Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Uma Investigação sobre o Uso de Práticas *Extreme Programming* no Desenvolvimento Global de Software

Roger Gonçalves Urdangarin

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Paulo Henrique Lemelle Fernandes

Porto Alegre
2008

Pontifícia Universidade Católica do Rio Grande do Sul

Dados Internacionais de Catalogação na Publicação (CIP)

U74i Urdangarin, Roger Gonçalves.
Uma investigação sobre o uso de práticas extreme programming no desenvolvimento global de software / Roger Gonçalves Urdangarin. – Porto Alegre, 2008.
118 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS
Orientador: Prof. Dr. Paulo Henrique Lemelle Fernandes

1. Informática. 2. Engenharia de Software.
3. Extreme Programming. 4. Desenvolvimento Global de Software. I. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Campus Central
Av. Ipiranga, 6681 - prédio 16 - CEP 90619-900
Porto Alegre - RS - Brasil
Fone: +55 (51) 3320-3544 - Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Uma Investigação sobre o Uso de Práticas *Extreme Programming* no Desenvolvimento Global de *Software*"**, apresentada por Roger Gonçalves Urdangarin, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 28/02/08 pela Comissão Examinadora:

Prof. Dr. Paulo Henrique Lemelle Fernandes – Orientador — PPGCC/PUCRS

Prof. Dr. Toacy Cavalcante de Oliveira – — PPGCC/PUCRS

Dr. Alberto Avritzer – — Siemens Corporate Research

Homologada em 09/05/08, conforme Ata No. 009/08 pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@inf.pucrs.br
www.pucrs.br/facin/pos

Agradecimentos

Ao Professor Dr. Paulo Fernandes pela oportunidade de ser seu aluno de mestrado e pela credibilidade em meu trabalho. Você é sem dúvida o grande amigo que fiz durante este curso de mestrado e a pessoa com quem pude aprender muito durante esta etapa de minha vida.

Ao Dr. Alberto Avritzer pelos valiosos ensinamentos de como elaborar uma pesquisa científica em engenharia de software. Por sua grande motivação e confiança no meu potencial, e finalmente por me guiar no caminho da conclusão da dissertação de mestrado. Sua participação e contribuições neste trabalho foram inestimáveis.

A minha querida esposa Ana Cristina, que em diversos momentos sempre me apoiou, chorou e sorriu comigo para a realização deste grande sonho em minha vida. Aninha te agradeço com muito carinho todo o esforço que tiveste comigo nestes dois anos.

Aos amigos Kleinner Oliveira e Luiz Mello que sem dúvida nenhuma ficarão guardados para sempre na minha memória, desejo sucesso a vocês e obrigado por suas contribuições neste trabalho.

Ao Centro de Desenvolvimento e Pesquisa da Dell, por ter financiado, parcialmente, meu curso de mestrado. Agradeço os professores do PPGCC e aos alunos entrevistados pelo tempo dispensado e por suas contribuições nesta pesquisa.

Resumo

Os desafios que a engenharia global de software vem enfrentando atualmente em função das grandes distâncias geográficas continuam cada vez mais complexos. A globalização de companhias tem afetado diretamente o mercado de desenvolvimento de software. Na busca por diferenciais competitivos que resultem em custos mais baixos e um alto índice de produtividade e qualidade no desenvolvimento de software, diversas empresas multinacionais optaram por expandir suas fronteiras e começaram a expandir o seu desenvolvimento de software nos países emergentes do bloco BRIC (Brasil, Rússia, Índia e China) em função dos incentivos fiscais favoráveis e mão de obra especializada abundante. Torna-se cada vez mais necessário identificar alternativas de processos de desenvolvimento de software que sejam mais leves e menos burocráticos que contribuam para a agilidade das equipes distribuídas e aliviem os efeitos negativos que a distribuição geográfica traz para o desenvolvimento global de software. Nesse sentido, esta dissertação de mestrado tem como objetivo avaliar os efeitos causados pela adoção de práticas de desenvolvimento ágil nos principais desafios enfrentados por projetos GSD. O método de pesquisa utilizado foi o estudo de caso tendo como unidade de análise um projeto de desenvolvimento global de software envolvendo a participação de três universidades situadas em dois continentes e um centro de pesquisas em engenharia de software localizado nos EUA. A pesquisa contribui no sentido de identificar quais as lições aprendidas sobre os efeitos produzidos pela aplicação da metodologia ágil no contexto da engenharia global de software, bem como, que novos desafios surgem a partir desta combinação.

Palavras-chave: Engenharia Global de Software, Desenvolvimento Global de Software, *Extreme Programming*, *Pair Programming*.

Abstract

The challenges facing global software engineering in today's world, due to large geographical area, became more complex. The rise in globalization of international companies have directly affected the software development market. In the search for low cost competitive advantages for companies, high productivity and quality in the system development, several organizations decided to extend their internal development to the BRIC emerging countries (Brazil, Russia, India and China) because these countries reduce their costs and have large specialized labor pools available. Making it necessary to search for new software development processes that are less bureaucratic, increase the distribution team's agility and minimizing the negative impacts that large geographic distribution causes in global software development. In this sense, the goal of this research is to identify and validate the effects caused by adopting the practice of agile development in the principal challenges faced by GSD projects. The research method chosen was the case study, conducted in three universities located in the two different continents, in addition to a software engineering research center from the United States. The contribution of this research is the identification of the lessons learned based on the affects that the adoption of agile methodology in the global software engineering context, as well as what would be new challenges that appear from this combination.

Keywords: Global Software Engineering, Global Software Development, *Extreme Programming*, *Pair Programming*.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

VoIP	Voz sobre IP	16
TI	Tecnologia da Informação	16
GSD	Desenvolvimento Global Software	20
TDD	*Test-Driven Development*	36
GSP	*Global Studio Project*	38
SCR	*Siemens Corporate Research*	38
AUP	*Agile Unified Process*	43
GSE	Engenharia Global de Software	50
EUA	Estados Unidos da América	51
SNA	Social Network Analysis	58

Sumário

Capítulo 1

Introdução

O Desenvolvimento Global de Software aumentou significativamente nos últimos anos. Alianças, terceirizações, aquisições e a demanda do mercado foram algumas das principais razões pelas quais grandes companhias do setor tecnológico distribuíram o desenvolvimento de suas aplicações pelo globo (Sharma 2003). O desenvolvimento de software vem sofrendo profundas transformações em função dos novos modelos de atuação incorporados pelas grandes multinacionais, na tentativa de se adaptarem as novas necessidades dos mercados emergentes.

Diversos autores (Sa & Maslova 2002, Sharma 2003, Damian et al. 2006, Carmel 1999, Espinosa et al. 2002, Herbsleb et al. 2001) destacam que devido ao notável avanço da Internet, a forma de se desenvolver software vem sofrendo constantemente mudanças. Com a globalização do desenvolvimento de software, aplicações inteiras estão sendo desenvolvidas por times remotos. Atualmente uma aplicação pode ter seus módulos sendo desenvolvidos paralelamente por diferentes equipes espalhadas em diversas partes do mundo. Tais times remotos podem até mesmo possuir diferenças na forma de aplicar a engenharia de software dentro de um mesmo projeto.

De acordo com os estudos de (Paulish et al. 2005), o desenvolvimento geograficamente distribuído é considerado como uma grande promessa e ao mesmo tempo um grande desafio. Os principais fatores motivadores são a redução dos custos, enorme capacidade de trabalho disponível e a possibilidade de entrega de produtos sob medida para mercados específicos. Sob o ponto de vista de (Paasivaara 2003), o desenvolvimento global de software vem se tornando muito comum nas grandes organizações.

1.1 Motivação

A globalização do desenvolvimento de software é uma importante marca na história da economia global. Porém quais foram as razões que impulsionaram o desenvolvimento global de software a um crescimento tão significativo? Na verdade não foi um único

fator isolado que desencadeou o crescimento deste novo fenômeno mundial, mas sim a combinação de seis forças atuantes. Estas seis principais forças convergiram, conforme ilustrado na Figura 1.1.

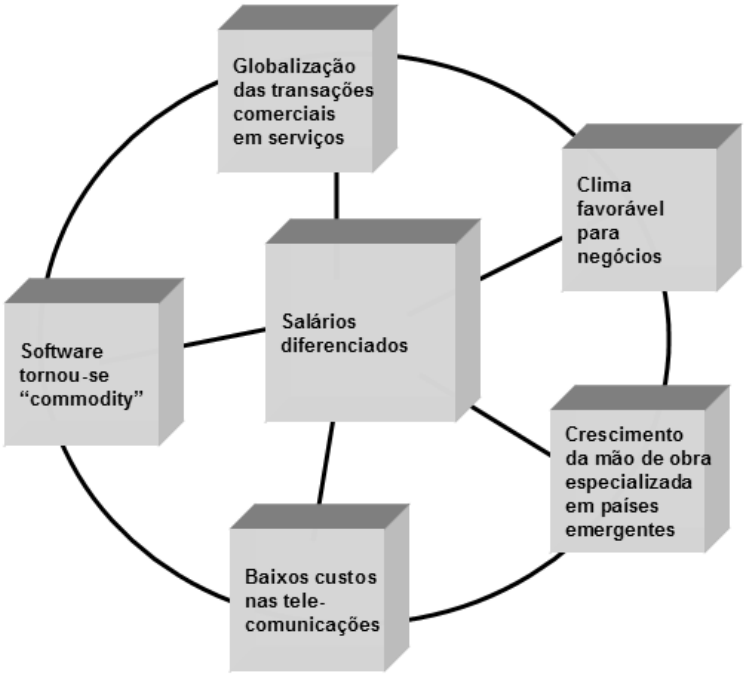

Figura 1.1: Forças que determinaram a expansão do fenômeno GSD.

A primeira força é uma velha conhecida, a globalização das transações comerciais, que mais recentemente chegou ao ramo de serviços. Com a abertura de fronteiras a partir do ano 1980 soluções baseadas no mercado ganharam uma ampla aceitação. O colapso do bloco da União Soviética contribuiu ainda mais para que este processo ganhasse força.

Nações que antes eram hostis para os negócios, ou indiferentes, estão agora competindo entre si para atrair investimento estrangeiro e desenvolver o seu setor de software, criando desta forma um clima favorável para os negócios. As nações estão oferecendo taxas de incentivos mais atrativas e também menos rígidas as regulamentações governamentais em seus países. Elas estão construindo parques tecnológicos para facilitar a montagem e a execução das operações. Enquanto isto, o número de engenheiros transborda das universidades e escolas técnicas da Índia, China, e outras nações que surgem neste âmbito. A China sozinha, forma profissionais qualificados em uma taxa anual quatro vezes maior que os EUA. Observa-se que a qualidade destes programas de graduação já foi bastante inferior aos programas de países industrializados, porém esta diferença esta cada vez menor. Esta elite de mão de obra *offshore* sempre esteve disponível, mas o que acabava muitas vezes ocorrendo era que estes talentos migravam para as nações industrializadas ou procuravam outros tipos de trabalho. Atualmente, as grandes multinacionais do ramo tecnológico contratam talentosos engenheiros e cientistas da computação onde quer que

estejam. No período de uma década aproximadamente, os custos com a comunicação caíram significativamente. Este fator fez surgir uma nova e impressionante realidade, isto é, nos dias de hoje, está muito fácil trabalhar com alguém do outro lado do mundo como se estivéssemos trabalhando com alguém do outro lado da cidade. Muitos engenheiros de software passaram a utilizar a tecnologia de telecomunicação (VoIP) à custos zero (Carmel & Tija 2005). Igualmente importante para o desenvolvimento de software, as capacidades de tráfego de dados nas redes de computadores expandiu na ordem de magnitudes, de algo próximo a zero na década de 90 para 4 *gigabits* por segundo na Índia em 2004.

