabilidade por pacote do software desenvolvido.

Para testar as hipóteses descritas anteriormente foram coletados os seguintes dados:

a) Prática de desenvolvimento: medido por PP ou TP (escala nominal);
b) Métricas disponíveis no *plugin* Metrics da IDE Eclipse:
    - Média do número de linhas de código dos métodos do software desenvolvido (escala da medida é do tipo razão): esse dado é obtido por meio da informação presente na coluna *Mean* da métrica *Method Lines of Code*, ou MLOC, do *plugin*;
    - Média da complexidade ciclomática de McCabe dos métodos do software desenvolvido (escala da medida é do tipo intervalar): esse dado é obtido por meio da informação presente na coluna *Mean* da métrica *McCabe Cyclomatic Complexity*, ou VG, do *plugin*;
    - Média da falta de coesão dos métodos por classe do software desenvolvido (escala da medida é do tipo intervalar): esse dado é obtido por meio da informação presente na coluna *Mean* da métrica *Lack of Cohesion of Methods*, ou LCOM, do *plugin*;
    - Média da instabilidade por pacote do software desenvolvido (escala da medida é do tipo intervalar): esse dado é obtido por meio da informação presente na coluna *Mean* da métrica *Instability*, ou RMI, do *plugin*;

Por outro lado, para examinar a questão *"Q: Qual a percepção dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par?"* foram coletados os seguintes dados:

a) A PP ajuda a manter os integrantes focados no desenvolvimento do software?: no questionário pós-experimento é solicitado ao desenvolvedor, que desenvolveu em dupla, que ele avalie a seguinte questão fraseada como afirmativa *"Meu (ou minha) colega ajudou a me manter focado no desenvolvimento do software".* Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de "discordo totalmente" a "concordo totalmente".
b) Relatos dos desenvolvedores sobre os Benefícios e Desafios (ou desvantagens) do uso da Programação em Par: no questionário pós-experimento é solicitado ao desenvolvedor, que desenvolveu em dupla, que

responda a seguinte questão "*Por favor, liste quaisquer benefícios ou desvantagens da prática Programação em Par*".

Por fim, além dos dados citados anteriormente, conforme literatura revisada, para subsidiar a realização e a avaliação da qualidade do próprio experimento foram coletados os seguintes dados:

a) Qualificação/habilidade do desenvolvedor. É medido pela introdução de uma classificação em quatro classes: 4 (rendimento baixo-médio), 3 (rendimento médio), 2 (rendimento médio-alto) e 1 (rendimento alto).
    - A classificação é realizada com antecedência e é a partir dos coeficientes de rendimento (CR)[6] acadêmico geral dos alunos que participarão do estudo. Em linhas gerais, cada classe corresponde a uma faixa de valores de coeficientes de rendimento. As faixas possuem o mesmo "tamanho". Uma vez encontrada a qual faixa de valores pertence o coeficiente de rendimento do aluno, a classe correspondente a essa faixa é o nível de qualificação/habilidade do participante.
b) Nível de adoção da PP por parte dos desenvolvedores: ele é avaliado a partir dos questionamentos/mecanismos definidos abaixo.
    - Disposição dos desenvolvedores para o trabalho em equipe: no questionário pré-experimento é solicitado ao desenvolvedor que ele avalie a seguinte questão fraseada como afirmativa *"Eu gosto de trabalhar com outras pessoas para alcançar uma meta"*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de "discordo totalmente" a "concordo totalmente".
    - Disposição dos desenvolvedores para a resolução de problemas/exercícios de modelagem e construção de software: no questionário pré-experimento é solicitado ao desenvolvedor que ele

[6] "O CR é índice de rendimento acadêmico, e leva em consideração tanto a média final quanto a carga horária/créditos da disciplina. Ele é calculado após o fim de cada semestre e é cumulativo. Assim, o coeficiente que aparece no sistema no seu segundo semestre leva em conta *as matérias cursadas* no primeiro(...)". Mais detalhes em http://dainf.ct.utfpr.edu.br/peteco/wp-content/uploads/2011/08/manual.pdf

avalie a seguinte questão fraseada como afirmativa *“Eu gosto de resolver problemas do tipo onde é dado o ‘quê’ é esperado pelo cliente/usuário e há liberdade por parte do desenvolvedor em definir o ‘como’”*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de “discordo totalmente” a “concordo totalmente”.

- Compatibilidade da dupla: no questionário pós-experimento é solicitado ao desenvolvedor, que desenvolveu em dupla, que ele avalie a seguinte questão fraseada como afirmativa “*Avalie quanto compatível foi trabalhar com o seu colega*”. Ele deve escolher uma alternativa dentre as seguintes opções: 1) “Incompatível / A dupla ‘não funcionou’”; 2) “OK / ‘Sem maiores problemas/dificuldades de interação’”; 3) “Muito compatível”.
- Aprovação quanto ao uso da PP: ela é avaliada a partir dos dois questionamentos descritos abaixo.
  - No questionário pré-experimento:
    - É apresentada uma definição da PP: *“Os desenvolvedores trabalham lado-a-lado em único computador, trocando periodicamente quem codifica (piloto) e quem revisa e acompanha (navegador) continua e ativamente o trabalho desenvolvido – por exemplo, pensando e sugerindo novas ideias ou estratégias”.*
    - É solicitado ao desenvolvedor que ele avalie a seguinte questão fraseada como afirmativa *“Eu gostaria de desenvolver um software utilizando a prática de Programação em Par”*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de “discordo totalmente” a “concordo totalmente”
  - No questionário pós-experimento:
    - É solicitado ao desenvolvedor, que desenvolveu em dupla, que ele avalie a seguinte questão fraseada como

afirmativa *“Eu gostei de utilizar a Programação em Par para resolver o tipo de problema apresentado”*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de “discordo totalmente” a “concordo totalmente”.

- Aprovação quanto ao uso das Histórias (descrição dos exercícios): ela é avaliada a partir dos dois questionamentos descritos abaixo.
  - No questionário pré-experimento:
    - É apresentada uma definição da prática História: *“Desenvolver o software a partir de unidades de funcionalidade ‘visíveis ao cliente’. As funcionalidades são brevemente descritas pelo cliente idealmente em cartões de papel, que podem ser manuseados facilmente por ele mesmo e pelos desenvolvedores. Nelas pode haver definição de regras de negócio e/ou testes de aceitação (o cliente/usuário valida se o software/funcionalidade está fazendo aquilo que ele espera)”.*
    - É solicitado ao desenvolvedor que ele avalie a seguinte questão fraseada como afirmativa *“Eu gostaria de desenvolver um software utilizando a prática Histórias”*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de “discordo totalmente” a “concordo totalmente”
  - No questionário pós-experimento:
    - É solicitado ao desenvolvedor, que desenvolveu em dupla, que ele avalie a seguinte questão fraseada como afirmativa *“Eu gostei de desenvolver o software a partir da prática Histórias”*. Ele deve escolher uma alternativa dentre uma escala de Likert de 5 pontos de “discordo totalmente” a “concordo totalmente”

- Adoção das regras e diretrizes da prática da PP por parte dos desenvolvedores. Antes do experimento (no questionário pré-experimento) há a definição da prática a todos os desenvolvedores, e eventuais dúvidas por parte deles são esclarecidas pelo pesquisador. E mais, durante o experimento, na atividade treino é entregue às duplas um guia com diretrizes para Programação em Par (mais detalhes na subseção Instrumentação). Além disso, a troca de papeis é garantida pelo pesquisador, pois ele solicita aos desenvolvedores a troca de piloto/navegador a cada 5 minutos, na atividade treino, e 10 minutos, na atividade desenvolvimento. Essas regras e diretrizes são medidas com a introdução de duas avaliações: na primeira o desenvolvedor dá uma nota de 0 a 10 para o seu próprio desempenho (auto avaliação); e na segunda o desenvolvedor avalia o desempenho de seu colega – ou parceiro de dupla.
    - No questionário pós-experimento:
        - É apresentada a seguinte descrição "*Para as duas questões apresentadas a seguir considere as seguintes referências: NOTA"0"-> não conseguiu seguir NENHUMA das regras e diretrizes da prática Programação em Par em NENHUMA das tarefas realizadas durante o desenvolvimento do software; NOTA "10"-> conseguiu seguir praticamente TODAS as regras e diretrizes da prática Programação em Par em TODAS as tarefas realizadas durante o desenvolvimento do software;"*
        - É solicitado ao desenvolvedor que ele avalie as seguintes questões "*1) Dê uma nota de 0 a 10 para o seu desempenho (auto avaliação)*" e "*2) Dê uma nota de 0 a 10 para desempenho do seu colega*".

Análise do nível de adoção da PP: uma negativa em todos estes itens por parte de um dos integrantes da dupla, ou uma negativa no item

"compatibilidade da dupla" por ambos, é um indício forte do não "funcionamento" da PP para uma determinada dupla. Logo, o software desenvolvido por ela é "descartado" da análise de qualidade do código, mas não da análise da percepção dos desenvolvedores quanto ao uso da PP.

c) Situação do software entregue pelos desenvolvedores: devido à limitação de tempo de desenvolvimento é esperado que alguns desenvolvedores não entreguem todas as histórias implementadas, sendo assim, é necessário definir critérios para identificar quais projetos são passíveis de análise de qualidade do código. A situação do software (projeto) dos desenvolvedores é medida a partir de uma avaliação realizada por dois especialistas considerando dois aspectos do projeto/código-fonte. Os aspectos são (ver Apêndice H para mais detalhes):
    - Situação do projeto Java: o projeto está com "Erro de compilação", ou "Executando sem interação", ou "Executando com interação";
    - "Intenção" de design do desenvolvedor[es] (projeto de classes):
        - Nota "0": Não avaliado.
        - Nota de "1" (design estruturado) a "5" (ótimo design OO).

Análise da situação do software entregue pelos desenvolvedores: o software (ou projeto Java) fará parte da análise de qualidade do código se ele (1º) está compilando e (2º) obtiver no mínimo nota 2 na história 1.

Desenho experimental

A variável investigada é a prática de desenvolvimento (PP ou PT), especificamente a eficácia da PP versus PT perante tarefas de modelagem e construção de software. As variáveis observadas são: tamanho dos métodos, complexidade estrutural dos métodos, falta de coesão dos métodos por classe e acoplamento/dependência entre pacotes.

O tipo de desenho experimental para a configuração especificada anteriormente é "um fator com dois tratamentos". Nele buscam-se comparar os dois tratamentos entre

si (no contexto deste trabalho, PP versus PT), confrontando as médias das variáveis dependentes (no contexto deste trabalho, tamanho dos métodos – por exemplo) de cada tratamento. Entretanto, o desenho experimental específico escolhido é o qual se usa os mesmos objetos (descrição do exercício de desenvolvimento de software) para ambos os tratamentos (PP e PT) e atribui-se os sujeitos (alunos) aleatoriamente a cada tratamento. E cada sujeito usa somente um tratamento perante um objeto (WOHLIN et al., 2012, p.93-96).

A seguir um exemplo com dados fictícios de como ocorreu a atribuição dos sujeitos entre os tratamentos, PP (grupo experimental) e PT (grupo de controle) – ou formação de grupos e pares.

1) Inicialmente, os sujeitos que irão efetivamente participar do experimento são agrupados e ordenados conforme suas classes de qualificação/habilidade;

Lista original		**Lista agrupada e ordenada por classe**	
Sujeito 1	3	Sujeito 2	1
Sujeito 2	1	Sujeito 4	1
Sujeito 3	2	Sujeito 8	1
Sujeito 4	1	Sujeito 3	2
Sujeito 5	2	Sujeito 5	2
Sujeito 6	2	Sujeito 6	2
Sujeito 7	4	Sujeito 1	3
Sujeito 8	1	Sujeito 7	4

Quadro 2 – Exemplo de agrupamento e ordenação dos sujeitos
Fonte: Autoria própria.

2) Cada sujeito de cada classe é atribuído alternadamente para os tratamentos (PP e PT), sendo que o primeiro tratamento a receber o primeiro sujeito é definido aleatoriamente. Entretanto, se a quantidade de integrantes do grupo experimental é ímpar, é sorteado aleatoriamente um número que corresponde à posição de um dos sujeitos na lista do grupo em questão para que ele possa ser transferido para o grupo de controle;

Grupo experimental (PP)		**Grupo de controle (PT)**	
Sujeito 2	1	Sujeito 4	1
Sujeito 8	1	Sujeito 3	2
Sujeito 5	2	Sujeito 6	2
Sujeito 1	3	Sujeito 7	4

Quadro 3 – Exemplo de agrupamento e classificação dos sujeitos
Fonte: Autoria própria.

3) Internamente no grupo experimental, os pares são formados combinando o primeiro sujeito da classe no topo da lista com o primeiro sujeito da classe subsequente.

Grupo experimental (PP)		Pares
Sujeito 2	1	1
Sujeito 5	2	
Sujeito 8	1	2
Sujeito 1	3	

Quadro 4 – Exemplo de formação dos pares
Fonte: Autoria própria.

Dessa maneira os grupos e pares formados respeitam os princípios gerais para o projeto experimental, que são balanceamento, agrupamento e aleatoriedade. E mais, para o desenho experimental descrito anteriormente, o número esperado de participantes descrito na subseção Delimitação e variáveis dependentes com medidas em escala da classe intervalar (no mínimo), com isso um teste estatístico que pode ser utilizado é o teste de Wilcoxon[7] para amostras independentes (WOHLIN et al., 2012).

Instrumentação

Nesta subseção é definida a instrumentação necessária para a realização da pesquisa experimental. Eles estão separados por Objetos (por exemplo, ambiente de desenvolvimento e descrição dos exercícios), Instrumentos de medição ou levantamento de dados (questionários) e, por fim, Diretrizes ou roteiros.

a) Objetos
    a. Descrição do exercício de manutenção de software (atividades treino): ver Apêndice G, Passo 3, item “Aplicação Treino: Calculadora de somar e subtrair Inteiros”;
    b. Código do exercício de manutenção de software (atividade treino): ver Apêndice F;

[7] Maiores detalhes em Wohlin et al. (2012, p.132-141) e http://www.portalaction.com.br/964-t%C3%A9cnicas-n%C3%A3o-param%C3%A9tricas.

c. Descrição do exercício de desenvolvimento de software: ver Apêndice E.
d. Ambiente de desenvolvimento:
    1) Eclipse Juno versão 4.2: IDE Java;
    2) JRE 7: ambiente de execução Java versão 7[8];
    3) Documentação da JRE 7: documentação para o ambiente de execução Java versão 7[9];

b) Instrumentos de medição ou levantamento de dados: os questionários foram elaborados, [auto]aplicados e acompanhados por meio de formulários do Google Drive[10].
    a. Questionário A – Levantamento inicial: ver Apêndice A;
    b. Questionário B – Levantamento pós-atividades de desenvolvimento: ver Apêndice B;
    c. *Plugin* Metrics da IDE Eclipse: mais detalhes em ECLIPSE METRICS PLUGIN (2012). Ver Apêndice M para exemplo (figura) da interface do *plugin*;
    d. Aplicação de exportação (transformação) dos arquivos XML do plugin para arquivos em formato *Comma-separared values* (ou .csv)[11] (planilha): foi desenvolvida uma aplicação para transformar os arquivos exportados do Metrics em arquivos .csv, os quais são importados para planilhas para tratamento/teste estatístico.

c) Diretrizes ou roteiros
    a. Diretrizes para Programação em Par: ver Apêndice I, documento elaborado a partir da tradução livre e adaptação de Pair ... (2008).

---

[8] Disponível em http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jre-7u21-oth-JPR
[9] Disponível em http://www.oracle.com/technetwork/java/javase/documentation/java-se-7-doc-download-435117.html
[10] Disponível em https://drive.google.com
[11] Mais detalhes em http://pt.wikipedia.org/wiki/Comma-separated_values

b. Situação do software entregue pelos desenvolvedores: ver Apêndice H.
c. Roteiro para execução da pesquisa experimental: ver Apêndice G.
d. Roteiro geral do estudo:

    1) Identificar e entrar em contato com os professores voluntários/candidatos.

    Uma vez confirmada a participação do professor/alunos na pesquisa.