Um outro aspecto foi a padronização do modo de se construir software. Com a adoção de processos e práticas bem definidas de desenvolvimento de software, pela primeira vez em 50 anos de história da computação, algumas atividades de desenvolvimento de software puderam ser suficientemente repetidas e automatizadas até que elas acabaram se tornando uma commodity. O termo *commodity* é o que se diz na indústria para produtos que atingiram um grau de maturidade em sua fabricação suficiente o bastante para serem produzidos em larga escala à custos muito baixos sem comprometer a sua qualidade final (Senge 1998). Finalmente, a força dominante que impulsionou consideravelmente a expansão do desenvolvimento global de software, como por exemplo o *Offshoring*, é a diferença entre os níveis salariais entre as nações envolvidas. É por este motivo que esta força foi ilustrada bem no centro das forças apresentadas pela Figura 1.1.

A diferença nos níveis salariais leva naturalmente a custos mais baixos. As pressões em cima dos custos fizeram com que o *Offshoring* se tornasse um ponto estratégico para muitas companhias, e é por esta razão que nos últimos anos muitos estudos vem sendo realizados na área de GSD, a fim de se obter melhores resultados na colaboração, comunicação e coordenação dos times remotos. Com certeza, existem muitos outros fatores que motivam a adoção cada vez mais freqüente de projetos GSD, uma destas forças secundárias seria o aparecimento de empresas de TI especializadas em *offshore*, especialmente na Índia. Além das vantagens em alguns casos isolados do desenvolvimento chamado *Following the sun*. Adicionalmente, o acesso a mercados também tem sido fator decisivo, como é o caso da China, onde as grandes empresas multinacionais são forçadas a investir em pesquisa e desenvolvimento para ter permissão de vender seus produtos no país (Carmel & Tija 2005).

1.2 Caracterização do Problema

Diversos problemas entre as equipes distribuídas nos projetos GSD são relatados por diversas pesquisas científicas (Herbsleb et al. 2001, Damian & Hargreaves 2004, Carmel 1999). A maior parte destes problemas são recorrentes nos projetos GSD. Problemas tais

como falhas na comunicação entre os membros dos times distribuídos ocorrem devido à fatores relacionados as diferenças culturais entre os times, bem como, as diferenças de idiomas e também em razão das grandes distâncias geográficas que separam os times. O problema que se pretende investigar nesta pesquisa diz respeito aos efeitos da aplicação das práticas de desenvolvimento de software da metodologia ágil *Extreme Programming* sobre os principais desafios enfrentados no desenvolvimento global de software. É esperado como resultado deste estudo que tais práticas ágeis ajudem a minimizar os efeitos negativos dos principais desafios provenientes das grandes distâncias geográficas nos projetos GSD.

1.3 Questão de Pesquisa

A questão de pesquisa no projeto de pesquisa aqui apresentado foi a seguinte: *Avaliar os efeitos na utilização das práticas da metodologia ágil Extreme Programming causados nos principais desafios enfrentados pelo Desenvolvimento Global de Software?*

1.4 Objetivos

Com base na contextualização na seção anterior, o objetivo geral desta pesquisa é identificar um conjunto de lições aprendidas que possam identificar quais os efeitos causados nos projetos GSD, quando utilizamos as práticas da metodologia ágil *Extreme Programming* sobre um projeto de desenvolvimento global de software. A fim de atender este objetivo geral emergem os seguintes objetivos específicos:

- Aprofundar o conhecimento na engenharia global software, na metodologia ágil *Extreme Programming* e em tópicos relacionados.
- Elaborar um conjunto de métricas para medir a produtividade da equipe de desenvolvimento que irá executar o processo XP.
- Investigar os efeitos das práticas XP nos desafios de GSD através das redes sociais formadas entre as equipes distribuídas.

1.5 Organização do Volume

Este volume está organizado em sete capítulos.

No capítulo 2 é apresentado o referencial teórico utilizado nesta pesquisa, são apresentados os principais conceitos sobre o desenvolvimento global de software e quais os principais desafios atualmente enfrentados nesta área de pesquisa. Ainda neste capítulo, são introduzidos os principais conceitos da metodologia ágil *Extreme Programming*, onde se descreve os conceitos de cada prática ágil XP adotada no projeto do estudo de caso

desta pesquisa. A apresentação deste referencial teórico é feita de forma abrangente, em virtude da natureza exploratória desta pesquisa (Yin 2005).

No capítulo 3, são apresentados os trabalhos relacionados. Os estudos descritos neste capítulo caracterizam-se por terem sido realizados utilizando o mesmo projeto de desenvolvimento global de software usado por esta pesquisa. Os trabalhos relacionados são descritos detalhadamente e comparados entre si, buscando-se identificar quais seus pontos fortes e fracos com relação a sua eficiência na redução dos impactos negativos dos desafios de GSD. Tais estudos inspiraram esta pesquisa sobretudo na intenção de entender mais sobre os problemas e sobre a dinâmica dos projetos GSD.

O capítulo 4 apresenta o método de pesquisa utilizado neste estudo, bem como apresenta também o projeto de pesquisa com o detalhamento das suas quatro fases. O capítulo encerra com uma breve introdução do projeto de desenvolvimento global de software utilizado como unidade de análise do estudo de caso.

No capítulo 5 é apresentado detalhadamente o cenário do estudo de caso, são descritos os papéis de cada um dos times distribuídos envolvidos no projeto, bem como, uma breve explicação sobre os componentes que constituíam a arquitetura da aplicação construída através da cooperação destas equipes. Descreve-se também, uma síntese da estratégia de testes de integração utilizada para validar a comunicação entre os componentes distribuídos. Finalmente, são apresentadas as etapas da metodologia do estudo de caso e uma análise comparativa entre as duas versões do projeto GSP, usando os gráficos das redes sociais, usadas para medir a intensidade da comunicação entre as equipes, as respostas das entrevistas com os membros das equipes e os resultados obtidos para as métricas de produtividade nas duas edições do projeto.

No capítulo 6 são listadas as lições aprendidas identificadas a partir da adoção da metodologia ágil XP sobre uma equipe de desenvolvimento co-localizada composta por estudantes de graduação em informática. São relatados todos os problemas enfrentados na condução das práticas de desenvolvimento empregadas pela equipe para a condução das atividades do projeto do estudo de caso. Além disto, as lições aprendidas destacam ainda os diversos pontos positivos que a metodologia ágil XP contribuiu com relação à comunicação, coordenação e colaboração das equipes distribuídas. São apresentados alguns novos desafios que acabaram surgindo em virtude da adoção das práticas XP na equipe de desenvolvimento, bem como, algumas boas práticas que precisam ser seguidas antes de adotar um processo de desenvolvimento de software baseado nos princípios de uma metodologia ágil.

Finalmente no capítulo 7, são apresentadas as conclusões da pesquisa sobre os efeitos causados pela adoção da metodologia ágil XP na área de engenharia global de software. Também são descritas as principais contribuições que o estudo trouxe para o preenchimento da lacuna existente sobre os impactos das metodologias ágeis no desenvolvimento global de software. Além disto, são descritas as diversas limitações deste

estudo durante a realização do estudo de caso. Ainda neste capítulo são descritas as sugestões de trabalhos futuros na linha de pesquisa da engenharia de software empírica.

Capítulo 2

Base Teórica

2.1 Desenvolvimento Global de Software (GSD)

O Desenvolvimento Global de Software (GSD) é um fenômeno que está recebendo um interesse considerável de diversas companhias em todo o mundo. Em GSD, o desenvolvimento de software tem o envolvimento de pessoas de diferentes nacionalidades e diferentes culturas organizacionais. Muitos benefícios podem ser obtidos a partir desta prática, como por exemplo, acesso a mão-de-obra abundante, custos menores e desenvolvimento vinte e quatro por sete (Herbsleb & Moitra. 2001).

As atividades do ciclo de desenvolvimento de software podem ser distribuídas para serem implementadas em diferentes localidades geográficas sendo coordenadas através das novas tecnologias de comunicação e informação (Sahay 2003). Segundo (Espinosa et al. 2002), os últimos anos testemunharam a globalização de várias organizações e indústrias. Como conseqüência, as colaborações entre equipes globalmente distribuídas e times virtuais tornaram-se muito comuns em diversas áreas, tais como, desenvolvimento de novos produtos e em sistemas de informação. De acordo com (Carmel 1999, Borchers 2003), os projetos de desenvolvimento globalmente distribuído de software consistem de times trabalhando juntos para atingir objetivos comuns em um mesmo projeto, porém situados em diferentes países. Além da dispersão geográfica, os times distribuídos enfrentam desafios com relação as diferenças culturais e temporais, que trazem problemas no entendimento entre as equipes, em função dos diferentes idiomas, tradições nacionais, valores e normas de comportamento. Conforme identificado por (Ågerfalk et al. 2005), GSD apresenta-se como sendo a alternativa mais adequada para muitos casos de *Outsourcing* de desenvolvimento de software, tendo baixo custo em países emergentes como China, Brasil e Índia (Holmstrom et al. 2006). Os projetos geograficamente dispersos são caracterizados por terem suas atividades de codificação e testes sendo desenvolvidas por uma unidade situada em uma determinada localização geográfica diferente da localidade onde uma outra unidade esteja desempenhado outras atividades do ciclo de desenvolvi-

mento (Sharma 2003).

De acordo com (Carmel 1999), as principais características que diferenciam o desenvolvimento distribuído de software do desenvolvimento co-localizado são a dispersão geográfica, a dispersão temporal e as diferenças culturais. Os projetos GSD possuem uma maior tendência do fluxo de comunicação ser mais intenso através das tecnologias de comunicação como e-mails, videoconferências e chat. Segundo (Espinosa & Carmel 2003), na ausência de contatos face a face, cabe aos times distribuídos escolherem quais as ferramentas utilizar para trazer os melhores resultados na comunicação entre as equipes.

2.1.1 Times distribuídos

Segundo os autores (Ramesh & Dennis 2002, Bradner & Mark 2002), times distribuídos podem ser definidos como equipes de trabalho que utilizam a tecnologia para se comunicar com um ou mais membros geograficamente remotos. Os times distribuídos também são conhecidos como times virtuais. Os membros das equipes remotas compartilham o mesmo propósito e estão localizados em pelos menos dois países distintos. Apesar dos membros dos times distribuídos poderem viajar para encontros face a face, a maior parte da rotina de comunicação entre eles ocorre de forma remota. Estudos apontam que o modelo de times globais está cada vez mais comum à medida que as companhias começam a competir globalmente. Um dos maiores desafios que envolvem os times virtuais globais concentra-se no gerenciamento dos processos e coordenação do trabalho.

2.1.2 Desenvolvimento de Software Offshore (DSO)

O DSO é o desenvolvimento que acontece fora da matriz da empresa contratante, em um país diferente por uma empresa prestadora de serviços (Khan et al. 2003, Carmel & Tija 2005). O termo *Offshore* inclui ambos, *Offshore* para uma empresa contratada também chamado de *Offshore Outsourcing*, como também, *Offshore* para um grupo interno dentro da organização global, também chamado de *Offshore Insourcing*.