    2) Solicitar planilha com a lista de alunos e respectivos coeficientes de rendimento e preparar lista de alunos agrupados e ordenados por nível de qualificação/habilidade;
    3) Visitar instituição/laboratórios;
    4) Preparar laboratórios: disponibilizar o material (ambiente de desenvolvimento, roteiro geral, descrição do software) para download;
    5) Executar a pesquisa experimental: ver item “Roteiro para execução da pesquisa experimental”;
    6) Copiar os projetos dos alunos;
    7) Enviar e-mail convite para preenchimento do Questionário B;
    8) Acompanhar o número de respostas ao Questionário B;

3.2.3 Validade

A validade de uma pesquisa experimental (ou experimento, neste caso) está relacionada ao nível de confiança que se pode ter no processo de investigação como um todo (TRAVASSOS et al., 2002; WAINER, 2007). Conforme Lima, Seca Neto e Emer (2012), um experimento de qualidade envolvendo a PP requer uma análise detalhada de pelo menos 21 ameaças, estando elas distribuídas dentre os quatro tipos de validades - interna, externa, de construção e de conclusão. Por isso, conforme Wohlin et al. (2012, p. 89) e Mafra e Travassos (2006, p. 7-9), neste trabalho a avaliação da validade não foi realizada apenas uma única vez após o final da etapa de planejamento, mas de maneira iterativa, onde, por exemplo, o planejamento é confrontado várias vezes por meio da constante análise das 21 ameaças citadas anteriormente. Esse confronto é motivado pelo resultado das revisões de outros pesquisadores ou da realização de um piloto (mais detalhes na subseção seguinte).

A seguir um quadro com o detalhamento da análise da validade desta pesquisa. A análise foi realizada com base nas 21 ameaças mapeadas em Lima et al. (2012).

Nº	**Ameaça**	**Resultado da análise** **(e definição do mecanismo de proteção, quando pertinente)**
Validade interna A validade interna, para Travassos et al. (2002), "define se o relacionamento observado entre o tratamento e o resultado é causal, e não é resultado da influência de outro fator – não controlado ou medido".		
1	*Testagem*: o desenho proporciona com que os participantes aprendam com seus próprios erros; está relacionada à sequência da coleta dos resultados versus o momento da intervenção, por exemplo.	*Neutralizada:* consequência do desenho experimental escolhido, com isso os sujeitos "passam" pelos tratamentos apenas uma única vez. Entretanto, há atividade treino, mas o problema abordado nela é diferente da atividade principal e nenhum dado dessa atividade é coletado e/ou utilizado na pesquisa.
2	*Influência de parte da intervenção:* aparece em diferentes domínios com diferentes nomes, logo, este efeito não tem um nome padrão, mas a ideia é que o efeito observado não é devido à intervenção como um todo, mas	*Minimizada:* os sujeitos passam e utilizam praticamente os mesmo objetos, instrumentos, diretrizes/roteiro, salvo, o tratamento PP que é foco deste estudo.

	devido a apenas parte dela.	
3	*História*: possibilidade de que o resultado do processo é consequência de um evento externo ao experimento.	*Minimizada*: consequência do desenho experimental escolhido, com isso os sujeitos "passam" pelos tratamentos apenas uma única vez. Mas, deve-se realizar o experimento em períodos onde não haja eventos (antes, durante ou depois) diretamente relacionados ao experimento. Logo, é uma ameaça que deve ser reavaliada após a realização do experimento.
4	*Efeito da expectativa do sujeito:* os participantes esperam ou buscam um determinado resultado – positivo ou negativo – para o experimento, ou ainda, pode haver um efeito positivo pelo simples fato dos participantes saberem que estão sendo observados – conhecido também por efeito Hawthorne.	*Minimizada:* os sujeitos não são informados quanto às hipóteses e/ou questões de pesquisa. Com isso, eles não o que é positivo e/ou negativo. E quanto ao efeito Hawthorne, vale ressaltar que participantes de ambos os grupos, PP e PT, estão sendo "observados", então se esse efeito ocorrer dificilmente irá favorecer um grupo apenas.
5	*Seleção*: os participantes não são selecionados de maneira aleatória dentre os grupos ou eles não foram divididos de maneira igualitária tanto no aspecto quantitativo quanto qualitativo.	*Neutralizada:* consequência da maneira como é realizada a formação dos grupos e pares e do desenho experimental escolhido.
6	*Mortalidade seletiva:* a evasão de participantes com características específicas e relevantes durante o experimento.	*Minimizada:* os sujeitos "passam" pelos tratamentos apenas uma única vez. Mas, é uma ameaça que deve ser reavaliada após a realização do experimento.
7	*Regressão à média:* fenômeno que pode trazer ameaça ao experimento. Em geral, conforme Bandeira (2012), essa ameaça está presente quando se trabalha apenas com um grupo (somente o grupo experimental) e seus integrantes são selecionados de maneira deliberada com base nos piores ou nos melhores escores do pré-teste, ou seja, não há um grupo de controle e seleção aleatória de participantes.	*Neutralizada*: há o grupo de controle.
8	*Instrumentação:* a diferença nos resultados é consequência de uma medição incorreta ou instrumento inadequado.	*Minimizada*: a medição é realizada de maneira automatizada (plugin Metrics) e os instrumentos são definidos a partir de referencias teóricos diretamente relacionados com o objeto de estudo, ou adaptados a partir do material utilizado em treinamentos de certificação sobre métodos e práticas ágeis

		cursados por um dos autores deste trabalho, ou adaptados de experiências vivenciadas por um dos autores deste trabalho[12]. E mais, os instrumentos, objetos e diretrizes/roteiros foram revisados por outros pesquisadores e profissionais de software (tanto na teoria quanto na prática - piloto do experimento). A aplicação deles é avaliada por meio de dados coletados antes e/ou após o experimento, tais como: (1) disposição dos desenvolvedores para a resolução de problemas/exercícios de modelagem e construção de software; (2) aprovação quanto ao uso da PP; (3) aprovação quanto ao uso das Histórias; (4) adoção das regras e diretrizes da prática da PP por parte dos desenvolvedores.
9	*Comportamento compensatório:* alguma pessoa, ou entidade externa ou não ao experimento, sente que o grupo de controle foi preterido e, com isso, cria medidas compensatórias para ele.	*Minimizada:* os sujeitos "passam" pelos tratamentos apenas uma única vez. Mas, devem-se acompanhar as reações dos sujeitos perante os dois diferentes tratamentos (PP e PT). Sendo assim, é uma ameaça que deve ser reavaliada após a realização do experimento.
10	*Contaminação:* por exemplo, participantes que fazem parte de um grupo ensinam, ou aprendem, com participantes do outro grupo.	*Minimizada:* consequência do desenho experimental escolhido, com isso os sujeitos "passam" pelos tratamentos apenas uma única vez. E mais, busca-se distribuir fisicamente os sujeitos dentre as estações de trabalhos de maneira que haja uma distância razoável entre eles, e monitorar pessoalmente suas atividades.
11	*Comportamento competitivo:* os membros do grupo de controle se sintam preteridos frente aos do outro grupo, e podem se mostrar motivados a competir com o grupo experimental.	*Minimizada:* os sujeitos não são informados quanto às hipóteses e/ou questões de pesquisa. Com isso, eles não sabem qual realmente é o grupo de controle e/ou experimental.
12	*Efeito da expectativa do experimentador:* o pesquisador interage intensamente com os participantes, as crenças dele causam um efeito no sujeito ou nos testes realizados por ele, ou ainda, o simples fato do pesquisador querer que o experimento "dê certo".	*Minimizada:* a instrumentação está estruturada de maneira a demandar o mínimo de participação por parte do pesquisador, quando ocorre são orientações pontuais e objetivas ("aluno1 você fará dupla com aluno2 nesta estação de trabalho"). Quando solicitado para sanar "dúvidas" da descrição do exercício de desenvolvimento de software, a fala padrão é "A

[12] Relato publicado em http://www.agilebrazil.com/2013/wbma/

		interpretação do exercício faz parte da atividade". Vale ressaltar que, devido ao desenho do experimento, pesquisador e sujeito se encontraram apenas uma única vez. E mesmo após o experimento, se procurado, o pesquisador "se nega" a esclarecer qualquer dúvida/questionamento sobre o experimento, isso até receber a resposta do questionário B, pelo menos. E mais, há participação de 4 especialistas durante a pesquisa, 2 na avaliação/seleção dos projetos e 2 na revisão da análise dos relatos dos alunos.
13	*Maturação*: os sujeitos podem tornar-se mais capazes ou ficarem desmotivados com o passar do tempo.	*Neutralizada:* consequência do desenho experimental escolhido, com isso os sujeitos "passam" pelos tratamentos apenas uma única vez.
Validade externa A validade externa, conforme Travassos et al. (2002), "define condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial".		
14	*Participantes:* selecionar participantes que não possuem relação ou refletem o comportamento da população, ou, trabalhar com uma amostra não representativa quantitativamente e qualitativamente da população.	Sabe-se da dificuldade em realizar experimentos em empresas, pois elas não querem arcar com os custos e riscos inerentes a uma pesquisa científica, tais como pessoal e infraestrutura. Sendo assim, uma alternativa é realiza-los com estudantes. Por exemplo, em Sfetsos et al. (2008) e Balijepally et al. (2009). De acordo com Höst et al. (2000), os estudantes podem ser usados sem necessariamente causar ameaça a validade do experimento. Neste trabalho, para *minimizar* tal ameaça, foi dada preferência a alunos dos últimos períodos – ou que no mínimo passaram pelas disciplinas de Lógica para Computação, Fundamentos de Programação (ou Algoritmos e Estrutura de Dados), Banco de Dados, Análise e Projeto de Sistemas (ou Desenvolvimento de Sistemas), Engenharia de Software, e pelo menos uma disciplina de programação utilizou a linguagem/tecnologia Java. E mais, espera-se realizar o experimento em diferentes turmas e possivelmente instituições.
15	*Tempo:* por exemplo, impor ou suprimir restrições de tempo.	A limitação de tempo é necessária, sendo assim, buscou-se *minimizar* tal ameaça por meio, por exemplo, da delimitação de escopo do exercício de desenvolvimento de software.

16	*Configuração do experimento:* realizar o experimento em um ambiente muito diferente do ideal, ou adotar materiais ou instrumentos distantes da realidade.	*Minimizada:* ver resultado da análise da ameaça 14, quanto a ambiente. E, quanto a materiais e instrumentos, ver resultado da análise da ameaça 8.
Validade de construção A validade de construção, para Travassos et al. (2002), "considera os relacionamentos entre a teoria e a observação, ou seja, se o tratamento reflete a causa bem e o resultado reflete o efeito bem".		
17	*Projeto do experimento:* em geral, má definição da base teórica ou da definição do processo de experimentação.	[Praticamente] *Neutralizada*: buscou-se diferentes referenciais bibliográficos tanto na definição da base teórica, quanto do processo de experimentação, e além da variedade, priorizou-se autores com publicações teóricas e práticas sobre o tema abordado neste estudo.
18	*Fatores humanos (ou sociais):* os participantes podem basear seus comportamentos nas hipóteses de pesquisa ou eles podem estar envolvidos em outros experimentos.	Conforme comentado anteriormente, os participantes não tem conhecimento das hipóteses de pesquisa, e busca-se realizar os experimentos em períodos onde eles não estejam envolvidos em outras pesquisas/experimentos. Mas é sabido que o pesquisador não possui controle sobre tais fatores, por isso que se buscou monitorar os que estão diretamente relacionados com a prática investigada neste estudo. Por exemplo, a questão da compatibilidade da dupla e aprovação quanto ao uso da PP e Histórias por parte dos sujeitos. Se um dos sujeitos apresentar um "valor" muito diferente da maioria para tais fatores, ele é excluído da análise dos dados. Logo, com isso essa ameaça é *minimizada* (ou até mesmo neutralizada), se ocorrer a exclusão ver possível impacto em consequência da ameaça 6.
Validade de conclusão A validade de conclusão está, segundo Travassos et al. (2002), "relacionada à habilidade de chegar a uma conclusão correta a respeito dos relacionamentos entre o tratamento e o resultado do experimento".		
19	*Análise e interpretação estatística:* o uso incorreto dos métodos/testes estatísticos durante a análise e interpretação dos dados coletados.	*Minimizada*: desde o início da etapa de planejamento, buscou-se suporte de especialistas em tal assunto junto ao Laboratório de Estatística Aplicada (LEA)[13] da Universidade Federal do Paraná (UFPR). Professores em conjunto com alunos do curso de Estatística revisaram o desenho experimental e auxiliaram (validaram) na escolha e aplicação do teste estatístico.

[13] Mais informações em http://www.lea.ufpr.br/

20	*Confiabilidade das medidas:* adoção de apenas uma medida ou tipo de medida por constructo. Ou, as medidas não refletem o que se busca observar. Ou ainda, elas são subjetivas.	*Minimizada*: neste estudo adotam-se vários tipos de medidas para avaliação do constructo qualidade de software (ver subseção 3.2.1 para mais deatlhes). Tal abordagem e medidas são adotadas em diferentes estudos, tais como: Nawrocki e Wojciechowski (2001), Hanks et. al. (2004), Vanhanen et al. (2005), Xu e Rajlich (2006), Sato (2007), Bipp et. al. (2008) e, por fim, Aniche (2012).
21	*Confiabilidade da implementação dos tratamentos:* a aplicação do tratamento não é realizada de maneira adequada pelos sujeitos.	*Minimizada*: o trabalho em dupla faz parte do dia a dia dos alunos no meio acadêmico, mas possivelmente a utilização de regras específicas como a troca de papeis (piloto e navegador) a cada 10 minutos e a formação aleatória de duplas de acordo com o nível de qualificação/habilidade de cada aluno. Para minimizar tal ameaça, é realizada uma atividade treino e avaliado o nível de qualificação/habilidade de cada aluno antes do experimento e avaliada, por exemplo, a “compatibilidade da par” ao final do experimento.

Quadro 5 – Análise detalhada da validade desta pesquisa
Fonte: Autoria própria.

3.2.4 Avaliação

A avaliação do experimento pode-se dar tanto pela análise de outros pesquisadores quanto por um experimento piloto (TRAVASSOS et al., 2006). Neste estudo realizamos ambas as avaliações.

A primeira ocorreu por meio da apresentação do projeto de pesquisa a outros pesquisadores nos Seminários de Acompanhamento do próprio PPGCA. Dentre as ponderações/considerações realizadas pelos pesquisadores, destacam-se as seguintes: (i) um dos pesquisadores solicitou melhorar a descrição da forma de coleta dos dados e das métricas – em geral, mais informação sobre a relação atributo do software, métrica e *plugin* – e (ii) outro colocou que o tema e a abordagem se demonstram interessantes, porém deve-se planejar bem e realizar um piloto, pois pesquisa experimental envolvendo pessoas é algo custoso e difícil de se realizar. E mais, como não foi possível definir junto aos pesquisadores um exemplo de software (exercício), ver Apêndices C e D, foi adaptado um exercício de materiais utilizados em treinamentos de certificação sobre métodos e práticas ágeis cursados por um dos autores deste trabalho.

A segunda ocorreu por meio da realização de um piloto do experimento com profissionais colegas de trabalho de um dos autores deste trabalho. Neste caso, destacam-se as seguintes ponderações/considerações: (i) praticamente todos os profissionais solicitaram um maior detalhamento das restrições e diretrizes gerais para o desenvolvimento do software (exercício) e (ii) um deles colocou a necessidade de se entregar um roteiro detalhado da execução da pesquisa.

4. EFICÁCIA DA PROGRAMAÇÃO EM PAR PERANTE TAREFAS DE MODELAGEM E CONSTRUÇÃO DE SOFTWARE

4.1 EXECUÇÃO DA INVESTIGAÇÃO

A pesquisa foi realizada com quatro turmas voluntárias de cursos da área de Computação de três instituições de ensino superior da cidade de Curitiba (ver Apêndice K para carta de apresentação). As atividades ocorreram entre os meses de junho e julho do ano de 2013 em diferentes dias da semana. Abaixo um quadro com o perfil dessas turmas e instituições de ensino.

Id. da turma	Curso	Perfil da instituição	Qtd. de participantes
A	Engenharia da Computação (EC)	Universidade A: instituição pública de grande porte	4
B	Bacharelado em Sistemas de Informação (BSI)		12
C	Análise e Desenvolvimento de Sistemas	Universidade B: instituição pública de grande porte.	25
D	Bacharelado em Sistemas de Informação (BSI)	Faculdade A: instituição particular de pequeno/médio porte.	3

Quadro 6 – Perfil dos participantes desta pesquisa
Fonte: Autoria própria.