Segundo (Carmel & Agarwal 2002), há mais de uma década, diversas organizações começaram a investir no desenvolvimento de software *Offshore* através de centros de desenvolvimento de software localizados em países emergentes. Tal fenômeno está associado aos custos mais baixos e disponibilidade de recursos técnicos habilitados. Os gerentes de TI estão sendo pressionados, sobretudo, para conter custos, iniciar os projetos com maior rapidez, e encontrar os recursos necessários nas tecnologias de ponta. Isto fez com que ocorresse uma rápida evolução na área de TI através do *Offshore Outsourcing*.

Existem muitas razões para o crescimento desta prática no segmento de TI, dentre as quais estão:

- Amadurecimento constante de tecnologias para o gerenciamento e coordenação das

atividades dos times geograficamente distribuídos.

- Modularização do desenho do software tem reduzido os custos de transação.
- Organizações *Offshore* (equipes internas e empresas contratadas) vêm melhorando significativamente o seu desenvolvimento de software e as suas capacidades de gestão de projetos.
- Mão-de-obra cada vez mais qualificada em abundância a baixo custo.

2.1.3 Site ou Unidade Organizacional

Site ou Unidade Organizacional são as empresas envolvidas em ambiente de desenvolvimento distribuído. A unidade ou site pode ser tanto a organização responsável pela prestação do serviço de desenvolvimento como também pode ser a organização contratante. Em um cenário de desenvolvimento de software distribuído, as empresas envolvidas no projeto que colaboram e compartilham esforços entre si para cumprir um objetivo comum são consideradas unidades ou sites. A unidade ou o site pode ser remota, isto é, separado por uma distância geográfica da matriz da companhia, ou pode ser a própria matriz.

2.1.4 *Outsourcing*

Segundo os estudos de (Kishore et al. 2003), o *Outsourcing* é definido como a contratação de vários serviços na área de sistemas de informação, tais como, gerenciamento de data centers, operações, suporte a hardware, manutenção de software, redes de computadores e até mesmo desenvolvimento de software realizado por empresas prestadores de serviços de TI, fora da matriz da empresa contratante. Do ponto de vista dos autores (Sparrow 2003, Narayanaswam & Henry 2005), *Outsourcing* pode ser definido como o processo de transferência de responsabilidade sobre o negócio, planejamento, gerenciamento e sobre as operações de determinadas funções de TI para uma terceira parte independente. Isto ocorre através de um acordo formal de prestações de serviço. Geralmente caracterizado por transferências de patrimônio do cliente para a organização prestadora de serviço.

Outsourcing é a prática de contratar uma organização externa para desenvolver um projeto, ao invés de desenvolver na própria matriz da organização contratante (*in-house*). As organizações que contratam serviços de *Outsourcing* podem se concentrar em seus *Outsourcing* podem se concentrar em seus negócios e reduzir a equipe de desenvolvimento de software. A combinação destes fatores resulta numa significativa redução no tempo e no custo de desenvolvimento (Mcconnell 1996). O princípio que estas organizações seguem é o de que terceirizando o processo, o trabalho poderá se tornar mais simplificado, o que muitas vezes na prática ocorre ao contrário. Existe uma perda de

visibilidade do progresso dos projetos quando eles são desenvolvidos em outro país por exemplo. E para compensar esta perda de visibilidade é necessário um gerenciamento ágil e eficaz.

2.2 Principais desafios em GSD

Do ponto de vista de diversos autores (Damian 2002, Herbsleb et al. 2000, Bradner & Mark 2002), os times distribuídos em GSD enfrentam desafios relacionados a fatores não técnicos como confiança, comunicação, conflitos e diferenças culturais. Também fatores técnicos de infraestrutura tais como, problema de conectividade de rede, diferenças entre os ambiente de desenvolvimento e testes nas localidades onde ocorre o desenvolvimento (Mockus & Herbsleb 2001). O GSD acrescentou novos fatores ao processo de desenvolvimento de software, tais como, distância temporal, dispersão geográfica, diferenças sócio-culturais, os quais amplificaram alguns dos desafios existentes na engenharia de software e que adicionou novas demandas que desafiam os processos de comunicação, coordenação e controle dos projetos (Damian et al. 2006, Ebert et al. 2001).

As equipes distribuídas têm poucas oportunidades de se beneficiar das vantagens existentes na comunicação informal. As grandes distâncias geográficas tornam difícil a possibilidade de se estabelecer e manter relacionamentos interpessoais essenciais para a construção de um espírito de equipe entre os times distribuídos. De acordo com (Damian & Hargreaves 2004), as negociações entre as equipes são caracterizadas por extrema cautela no momento de se estabelecer acordos, em particular, confiar nos argumentos de um colega remoto para perceber seu grau de honestidade e a partir disto antecipar e resolver conflitos não é uma tarefa fácil devido as grandes distâncias geográficas envolvidas.

Adicionalmente, os times distribuídos raramente chegam a um acordo sobre quais práticas de comunicação e de processo de desenvolvimento devem ser utilizadas como padrões no projeto. Os projetos GSD podem ser de vários tamanhos e formatos. Não é incomum estes terem múltiplas divisões, organizações, culturas e idiomas envolvidas no projeto (Sangwan et al. 2006). A maior parte das vezes os participantes nunca se viram antes, muitas vezes estes têm diferentes níveis de experiência sobre o domínio e podem possuir motivações que conflitem com os objetivos do projeto. Todos estes fatores conspiram para aumentar a dificuldade de coordenação sobre os times distribuídos, sobre o monitoramento do progresso das atividades. Estudos passados mostraram que as atividades levam cerca de 2.5 vezes mais tempo de serem realizadas em um ambiente de desenvolvimento distribuído comparado com ambientes co-localizados (Mockus & Herbsleb 2001). Outra questão também identificada através dos estudos é o de se torna difícil coordenar tarefas complexas com os times remotos até que haja um primeiro encontro face a face entre as pessoas envolvidas no projeto. Quando um problema ou uma

Tabela 2.1: Principais desafios enfrentados em GSD.

Desafio	**Questões envolvidas**
Diferenças Culturais	Costumes Locais
	Interpretação dos Requisitos
	Liderança
	Tempo de Resposta na Resolução de Problemas
	Perspectiva de Tempo
	Reações e Estilos Individuais e de Grupo
Dispersões Geográficas	Gestão dos Recursos
	Fusos Horários
	Legislação
Coordenação e Controle	Interdependência
	Processos
	Consciência
Comunicação	Disseminação do Conhecimento
	Meio
	Estilos de Comunicação
Espírito de equipe	Sinergia
	Confiança
	Tamanho
	Comprometimento

questão aparece, é quase sempre complicado saber a quem recorrer. Um indivíduo em uma localidade remota geralmente não tem a visibilidade sobre as atividades das outras pessoas que não estão na sua localidade. Por causa disto, muito tempo é gasto tentando encontrar alguém que possa ajudar. Uma outra situação que ocorre é que na ausência da disponibilidade de informações, suposições serão feitas. Tais suposições podem algumas vezes estarem conflitantes com os requisitos ou ainda, mais tarde, serem motivo de falhas no desenvolvimento do software. A Tabela 2.1 são apresentados os principais desafios encontrados em GSD, de acordo com a visão de diversos autores (Rockenbach 2003, Herbsleb et al. 2001, Damian 2002).

2.2.1 Diferenças Sócio-Culturais

Diferenças culturais geralmente complicam a comunicação, e podem levar a estágios de frustração, descontentamento e até mesmo desentendimento entre as equipes. Segundo os estudos de (Holmstrom et al. 2006), a cultura pode ter um efeito determinante na forma como as pessoas interpretam determinadas situações, e como elas reagem à estas situações. Estes fatores não técnicos envolvem a cultura organizacional, cultura nacional, idioma, política, motivação individual e a ética no trabalho das equipes distribuídas (Prikladnicki & Yamaguti 2004, Carmel 1999). Um exemplo de conflito cultural foi descrito por (Paulish

et al. 2005), um indivíduo relatou em uma entrevista que apesar da sua organização ter a cultura de comunicação aberta, ele estava trabalhando com uma organização prestadora de serviços que demonstrava ter uma cultura, na qual as pessoas eram extremamente relutantes a admitir seus erros. O entrevistado se mostrou desapontado pela impossibilidade de obter as informações necessárias de forma precisa. Certamente, as distâncias geográficas podem implicar no aumento das diferenças culturais. Entretanto, a diferença cultural pode ser imensa até mesmo se a distribuição geográfica for pequena (Holmstrom et al. 2006).

2.2.2 Distâncias Geográficas

Conforme apresentado nos estudos de (Carmel & Agarwal 2001, Bradner & Mark 2002, Damian & Hargreaves 2004), as distâncias geográficas acentuam os problemas de coordenação e controle, de forma direta ou indireta através dos efeitos negativos que causam na comunicação. A Figura 2.1 ilustra estes desafios e suas correlações, as linhas tracejadas apontam os desafios amplificados por outros desafios e as linhas contínuas apontam para os problemas decorrentes em função desta amplificação. As grandes distâncias geográficas afetam negativamente a comunicação, a qual afeta negativamente a coordenação e o controle (Clerc et al. 2007).

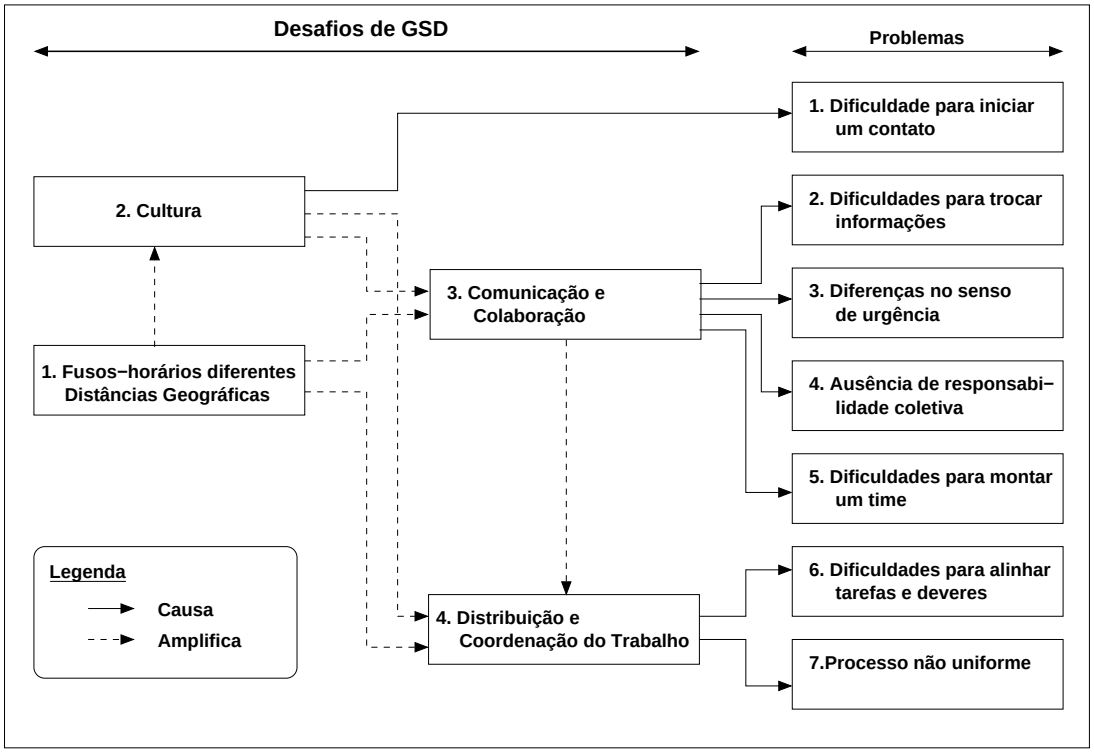

Figura 2.1: Visão geral dos desafios em GSD e suas correlações.
(Clerc et al. 2007)

2.2.3 Coordenação e Controle

De acordo com (Carmel & Agarwal 2001), a coordenação é o ato de integrar cada atividade com cada unidade organizacional. A condução desta integração geralmente requer comunicação intensa e constante. O controle é o processo de adesão aos objetivos, políticas, padrões e níveis de qualidade. O controle pode ser formal (tais como orçamentos e regras explícitas) ou informal (tais como pressão do colega de trabalho). Sob o ponto de vista de (Mockus & Herbsleb 2001), a coordenação se torna um problema porque os processos de cada time distribuído são diferentes, isto é, não existe um processo uniforme. Por exemplo, o que se entende como um teste unitário por uma equipe pode ser algo completamente diferente para outra equipe, este desentendimento pode ocasionar conflitos e desapontamentos entre os membros dos times distribuídos. Um outro problema ocorre quando uma determinada equipe atrasa a entrega de um módulo, este atraso na entrega pode afetar as datas de entrega das outras equipes, caso não sejam comunicadas a tempo.

2.2.4 Comunicação

Sob o ponto de vista dos autores (Carmel & Agarwal 2001), a comunicação é o fator de mediação que afeta diretamente a coordenação e o controle. Ela representa a troca de informações não ambíguas e completas, isto é, o emissor e o receptor conseguem chegar a um entendimento comum. De acordo com (Paulish et al. 2005), é fortemente recomendado que os times de desenvolvimento de todas as localidades remotas em um projeto GSD interajam diretamente, pois foi observado que devido a separação geográfica das equipes os níveis de comunicação informal caem consideravelmente, comparado com equipes colocalizadas (Herbsleb et al. 2001). Todos precisam saber, ou devem ser capazes de encontrar, quem deve ser contatado de acordo com a natureza da situação, e que todos tenham em suas mentes que tal tentativa de comunicação será prontamente atendida. Jamais permitir a criação de gargalos de comunicação na fase de desenvolvimento, ou seja, afunilamentos da comunicação através de uma única pessoa dentro da equipe. Conforme os estudos de (Damian et al. 2006), todos os desafios enfrentados em GSD parecem apontar para o prelúdio do uso de processos dependentes de comunicação informal.

A comunicação informal exerce um papel chave no sucesso das equipes distribuídas (Carmel & Agarwal 2001, Herbsleb et al. 2001). A comunicação é considerada o componente essencial de todas as práticas de colaboração e de processo no desenvolvimento de software. O impacto que a falta de comunicação eficiente pode causar em ambientes GSD é o resultado de um baixo nível de confiança entre as equipes e a perda da visibilidade sobre o andamento dos trabalhos nos sites remotos (Damian et al. 2006).

2.2.5 Espírito de equipe

A palavra equipe, em geral, tende a subentender co-localização, homogeneidade cultural, confiança, padrões de comunicação, um pequeno número de membros e, mais importante, coesão. Equipes são unidades sociais frágeis, que podem facilmente ser quebradas (Marquardt & Horvath 2001). Quando dificuldades tais como, distância, diferenças culturais e de fuso horário aparecem, o espírito de equipe geralmente é comprometido. O espírito de equipe é o efeito sinérgico que torna a equipe uma unidade coesa (Carmel 1999). Em relação a confiança, importantes características foram observadas na pesquisa feita por (Paulish et al. 2005). A primeira é a de que se leva muito tempo para se estabelecer relações de confiança entre as equipes. O estabelecimento de confiança ao longo do tempo é fator chave para o sucesso dos projetos, dado a extrema dificuldade existente para se julgar talentos em equipes separadas por grandes distâncias geográficas. A confiança é essencial quando as pessoas dependem umas das outras para atingirem objetivos comuns, uma equipe sem confiança não consegue atingir seus compromissos de maneira eficaz (Carmel 1999).

Ele ainda define que a confiança baseia-se no indivíduo em acreditar no caráter, habilidade, força e confiança de outra pessoa. Como essas características são complexas de serem estabelecidas em um único contato, elas levam tempo até tornarem-se consistentes, logo, a confiança leva tempo para ser construída (Whitney 1994). O tamanho do grupo é importante para assegurar que a comunicação de todos os membros irá ocorrer de maneira efetiva. Quanto menor for o número de integrantes de uma equipe, menor será o número de canais de comunicação necessários.

Por exemplo, com nove membros, uma equipe precisa gerenciar 36 canais de comunicação, que crescem de forma geométrica com a adição de membros ao grupo conforme ilustrado na Figura 2.2. Por isso é importante que as equipes permaneçam pequenas, preferencialmente não contendo mais de 10 pessoas (Carmel 1999).

2.3 *Extreme Programming (XP)*

Segundo os autores (Marchesi et al. 2002, Beck & Andres 2004) a *Extreme Programming* é a metodologia ágil com a maior documentação disponível atualmente na literatura, ela também é a que oferece maior suporte, como ferramentas de apoio por exemplo, e conseqüentemente a metodologia ágil mais usada. Ela foi inventada entre os anos de 1998 à 1999 e começou a se espalhar no ano de 2000. XP é fortemente indicada para times com equipes pequenas de desenvolvimento com até no máximo 15 pessoas, que concordem em seguir as práticas. A maioria das práticas XP também podem ser adotadas por desenvolvedores individualmente. Devido ao fato da metodologia XP ser muito bem detalhada e explicado, e não existir grandes dificuldades em seguir suas práticas, entretanto, é ex-

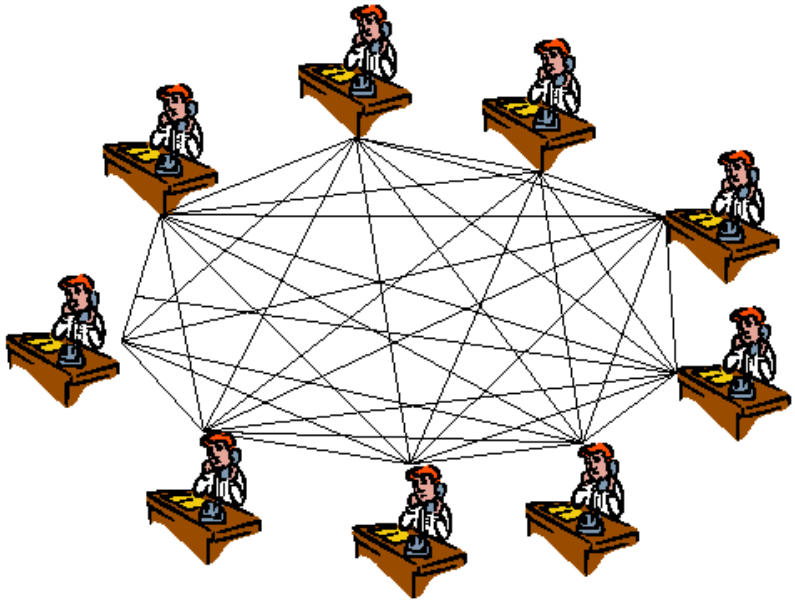

Figura 2.2: Número de canais de comunicação em uma equipe com 9 membros.

tremamente difícil e arriscado seguir todas as práticas rigorosamente e ao pé da letra de uma única vez (Marchesi et al. 2002).

Segundo (Beck & Andres 2004), a *Extreme Programming* (XP) é um processo leve, eficiente, de baixo risco, previsível, científico, e divertido de desenvolver software. Ele se distingue dos outros processos pelas seguintes razões:

- O cliente tem a possibilidade de passar o seu *feedback* com relação aos trabalhos que estão sendo entregues a cada final de ciclo.
- É um processo baseado em um plano incremental, o qual vai rapidamente se transformando em um plano mais amplo que gradualmente evolui durante o ciclo de vida do projeto.
- Possui um calendário flexível de implementação das funcionalidades, e por esta razão é capaz de atender rapidamente as necessidades dos clientes.
- Baseia-se em testes automatizados, escritos por programadores e clientes, servindo como uma forma eficiente de monitorar o progresso do desenvolvimento, permitindo que o sistema evolua, e que os defeitos sejam descobertos muito antes de causarem danos mais sérios ao projeto.
- Baseia-se fortemente na comunicação oral, testes e código fonte para transmitir a estrutura do sistema e intenções.
- Baseia-se em um processo de *design* evolutivo que persiste tanto tempo quanto o sistema persistir.

- Baseia-se em uma relação forte entre os programadores com habilidades técnicas dentro da média.

- Baseia-se em práticas que manipulam ambos os instintos a curto prazo dos programadores quantos os interesses a longo prazo do projeto.

XP é uma disciplina de desenvolvimento de software. Ela é considerada uma disciplina porque existem certas coisas que precisam ser feitas para poder se garantir que realmente esta sendo usado XP. Você não poderá escolher entre escrever ou não escrever os testes, se os testes não forem escritos você não estará sendo *extreme*. XP é projetado para atender equipes de projeto que envolvam de 2 a até 10 programadores, que não estejam extremamente dependentes da existência de um ambiente de programação, e onde seja possível rodar um conjunto de testes em uma fração do dia. A inovação que XP trouxe é que todas as técnicas que tiveram sua eficiência comprovada durante décadas foram trazidas para baixo do guarda-chuva de XP (Beck & Andres 2004, Warden et al. 2003). A metodologia XP baseia-se em 4 valores que são:

- **Comunicação** - O primeiro e mais importante valor da XP é a comunicação. A maioria dos problemas que ocorrem nos projetos invariavelmente tem sua causa associada ao fato de alguém não ter informado alguma coisa muito importante para uma pessoa que precisava saber. Algumas vezes um programador não conta a ninguém sobre uma mudança bastante crítica no design de uma solução.

- **Simplicidade** - O segundo valor é a simplicidade. A simplicidade não é fácil. Uma das coisas mais difíceis de se fazer é não olhar para as coisas que serão necessárias implementar no dia seguinte, na semana seguinte e no mês seguinte. Apenas implemente aquilo que é necessário e realmente importa ser construído. Isto significa dizer que, apenas se preocupe em solucionar hoje os problemas de hoje. A complexidade custa muito caro e tentar prever o futuro é muito difícil. É necessário aguardar o futuro para ver se está certo ou não.

- **Feedback** - O terceiro valor é o *feedback*. A veracidade dos relatórios do estado atual das atividades é extremamente importante em XP. *Feedback* significa perguntar e aprender com as respostas. A única forma de saber a necessidade do cliente é perguntando a ele. O único modo de saber se um código faz o que ele se destina fazer é testando-o. Quanto mais cedo se obter o *feedback*, mais tempo se terá disponível para reagir. A metodologia XP fornece um *feedback* rápido e freqüente por parte dos seus seguidores.

- **Coragem** - Depois de se falar nos três valores, comunicação, simplicidade e *feedback*, é hora de se esforçar como nunca antes. Se o trabalho não for desempenhado na

> sua velocidade mais rápida possível, alguém mais o irá fazer, e ele vai lucrar em seu lugar. A coragem significa tomar as decisões na hora em que elas precisam ser tomadas. Se uma funcionalidade não está funcionando , conserte-a. Se algum código não parece estar bem escrito, faça o *refactoring* dele. Se não será possível entregar tudo o que se havia prometido dentro do prazo acordado, informe o cliente. A coragem é uma virtude difícil de se aplicar. Ninguém gosta de estar errado ou quebrar um acordo. O único modo de consertar um engano é admitir que houve um engano e consertá-lo.