Dentre essas quatro turmas, buscando manter a qualidade da pesquisa, três foram excluídas completamente, pois nas turmas A e D houve mortalidade seletiva – um participante em cada turma não finalizou e cumpriu com o planejado para a atividade de desenvolvimento – e na turma C houve contaminação – todos os participantes do grupo de controle ficaram muito próximos um do outro. Com isso, a seguir a análise, apresentação e interpretação dos dados da turma B.

4.2 RESULTADOS

Para análise, apresentação e interpretação dos benefícios e desafios (ou desvantagens) da PP foram coletados dados de 6 sujeitos, sendo que não foi excluída nenhuma dupla. Nesse caso a análise é qualitativa e, tanto os dados, quanto o tratamento (método) aplicado a eles, estão descritos em detalhes no Apêndice L.

Por outro lado, para análise, apresentação e interpretação da qualidade do software desenvolvido pelos alunos dos 9 projetos (3 duplas e 6 individuais), 2 projetos de desenvolvedores individuais foram excluídos – ver Apêndice H para critérios de exclusão – com isso ficaram 3 projetos realizados em dupla (experimental) e 4 realizados individualmente (controle). Nesse caso a análise é quantitativa e foi realizada sobre os dados que estão no Apêndice J.

Sendo assim, a seguir, primeiramente o resultado da análise quantitativa de 3 projetos (3 duplas) versus 4 projetos (de 4 sujeitos que realizaram as atividades individualmente) e na sequência o resultado da análise qualitativa dos benefícios e desafios (ou desvantagens) relatados pelos 6 sujeitos que realizaram as atividades em dupla.

4.2.1 Qualidade do software desenvolvido utilizando PP versus PT

Tamanho dos métodos (HP.tm)

Abaixo o quadro com o resultado do teste de Wilcoxon para os dados (média) referentes ao tamanho dos métodos do software desenvolvido pelas duplas versus desenvolvedores individuais.

Hipótese alternativa (HP.tma)	**Grupo experimental**		**Grupo de controle**		**p-valor**
	Projeto	**Média**	**Projeto**	**Média**	
O tamanho médio (em linhas de código) dos métodos desenvolvidos pelos pares é menor que o dos desenvolvedores individuais.	C	4,538	F	6,8	0,0286
	B	4,846	D	7	
	A	5,25	G	13,667	
			E	20,5	

Quadro 7 - Resultado do teste de Wilcoxon para os dados (média) referentes ao tamanho dos métodos
Fonte: Autoria própria.

A partir do p-valor apresentado no Quadro 6, que é menor que 0,05 (5%), rejeitamos a hipótese nula, ficamos com a hipótese alternativa. Ou seja, o tamanho médio (em linhas de código) dos métodos desenvolvidos pelos pares é menor que o dos métodos desenvolvidos pelos desenvolvedores individuais.

Complexidade estrutural dos métodos (HP.cem)

Abaixo o quadro com o resultado do teste de Wilcoxon para os dados (média) referentes à complexidade ciclomática média dos métodos do software desenvolvido pelas duplas versus desenvolvedores individuais.

Hipótese alternativa (HP.cema)	Grupo experimental		Grupo de controle		p-valor
	Projeto	Média	Projeto	Média	
A complexidade ciclomática média dos métodos desenvolvidos pelos pares é menor que a dos desenvolvedores individuais	A	1,5	D	2,1	0,0286
	C	1,538	F	2,4	
	B	1,769	G	3,333	
			E	5	

Quadro 8 - Resultado do teste de Wilcoxon para os dados (média) referentes à complexidade ciclomática média dos métodos
Fonte: Autoria própria.

A partir do p-valor apresentado no Quadro 7, que é menor que 0,05 (5%), rejeitamos a hipótese nula, ficamos com a hipótese alternativa. Ou seja, A complexidade ciclomática média dos métodos desenvolvidos pelos pares é menor que a dos desenvolvedores individuais.

Falta de coesão dos métodos por classe (HP.fcc)

Abaixo o quadro com o resultado do teste de Wilcoxon para os dados (média) referentes à falta de coesão dos métodos por classe desenvolvidos pelos pares versus desenvolvedores individuais.

Hipótese alternativa (HP.fcca)	Grupo experimental		Grupo de controle		p-valor
	Projeto	Média	Projeto	Média	
A média da falta de coesão dos métodos por classe desenvolvidos pelos pares é menor que a dos desenvolvedores individuais.	A	0,125	E	0	0,6445
	B	0,125	F	0,125	
	C	0,611	D	0,25	
			G	0,25	

Quadro 9 - Resultado do teste de Wilcoxon para os dados (média) referentes à falta de coesão dos métodos por classe
Fonte: Autoria própria.

A partir do p-valor apresentado no Quadro 8, que é maior que 0,05 (5%), não podemos rejeitar a hipótese nula. Ou seja, a média da falta de coesão dos métodos por classe desenvolvidos pelos pares é estatisticamente igual a dos desenvolvedores individuais.

Acoplamento/dependência entre pacotes (HP.ap)

Abaixo o quadro com o resultado do teste de Wilcoxon para os dados (média) referentes à média da instabilidade por pacote – ou acoplamento/dependência entre pacotes – do software desenvolvido pelos pares versus desenvolvedores individuais.

Hipótese alternativa (HP.fcca)	Grupo experimental		Grupo de controle		p-valor
	Projeto	Média	Projeto	Média	
A média da instabilidade por pacote do software desenvolvido pelos pares é menor que a dos desenvolvedores individuais.	A	0,556	D	0,5	0,5
	B	1	E	1	
	C	1	F	1	
			G	1	

Quadro 10 - Resultado do teste de Wilcoxon para os dados (média) referentes à média da instabilidade por pacote
Fonte: Autoria própria.

A partir do p-valor apresentado no Quadro 9, que é maior que 0,05 (5%), não podemos rejeitar a hipótese nula. Ou seja, a média da instabilidade por pacote – ou

acoplamento/dependência entre pacotes – do software desenvolvido pelos pares é estatisticamente igual a dos desenvolvedores individuais..

4.2.2 Benefícios e Desafios (ou desvantagens) da PP

Benefícios da PP

Abaixo, o quadro com os benefícios relatados pelos participantes do experimento quanto ao uso da PP.

Benefícios relatados pelos participantes do experimento	Código(s) do participante(s)
'Ideias fluem mais facilmente; [...]'	AlunoH
'Discute-se mais antes de programar. [...]'	AlunoA
'[...] Enquanto um aluno programava o outro podia pensar em outros fatores faltantes no código que estava sendo desenvolvido. [...]'	
'[...] Vendo outro programar, você observa um padrão diferente do seu que as vezes ajuda a melhorar seu próprio nível de programação. [...]'	
'Há a troca de conhecimento. Com a mudança entre quem codifica, mescla-se os dois estilos em um mesmo código [...]'	AlunoJ
'[...] houve situações em que, enquanto um codificava, o outro percebia possíveis problemas que poderiam acontecer no futuro, que passariam despercebidos ou demorariam mais para serem percebidos, pois o codificador está focado no que está fazendo no momento.'	
'Mantém o foco [...]'	AlunoC
'[...] abre discussão para melhor solução'.	
'[...] amplia a possibilidade de formas de resolver o problema.'	AlunoI
'O maior benefício é quando existe um problema de difícil solução [...]'	AlunoB
'O maior benefício é quando [...] é necessário uma tomada de decisão muito grande no sistema, é possível ter uma visão diferente do assunto. [...]'	

Quadro 11 – Benefícios relatados pelos participantes do experimento
Fonte: Autoria própria.

E mais, abaixo um quadro com o consolidado das respostas dos participantes a declaração *“Meu (ou minha) colega ajudou a me manter focado no desenvolvimento do software (QB3)”*.

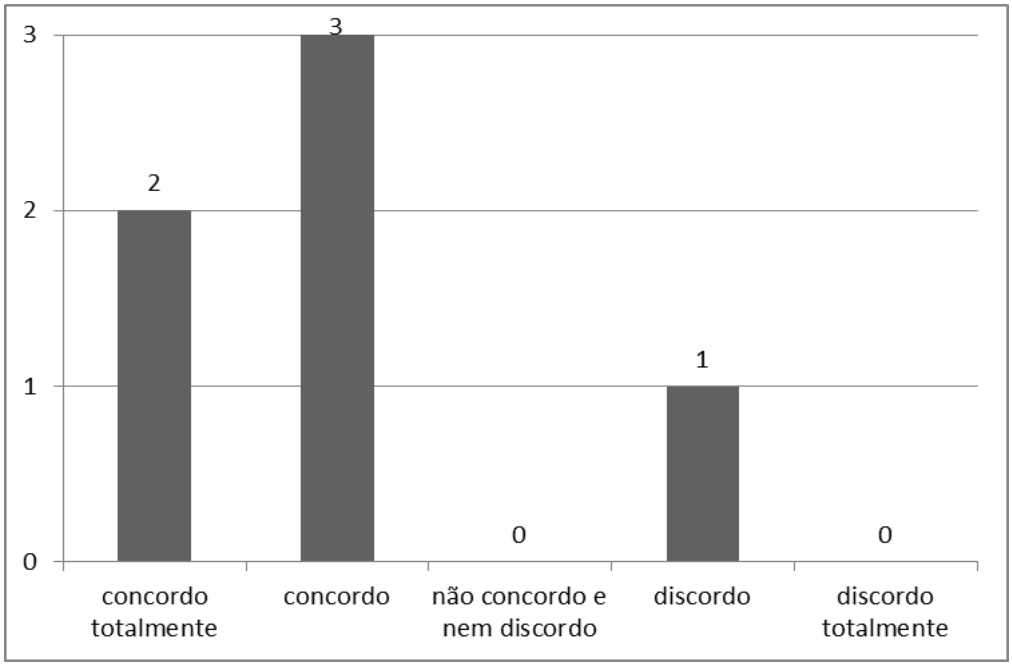


Figura 4 – A PP me mantém focado no desenvolvimento do software
Fonte: Autoria própria.

Cinco dos seis participantes concordam ou concordam totalmente que seu colega ajudou manter o foco no desenvolvimento do software. Somente um participante discordou.

Desafios (ou desvantagens) da PP

Abaixo, o quadro com os benefícios relatados pelos participantes do experimento quanto ao uso da PP.

Desafios (ou desvantagens) relatados pelos participantes do experimento	Código(s) do participante(s)
‘[...] Quando as duplas possuem conhecimentos muito diferentes, um acaba atrapalhando o outro e enquanto um pode crescer como programador, o outro fica estagnado empurrando o outro pra frente.’	AlunoB

‘Diminui a velocidade de resolução do problema, pois as vezes surgem discussões de implementação. [...]’	AlunoI
'[...] A maior desvantagem está presente nos momentos em que se apresenta soluções simples em que o outro pode mais atrapalhar do que ajudar.'	AlunoB

Quadro 12 – Desvantagens relatados pelos participantes do experimento
Fonte: Autoria própria.

4.3 OBSERVAÇÕES E DISCUSSÃO

Diferente do resultado “misto” reportado em Hanks et al. (2004) quanto ao tamanho médio dos métodos em linhas de código, observou-se neste estudo que pares podem desenvolver métodos menores se comparado aos desenvolvidos individualmente. E mais, ainda para Hanks et al. (2004) o resultado “misto” também se aplica à complexidade dos métodos quando medida pela complexidade ciclomática. Mas, segundo Vanhanen e Lassenius (2005), quando se usa PP há uma “ligeira” melhora na métrica em questão, algo que corrobora com o que encontramos neste trabalho, que pares podem desenvolver métodos menos complexos se comparado aos desenvolvidos individualmente.

Por outro lado, não se observou diferença significativa quando a qualidade do software é avaliada pela métrica falta de coesão dos seus métodos, diferente do resultado reportado em Bipp, Lepper e Schmedding (2008), onde os autores encontraram uma melhora na qualidade do software por meio do uso da PP quando avaliada pela métrica citada anteriormente. Além disso, não se observou diferença significativa quando a qualidade do software é avaliada pela métrica instabilidade – ou acoplamento/dependência entre pacotes, algo que corrobora os resultados encontrados em Madeyski (2006).

Além dos efeitos na qualidade do software produzido, a maioria dos alunos concordou que PP faz com que eles fiquem mais focados nas tarefas de modelagem e construção do software, inclusive um deles declarou explicitamente isso quando solicitado sobre quais eram os benefícios e/ou desafios quanto ao uso da PP.

E mais, ela pode auxiliar no entendimento do problema a ser resolvido e/ou planejamento da solução, pois um participante relatou que com PP “Ideias fluem mais facilmente; [...]” e outro que “Discute-se mais antes de programar. [...]”, ou ainda, parafraseando os relatos de dois outros participantes, tal prática ‘amplia as possibilidades de resolução do problema, principalmente se ele é de difícil solução’.

Além disso, a PP é um meio em potencial para o aprendizado, pois como declararam dois participantes, “Vendo outro programar, você observa um padrão diferente do seu que as vezes ajuda a melhorar seu próprio nível de programação” e “Há a troca de conhecimento. Com a mudança entre quem codifica, mescla-se os dois estilos em um mesmo código [...]”.

Agora, ela pode levar a despender um tempo desnecessário porque para um participante, “[...] A maior desvantagem está presente nos momentos em que se apresenta soluções simples em que o outro pode mais atrapalhar do que ajudar” e para outro participante, “Quando as duplas possuem conhecimentos muito diferentes, um acaba atrapalhando o outro e enquanto um pode crescer como programador, o outro fica estagnado empurrando o outro pra frente”. E mais, para um terceiro participante a PP “...diminui a velocidade de resolução do problema, pois as vezes surgem discussões de implementação. [...]”.

4.4 LIMITAÇÕES

Mesmo buscando diferentes lugares para realização da pesquisa experimental (investigação), uma das limitações deste estudo está relacionada à representatividade quantitativa, isso devido à análise e interpretação de um experimento com 12 sujeitos, pois quanto à representatividade qualitativa, de acordo com Höst et al. (2000), os estudantes podem ser usados sem necessariamente causar ameaça à validade do experimento. Por exemplo, em Sfetsos et al. (2008) e Balijepally et al. (2009). Com relação à limitação quantitativa, pode-se citar o estudo realizado por Nawrocki e Wojciechowski (2001), onde a pesquisa experimental envolveu 21 sujeitos.

Uma limitação pode estar associada à complexidade e/ou tipo de software utilizado na pesquisa experimental em questão. E outra à escolha do modelo de avaliação da qualidade do software desenvolvido, pois considera basicamente medidas de qualidade interna do produto. Porém, abordagens semelhantes são adotadas em Hanks et al. (2004), Vanhanen e Lassenius (2005), Bipp, Lepper e Schmedding (2008) e Madeyski (2006).

5. CONSIDERAÇÕES FINAIS

Por meio da pesquisa experimental realizada neste trabalho, podemos concluir que a prática Programação em Par mostrou-se mais eficaz perante tarefas de modelagem e construção de software que a programação individual se avaliada a qualidade do software desenvolvido, considerando o tamanho e a complexidade dos métodos. Mas, não se mostrou mais eficaz se considerada a coesão dos métodos por classe e o acoplamento/dependência entre pacotes. Os alunos perceberam mais benefícios do que desafios (ou desvantagens) quando se adota tal prática para o desenvolvimento de um software, com destaque para os seguintes benefícios: (a) auxilia no entendimento do problema a ser resolvido e/ou planejamento da solução e (b) ajuda a manter o foco na atividade de desenvolvimento. Tais descobertas podem levar ao desenvolvimento de soluções menores e menos complexas. Essas características fazem com que o software fique mais fácil de manter, por consequência há uma redução no custo de desenvolvimento e/ou manutenção do software.

Este trabalho contribuiu com o cenário descrito no início deste documento, pois gerou evidências empíricas que somadas às já existentes em outros estudos, servirão de base para novas pesquisas na área de Engenharia de Software. Além disso, o próprio método (protocolo de estudo) pode ser utilizado para replicar o experimento envolvendo a prática PP perante outros fatores ou servir de base para delinear uma investigação sobre eficácia de outras práticas ágeis ou de desenvolvimento de software em geral. Por fim, a aplicação[14] de exportação dos arquivos XML para arquivos em formato CSV desenvolvida pode facilitar o trabalho de outros pesquisadores que decidam adotar o *plugin* Metrics.

Ao longo do mestrado realizamos as seguintes produções técnicas que nos ajudaram a planejar e aprimorar este trabalho:

- Artigo resumido (*short paper*) intitulado “Experiência bem-sucedida de adoção de Métodos Ágeis em uma Empresa Pública de Tecnologia da

[14] Disponível em http://www.dainf.ct.utfpr.edu.br/~adolfo/Pesquisa/Agile/VagnerLima/metricstocsv.html

Informação e Comunicação: um relato preliminar" aceito no 4º Workshop Brasileiro de Método Ágeis, ou WBMA´2013 (LIMA et al., 2013);

- Relatório técnico intitulado "*AGILE TOUR* 2012 CURITIBA: dados sobre a adoção de Tecnologias de Desenvolvimento de Software e a relação Empresa e Pesquisas Científicas" (LIMA et al., 2013r);
- Artigo intitulado "Investigação experimental e Práticas ágeis: ameaças à validade de experimentos envolvendo a prática ágil Programação em Par" aceito e apresentado no WBMA´12 (LIMA et al., 2012);
- Resumo intitulado "Pair Programming: investigando sua eficácia perante tarefas de modelagem e construção de software" aceito e publicado na IV Mostra de Pesquisa e Pós-Graduação da UTFPR/Curitiba, ou MOPP´2012 (LIMA et al., 2012r).

Como trabalho futuro sugere-se replicar essa pesquisa experimental da maneira como está, e/ou investigar tarefas de modelagem onde o resultado esperado e avaliado é um modelo UML, por exemplo, diagrama de classes, ou até uma especificação de caso de uso. Entretanto, vale a ressalva da grande dificuldade para realizar essa pesquisa em uma empresa e até mesmo nas universidades. Um caminho é buscar professores/disciplinas onde não há muito conteúdo a ser ministrado – em geral, "disciplinas de acompanhamento" das atividades dos alunos, isso no âmbito de graduação. No âmbito de pós-graduação, onde em geral há profissionais de software matriculados, buscar por professores/disciplinas ditas como "tópicos/estudos avançados" que possuem um formato diferenciado. Nestas disciplinas as pesquisas poderiam ser realizadas como se fosse um "seminário" e/ou "módulo" – dependendo da duração da pesquisa. E, por fim, nas empresas deve-se buscar apoio no mínimo do gestor/superior imediato e, na sequência, com o planejamento/projeto da pesquisa pronto, negociar junto ao RH da empresa a realização da pesquisa dentro da empresa – ou no laboratório da universidade – em formato de "treinamento prático" – em geral, de curta duração.

6. REFERÊNCIAS

ANICHE, M. F. **Como a prática de TDD influencia o projeto de classes em sistemas orientados a objetos**. 2012. 75 f. Dissertação (Mestrado em Ciência da Computação) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

BALIJEPALLY, V.; MAHAPATRA, R.; NERUR, S.; Price, K. H. Are two heads better than one for software development? the productivity paradox of pair programming. **MIS Q.**, v. 33, p. 91-118, mar. 2009.

BANDEIRA, M. **Validade interna e externa de uma pesquisa: vieses**. 2012. Disponivel em: <http://www.ufsj.edu.br/portal2-repositorio/File/lapsam/Texto%204-VALIDADE.pdf>.

BASSI FILHO, D. L. **Experiências com desenvolvimento ágil**. 2008. 170 f. Dissertação (Mestrado em Ciências) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2008.

BECK, K. **Programação Extrema (XP) Explicada:** acolha as mudanças. Porto Alegre: Bookman, 2000.

BECK, K.; ANDRES, C. **Extreme Programming Explained:** Embrace chance. Boston: Addison-Wesley, 2004.

BIPP, T; LEPPER, A.; SCHMEDDING, D. Pair programming in software development teams – An empirical study of its benefits. **Information and Software Technology**, v. 50, n. 3, p. 231–240, Fevereiro. 2008.

BRAUGHT, G.; WAHLS, T.; EBY, L. M. The Case for Pair Programming in the Computer Science Classroom. **Trans. Comput. Educ.**, v. 11, p. 2-21, fev. 2011.

CORBUCCI, H.; GOLDMAN, A; KATAYAMA, E.; KON, F.; MELO, C.; SANTOS, V. Genesis and Evolution of the Agile Movement in Brazil – Perspective from Academia and Industry. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 25., 2011, São Paulo. **Anais…** São Paulo, 2011. p. 98-107.

COSÍN, J. J. D.; SANZ, L. F. **Medición para la gestión em la Ingeniería del Software**. Madrid: RA-MA, 2000.

DELAMARO, M. E.; JINO, M.; MALDONADO, J. C. **Introdução ao Teste de Software.** Rio de Janeiro: Campus, 2007.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and Software Technology**, v. 50, n. 9-10, p. 833-859, ago. 2008.

DYBÅ, T.; DINGSØYR, T. What Do We Know about Agile Software Development ? **IEEE Software**, v. 26, n. 5, p. 6-9, 2009.

ECLIPSE METRICS PLUGIN. 2012. Disponível em: <http://sourceforge.net/projects/metrics/ >.

FARIA, E. S. J. de. **Método trifásico de ensino-aprendizagem baseado na taxionomia de objetivos educacionais de Bloom**: uma aplicação no ensino de programação de computadores. 2010. 295 f. Tese (Doutorado) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2010.

FARIA, E. S. J. de; YAMANAKA, K. Programação em Duplas: Estado da Arte. **Revista Ciências Exatas e Naturais**, v. 12, n. 1, p. 145-182, jan./jun. 2010.

FLICK, U. **Introdução à pesquisa qualitativa.** 3. ed. Porto Alegre: Artmed, 2009.

HANKS, B.; MCDOWELL, C.; DRAPER, D.; KRNJAJIC, M. Program quality with pair programming in CS1. **Proceedings…** ITiCSE'04. New York, NY, USA: ACM, 2004.

HAYWARD, L. J. **Students' Perceptions of Distributed Pair Programming in an Upper-Level Undergraduate Software Engineering Course**. 2009. 88 f. Dissertação (Mestrado) - Graduate Faculty of North Carolina State University, Raleigh, North Carolina, 2008.

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. **Proceedings…** Frontiers in Education, 34th annual. 2004.

HÖST, M., REGNELL, B. e WOHLIN, C. Using students as subjects - a comparative study of students and professionals in lead-time impact assessment. In: **Empirical Software Engineering**, v. 5, n. 3, p. 201-214, 2000.

KOSCIANSKI, A.; SOARES, M. S. **Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** 2. ed. São Paulo: Novatec, 2007.

LARMAN, C. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**, 3. ed. Porto Alegre: Bookman, 2007.

LIMA, V. C. M.; SECA NETO, A. G. S. e EMER, M. C. F. P. Investigação experimental e práticas ágeis: ameaças à validade de experimentos envolvendo a prática ágil Programação em par. **Anais** do 3º Workshop Brasileiro de Métodos Ágeis, WBMA´2012, 2012. Disponível em: <http://www.ime.usp.br/~kon/wbma2012.pdf>.

_____. **Pair Programming:** investigando sua eficácia perante tarefas de modelagem e construção de software. IV Mostra de Pesquisa e Pós-Graduação da UTFPR/Curitiba, MOPP´2012, 2012r. Disponível em: <http://www.utfpr.edu.br/curitiba/estrutura-universitaria/diretorias/dirppg/eventos/mopp/2012/bannervagner.pdf/at_download/file>.

_____. **Experiência bem-sucedida de adoção de Métodos Ágeis em uma Empresa Pública de Tecnologia da Informação e Comunicação:** um relato preliminar. 4º Workshop Brasileiro de Métodos Ágeis, WBMA´2013. Disponível em: < http://vagnercml.files.wordpress.com/2013/10/wbma2013_experienciaagil_empresapublicatic_limasecanetoemer.pdf>

_____. **AGILE TOUR 2012 CURITIBA:** dados sobre a adoção de Tecnologias de Desenvolvimento de Software e a relação Empresa e Pesquisas Científicas. PPGCA/UTFPR. Curitiba. Relatório técnico. 2013, 2013r. Disponível em: <http://www.dainf.ct.utfpr.edu.br/~adolfo/Pesquisa/Agile/VagnerLima/PPGCA_2013_RelTecnico_ESW_EmpPesq_Lima_SecaNeto_Emer.pdf>.

MADEYSKI, L. The impact of pair programming and test-driven development on package dependencies in object-oriented design - an experiment. **Proceedings…** PROFES'06. Berlin, Heidelberg: Springer-Verlag, 2006.

MAFRA, S. N. e TRAVASSOS, G. H. **Estudos Primários e Secundários apoiando a busca por Evidência em Engenharia de Software**. COPPE/UFRJ. Rio de Janeiro. Relatório técnico: RT-ES 687/06. 2006.

MCCABE, T. A Complexity Measure. **IEEE Transactoins of Software Engineering** SE-2 (4): 308-320, 1976.

MEIRELLES, P. R. M. **Levantamento de Métricas de Avaliação de Projetos de Software Livre.** Centro de Competência em Software Livre do Departamento de Ciência da Computação do IME/USP. São Paulo: Relatório técnico. 2008. Disponível em < http://ccsl.ime.usp.br/files/relatorioPauloMeirelles_final.pdf>.

MELO, C.; SANTOS, V. A.; CORBUCCI, H.; KATAYAMA, E.; GOLDMAN, A.; KON, F. **Métodos ágeis no Brasil: estado da prática em times e organizações.** Departamento de Ciência da Computação do IME/USP. São Paulo: Relatório técnico: RT-MAC-2012-03. 2012.

NAWROCKI, J.; WOJCIECHOWSKI, A. **Experimental Evaluation of Pair Programming**. 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.1689>.

NOSEK, J. T. The Case for Collaborative Programming. In: Communications of the **ACM**. 1998, p. 105-108.

PAIR LEARNING: THE USE OF PAIR PROGRAMMING IN EDUCATION. **Fun with Pair Programming.** 2008. Disponível em: < http://www.realsearchgroup.org/pairlearning/worksheet.pdf>.

POPPENDIECK, M.; POPPENDIECK, T. **Implementando o Desenvolvimento Lean de Software:** Do Conceito ao Dinheiro. 1. ed. Porto Alegre: Bookman, 2011.

PRESTON, D. Using collaborative learning research to enhance pair programming pedagogy. **ACM SIGITE Newsletter**, v. 3, n. 1, p. 16–21, jan. 2006.

ROCHA, A. R. C. da; MALDONADO, J. C.; WEBER, K. C. **Qualidade de Software:** teoria e prática. São Paulo: Prentice Hall, 2001.

SALLEH, N.; MENDES, E.; GRUNDY, J.; BURCH, G. An empirical study of the effects of personality in pair programming using the five-factor model. **Proceedings…** 3rd International Symposium on Empirical Software Engineering and Measurement, 2009.

SALLEH, N.; MENDES, E.; GRUNDY, J. Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. **IEEE Transactions on Software Engineering**, v. 37, n. 4, p. 509-525, ago. 2011.

SATO, D. T. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. 2007. 155 f. Dissertação (Mestrado em Ciências) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2007.

SAWICKI, S. **Projeto cooperativo no Ambiente Cave baseado em espaço compartilhado de objetos**. 2002. 156 f. Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

SFETSOS, P.; STAMELOS, I.; ANGELIS., L.; DELIGIANNIS, I. An experimental investigation of personality types impact on pair effectiveness in pair programming. **Empirical Software Engineering**, v. 14, p. 187-226, nov. 2008.

SFETSOS, P.; STAMELOS, I. Empirical Studies on Quality in Agile Practices: A Systematic Literature Review. **Proceedings…** Quality of Information and Communications Technology (QUATIC), 2010.

SHORE, J.; WARDEN, S. **A Arte do Desenvolvimento Ágil.** Tradução: Bianca Capitânio. Rio de Janeiro: Alta Books, 2008.

SILVA, A. F. da. **Reflexões sobre o ensino de metodologias ágeis na academia, na indústria e no governo**. 2007. 151 f. Dissertação (Mestrado em Ciências) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2007.

TELES, V. M. **Extreme Programming:** aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec, 2004.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. **Introdução à Engenharia de Software Experimental.** COPPE/UFRJ. Rio de Janeiro. Relatório técnico: RT-ES-590/02. 2002.

VANHANEN, J.; LASSENIUS, C. Effects of pair programming at the development team level: an experimente. **Proceedings...** International Symposium on Empirical Software Engineering, 2005.

VERSIONONE. **State of Agile Development.** 2012. Disponível em: <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf>.

WAINER, J. **Métodos de pesquisa quantitativa e qualitativa para a Ciência da Computação.** 2007. Disponível em: <http://www.ic.unicamp.br/~wainer/papers/metod07.pdf>.

WELLS, D. **Acceptance Tests**. 1999. Disponível em: <http://www.extremeprogramming.org/rules/functionaltests.html>.

WILLIAMS, L. Integrating Pair Programming into a Software Development Process. **Proceedings...** CSEET '01.Washington, DC, USA: IEEE Computer Society, 2001.

WILLIAMS, L. A.; KESSLER, R. R. Experimenting with Industry's "Pair-Programming" Model in the Computer Science Classroom. **Journal on Software Engineering Education**, 2000a.

WILLIAMS, L. e KESSLER, R. R. All I Really Need to Know about Pair Programming I Learned In Kindergarten. In: **Communications of the ACM**. 2000b.

WILLIAMS, L.; KESSLER, R. R.; CUNNINGHAM, W.; JEFFRIES, R. Strengthening the case for pair programming. **IEEE Software**, v. 17, n. 4, p. 19-25, ago. 2000.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B. e WESSLÉN, A. **Experimentation in Software Engineering**. Springer, 2012.

XU, S.; Rajlich, V. Empirical Validation of Test-Driven Pair Programming in Game Development. **Proceedings...** ICIS/COMSAR´2006, IEEE Computer Society, 2006.

7. APÊNDICE

APÊNDICE A – QUESTIONÁRIO A: LEVANTAMENTO INICIAL

Descrição inicial

Apresentação da pesquisa
(consentimento de participação)

Este questionário faz parte de uma pesquisa de mestrado do Programa de Pós-graduação em Computação Aplicada (PPGCA) do DAINF/UTFPR. A pesquisa é de cunho estritamente científico e é sobre práticas de desenvolvimento de software. Ela envolve responder a dois questionários e realizar uma atividade treino e um exercício de desenvolvimento de software.

Hoje você será convidado a preencher um dos questionários (Questionário A) e realizar as duas atividades de desenvolvimento de software. Essas atividades tem duração de duas aulas (aproximadamente 1h40). O segundo questionário (Questionário B) será enviado por e-mail, podendo ser preenchido em outro momento.

Pré-requisitos
- Ter programado ou ter conhecimento básico em Java;
- Ter utilizado alguma IDE de desenvolvimento (Eclipse ou Visual Studio, por exemplo);
- Estar disposto a realizar um trabalho colaborativo, porque talvez tenha que trabalhar em dupla;

Por que participar?
- Para contribuir com a pesquisa científica na área da Computação, especialmente na área da Engenharia de Software;
- Para ter a oportunidade de aplicar técnicas de Engenharia de Software que são adotadas por empresas do mercado de software;

Importante
Os pesquisadores se comprometem a:
- Não divulgar, em qualquer meio, os dados ou informações pessoais ou acadêmicos dos participantes;
- Divulgar, em formato de dissertação, artigos e apresentações, apenas os dados agregados (ou resumidos), de maneira que não se possa retirar ou inferir a identificação de qualquer participante.

Questões

Id	Questão/afirmativa	Considerações sobre a resposta
QA1	Idade	Resposta aberta.
QA2	Gênero	Opções: 1) Masculino; 2) Feminino.
QA3	Eu gosto de trabalhar com outras pessoas para alcançar uma meta	Likert de 5 pontos: de “discordo totalmente” a “concordo totalmente”.
QA4	Eu gosto de resolver problemas do tipo onde é dado o ‘quê’ é esperado pelo cliente/usuário e há liberdade por parte do desenvolvedor em definir o ‘como’	Likert de 5 pontos: de “discordo totalmente” a “concordo totalmente”.
QA5	Programação em Par (ou Pair Programming): abaixo a definição dessa prática de desenvolvimento. Os desenvolvedores trabalham lado-a-lado em único computador, trocando periodicamente quem codifica (piloto) e quem revisa e acompanha (navegador) continua e ativamente o trabalho desenvolvido – por exemplo, pensando e sugerindo novas ideias ou estratégias. Considerando a definição anterior, avalie a seguinte afirmação: *Eu gostaria de desenvolver um software utilizando a prática de Programação em Par.*	Likert de 5 pontos: de “discordo totalmente” a “concordo totalmente”.
QA6	Histórias: abaixo a definição dessa prática de desenvolvimento. Desenvolver o software a partir de unidades de funcionalidade "visíveis ao cliente". As funcionalidades são brevemente descritas pelo cliente idealmente em cartões de papel, que podem ser manuseados facilmente por ele mesmo e pelos desenvolvedores. Nelas pode haver definição de regras de negócio e/ou testes de aceitação (o cliente/usuário valida se o software/funcionalidade está fazendo aquilo que ele espera). Considerando a definição anterior, avalie a seguinte afirmação: *Eu gostaria de desenvolver um software utilizando a prática Histórias.*	Likert de 5 pontos: de “discordo totalmente” a “concordo totalmente”.