2.3.1 As Práticas *XP*

A metodologia ágil XP é constituída por doze práticas que cooperam entre si. As informações coletadas por cada uma delas ajudam na tomada de decisão do projeto. As interações entre as práticas ajudam também na obtenção de altos níveis de produtividade e qualidade. É possível codificar um software de boa qualidade usando apenas algumas das práticas, mas XP trabalha melhor se cada parte for executada, o todo é muito melhor do que a soma das partes (Warden et al. 2003).

Cada prática pode desempenhar vários papéis. Por exemplo, o teste influencia o projeto da solução e encoraja a execução de pequenos experimentos controlados. Apenas escolher algumas práticas XP ao acaso, sem antes entender a relação existente entre elas, pode levar à resultados desastrosos, por exemplo, um refinamento de código sem que tenha sido realizada uma rigorosa fase de testes anteriormente poderá resultar na introdução de defeitos tão sutis dentro do código que poderiam levar à extensas sessões de depuração para sua correção.

A melhor maneira para se entender XP é exercitando inteiramente todas as práticas que compõem a base do processo. Executar rigorosamente todas as etapas do processo requer muita disciplina (Warden et al. 2003, Beck & Andres 2004). Se a condução total da metodologia XP for algo impraticável para o projeto alvo, é necessário ter muita cautela na hora de se escolher qual subconjunto de XP utilizar, pois tal decisão poderá levar a conseqüências imprevisíveis. A Figura 2.3 ilustra as práticas que constituem a estrutura da *Extreme Programming*.

A aplicação de todas as práticas XP em um projeto de desenvolvimento de software é uma tarefa muito difícil (Beck & Andres 2004), nenhum relato foi encontrado na literatura envolvendo um projeto GSD. Por este motivo, foram selecionadas algumas práticas XP, visando não extrapolar o escopo deste trabalho, apesar dos riscos envolvidos. Não se busca limitar ou restringir o processo e seu conjunto de práticas. No entanto, os resultados deste estudo poderão servir para trabalhos futuros envolvendo as demais práticas disponíveis que não foram abordadas neste trabalho. As práticas utilizadas neste estudo foram:

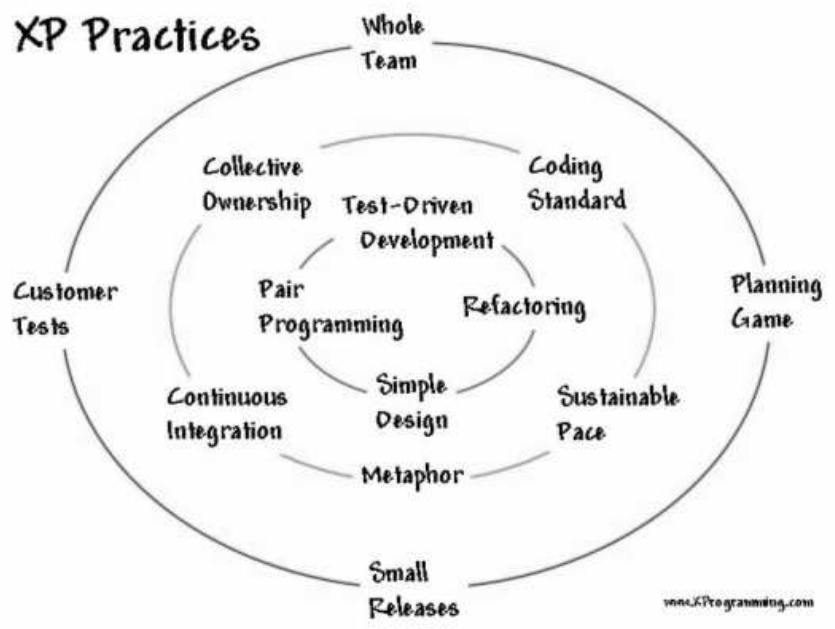

Figura 2.3: Conjunto das práticas que constituem a metodologia *Extreme Programming*.

Planning Game

Esta prática serve para o planejamento da próxima entrega, nela definidas as datas, as prioridades e as estimativas. O objetivo principal desta prática é o definir o cronograma. A metodologia XP utiliza a fase de *Planning Game* para determinar quais funcionalidades serão implementadas e em qual seqüência. O objetivo é sempre maximizar o valor agregado das funcionalidades desenvolvidas. Segundo os autores (Beck & Andres 2004, Warden et al. 2003) os cronogramas de projetos XP sempre são baseados em iterações. Uma interação é uma fotografia de um ciclo inteiro de desenvolvimento. O ciclo começa quando o cliente requisita as funcionalidades. O time de desenvolvimento planeja como irá implementar as novas funcionalidades, e então as funcionalidades são codificadas, testadas e entregues ao cliente. Todo o ciclo leva de uma a quatro semanas antes de recomeçar. A Figura 2.4 ilustra um ciclo de desenvolvimento XP.

Pair Programming

Segundo (Warden et al. 2003), o objetivo principal da *Pair Programming* é disseminar o conhecimento, a experiência e as idéias. A programação aos pares funciona da seguinte forma, quando você inicia uma nova atividade, você solicita que outro desenvolvedor venha trabalhar com você. Os pares geralmente trabalham juntos apenas em uma atividade, porém isto pode levar uma tarde inteira ou até mesmo alguns dias, e então novos pares se formam com novos parceiros. Esta prática promove a disseminação do conhecimento sobre o sistema para todo o time. Em *Pair Programming*, dois desenvolvedores trabalhando juntos para concluir uma única atividade. A pessoa que esta no teclado é chamada

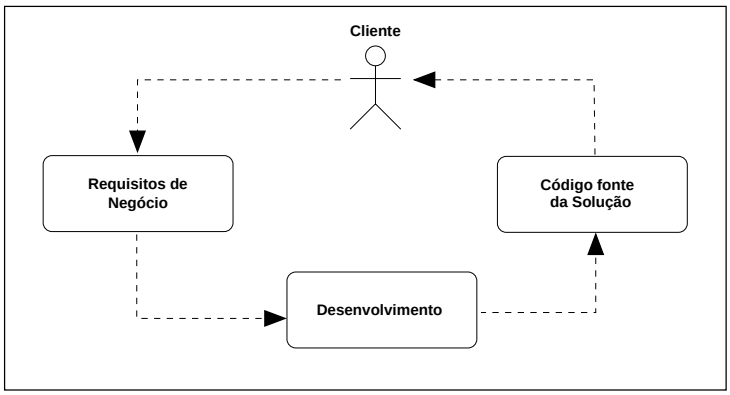

Figura 2.4: Ciclo de desenvolvimento XP.

de *driver*. Esta pessoa tem sua atenção dirigida para os detalhes da atividade. A outra pessoa chamada de *navigator* tem a responsabilidade de manter todo o projeto na cabeça, para garantir que a atividade esteja complacente com o projeto e também que ela esteja sendo desenvolvida seguindo os padrões de projeto definidos pelo time. O *navigator* precisa pensar de forma estratégica. Ambas as atribuições são importantes. As funções de *navigator* e *driver* podem mudar sempre que necessário. *Pair Programming* produz código através da conversação. A utilização da programação aos pares reforça bons hábitos de programação, cada parceiro exerce uma pressão positiva no seu par. A presença do parceiro reduz a tentação de pular a fase de testes, refinamento do código ou simplificações. Duas cabeças trabalhando em uma mesma atividade e dois pares de olhos no código produz poucos defeitos, guiando para um código mais limpo que respeita os padrões de codificação do projeto. Aos pares é gasto um pouco mais de comunicação enquanto se codifica e o resultado é a produção de códigos de melhor qualidade e que necessitam de muito menos depuração.

Os autores (Warden et al. 2003, Marchesi et al. 2002) destacam que trabalhando aos pares e mantendo os padrões de código em mente também reduz a tentação de reescrever código desnecessário. Durante cada atividade, o *navigator* sempre mantém em mente a idéia geral do trabalho, continuamente perguntando se modificações grandes são necessárias. Os casos de testes funcionam tanto para documentação quanto como guias para manter o sistema funcionando adequadamente. O *Pair Programming* é altamente informal, ele acontece naturalmente, explorando todas as possibilidades de agrupamento entre os desenvolvedores. Ao invés de trabalhar em duplas previamente determinada, o objetivo da prática é disseminar o conhecimento por todo o time, aprendendo com os colegas e ao mesmo tempo ensinando. Em algumas situações, entretanto, é necessário trocar as duplas com maior freqüência para evitar que os parceiros fiquem estagnados ou que sempre os mesmos se juntem. A programação aos pares pode parecer desconfortável logo no início, porém ela suporta a maioria das demais práticas de desenvolvimento da *Extreme Programming*. Por exemplo, ela diminui o sentimento de individualidade sobre

a responsabilidade do código, na verdade o código já é compartilhado desde o começo. O desenvolvedor que estiver trabalhando com esta prática precisa ser altamente disciplinado para seguir por conta própria o ciclo de testes, codificação e otimização do código.

Estudos mostram que desenvolvedores que se habituaram a codificar aos pares não se imaginam trabalhando de outra forma. O *Pair Programming* é um investimento em treinamento contínuo. Ele tem uma boa utilidade para pessoas novas no projeto assim como desenvolvedores com pouca experiência em programação. Um desenvolvedor junior poderá aprender com um desenvolvedor sênior e vice-versa. Também, se você é familiar com uma técnica ou tecnologia, você poderá demonstrá-la para seu parceiro de dupla. Como regra, os parceiros da dupla precisam auxiliar um ao outro. A programação aos pares necessita como requisito que os dois desenvolvedores sentem-se e trabalhem juntos na mesma estação de trabalho. Os desenvolvedores precisam demonstrar interesse em trabalhar com um colega ao lado. É importante que os gerentes de projeto também apóiem a prática (Beck & Andres 2004). Na *Extreme Programming* a *Pair Programming* tem a função chave de suportar outras práticas do processo.

Os autores (Marchesi et al. 2002, Beck & Andres 2004) destacam algumas razões importantes na motivação da equipe de projeto quando esta utiliza prática de *Pair Programming*. São elas:

- Redução do Risco da Perda

  Com a programação aos pares o impacto na perda de um desenvolvedor chave para o projeto é muito baixo porque diversas pessoas já estão familiarizadas com cada parte do sistema. Se uma dupla trabalha consistentemente, duas pessoas tornam-se bastante familiares com uma parte específica do código da aplicação. Se os pares rotarem regularmente, gradualmente todos os desenvolvedores acabam adquirindo conhecimento sobre todos os módulos da aplicação.

  Como os desenvolvedores rotam entre os grupos, eles podem ter a chance de conhecer pessoalmente muitas pessoas em seus times. Este tipo de aproximação ajuda a quebrar muitas barreiras de comunicação. A rotação das duplas permite um compartilhamento face-a-face de um conhecimento tácito, de idéias, e formas criativas de solucionar os problemas muitas vezes sequer constam na documentação e difíceis de implementar. Através da programação aos pares, o compartilhamento do conhecimento tácito ocorre naturalmente no dia a dia dos desenvolvedores. Não é necessário alocar nenhum tipo especial de recurso, sistema ou repositório para fomentar o compartilhamento do conhecimento na equipe. A programação aos pares funciona como um veículo que permite a continuidade das conversas entre os desenvolvedores que formaram duplas antigas, bem como nas duplas atuais.