APÊNDICE B – QUESTIONÁRIO B: LEVANTAMENTO PÓS-ATIVIDADES DE DESENVOLVIMENTO

Questões

Id	Questão/afirmativa	Considerações sobre a resposta
QB1	Você desenvolveu em dupla ?	Opções: 1) Sim; 2) Não.
[sim, desenvolvi em dupla]		
Desconsiderar a atividade treino (calculadora...) quando estiver respondendo as questões a seguir.		
QB2	Eu gostei de desenvolver o software a partir da prática Histórias. *[texto lembrete/ajuda] Prática Histórias: desenvolver o software a partir de unidades de funcionalidade "visíveis ao cliente" [...] Nelas pode haver definição de regras de negócio e/ou testes de aceitação.*	Likert de 5 pontos: de "discordo totalmente" a "concordo totalmente".
QB3	Meu (ou minha) colega ajudou a me manter focado no desenvolvimento do software.	Likert de 5 pontos: de "discordo totalmente" a "concordo totalmente".
QB4	Avalie quanto compatível foi trabalhar com o seu colega.	Opções: 1) "Incompatível / A dupla 'não funcionou'"; 2) "OK / 'Sem maiores problemas/dificuldades de interação'"; 3) "Muito compatível"
QB5	Eu gostei de utilizar a Programação em Par para resolver o tipo de problema apresentado. *[texto lembrete/ajuda] Tipo de problema: é dado o "quê" é esperado pelo cliente/usuário e há liberdade por parte do desenvolvedor em definir o "como".*	Likert de 5 pontos: de "discordo totalmente" a "concordo totalmente".
QB6.a	Avaliação da adoção das regras e diretrizes da prática Programação em Par. Para as duas questões apresentadas a seguir considere as seguintes referências: NOTA"0"-> não conseguiu seguir NENHUMA das regras e diretrizes da prática Programação em Par em NENHUMA das tarefas realizadas durante o desenvolvimento do software; NOTA "10"-> conseguiu seguir praticamente TODAS as regras e diretrizes da prática Programação em Par em TODAS as tarefas realizadas durante o desenvolvimento do software; *a) Dê uma nota de 0 a 10 para o seu desempenho (auto avaliação).*	Opções: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
QB6.b	*[continuação da questão anterior]* *b) Dê uma nota de 0 a 10 para desempenho do seu colega.*	Opções: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
QB7	Por favor, liste quaisquer benefícios ou desvantagens da prática Programação em Par.	Resposta aberta.
[não desenvolvi em dupla]		
Desconsiderar a atividade treino (calculadora...) quando estiver respondendo as questões a seguir.		
QB8	Eu gostei de resolver o tipo de problema apresentado. *[texto lembrete/ajuda] Tipo de problema: é dado o "quê" é esperado pelo cliente/usuário e há liberdade por parte do desenvolvedor em definir o "como".*	Likert de 5 pontos: de "discordo totalmente" a "concordo totalmente".
QB9	Eu gostei de desenvolver o software a partir da prática Histórias. *[texto lembrete/ajuda] Prática Histórias: desenvolver o software a partir de unidades de funcionalidade "visíveis ao cliente" [...] Nelas pode haver definição de regras de negócio e/ou testes de aceitação.*	Likert de 5 pontos: de "discordo totalmente" a "concordo totalmente".

APÊNDICE C – QUESTIONÁRIO DE AVALIAÇÃO E SELEÇÃO DE EXEMPLOS DE SOFTWARE

Descrição

Com base na definição e planejamento do experimento apresentado anteriormente, pede-se que o avaliador preencha a lista apresentada a seguir.

Obrigado!

Lista de Exemplos de Software

Por favor, considere os seguintes valores para o preenchimento do campo 'Preferência':

1 – para o melhor exemplo de software
2 – para o segundo melhor exemplo de software
3 – para o terceiro melhor exemplo de software

Ou, caso considere nenhum software pertinente ao contexto do experimento, utilize o campo 'Comentários e/ou sugestões' ao final do questionário para deixar suas considerações.

Id	Titulo	Descrição	Preferência
1	Sistema de locação de filmes	Nesse caso há uma especificação simplificada com algumas funcionalidades, o foco neste caso era manutenção, logo, aqui é necessário elaborar a especificação. Resumo da tarefa: prover uma forma de manter uma lista de filmes e uma forma de mostrar um filme dado o seu titulo. Assume-se que o titulo do filme é único. Fonte: Artigo de Balijepally et al. (20009).	
2	Clube do DVD	Nesse caso há uma especificação, inclusive uma lista de funcionalidades. *Construir um software para controlar empréstimos de DVDs de um grupo de amigos que trabalham numa certa empresa. No Clube do DVD, qualquer pessoa pode participar como sócio, oferecendo seus DVDs particulares para o grupo e emprestando DVDs disponibilizados pelos outros colegas. No Clube do DVD são permitidos apenas DVDs originais.* Fonte: Trabalho final da disciplina de Análise Orientada a Objetos com UML da Pós-graduação em Informática da UFPR/2008.	

3	Estudantes e Repúblicas	Nesse caso não há especificação, logo, aqui é necessário elaborar a especificação. *Questão 23:* *Considere o esquema de banco de dados relacional apresentado a seguir, formado por 4 relações, que representa o conjunto de estudantes de uma universidade que podem, ou não, morar em repúblicas (moradias compartilhadas por estudantes). A relação Estudante foi modelada como um subconjunto da relação Pessoa. Considere que os atributos grifados correspondam à chave primária da respectiva relação e os atributos que são seguidos da palavra referencia sejam chaves estrangeiras.* *Pessoa(IdPessoa:integer, Nome:varchar(40), Endereco:varchar(40))* *FonePessoa(IdPessoa:integer referencia Pessoa, DDD:varchar(3), Prefixo:char(4), Nro:char(4))* *Republica(IdRep:integer, Nome:varchar(30), Endereco:varchar(40))* *Estudante(RA:integer, Email:varchar(30), IdPessoa:integer referencia Pessoa, IdRep:integer referencia Republica)* Fonte: Prova de Ciência da Computação do ENADE Computação 2008.	
4	Sistema de Contas Bancárias	Nesse caso não há especificação, há apenas o diagrama de classe com 4 classes de negócio: Banco, Agência, Cliente e Conta, logo, aqui é necessário elaborar a especificação. *Questão 80 – Discursiva.* Fonte: Prova de Sistemas de Informação do ENADE Computação 2008.	
5	Publicação acadêmica	Nesse caso não há especificação, logo, aqui é necessário elaborar a especificação. *Questão 46.* *Em um modelo de dados que descreve a publicação acadêmica de pesquisadores de diferentes instituições em eventos acadêmicos, considere as tabelas abaixo.* *DEPARTAMENTO (#CodDepartamento, NomeDepartamento)* *EMPREGADO (#CodEmpregado, NomeEmpregado, CodDepartamento, Salario)* *[...]* Fonte: Prova de Ciência da Computação do ENADE Computação 2011	
6	Sistema de venda de produtos nacionais e importados	Nesse caso não há especificação, há apenas um diagrama de caso de uso, logo, aqui é necessário elaborar a especificação. *Questão 48.* Fonte: Prova de Ciência da Computação do ENADE Computação 2011	

Comentários e/ou sugestões

_____
_____
_____
_____
_____
_____
_____
_____
_____

APÊNDICE D – RESPOSTAS DO QUESTIONÁRIO DE AVALIAÇÃO E SELEÇÃO DE EXEMPLOS DE SOFTWARE

Descrição

Abaixo as respostas dos pesquisadores que formaram a banca avaliadora do Seminário de Acompanhamento do PPGCA/UTFPR.

Id	Titulo	Avaliador 1	Avaliador 2	Avaliador 3	Avaliador 4
1	Sistema de locação de filmes	2			
2	Clube do DVD	3			
3	Estudantes e Repúblicas	1	3		
4	Sistema de Contas Bancárias		1		
5	Publicação acadêmica		2		
6	Sistema de venda de produtos nacionais e importados				
Comentários e/ou sugestões		Nenhum	Nenhum		

Ocorrências

Dois pesquisadores não preencheram os questionários.

Resultado

Não foi possível identificar um exemplo de software de aplicação que fosse consenso entre os pesquisadores.

APÊNDICE E – DESCRIÇÃO DO EXERCÍCIO DE DESENVOLVIMENTO DE SOFTWARE

Agenda de Contatos

(Agenda Emergência)

O software em questão deve permitir a manutenção (ou gestão) de uma lista de contatos pessoais (nome e telefone). As funcionalidades (histórias) e respectivas regras de negócio (RN) e testes de aceitação estão descritos a seguir.

Restrições ou diretrizes gerais para o desenvolvimento do software:

- As informações manipuladas pela aplicação existirão enquanto o software estiver em execução. Ou seja, as informações ficaram em memória, nenhum dado deverá ser armazenado em disco (arquivo, banco de dados,...).
- O teste de aceitação retrata o comportamento da aplicação/funcionalidade que é esperado pelo cliente. Ele deve ser usado pelo desenvolvedor(es) para validar a funcionalidade implementada.
    - Os testes classificados como INSUCESSO são testes onde o cliente espera algum tipo de aviso por parte da aplicação, o qual é especificado na seção “Telas/Layouts para cada teste de aceitação...”.
- A interface com o usuário deve ser semelhante aos layouts definidos para cada teste de aceitação. Para isso, deve-se usar essencialmente o mesmo mecanismo que foi utilizado na atividade treino. Ou seja, não é necessário usar HTML, JSP, JSF, Swing, SWT, ou algo do gênero!
- A meta de desenvolvimento é o software como um todo, mas o objetivo inicial é a história 1 (incluir contato). Ela é prioritária para o cliente. Após finalizar essa história, deve-se desenvolver a história 2 (listar contatos) e, por fim, a história 3 (excluir contato).
- O objetivo é um “código de produção”, seguindo os conceitos/princípios da Programação Orientado a Objetos. Ou seja, uma aplicação (ou parte dela) que, uma vez finalizada, poder ser disponibilizada ao cliente e que é de fácil manutenção por parte de outros desenvolvedores!

- Por fim, a aplicação será executada por meio do método *main* da classe Executar que está dentro do pacote principal/src. Essa é única estrutura que obrigatoriamente deverá ser mantida, acrescentar ou não novo “pacotes” fica a cargo do desenvolvedor(es).

Quando considerar finalizado uma história (funcionalidade):

1) Todos os testes de aceitação estão “passando” (ou seja, a funcionalidade apresenta todos os resultados especificados pelo cliente nos testes de aceitação).

História 1 – Incluir contato

Como usuário da agenda gostaria de poder incluir uma pessoa (nome e número de telefone) na lista de contatos.

Regras:
RN2: O nome e o número de telefone são informações obrigatórias.
RN3: O número de telefone informado deve ter pelo menos 8 dígitos.
RN4: O número de telefone informado deve ser um valor numérico.
RN5: O novo contato deve ser inserido no final da lista.
RN6: O número de telefone informado para um novo contato deve ser diferente dos números que estão cadastrados.

Testes de aceitação (a seguir neste documento, a sequência de telas/layouts para cada teste):

a) [sucesso] Abrir a aplicação, visualizar a lista vazia, selecionar incluir contato, incluir uma pessoa (nome e número) e, em seguida, visualizar a lista com esse novo contato.
b) [sucesso] Continuando o teste anterior, incluir mais um contato com nome e número de telefone.
c) [insucesso] Ainda com os dados dos testes anteriores, incluir um contato sem nome e número.
d) [insucesso] Abrir a aplicação, visualizar a lista vazia, selecionar incluir contato, incluir uma pessoa sem nome e com o número de telefone com a quantidade incorreta de dígitos.
e) [insucesso] Abrir a aplicação, visualizar a lista vazia, selecionar incluir contato, incluir uma pessoa com nome, mas com o número de telefone com a quantidade de dígitos correta, porém não é um valor numérico.
f) [insucesso] Abrir a aplicação, visualizar a lista vazia, selecionar incluir contato, incluir duas pessoas com nome e número de telefones diferentes e, na sequência, incluir uma pessoa com número de telefone que já está cadastrada na lista de contatos.

História 2 – Listar contatos

Como usuário da agenda gostaria de visualizar a lista de contatos já cadastrados para saber qual eu devo incluir ou excluir.

Regras:

RN1: A quantidade máxima de contatos é igual 5.

Testes de aceitação (a seguir neste documento, a sequência de telas/layouts para cada teste):

a) *[sucesso]* Abrir a aplicação e visualizar a lista vazia.
b) *[sucesso]* Abrir a aplicação e visualizar como ficaria a lista com 5 contatos cadastrados.

História 3 – Excluir contato

Como usuário da Agenda Emergência gostaria de poder excluir uma pessoa da lista de contatos utilizando o seu número de telefone.

Regras:
RN7: Para excluir um contato deve ser informado um número de telefone, respeitando as RN3 e RN4.
RN8: Não é necessário informar que o contato foi excluído ou que não foi encontrado na lista de contato.

Testes de aceitação (a seguir neste documento, a sequência de telas/layouts para cada teste):

a) *[sucesso] Abrir a aplicação, visualizar a lista vazia, incluir duas pessoas com nome e número de telefones diferentes e, na sequência, excluir uma delas informando o número de telefone.*
b) *[insucesso] A partir do resultado do teste anterior, informar um número de telefone com a quantidade incorreta de dígitos e com um valor não numérico.*
c) *[sucesso] Abrir a aplicação, visualizar a lista vazia, informar um número de telefone e tentar excluir um contato.*

Telas/Layouts para cada teste de aceitação da história 1 – incluir contato

a) *[sucesso]* Abrir a aplicação, visualizar a lista vazia, *selecionar incluir contato,* incluir uma pessoa (nome e número) e, em seguida, visualizar a lista com esse novo contato.

- Abrir a aplicação, visualizar a lista vazia, incluir uma pessoa (nome e número)

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: xxxxxxxxx <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato 1
Número: 99999999 <enter>
```

- Visualizar a lista com o novo contato

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

b) *[sucesso]* Continuando o teste anterior, incluir mais um contato com nome e número de telefone.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

- Resultado final/esperado após incluir mais um contato com nome e número de telefone.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

c) *[insucesso]* Ainda com os dados dos testes anteriores, incluir um contato sem nome e número.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ? I <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: <enter>
Número: <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome:
Número:
## AVISO: A informação Nome é obrigatória. A informação Número é obrigatória.
(Pressione ENTER para continuar...)
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (88888888)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

d) *[insucesso]* Abrir a aplicação, visualizar a lista vazia, *selecionar incluir contato,* incluir uma pessoa sem nome e com o número de telefone com a quantidade incorreta de dígitos.

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome:
Número: 9999 <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome:
Número: 9999
## AVISO: A informação Nome é obrigatória. O número de telefone deve ter no mínimo 8 dígitos.
(Pressione ENTER para continuar...)
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

e) *[insucesso]* Abrir a aplicação, visualizar a lista vazia, *selecionar incluir contato,* incluir uma pessoa com nome, mas com o número de telefone com a quantidade de dígitos correta, porém não é um valor numérico.

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato xyz <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato xyz
Número: 999X9999 <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato xyz
Número: 999X9999
## AVISO: A informação Número deve ser um valor numérico.
(Pressione ENTER para continuar...)
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

f) *[insucesso]* Abrir a aplicação, visualizar a lista vazia, *selecionar incluir contato,* incluir duas pessoas com nome e número de telefones diferentes e, na sequência, incluir uma pessoa com número de telefone que já está cadastrada na lista de contatos.

- Abrir a aplicação, visualizar a lista vazia, incluir duas pessoas com nome e número de telefones diferentes.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ? I <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato 3 <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato 3
```

```
Número: 99999999 <enter>
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ? I
Nome: Contato 3
Número: 99999999
## AVISO: A informação Número já está cadastrada.
(Pressione ENTER para continuar...)
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

Telas/Layouts para cada teste de aceitação da história 2 – listar contatos

a) *[sucesso]* Abrir a aplicação e visualizar a lista vazia.

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir um contato ou [S]air ?
```

b) *[sucesso]* Abrir a aplicação e incluir e visualizar como ficaria a lista com 5 contatos cadastrados.

```
# Agenda Emergência #
Lista de Contatos
====================
Nome contato 1 (99999999)
Nome contato 2 (88888888)
Nome contato 3 (77777777)
Nome contato 4 (11111111)
Nome contato 5 (0414599999999)
====================
Deseja [I]ncluir um contato ou [S]air ?
```

Telas/Layouts para cada teste de aceitação da história 3 – excluir contato

a) *[sucesso]* Abrir a aplicação, visualizar a lista vazia, incluir duas pessoas com nome e número de telefones diferentes e, na sequência, excluir uma delas informando o número de telefone.