- Times Satisfeitos e Confiantes

O problema de perda de uma pessoa na equipe é muito caro, tanto para recrutar um substituto quanto para treinar esta nova pessoa. Pessoas mais felizes, menos frustradas são mais positivas em suas responsabilidades e conseqüentemente mais motivadas em ajudar o time a alcançar os objetivos.

Mais de metade dos desenvolvedores resistem a idéia de começar a trabalhar com a prática de *Pair Programming*. Entretanto, uma vez que estes tentam quase sempre aprovam a prática. Estudos apresentados por (Marchesi et al. 2002) mostram que a adoção da *Extreme Programming* resultou em uma sensível melhoria na satisfação dos engenheiros de software com os seus cargos e também o nível de qualidade do trabalho realizado. Os pares foram entrevistados seis vezes perguntando se eles estavam gostando mais de trabalhar desta forma. Primeiramente foi aplicada uma pesquisa anônima usando a Internet. Ambas os times da universidade de Utah três vezes. Consistentemente, por volta de 90% concordaram que estavam gostando mais de realizar seu trabalho quando dispostos aos pares. Os grupos também foram entrevistados se a forma de trabalho colaborativa tornavam-os mais confiantes em seus trabalhos. Os resultados foram ainda mais positivos, com 96% dos entrevistados respondendo que o trabalho realizados em duplas tornavam-se os mais confiantes.

- Redução da Curva de Aprendizado

  Tradicionalmente, as pessoas quando entram uma organização, são introduzidas as aplicações pelo time sênior de desenvolvedores da companhia. Para isto, precisam ser reservadas horas de treinamento do pessoal sênior, o que geralmente tem um custo bem elevado para a empresa, pois tratam-se de horas não produtivas em que nem o desenvolvedor mais experiente do time nem o novo desenvolvedor estarão trabalhando nas aplicações que geram receita para a companhia. Se a *Pair Programming* fosse adotada, o desenvolvedor sênior ensinaria as aplicações na prática e não apenas apresentando-as, até mesmo contribuições diretas do aprendiz podem acontecer durante o seu período de treinamento. Além disto, o treinamento parece ser mais rápido, e o iniciante aprende o sistema muito melhor.

Collective Ownership

Qualquer indivíduo da equipe de desenvolvimento tem condições de alterar qualquer parte do sistema a qualquer instante. A idéia por traz desta prática é que não deve existir um único responsável por uma parte de código, é necessário que o time desenvolva um senso de responsabilidade coletiva. A *Pair Programming* auxilia os colegas de dupla a demonstrar seu comprometimento com a qualidade do trabalho realizado em parceria.

Coding Standards

Segundo , a prática *Coding Standards* auxilia os desenvolvedores a se comunicarem através do código. O código fonte é a forma primária de comunicação dentro de um projeto. As requisições por funcionalidades e relatórios vão e vêm, mas o projeto nasce e morre com o software. Enquanto houver alguém realizando a manutenção do software o projeto se manterá vivo. O *Coding Standards* é na verdade um conjunto de convenções que regulamentam a formatação e a estrutura dos códigos. Eles descrevem as boas práticas de programação que cada membro do time adiciona ao longo dos projetos. Os padrões quando necessário podem ser suspensos, isto vai depender do bom senso dos desenvolvedores sobre qual o momento em que as regras precisam ser quebradas. As vantagens dos padrões de código é que eles representam o estilo de programação de toda a equipe de desenvolvimento, por esta razão utilizar os padrões de código agilizam a programação economizando tempo se compararmos com o tempo gasto para a compreensão de um código sem endentação e nomes de variáveis sem significado algum para o contexto do programa.

Muitas linguagens possuem seus próprios *Coding Standards*, o ideal é começar por estes padrões e gradualmente ir adaptando de acordo com as necessidades do projeto. Acima de tudo o time de desenvolvimento precisa assumir o compromisso em seguir os padrões para que esta prática realmente tenha o efeito esperado. A prática de *Coding Standards* coopera para as práticas de *Refactoring*, *Pair Programming* e *Collective Code Ownership*.

Simple Design

As equipes XP devem sempre buscar por soluções mais simples possíveis. Para isto devem ser utilizados quatro critérios para avaliar o grau de simplicidade de uma solução proposta. Quando se trabalha em duplas fica mais fácil identificar qual a solução mais adequada para um determinado problema.

Refactoring

O *refactoring* é o refinamento ou melhoramento do código atual. Refaça a estrutura de um programa novamente sempre que alguma coisa não estiver demonstrando ter a qualidade esperada. O objetivo desta prática é refazer partes do programa sem alterar o comportamento da aplicação removendo código redundantes ou desnecessários, adicionando comentários no código, simplificando e tornando as funções cada vez mais genéricas e flexíveis.

Test-Driven Development

O *Test-Driven Development* (TDD) é uma prática de desenvolvimento incremental de software. A idéia desta prática baseia-se na construção de testes de regressão automatizados com o objetivo de nortear as atividades do desenvolvimento. Ela se tornou popular através da *Extreme Programming* (Beck & Andres 2004).

O TDD é uma disciplina de projeto e programação onde cada nova linha de código é escrita em resposta a um teste que o programador escreveu antes de iniciar a sua codificação. Segundo os autores (Jeffries & Melnik 2007), a idéia é escrever uma especificação de testes da mesma forma que o programa deverá chamar a funcionalidade a ser implementada e como seus resultados deverão ser retornados. Quando um teste falha este expõe o fato de que a funcionalidade ainda não foi implementada. Quando um teste não falha, significa dizer que o programa passou no teste. Desta forma, a codificação de um programa, seguindo a prática TDD, tem por objetivo passar no teste verificando também se todos os testes realizados anteriormente contra o programa continuam passando.

Finalmente o código é revisado da forma como ele se encontra construído, executando desta forma a prática *refactoring*. Então o processo é repetido, elaborando-se um outro teste para uma outra pequena adição seja incorporada ao programa.

Seguindo este ciclo incremental ilustrado pela Figura 2.5, o programa começa a crescer sobre si mesmo e à medida que são finalizados mais ciclos o código vai conseqüentemente sendo aprimorado pelo desenvolvedor. No início de cada ciclo, a intenção é para que todos os testes passem exceto o novo teste, o qual está guiando o novo código atualmente em franco desenvolvimento. E ao final de cada ciclo, o desenvolvedor executa todos os testes, para ter certeza que cada um deles continua passando, o que garante com segurança que todas as funcionalidades implementadas até o momento continuam funcionando.

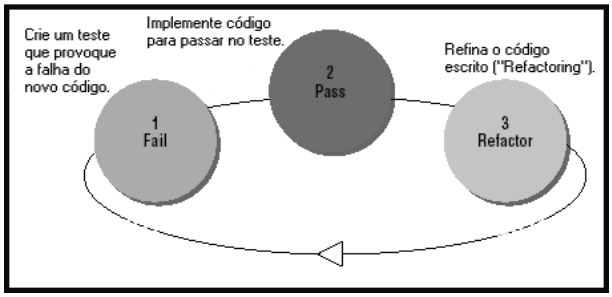

Figura 2.5: Ciclo de passos do *Test-Driven Development*

TDD é uma atividade de design e programação juntas, e não puramente uma atividade de testes. Sua relação com a área de Testes é meramente para a obtenção de uma comprovação através do conjunto de testes de regressão automatizado. Isto não significa que os analistas de testes não precisem mais continuar realizando seus testes

investigativos sobre a aplicação (Jeffries & Melnik 2007). Atualmente algumas confusões vem sendo feitas com TDD e Testes Unitários, TDD é na verdade uma disciplina de desenvolvimento incremental baseada em testes de regressão automatizados que devem ser executados pelo desenvolvedor (Erdogmus 2007).

Como resultado, TDD auxilia no alivio do medo que existe em mudanças no código. No passado a maioria dos desenvolvedores programavam primeiro escrevendo o código e então depois testando-o. Os desenvolvedores que trabalham com TDD tornam-se confiantes com seus códigos. Uma vez que o sentimento de confiança esteja presente em cada desenvolvedor da equipe, todos podem relaxar mais, sentir-se menos estressado e conseqüentemente produzindo um código mais focado em qualidade. Com o tempo todos acabam se habituando a forma de pensar em TDD, sobre quais testes não funcionariam e o que deve ser feito para fazê-los funcionar.

Capítulo 3

Trabalhos Relacionados

3.1 O Processo *Extended Workbench*

Para aumentar o entendimento sobre os problemas e as práticas do desenvolvimento global de software, o instituto de pesquisa em engenharia de software *Siemens Corporate Research* (SCR) executou um experimento durante dois anos utilizando times remotos formados por estudantes de diversas universidades espalhadas por cinco países em 3 continentes e com 11 fusos-horários diferentes. Os objetivos do experimento realizado pelo SCR foram:

- Entender as dinâmicas do desenvolvimento global de software.
- Identificar e documentar metodologias e boas práticas descobertas durante o estudo.

De forma a validar o processo *Extended Workbench*, o SCR iniciou o projeto *Global Studio Project* (GSP).

O experimento possibilitou identificar áreas chave em projetos de desenvolvimento de software envolvendo diversos times distribuídos geograficamente, entre as áreas chaves identificada está a coordenação. Segundo (Avritzer et al. 2007), o esforço de coordenação leva a utilização de táticas específicas destinadas a atender somente as necessidades provenientes de projetos GSD.

O processo de desenvolvimento de software utilizado no projeto GSP foi estruturado como se simulasse um modelo *hub-and-spoke* comumente conhecido internamente no SCR como modelo de processo *Extended Workbench*. Uma tradução para este nome seria a extensão da mesa de trabalho. Usando de uma analogia, a mesa de trabalho seria o time central, no caso o SCR, e as extensões desta mesa seriam os times remotos representados pelas universidades (Sangwan et al. 2006). A Figura 3.1 ilustra a estrutura descrita acima.

De acordo com o processo *Extended Workbench*, o time central era responsável pela definição dos papéis e responsabilidades de todos os times remotos, inclusive de si próprio. Segundo este processo, o time central realiza atividades no início de cada fase tais como, análise de requisitos, projetos de arquitetura do sistema e o plano de

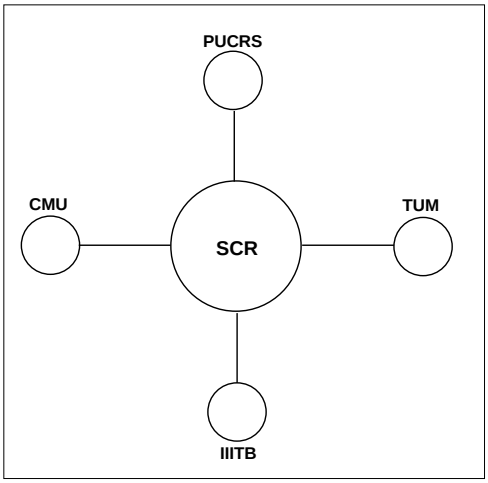

Figura 3.1: Modelo *hub and spoke*, o *hub* é o time central e os *spokes* são as universidades.

projeto. Enquanto que os times remotos ficavam com a incumbência das atividades de desenvolvimento e testes. Todo o gerenciamento do projeto deveria ser feito pelo time central (Avritzer et al. 2007, Paulish et al. 2005). A Figura 3.2 ilustra um diagrama de atividades que representa as responsabilidades de cada time.

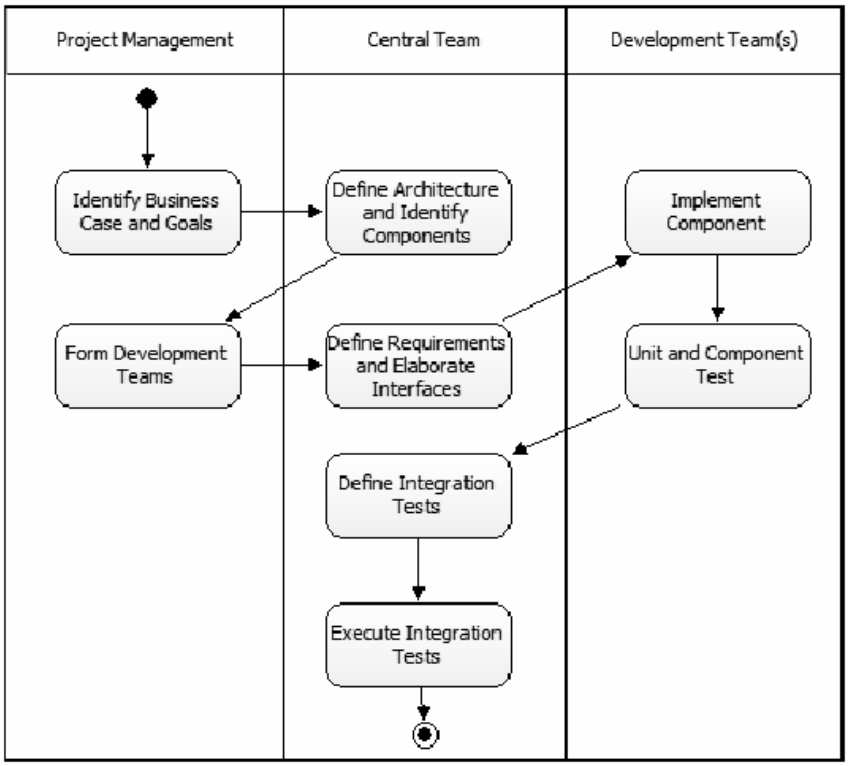

Figura 3.2: Responsabilidades dos times distribuídos segundo o processo *Extended Workbench*.

Do ponto de vista dos autores (Avritzer et al. 2007, Mullick et al. 2006) as principais práticas definidas pelo processo *Extended Workbench* foram as seguintes:

- Gerenciamento Centralizado - o processo de desenvolvimento de software era cen-

tralizado, os requisitos do sistema, arquitetura e testes do sistema eram executados de forma centralizada.

- Desenvolvimento Iterativo - Foi utilizado uma estratégia de ciclos de iterações curtos com duração quinzenal.
- Diminuição da Comunicação entre Times - O time central era quem gerenciava todas as comunicações entre os times. A maior parte das comunicações dentro do projeto ocorreu entre o time central e os times remotos.
- Especificação Formal de Requisitos - O objetivo foi de minimizar a comunicação entre os times remotos e o time central para esclarecimento de dúvidas, através do fornecimento de especificações formais dos requisitos para os times remotos.

Os papéis sob a responsabilidade do time central segundo o processo estão listados abaixo:

- Engenharia de Requisitos
- Arquitetura
- Engenharia de Integração
- Gerência de Infra-estrutura
- Administração da ferramenta de colaboração wiki

E os papéis sob a responsabilidade dos time remotos de acordo com o processo eram os seguintes:

- Desenvolvedores
- Líder de Testes
- Engenheiro de Requisitos
- Arquiteto de Sistemas
- Gerente de Qualidade
- Gerente de Infra-estrutura
- Engenheiro de Processo e Documentação

Ficava a critério dos times remotos decidir quais pessoas iriam assumir quais papéis dentro da equipe. As pessoas eventualmente poderiam alternar os papéis durante o projeto, porém era necessário comunicar o time central sobre todas as alterações. Para assegurar a visibilidade dos trabalhos das equipes distribuídas, o time central solicitava que fossem preenchidos duas pesquisas usando formulários *on-line*, onde todos os membros das equipes distribuídas deveriam responder a cada duas semanas completadas de projeto. Tais questionários estão descritos em detalhes a seguir:

- **Questionário das redes sociais (SNA):**

  O questionário SNA era estruturado com uma série de questões dedicadas a determinar uma análise das redes sociais dentro do projeto. Estas questões eram referentes a quem o indivíduo conhecia no projeto, com quem os indivíduos estiveram em contato desde a última vez que o questionário foi respondido, isto é, nas últimas duas semanas até o momento. Eram questionados como os indivíduos haviam comunicado entre si e qual a forma utilizada para realizar a comunicação. Estes dados eram utilizados para analisar a quantidade de coordenação necessária no projeto, verificar se os times que receberam atividades que necessitavam de comunicação estavam realmente se comunicando com os outros times. Identificar também, áreas onde a comunicação não estava sendo realizada de maneira efetiva. Os dados eram coletados a cada duas semanas de forma a analisar os resultados das táticas de coordenação aplicadas ao projeto. Os resultados obtidos nas análises levaram a conclusão que era necessário ajustar as táticas se as necessidades de coordenação fossem diferentes do que as estabelecidas no perfil de coordenação.

- **Questionário de estimativas de esforço de trabalho:**

  O questionário destinado as estimativas, era constituído de uma série de perguntas que buscavam extrair dados capazes de mostrar qual fora o esforço médio despendido por cada indivíduo dos times dentro do projeto no período compreendido desde o último questionário até a solicitação atual de preenchimento do mesmo. Esta informação que indicava qual teria sido o esforço gasto por cada indivíduo era utilizada monitorar o status ou andamento do projeto, além é claro de informar o esforço despendido pelas equipes individualmente nas duas semanas anteriores. Cada indivíduo era questionado com uma série de perguntas dirigidas ao desenvolvimento. Estas incluíam sobretudo questões a respeito de linhas de código produzidas (LOC), quão tivera sido difícil para o time remoto implementar o componente proposto pelo time central, e o quanto estavam claros os requisitos referentes e estes componentes. As análises feitas nos resultados deste questionário permitiram que a documentação e os treinamentos fossem aperfeiçoados de forma a sanar problemas identificados através das respostas dos indivíduos das equipes remotas. Bem como, o questionário

serviu como um auxílio para que os pesquisadores entendessem melhor como os indivíduos do projeto trabalhavam.

Infra-estrutura de Colaboração

O projeto Global Studio Project (GSP) contou com a infra-estrutura listada abaixo:

- Apache, MySql, PHP - Web Browser, Banco de Dados, e tecnologia para a camada cliente.
- MediaWiki - Principal meio de colaboração entre as equipes distribuídas.
- Manits - Ferramenta para registrar os defeitos encontrados pelas equipes de testes.
- Simple Machine Forum - Fórum de discussão.
- Subversion - Repositório central dos códigos entregues pelas equipes remotas.
- CruiseControl .NET - Servidor para integrar os módulos da aplicação.
- nAnt, nUnit, nCover, nDoc - Usado respectivamente como repositório dos builds da aplicação, testes unitários, resultados dos testes e documentação.

Apache, MySql e PHP são as tecnologias necessárias para suportar os software mencionados acima, de modo que estes pudessem funcionar corretamente. Eles foram escolhidos porque as outras ferramentas selecionadas eram capazes de serem suportadas por eles e o custo de sua utilização estava dentro do orçamento, isto é, custo zero.

O MediaWiki foi o principal meio de colaboração utilizados pelas equipes distribuídas trocarem informações com o time central, bem como com outras equipes remotas. O MediaWiki possibilitou que os requisitos, o design da arquitetura, o desenvolvimento e os testes pudessem ficar todos localizados em um mesmo lugar, possível de ser acessado por qualquer indivíduo das equipes remotas a qualquer momento. o MediaWiki também fornecia uma funcionalidade que permitia associar áreas de desenvolvimento com os requisitos. Isto provou ser muito benéfico pois quando os times remotos precisavam de esclarecimentos em requisitos específicos para o componente que estes estavam desenvolvendo localmente. Além disto a funcionalidade tinha uma outra vantagem, através do MediaWiki era possível rastrear os requisitos solicitados para um determinado módulo, e verificar se o módulo implementado realmente atendia aos requisitos solicitados, quando isto se fazia necessário.

O Mantis era o software utilizado para registrar os defeitos encontrados nos módulos implementados pelas equipes remotas. Ele permitia que os times remotos entrassem com os defeitos que eles identificavam ao longo da implementação ou testes de um módulo

e os propagavam para o time responsável por aquele módulo onde o defeito havia sido detectado. Seu uso era extremamente simples e o seu custo era zero.

Simple Machine Forum era utilizado primeiramente como um mecanismo de coordenação dentro do projeto. Este era um fórum de discussão o qual permitia aos times remotos adicionarem tópicos os quais precisavam ser melhor explicados e definidos, os quais não haviam sido considerados na definição atual da arquitetura disponível nas seções do MediaWiki. Os tópicos eram adicionados por um time remoto e atribuídos para outro time remoto, como conseqüência disto, os times remotos precisavam entrar em um debate para resolver os problemas postados no fórum. O benefício geral desta ferramenta foi o de permitir que a coordenação entre os times fosse realizada de forma ágil sem a necessidade de teleconferências ou numerosas trocas de e-mails. Mas o principal benefício do software era a possibilidade que cada time remoto tinha em revisar o tópico do fórum, por exemplo, escrevendo comentários ou informações pertinentes ao problema, isto passava para todos os times uma idéia geral sobre de que forma as equipes estavam resolvendo determinados problemas e o quais problemas estavam impedindo dos times em progredir com seus trabalhos. Subversion era a ferramenta de controle de versões dos códigos desenvolvidos pelas equipes remotas. Ele é um software bastante simples de ser utilizado e muito eficiente no seu propósito. Tinha também a agradável vantagem de ser uma ferramenta de custo zero para o projeto.

O restante da infra-estrutura foi utilizada apenas no primeiro ano do projeto GSP. No ano seguinte ela foi deixada de lado. Segundo os pesquisadores, o principal objetivo pela escolha da infra-estrutura foi o de aumentar as capacidades de atendimento das necessidades do projeto global. Os pesquisadores também identificaram uma área chave no desenvolvimento global de software que é o monitoramento. O Monitoramento é o processo que deve ser feito durante todo o projeto. Por exemplo, através de um monitoramento da infra-estrutura no primeiro ano do GSP, os pesquisadores perceberam que as necessidades do projeto haviam mudado e então uma mudança também na infra-estrutura precisava ser feita.

3.2 O Processo LAGPRO

O acrônimo LAGPRO significa *Local Agile Game-based Process*, trata-se de uma proposta para uma nova metodologia de coordenação de equipes locais dentro de um contexto de desenvolvimento de software distribuído. Segundo (Blois et al. 2006), o LAGPRO é uma proposta de processo para ser empregado dentro de equipes co-localizadas inseridas em projetos de desenvolvimento GSD. Ele é baseado em um processo de desenvolvimento iterativo, onde cada iteração global é dividida em fases, cada fase tem a duração de duas semanas. Cada fase então é avaliada através de um conjunto de métricas que compõem

o chamado *Local Scores*, que é na verdade uma tabela que contém a classificação de cada membro da equipe co-localizada, de acordo com um número de pontos ganhos em cada iteração. Estes pontos atribuídos para cada indivíduo, são gerados a partir da aplicação de métricas como uma forma de avaliação de desempenho de cada indivíduo do time de desenvolvimento e time de testes. As métricas abordaram os seguintes aspectos dentro do projeto: modelagem, implementação, *Status* do trabalho, infra-estrutura e auto-avaliação.