- Abrir a aplicação, visualizar a lista vazia, incluir duas pessoas com nome e número de telefones diferentes.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
Contato 2 (98989898)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E
Número: 98989898 <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

b) [insucesso] A partir do resultado do teste anterior, informar um número de telefone com a quantidade incorreta de dígitos e com um valor não numérico.

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E
Número: 989AA <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E
Número: 989AA
## AVISO: O número de telefone deve ter no mínimo 8 dígitos. A informação Número deve ser um valor numérico.
(Pressione ENTER para continuar...)
```

```
# Agenda Emergência #
Lista de Contatos
====================
Contato 1 (99999999)
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

c) *[sucesso]* Abrir a aplicação, visualizar a lista vazia, informar um número de telefone e tentar excluir um contato.

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ? E
Número: 8888777 <enter>
```

```
# Agenda Emergência #
Lista de Contatos
====================
====================
Deseja [I]ncluir ou [E]xcluir um contato ou [S]air ?
```

APÊNDICE F – CÓDIGO DO EXERCÍCIO DE MANUTENÇÃO DE SOFTWARE (ATIVIDADE TREINO)

Calculadora de somar e subtrair inteiros

Abaixo o código da classe/aplicação treino.

```
package principal;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Executar {

	public static void main(String[] args) throws Exception {

		Integer primeiroNumero = 0;
		Integer segundoNumero = 0;
		Integer soma = 0;
		String valorTeclado = "";

		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

		System.out.println("Calculadora de somar e subtrair Inteiros");
		System.out.println("=======================================");

		System.out.print("Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: ");
		valorTeclado = br.readLine();

		if(valorTeclado.equalsIgnoreCase("1")) {
			System.out.println("Soma de dois números inteiros");
			System.out.println("Informe...");

			System.out.print("	Primeiro Número: ");
			valorTeclado = br.readLine();
			primeiroNumero = Integer.valueOf(valorTeclado);

			System.out.print("	Segundo Número: ");
			valorTeclado = br.readLine();
			segundoNumero = Integer.valueOf(valorTeclado);

			soma = primeiroNumero + segundoNumero;
			System.out.println("Resultado: " + soma);

			System.out.println("(Pressione ENTER para continuar...)");
			br.readLine();

			// "Limpa tela"
			for (int i = 0; i < 20; i++) {
				System.out.println("");
			}
		} else if(valorTeclado.equalsIgnoreCase("2")) {

			System.out.println("");
			System.out.println("## Opção indisponível!");
		} else if(valorTeclado.equalsIgnoreCase("S")) {

			System.out.println("");
		} else {

			System.out.println("");
			System.out.println("## Opção inválida!");
		}

		System.out.println("## Aplicação encerrada!");
	}

}
```

APÊNDICE G – ROTEIRO PARA EXECUÇÃO DA PESQUISA

Roteiro para execução da pesquisa experimental

Passo 0) Baixar o material da pesquisa

Passo 1) Preenchimento do Questionário A

Passo 2) Formação dos grupos e dos pares

Passo 3) Atividade treino (manutenção de software)

- *A partir de agora...*
- *Preparação: Acesso ao ambiente de desenvolvimento*
- *Preparação: Com executar a aplicação treino*
- *Preparação: Acesso a documentação da API Java*
- *Aplicação Treino: Calculadora de somar e subtrair Inteiros*
- *Telas/Layouts para o teste de aceitação da história*

Passo 4) Desenvolvimento do software (exercício de programação)

Passo 0) Baixar o material da pesquisa

1) Informar o endereço onde está o material da pesquisa
2) Orientar o acesso ao material
    a. Salvar e descompactar o arquivo, preferencialmente na área de trabalho (ou em lugar de fácil acesso ou de lembrar sua localização)!
3) Orientar o acesso a esta documentação
    a. Após descompactar o arquivo, acesse o arquivo “inicio.pdf” na pasta “.../eclipse_pesquisa/documentacao”;
    b. Iniciar a partir do PASSO 1.

Observação:

- Tempo previsto: ***5~10 minutos.***

Passo 1) Preenchimento do Questionário A

- O início do preenchimento do questionário pode ser realizado mesmo sem o seu ID em mãos, porém o envio do formulário não, para isso aguarde o pesquisador entregar o seu ID!
- *O questionário está disponível no link abaixo:*
    - < *link para o questionário descrito no Apêndice A no Google Drive* >
- Após preenchê-lo, continue a partir do passo 2 deste documento!

Observação:

- Tempo previsto: ***~10 minutos.***

Passo 2) Formação dos grupos e dos pares

Aguarde orientação do pesquisador :-]

Observação:

- Tempo previsto: ***4~8 minutos.***

Passo 3) Atividade treino (manutenção de software)

Observação:

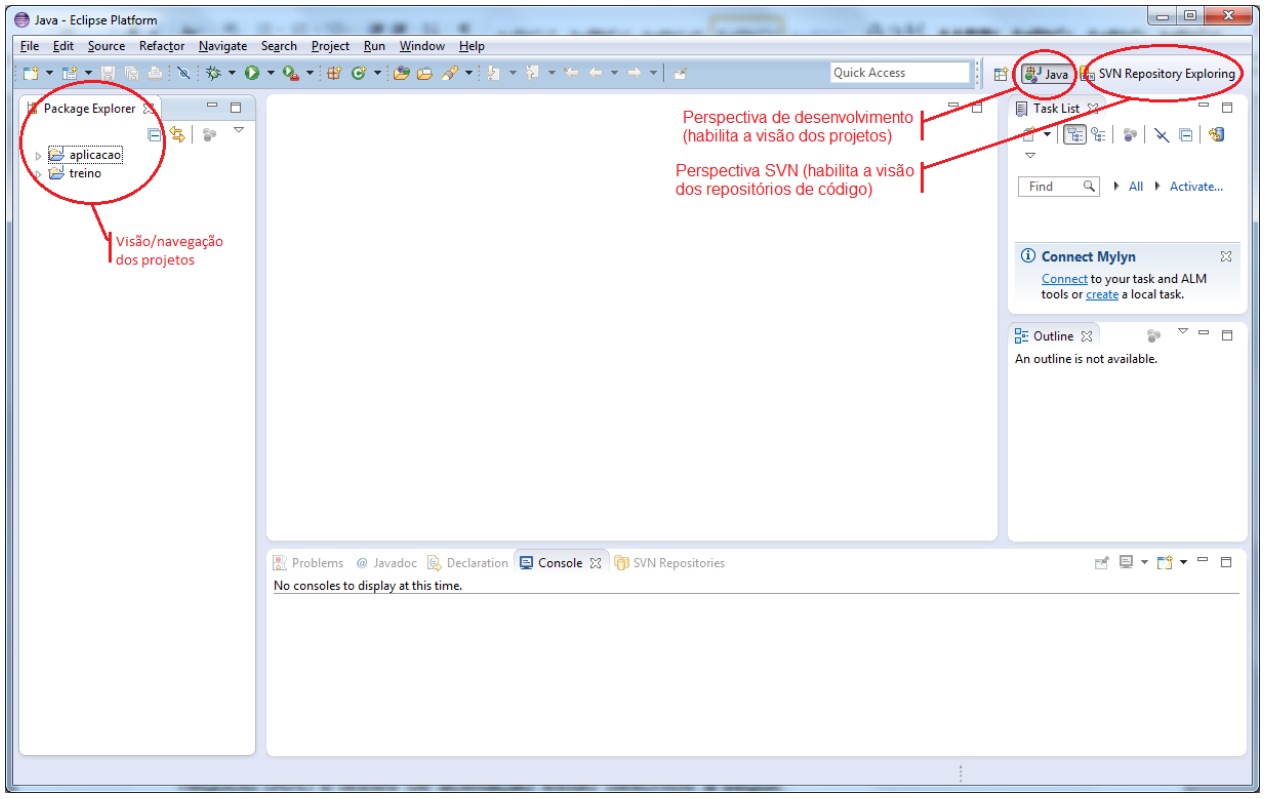
- Tempo previsto: ***10~20 minutos.***

A partir de agora...

- Desligue o celular ou coloque o aparelho no “modo silencioso” - caso esteja esperado alguma ligação importante.
- Não acesse a Internet, tanto pelos computadores do laboratório quanto por qualquer dispositivo eletrônico.
    - A documentação da API Java está disponível localmente e integrada a IDE Eclipse (mais detalhes a seguir).
    - Devem-se usar apenas as ferramentas, documentação e materiais disponibilizados pelo pesquisador;
- Não converse ou discuta sobre a solução dos exercícios com o participante próximo ou dupla próxima de você(s). Cada um ou cada dupla é responsável por sua atividade ou exercício.
- Não coloque nenhuma informação/comentário dentro do código que possa identificar você (nome, sobrenome,...). Se achar pertinente, use somente o ID da pesquisa: aluno[X]!
- E para quem está trabalhando em dupla, leia atentamente o documento *“diretrizes_pp.pdf”* que está na pasta “.../eclipse_pesquisa/documentacao”.

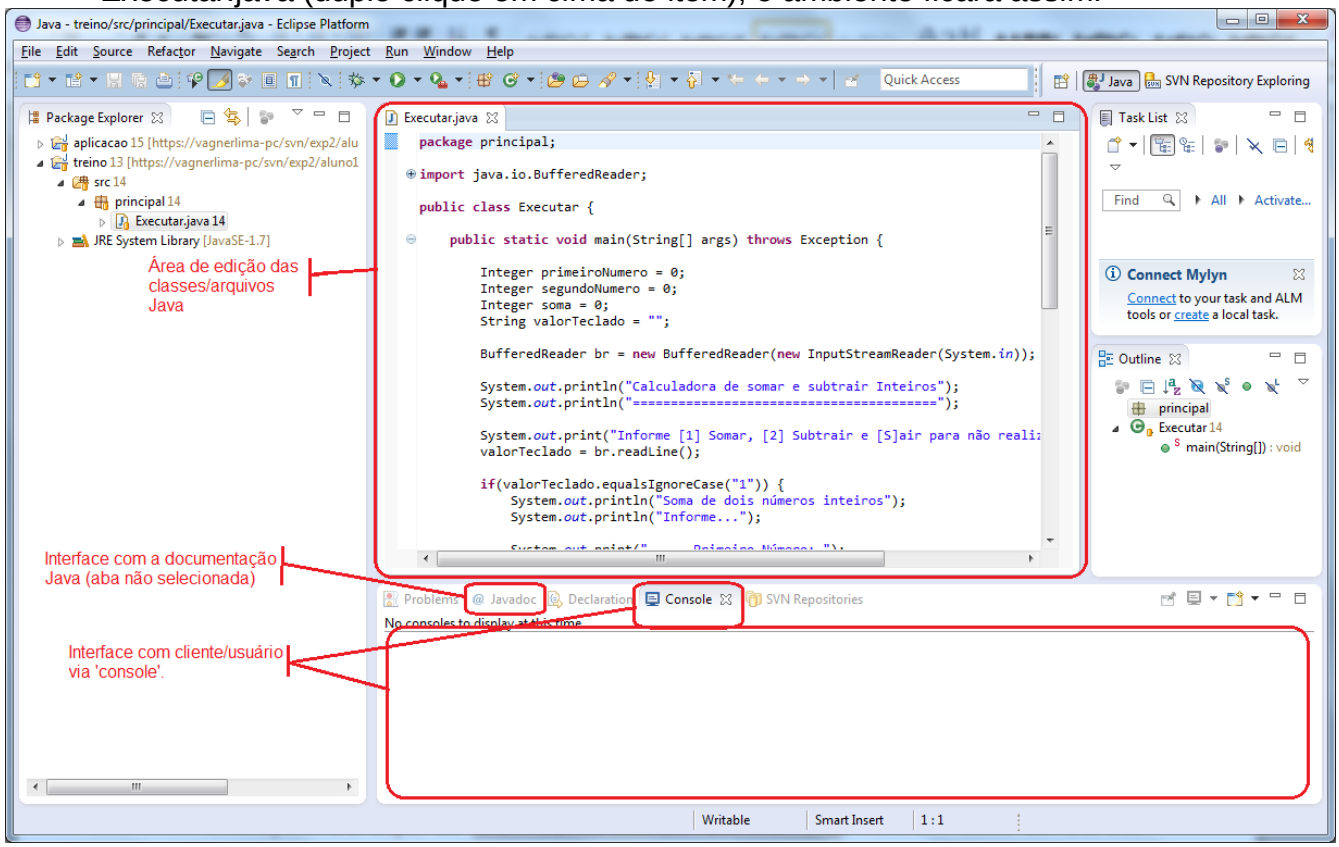
Preparação: Acesso ao ambiente de desenvolvimento

1) Criar link simbólico, aguarde orientação do pesquisador!
2) Dentro da pasta ".../eclipse_pesquisa", execute o arquivo 'eclipse_aplicativo';
3) Navegue pelo botão 'Browse' e selecione uma pasta chamada 'projetos' que está dentro do próprio diretório do eclipse (....\eclipse_pesquisa\projetos);
4) Clique em OK. Após abrir a ferramenta, você deve visualizar algo assim:

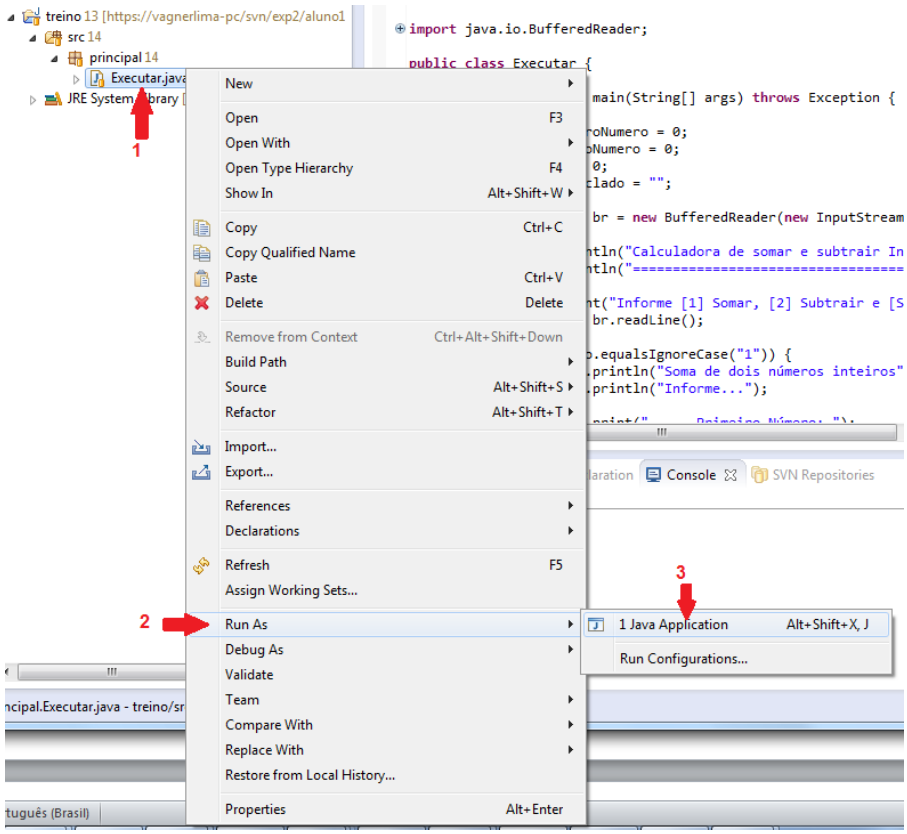


Preparação: Com executar a aplicação treino

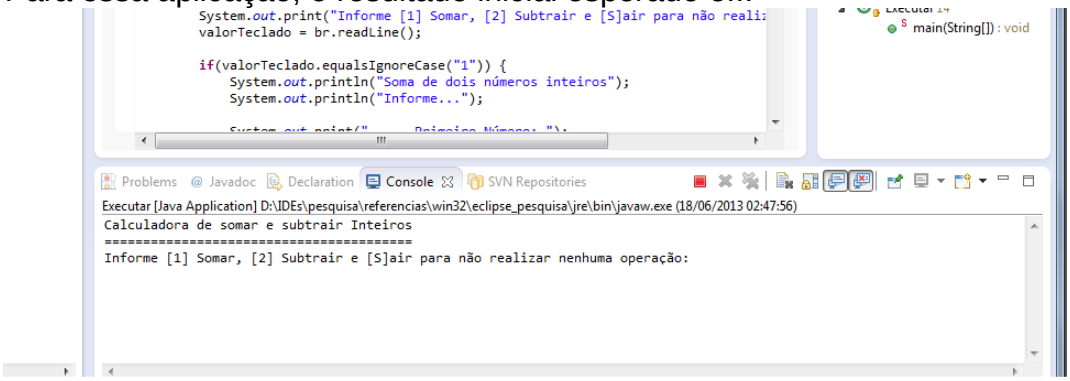
Após expandir a “árvore” itens e subitens do projeto **treino** e abrir a classe Executar.java (duplo clique em cima do item), o ambiente ficará assim:



1) Clique com o botão direito do mouse na classe Executar.java e selecione ‘Run As’ > ‘Java Application’



2) Para essa aplicação, o resultado inicial esperado é...

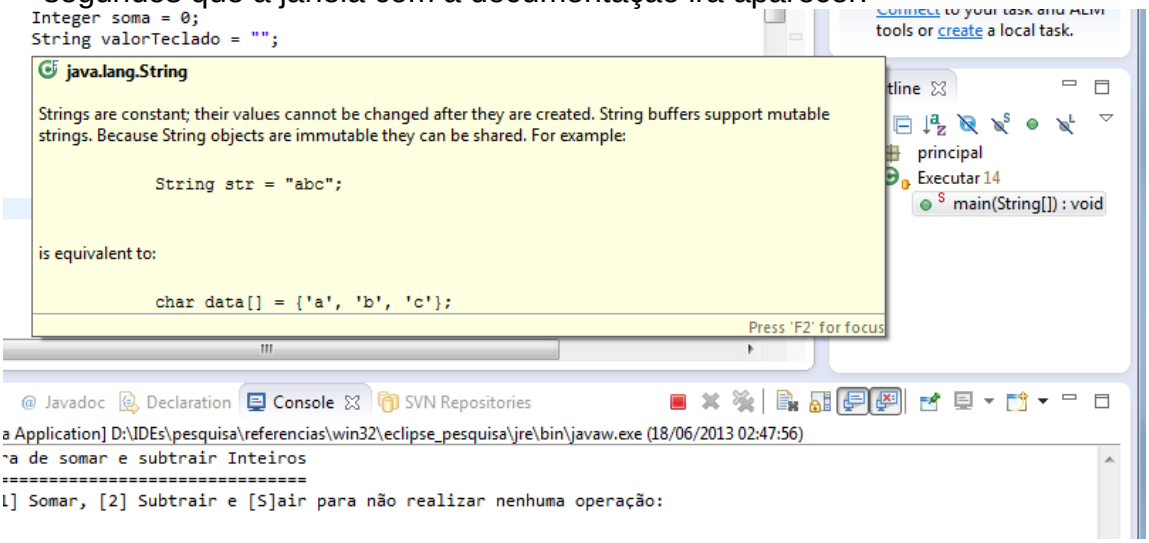


A seguir mais detalhes sobre o funcionamento dessa aplicação treino.

Preparação: Acesso a documentação da API Java

Primeira forma:

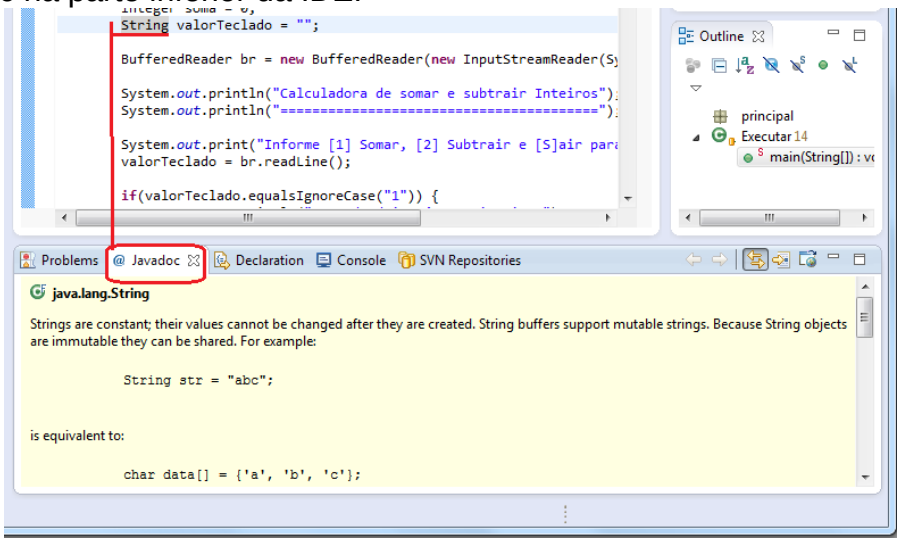
1) Coloque o cursor do mouse em cima da classe ou método e aguarde alguns segundos que a janela com a documentação irá aparecer.



2) E se na sequência pressionar F2, pode-se navegar no conteúdo.

Segunda forma:

1) Coloque o cursor do mouse em cima da classe ou método e selecione a aba Javadoc na parte inferior da IDE.



Aplicação Treino: Calculadora de somar e subtrair Inteiros

O software em questão deve possibilitar a operação de soma e subtração dois valores inteiro.

Na realidade ela está apenas somando dois inteiros e permitindo que o usuário desista de realizar as operações (sair). O aluno deve alterar a aplicação de tal forma que o usuário consiga somar, subtrair e desistir de realizar as operações. Para isso, é proposta a inclusão de mais uma história no projeto.

Observação: o teste de aceitação representa o que o cliente espera do software/funcionalidade implementada pelo desenvolvedor.

Quando considerar finalizado uma história (ou funcionalidade):

2) Todos os testes de aceitação estão "passando" (ou seja, a funcionalidade apresenta todos os resultados especificados pelo cliente nos testes de aceitação).

História – Subtrair dois números inteiros

Como usuário da calculadora gostaria de poder subtrair dois números inteiros.

Regras: não há restrições.

Testes de aceitação (a seguir neste documento, a sequência de telas/layouts para cada teste):

g) [sucesso] Abrir/executar a aplicação, selecionar subtração de dois números, informar o primeiro número, informar o segundo número e visualizar o resultado.

Telas/Layouts para o teste de aceitação da história

g) *[sucesso] Abrir/executar a aplicação, selecionar subtração de dois números, informar o primeiro número, informar o segundo número e visualizar o resultado.*

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação:
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2 <enter>
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2
Subtração de dois números inteiros
Informe...
    Primeiro Número:
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2
Subtração de dois números inteiros
Informe...
    Primeiro Número: 2 <enter>
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2
Subtração de dois números inteiros
Informe...
    Primeiro Número: 2
    Segundo Número:
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2
Subtração de dois números inteiros
Informe...
    Primeiro Número: 2
    Segundo Número: 1 <enter>
```

```
Calculadora de somar e subtrair Inteiros
=======================================
Informe [1] Somar, [2] Subtrair e [S]air para não realizar nenhuma operação: 2
Subtração de dois números inteiros
Informe...
    Primeiro Número: 2
    Segundo Número: 1
Resultado: 3
(Pressione ENTER para continuar...)
```

Passo 4) Desenvolvimento do software (exercício de programação)

Aguarde orientação do pesquisador :-]

Observação:

- Utilizar o projeto chamado “aplicacao” dentro do Eclipse!
- Tempo previsto: ***30~40 minutos.***

APÊNDICE H – ROTEIRO PARA AVALIAÇÃO DOS PROJETOS (SOFTWARE) DOS DESENVOLVEDORES

Roteiro

1. **Avaliar a situação do projeto Java:** deve-se avaliar e classificar a situação geral do projeto Java considerando as classes (situações) descritas abaixo.

- Classe: Erro de compilação.
  Mesmo após resolver problemas de classpath/jre, ainda assim, aparecem erros na lista de problemas da IDE (X´s).
- Classe: Executando sem interação.
  O projeto está sem erro de compilação, mas não aparece nada no console, nenhuma mensagem, cursor esperando entrada de valores,...
- Classe: Executando com interação.
  Há uma interface com o usuário, mesmo que mínima (por exemplo, a apresentação da tela inicial da aplicação, ou solicitação de entrada do nome de um novo contato,...).

2. **Avaliar o design da aplicação (software):** como alguns desenvolvedores não conseguiram implementar a interface e/ou todas funcionalidades (histórias), pede-se que avalie a "intenção" de design (projeto de classes) do desenvolvedor(es) por meio do código-fonte, eventualmente desconsiderando se aplicação está executando ou não. *Deve-se atribuir uma nota para cada história (funcionalidade) considerando a classificação descrita abaixo.*

- Nota "0": Não avaliado.
  Não é possível avaliar, pois não há praticamente nenhuma "intenção" de design para história em questão.
- Nota de "1" a "5", onde:
  - "1" A intenção de design é de código estruturado: por exemplo, não há nenhuma classe além da classe Executar, e há apenas o método main (da própria classe Executar).
  - "2" A intenção de design é de um código OO "mínimo": por exemplo, apenas duas classes, contando com a classe Executar;

- “5” A intenção de design é de um código OO com aspectos diferenciados: por exemplo, é possível identificar uma intenção de “diluir” as responsabilidades em um número equilibrado de classes – inclusive usando pacotes diferentes – e há uso explícito de algum padrão de projeto e/ou modelo arquitetural (MVC).

Planilha para registro da avaliação

Nome do projeto	Situação do projeto	Nota História 1	Nota História 2	Nota História 3

Resultado (após reunião de discussão/consenso entre os especialistas)

Id projeto	Situação do projeto	Testes de aceitação que passaram da H1	Nota História 1 (H1)	Nota História 2 (H2)	Nota História 3 (H3)
ProjetoA	Executando c/ interação	a; b;	4*	4*	4*
ProjetoF	Executando c/ interação	d;f	3	3	0
ProjetoG	Executando c/ interação	a;b;	2	2	2
ProjetoE	Executando c/ interação	a (parcial)	2	0	0
ProjetoC	Executando c/ interação	f	4	4	0
**	*Executando c/ interação*	--	*1*	*1*	*0*
**	*Erro de compilação*	--	*1**	*0*	*0*
ProjetoD	Executando c/ interação	a;b;	3	3	0
ProjetoB	Executando c/ interação	a;b;f	4	4	0

* Notas onde os especialistas discordaram por uma pequena variação antes da discussão/consenso.
** Projetos que não foram considerados na análise de qualidade do software, pois os critérios de inclusão são: i) o projeto deve estar compilando e ii) o projeto deve obter no mínimo nota 2 na história 1.

Perfil dos especialistas

- Especialista 1: Mestre em Ciência da Computação com pelo menos 4 anos de experiência em desenvolvimento de software;
- Especialista 2: Graduado em Tecnologia em Desenvolvimento de Sistemas com certificações em Java (SCJP 6.0 e OCWCD 5.0) e com experiência de pelo menos 6 anos em desenvolvimento de software;

APÊNDICE I – DIRETRIZES PARA PROGRAMAÇÃO EM PAR

Durante a Programação em Par...

FAZER

Fale

Nem um minuto deve se passar sem que você esteja se comunicando de alguma forma.

Ser paciente

É uma virtude! Explicando-se você ajuda você mesmo a aprender.

Escute

Cada parceiro deve estar totalmente envolvido no projeto

Respeite

Todo mundo tem algo a oferecer.

Troque os papéis

Troque de lugar para obter novas perspectivas e compartilhar o trabalho, isso ocorrerá:

- A cada 5 minutos na atividade treino;
- A cada 10 minutos no exercício de desenvolvimento de software;

Não se preocupe com isso porque o pesquisador irá avisar quando essa troca deve ocorrer!

Prepare-se

Faça qualquer trabalho preliminar antecipadamente. Por exemplo, não é necessário "sair codificando", vocês podem antes discutir a solução entre vocês e depois começar a codificar.

Limpe-se

Como você se sentiria se a pessoa com quem você fosse passar horas cheirasse mal?!

NÃO FAÇA!

Ser mandão

Você não quer ser a única pessoa na turma com a qual ninguém quer trabalhar!

Ser [muito] quieto

Não tenha medo de falar se você não concorda com seu parceiro.

Ficar intimidado

Você sabe muito mais do que você acha que sabe.

Sofrer em silêncio

Se você não está a vontade, fale com o pesquisador.

APÊNDICE J – DADOS EXTRAÍDOS COM PLUGIN METRICS DOS PROJETOS DA TURMA B

Id projeto	Aluno/Dupla		VG		LCOM		RMI		MLOC	
			média	*desvio padrão*	*média*	*desvio padrão*	*média*	*desvio padrão*	*média*	*desvio padrão*
ProjetoA	AlunoA	AlunoH	1,5	1,118	0,125	0,217	0,556	0,416	5,25	9,431
ProjetoB	AlunoB	AlunoI	1,769	1,423	0,125	0,217	1	0	4,846	6,455
ProjetoC	AlunoC	AlunoJ	1,538	0,929	0,611	0,079	1	0	4,538	4,651
ProjetoD	AlunoD		2,1	2,022	0,25	0,25	0,5	0,5	7	10,817
ProjetoE	AlunoE		5	4	0	0	1	0	20,5	18,5
ProjetoF	AlunoF		2,4	1,497	0,125	0,125	1	0	6,8	5,997
ProjetoG	AlunoG		3,333	5,217	0,25	0,25	1	0	13,667	27,879

Quadro 1 – Dados extraídos com plugin metrics dos projetos da turma B

APÊNDICE K – CARTA DE APRESENTAÇÃO

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Pró-Reitoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Programa de Pós-Graduação em Computação Aplicada
Campus Curitiba

CARTA DE APRESENTAÇÃO

Curitiba, 20 de junho de 2013

Prezado (a)

Apresento, por meio desta, o aluno e pesquisador Vagner C. M. Lima, regularmente matriculado no Programa de Pós-Graduação em Computação Aplicada (PPGCA) do Departamento Acadêmico de Informática (DAINF) da Universidade Tecnológica Federal do Paraná (UTFPR).

Solicito cordialmente sua colaboração e de sua instituição na pesquisa do aluno sob minha orientação, cujo caráter é exclusivamente científico. A privacidade será respeitada por meio do anonimato, ou seja, qualquer informação ou elemento que possa identificar os participantes da pesquisa serão mantidos em sigilo, evitando qualquer exposição pessoal, social ou profissional.

Coloco-me a disposição para eventuais esclarecimentos ou informações complementares por meio do seguinte endereço eletrônico: adolfo@dainf.ct.utfpr.edu.br.

Cordialmente.

Prof. Adolfo G. Serra Seca Neto, Dr.
PPGCA – DAINF – UTFPR

APÊNDICE L – ANÁLISE DOS RELATOS DOS ALUNOS SOBRE O USO DA PP

A análise das repostas dos participantes a questão *"QB7: Por favor, liste quaisquer benefícios ou desvantagens da prática Programação em Par"* do Questionário B foi realizada conforme o método de Análise Qualitativa de Conteúdo definido em Flick (2009, p. 291-293), abaixo as etapas definidos pelo método. O objetivo desse método é reduzir/categorizar o material coletado.

1) **Definir o material e selecionar as entrevistas ou aquelas partes que são relevantes para a questão de pesquisa**
2) **Analisar a situação da coleta de dados**
3) **Caracterizar formalmente o material**
4) **Definir a direção da análise para os textos selecionados e "o que de fato se quer interpretar a partir deles"**
5) **Diferenciar a questão de pesquisa com base na teoria/revisão bibliográfica**
6) **Definir a técnica de análise qualitativa de conteúdo**
7) **Definir as unidades analíticas**
8) **Conduzir as análises**
9) **Avaliar as análises**

Ao final dessas etapas elaboramos um consolidado das análises, buscando a apresentação do corpo principal deste trabalho.

A seguir o encaminhamento dado para cada etapa citada anteriormente considerando o contexto deste trabalho.

1) **Definir o material e selecionar as entrevistas ou aquelas partes que são relevantes para a questão de pesquisa:**
   a. Material: repostas dos participantes a questão "QB7: Por favor, liste quaisquer benefícios ou desvantagens da prática Programação em Par" do Questionário B;
   b. Partes relevantes: todo o texto redigido pelo participante no campo de resposta correspondente a pergunta sobre benefícios e/ou desvantagens da PP;

2) Analisar a situação da coleta de dados:

Os dados foram coletados por meio de um formulário eletrônico. O link de acesso a esse formulário foi enviado no dia seguinte ao e-mail pessoal do participante.

3) Caracterizar formalmente o material:

Conforme descrito anteriormente, o material foi coletado por meio de um formulário eletrônico, logo, não foi necessário nenhum tipo de transcrição de gravação de áudio.

4) Definir a direção da análise para os textos selecionados e "o que de fato se quer interpretar a partir deles":

Os textos selecionados serão analisados considerando se o que foi relatado pelo participante do experimento pode ser interpretado e categorizado como um benefício ou desvantagem quanto ao uso da PP.

5) Diferenciar a questão de pesquisa com base na teoria/revisão bibliográfica

Para auxiliar a análise/interpretação do material, deve-se diferenciar – ou escalonar – ainda mais a questão de pesquisa com base na teoria estudada até o momento (revisão bibliográfica). Vale relembrar aqui a questão de pesquisa, que é *"Q: Qual a percepção dos desenvolvedores quanto aos benefícios e desafios (ou desvantagens) no uso da Programação em Par?"*.

No Quadro 1 a seguir, o resultado da diferenciação ou escalonamento inicial do conceito "Benefícios" quanto ao uso da PP em [sub]categorias.

Descrição da [sub]categoria
Conhecimento geral da equipe fica mais **homogêneo,** por causa da disseminação do conhecimento causada pela troca de papeis e do rodízio dos pares.
Código com **menos defeitos**, devido, por exemplo, a frequente revisão do código – ora piloto, ora navegador
Desenvolvimento de **soluções mais complexas** em menos tempo e com melhor qualidade, devido, por exemplo, ao efeito "duas cabeças pesam melhor do que uma".

Foco quase que total na atividade em desenvolvimento devido à pressão do par.
Maior **produtividade,** em geral, no médio e longo prazo, em consequência, por exemplo, do menor número de defeitos e soluções de mais simples e eficazes.
Maior **satisfação no trabalho** realizado, devido, por exemplo, a maior colaboração entre os membros da equipe e maior volume de trabalho com qualidade realizado.
Um meio em potencial para o **aprendizado** e por consequência nivelamento do conhecimento e experiência dentre os integrantes da equipe, devido a constante comunicação entre os parceiros.

Quadro 1 – Lista inicial de [sub]categorias do conceito/código *benefícios*

No Quadro 2 a seguir, o resultado da diferenciação ou escalonamento inicial em [sub]categorias do conceito “Desvantagens” (ou inconveniências) quanto ao uso da PP. Elas foram elaboradas com base nos desafios inerentes a adoção da prática em questão.

Descrição da [sub]categoria
Baixa receptividade por parte dos **gerentes “tradicionais”**: em geral os gerentes das empresas, devido a sua formação fortemente baseada no modelo “produção em massa”, associam o projeto a uma fábrica e os desenvolvedores a máquinas, logo, é inconcebível ter duas “máquinas” fazendo o que apenas uma faria.
Maior atenção à **higiene** pessoal: por exemplo, há alimentos – tais como, o alho – que podem causar o mau-hálito.
Necessidade de **infraestrutura específica**: por exemplo, em diversas empresas as mesas são posicionadas nos cantos das baias, normalmente nesse caso a mobília tem um formato em “L”, algo que pode dificultar ou até inviabilizar o uso da PP.
Desapego quanto a **ferramentas** de desenvolvimento: em geral, cada desenvolvedor possui uma ou mais ferramentas de sua preferência. Sendo assim, podem ocorrer “guerras” de quais ferramentas é melhor ou não para o desenvolvimento do software, ou mesmo, dificuldade por parte do desenvolvedor em utilizar uma ferramenta que não é de sua preferência.
Maior atenção à **limpeza** da estação de trabalho: a mesa/estação de trabalho não está limpa/higienizada.
Desapego quanto ao uso de **padrões próprios de codificação**: em geral, cada desenvolvedor possui um padrão de codificação de sua preferência. Sendo assim, podem ocorrer “guerras” de qual padrão é melhor para o desenvolvimento do software.
Maior atenção à **etiqueta**: por exemplo, não cobrir a boca quando tossir, ou, usar perfume forte.
Necessidade de manter o **nível de qualificação (conhecimento/experiência) equilibrado** entre os integrantes das duplas: deve-

se evitar formar duplas com configuração do tipo “professor-estudante” - ou seja, não é recomendado um desenvolvedor sênior e júnior e uma mesma dupla.
Maior atenção à **saúde**: por exemplo, devido a proximidade, não ir ao trabalho quando estiver doente.
Gerenciar o **relacionamento humano**, em virtude das diferenças individuais: por exemplo, a) conciliar o sentimento de insegurança gerado por ter que mostrar a solução para o parceiro; b) ter humildade para aceitar facilmente as críticas; c) paciência para esperar o parceiro terminar o raciocínio para depois discutir uma solução que é a soma das ideias, isso principalmente quando se conhece outra solução potencialmente melhor; e por fim, d) não pode haver competição entre os membros da dupla, isso não favorece o trabalho colaborativo. Ou ainda, quando se necessita separar um desenvolvedor imaturo emocionalmente de seu parceiro de gênero oposto porque ele não soube discernir o que é trabalho do que é pessoal.

Quadro 2 – Lista inicial de [sub]categorias do conceito/código *desvantagens* (ou inconveniências)

6) Definir a técnica de análise qualitativa de conteúdo:

Será utilizada a técnica de Análise Estruturadora de Conteúdo. Nela buscam-se tipos ou estruturas formais no material, o qual é estruturado com base nas seguintes diretrizes:

1) Estruturação formal: por meio de aspectos formais, define-se como separar gradualmente uma estrutura (ver etapa 7);
2) Estruturação tipificadora: no material pode haver aspectos isolados e destacados que necessitam de uma descrição com maior precisão;
3) Estruturação quanto ao conteúdo: o material pode ser extraído e condensado a certos domínios de conteúdo (benefício e/ou desvantagens);

Para este trabalho definimos uma estruturação escalonada prévia quanto as [sub]categorias dos dois grandes conceitos possíveis para o conteúdo do material antes de iniciarmos o processo de análise propriamente dito (ver quadros 1 e 2).

Após essa estruturação inicial, a **análise ocorrerá em três momentos**, a saber:

Primeiro momento

Para cada participante do experimento:

1) Copiamos para as respectivas colunas do Quadro 3 o texto original redigido pelo participante como resposta a questão do formulário de pesquisa;
2) Analisamos uma unidade analítica após a outra, verificando se a unidade faz parte do domínio/categoria Benefício ou Desvantagem, buscando identificar possíveis tipificações;

Segundo momento:

1) Os especialistas avaliaram/revisaram o resultado, colocando suas interpretações/considerações quanto ao resultado definido;

Terceiro momento:

1) Analisamos as interpretações/considerações dos especialistas, considerando eventuais reclassificações e/ou tipificações;

7) Definir as unidades analíticas[15]:

As unidades de análise são as seguintes:

a. *Unidade de codificação* (qual é o menor elemento de material que pode ser analisado e enquadrado em uma categoria): palavra ou grupo de palavras que concorre para exprimir uma ideia ou conjunto de ideias, ou expressão ou ainda locução (frase, um membro de período)
b. *Unidade contextual* (qual é o maior elemento no texto que pode ser enquadrado em uma categoria): pequena parte ou seção de discurso, capítulo etc. que forma sentido completo e independente (parágrafo);
c. *Unidade analítica* (quais trechos são analisados um após o outro): frase composta de muitos membros, cuja reunião forma um sentido completo e independente (período);

[15] A delimitação das unidades analíticas foi realizada com base nas definições disponíveis em http://michaelis.uol.com.br/moderno/portugues/index.php para os seguintes conceitos: Frase, Período e Parágrafo.

8) Conduzir as análises

A seguir o resultado das análises das respostas dos participantes do experimento considerando os conceitos Benefícios e Desvantagens de uso da PP no desenvolvimento de software.

Perfil dos especialistas:

- Especialista 1: Especialista em Gestão de Projetos de Software com certificação Scrum com pelo menos 5 anos de experiência em desenvolvimento de software.
- Especialista 2: Especialista em Métodos para Engenharia de Software com pelo menos 8 anos de experiência em desenvolvimento de software (foco mais qualidade de software).

Código do participante	Texto original por participante do exp.	Unidade analisada	Tipificação	Resultado	Avaliação/revisão do especialista 1	Avaliação/revisão do especialista 2
AlunoH	'Ideias fluem mais facilmente; dificuldade de conciliação das atividades.'	'Ideias fluem mais facilmente; [...]'	Se ideias fluem mais facilmente, é razoável considerarmos que a PP permite um melhor entendimento do problema a ser resolvido e/ou planejamento da solução.	Benefício	Concordo	Concordo
		'[...] dificuldade de conciliação das atividades.'	O aspecto/termo 'atividades' dá margem para várias interpretações. Poderia ser, por exemplo, as atividades de responsabilidade do desenvolvedor no papel do piloto, ou ainda, no papel de navegador. Sendo assim, achamos razoável não classifica-la.	Não classificada	Discordo, pois eu vejo como um desafio relacionado a "Gerenciar relacionamento humano...".	Para mim não está claro o relato, concordo com a não classificação.
AlunoA	'Discute-se mais antes de programar.	'Discute-se mais antes de programar. [...]'	--	Benefício	Concordo	Concordo

	Enquanto um aluno programava o outro podia pensar em outros fatores faltantes no código que estava sendo desenvolvido. Vendo outro programar, você observa um padrão diferente do seu que as vezes ajuda a melhorar seu próprio nível de programação. Quando as duplas possuem conhecimentos muito diferentes, um acaba atrapalhando o outro e enquanto um pode crescer como programador, o outro fica estagnado empurrando o outro pra frente.’	‘[...] Enquanto um aluno programava o outro podia pensar em outros fatores faltantes no código que estava sendo desenvolvido. [...]’	O aspecto/termo ‘fatores’ dá margem para várias interpretações. Sendo assim, achamos razoável não classifica-la.	Benefício	Concordo, poderia ser um benefício associado a “código com menos defeito...”.	Concordo.
		‘[...] Vendo outro programar, você observa um padrão diferente do seu que as vezes ajuda a melhorar seu próprio nível de programação. [...]’	--	Benefício	Concordo	Concordo
		‘[...] Quando as duplas possuem conhecimentos muito diferentes, um acaba atrapalhando o outro e enquanto um pode crescer como programador, o outro fica estagnado empurrando o outro pra frente.’	É razoável interpretar que quando o participante relata algo do tipo “...possuem conhecimentos muito diferentes...” e “...empurrando o outro para frente”, ele evidencia a necessidade de se manter o nível de qualificação (conhecimento/experiência) equilibrado entre os integrantes das duplas.	Desvantagem	Concordo	Concordo
AlunoJ	‘Há a troca de conhecimento. Com a mudança entre quem codifica, mescla-	‘Há a troca de conhecimento. Com a mudança entre quem codifica, mescla-	Como na sequência do texto o participante coloca essa ‘mescla’ de estilos como algo positivo, imagina-se que ele se	Benefício	Concordo	Concordo

	se os dois estilos em um mesmo código, houve situações em que, enquanto um codificava, o outro percebia possíveis problemas que poderiam acontecer no futuro, que passariam despercebidos ou demorariam mais para serem percebidos, pois o codificador está focado no que está fazendo no momento.'	se os dois estilos em um mesmo código [...]'	referiu a design da solução. É plausível inferir que a troca que ele diz no início se refere a uma troca de experiência e/ou nivelamento de conhecimento.			
		'[...] houve situações em que, enquanto um codificava, o outro percebia possíveis problemas que poderiam acontecer no futuro, que passariam despercebidos ou demorariam mais para serem percebidos, pois o codificador está focado no que está fazendo no momento.'	--	Benefício	Concordo, poderia ser um benefício associado a "código com menos defeito...".	Concordo, um benefício associado a "...menos defeito...".
AlunoC	'Mantém o foco, e abre discussão para melhor solução'	'Mantém o foco [...]'	É plausível considerarmos que o participante se referiu ao foco na atividade atual, logo, no desenvolvimento do software (exercício de programação).	Benefício	Concordo	Concordo
		'[...] abre discussão para melhor solução'.	--	Benefício	Concordo	Concordo
AlunoI	'Diminui a velocidade de resolução do problema, pois as vezes surgem	'Diminui a velocidade de resolução do problema, pois as vezes surgem	--	Desvantagem	Concordo, na realidade vejo isso como um problema relacionado a "...gerenciar	Concordo, e inclusive classificaria como "Gerenciar relacionamento

	discussões de implementação. Por outro lado amplia a possibilidade de formas de resolver o problema.’	discussões de implementação. [...]’			relacionamento humano...”	humano...”.
		‘[...] amplia a possibilidade de formas de resolver o problema.’	--	Benefício	Concordo	Concordo
AlunoB	O maior benefício é quando existe um problema de difícil solução ou quando é necessário uma tomada de decisão muito grande no sistema, é possível ter uma visão diferente do assunto. A maior desvantagem está presente nos momentos em que se apresenta soluções simples em que o outro pode mais atrapalhar do que ajudar.	‘O maior benefício é quando existe um problema de difícil solução [...]’	--	Benefício	Concordo	Concordo
		‘O maior benefício é quando [...] é necessário uma tomada de decisão muito grande no sistema, é possível ter uma visão diferente do assunto. [...]’	--	Benefício	Concordo	Concordo
		‘[...] A maior desvantagem está presente nos momentos em que se apresenta soluções simples em que o outro pode mais atrapalhar do que ajudar.’	--	Desvantagem	Concordo e vejo isso como algo relacionado “gerenciar o relacionamento humano...”.	Concordo

Quadro 3 – Resultado da análise das respostas dos participantes versus benefícios da PP

9) Avaliar as análises

Nesta etapa apresentamos uma avaliação das análises realizadas na etapa anterior, considerando os relatos relacionados ao domínio Benefício e Desafio (ou desvantagens).

Benefícios relatados pelos participantes do experimento	Código(s) do participante(s)
'Ideias fluem mais facilmente; [...]'	AlunoH
'Discute-se mais antes de programar. [...]'	AlunoA
'[...] Enquanto um aluno programava o outro podia pensar em outros fatores faltantes no código que estava sendo desenvolvido. [...]'	
'[...] Vendo outro programar, você observa um padrão diferente do seu que as vezes ajuda a melhorar seu próprio nível de programação. [...]'	
'Há a troca de conhecimento. Com a mudança entre quem codifica, mescla-se os dois estilos em um mesmo código [...]'	AlunoJ
'[...] houve situações em que, enquanto um codificava, o outro percebia possíveis problemas que poderiam acontecer no futuro, que passariam despercebidos ou demorariam mais para serem percebidos, pois o codificador está focado no que está fazendo no momento.'	
'Mantém o foco [...]'	AlunoC
'[...] abre discussão para melhor solução'.	
'[...] amplia a possibilidade de formas de resolver o problema.'	AlunoI
'O maior benefício é quando existe um problema de difícil solução [...]'	AlunoB
'O maior benefício é quando [...] é necessário uma tomada de decisão muito grande no sistema, é possível ter uma visão diferente do assunto. [...]'	

Quadro 4 – Lista de benefícios relatados pelos participantes do experimento

Desafios (ou desvantagens) relatados pelos participantes do experimento	Código(s) do participante(s)

‘[...] Quando as duplas possuem conhecimentos muito diferentes, um acaba atrapalhando o outro e enquanto um pode crescer como programador, o outro fica estagnado empurrando o outro pra frente.’	AlunoB
‘Diminui a velocidade de resolução do problema, pois as vezes surgem discussões de implementação. [...]’	AlunoI
'[...] A maior desvantagem está presente nos momentos em que se apresenta soluções simples em que o outro pode mais atrapalhar do que ajudar.'	AlunoB

Quadro 5 – Lista de desafios (ou desvantagens) relatados pelos participantes do experimento

APÊNDICE M – EXEMPLO PLUGIN METRICS E ECLIPSE