A idéia principal foi simplificar o processo de desenvolvimento interno fazendo uso de um número bastante pequeno de disciplinas e fases do processo *Agile Unified Process* (AUP) (Ambler 2005). A disciplina *Model*, foi delineada através de modelos simplificados construídos no início de cada nova iteração e através de modelos de revisões ao final de cada iteração. Para a disciplina de implementação foi escolhida a metodologia *Test-Driven Development* (TDD), onde os times de testes definiam os testes unitários automatizados e os time de desenvolvimento implementavam as regras de negócio (Blois et al. 2006).

O processo LAGPRO foi baseado no *Agile Unified Process (AUP)* e foi customizado de acordo com as necessidades internas da equipe. O *AUP*, é uma versão mais simplificada do Processo Unificado criado pela empresa Rational (RUP) que passou a adotar técnicas da metodologia *Agile* tais como *Test-Driven Development (TDD)* , *Refactoring*, e *Agile Model Driven Development*. Os objetivos da adoção do *AUP* foram:

- Aumentar a performance e a produtividade do time da PUCRS.
- Construir um ambiente de desenvolvimento de software ágil e otimizado.

De acordo com o processo *AUP* deve se utilizar apenas um artefato o qual seja mais apropriado para a realidade do projeto. Por esta razão, o time da universidade PUCRS assumiu algumas premissas:

- O time seria o sempre o mesmo.
- Cada iteração do projeto global em que o time remoto estava participando era considerada como um novo projeto para o processo local executado.
- Toda a comunicação com o time central deveria ser feita apenas pelo gerente de projetos.
- As reuniões deveriam ser agendadas usando a lista de e-mails do projeto, e no e-mail constaria a ata da reunião.
- Era aberta a participação para todos os membros da equipe.
- Para efeitos de planejamento todas as atividades foram organizadas de modo a serem feitas com duração de duas semanas.

- No final de cada duas semanas, eram computados as posições de classificação de cada indivíduo através da planilha de métricas em reunião exclusivamente marcada para este fim.

A idéia do jogo que envolvia o processo era baseada no sistema de pontuação das competições de xadrez. Assim como acontece nas partidas de xadrez dois adversários se enfrentam, cada um dos adversários possui um certo número de pontos inicialmente atribuído, o qual os deixa em uma determinada posição de classificação perante aos outros jogadores. O fenômeno que vai ocorrendo após inúmeras partidas jogadas por um determinado participante, é que o número de pontos começa a oscilar à medida que o número de partidas realizadas aumenta. Por exemplo, caso um enxadrista com 1200 pontos jogue contra um adversário que possui 1500 pontos, e ganhe o embate, ele automaticamente recebe um determinado número de pontos do participante derrotado. Com o passar do tempo, o número de pontos de cada um dos jogadores vai começar a se estabilizar em uma determinada faixa de pontos, e este então é o fenômeno natural que acontece, esta faixa de pontos é a que vai indicar o grau de expertise do jogador. Desta forma, o processo baseado em jogos, LAGPRO, tenta reproduzir o mesmo efeito obtido nos campeonatos de xadrez para extrair de cada desenvolvedor ou testador qual o seu grau de conhecimento técnico dentro do projeto. Com estas informações diversas decisões dentro de um projeto podem basear-se ou não na posição de classificação dos indivíduos.

Segundo o autor (Blois et al. 2006), as tentativas de execução do processo LAGPRO falharam devido a ocorrência de uma total rejeição por parte dos membros do time com relação a um ponto específico do processo, a sua característica competitiva. Os membros da equipe não apreciaram a idéia de uma competição interna ser o meio de avaliação da produtividade da equipe.

3.3 O Processo *LACOI-PD*

O processo *Large Components Integration-Per Domain (LACOI-PD)* define que o trabalho de desenvolvimento de software precisa contar com a colaboração de diversos times de trabalho. Por exemplo, o time que for responsável pela coordenação dos trabalhos será definido como o time central. É necessário que as definições de arquitetura e de requisitos sejam feitas por um time distribuído. A implementação dos componentes deve ser realizada por diversos times remotos de desenvolvimento e finalmente estes componentes precisam ser testados por uma equipe remota de testes. A Figura 3.3 ilustra esta dinâmica de trabalho entre os times.

- O time central forma times que serão responsáveis pela arquitetura e requisitos para a definição de uma interface comum para os componentes de integração. No projeto

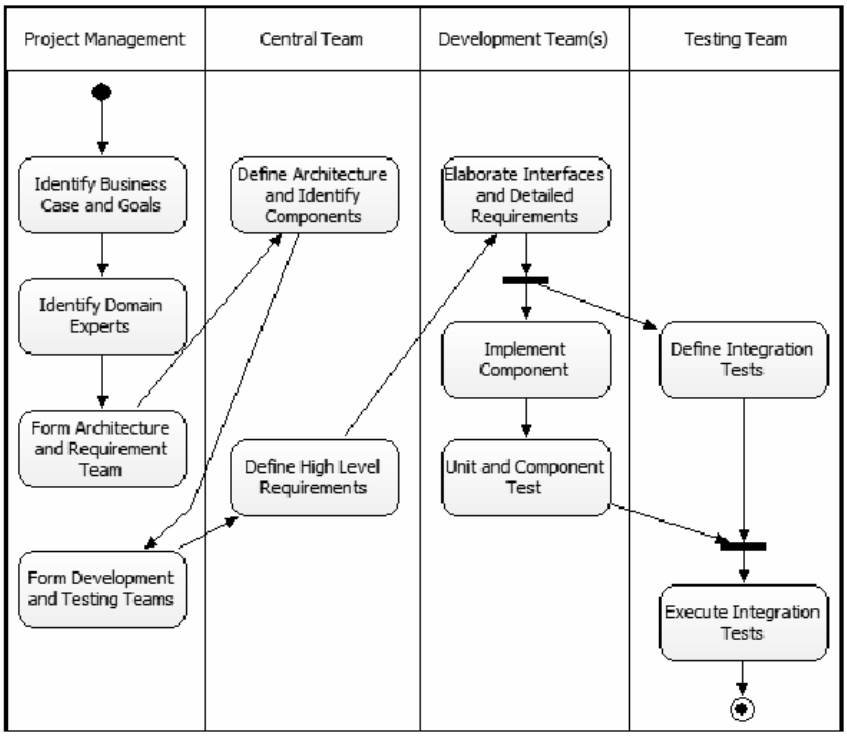

Figura 3.3: Responsabilidades de cada time segundo o processo *LACOI-PD*.

onde este processo foi aplicado a interface comum identificada para os implementar componentes de integração foi o padrão PMIF (Smith & Llado. 2004).

- O time central identifica um domínio de expertise para cada um dos componentes. Estes podem ser grupos de pesquisa localizados em universidades espalhadas pelo mundo.
- O time central identifica um time de desenvolvimento remoto capaz de implementar o componente.
- O time central identifica um time de testes remoto para executar os testes de integração do produto.

3.4 Análise Comparativa

A seguir serão apresentadas análises comparativas entre os processos de desenvolvimento de software apresentados nas seções 3.1, 3.2 e 3.3 com o processo de desenvolvimento ágil *Extreme Programming*, utilizado no projeto do estudo de caso desta pesquisa. Para isto, foi identificado um conjunto de critérios de comparação baseando-se nos principais desafios enfrentados atualmente pelos projetos GSD segundo a visão de diversos autores entre eles (Damian 2002, Carmel & Tija 2005, Borchers 2003). A Tabela 3.1 apresenta os critérios sob uma perspectiva dos desafios de GSD. Visando identificar qual dos processos

Tabela 3.1: Critérios de análise dos processos sob a perspectivas dos desafios de GSD

A - Diferenças Sócio-Culturais	
Critério	**Descrição**
1. Diversidade Cultural	Este critério busca identificar se o processo analisado tem a preocupação de observar as diferentes culturas envolvidas no projeto que podem em alguns casos ser a origem de falhas na comunicação, desconfiança e. frustrações entre as equipes remotas.

B - Dispersão Geográfica	
Critério	**Descrição**
1. Distâncias geográficas	Este critério busca identificar se o processo analisado permite que os times distribuídos explorem novos mecanismos para minimizar os impactos dos problemas gerados pelas grande distâncias.

C - Coordenação e Controle	
Critério	**Descrição**
1. Centralizado	Este critério busca identificar se o processo analisado caracteriza-se por uma centralização do controle das atividades do projeto.
2. Descentralizado	Este critério busca identificar se o processo analisado caracteriza-se por uma descentralização do controle das atividades do projeto.
3. Atribuição dos Papéis	Identifica se o processo possui uma organização rígida na definição dos papéis a serem desempenhado por cada time.
4. Métricas	Identifica se o processo trabalha com métricas de qualidade.
5. Acompanhamento	Identifica se o processo caracteriza-se por acompanhar periodicamente a evolução dos trabalhos dos times remotos através de reuniões semanais, relatórios e pesquisas de assiduidade e participação dos times.

D - Comunicação	
Critério	**Descrição**
1. Informalidade	Este critério busca caracterizar se o processo analisado promovia a comunicação informal entre os membros dos times remotos.
2. Comunicação aberta	Este critério identifica se o processo permitia que os problemas pudessem ser solucionados por qualquer indivíduo do time que tivesse detectado o problema, diretamente com o colega de outro time ou mesmo com o gerente central do projeto.
3. Colaboração	Este critério verifica se o processo permite o uso de ferramentas de colaboração entre os times como meio de compartilhamento de informações de projeto e detalhamentos sobre o domínio de negócio.
4. Requisitos	Este critério busca caracterizar se o processo analisado trabalhava a questão dos requisitos através de especificações formais.

E - Espírito de equipe	
Critério	**Descrição**
1. Comprometimento	Este critério busca caracterizar se o processo demonstra-se capaz de manter os times comprometidos e alinhados com os objetivos do projeto.
2. Confiança	Identifica se o processo é capaz de promover confiança entre os membros dos diferentes time remotos com o time central.

consegue reduzir os impactos dos problemas enfrentados pelos projetos de desenvolvimento de software em um ambiente GSD. Além disto busca-se também identificar qual dos processos melhor atende as necessidades dos projetos GSD.

Estruturar critérios de análise sempre é uma atividade que requer um considerável esforço intelectual, que por muitas vezes dá margem para questionamentos que podem ir além do escopo deste trabalho. Não se espera limitar os processos a este conjunto de critérios de análise. Porém esta primeira formulação pode ser utilizada como fonte de inspiração para estudos futuros sobre estes mesmos processos uma vez que os critérios foram derivados tomando-se por base as características presentes em cada um dos processos.

Utilizaram-se como base teórica, informações provenientes de artigos científicos que comparavam projetos de desenvolvimento global que utilizaram os processos. Foram realizadas entrevistas com os indivíduos que tiveram a oportunidade de executar os processos tanto do lado do time central quanto dos times remotos.

A Tabela 3.2 apresenta a análise comparativa entre os processos. No cabeçalho ela possui a numeração dos respectivos critérios utilizados na análise crítica dos processos. Quando o processo contempla o critério, um símbolo ($\surd$) é introduzido na relação processo e critério. Quando não há o símbolo, o processo não contempla o critério analisado.

Tabela 3.2: Critérios de análise dos processos sob a perspectivas dos desafios de GSD

Processo	**Critérios de Análise**												
	A1	B1	C1	C2	C3	C4	C5	D1	D2	D3	D4	E1	E2
Extended Workbench			$\surd$	$\surd$		$\surd$	$\surd$				$\surd$		
LAGPRO			$\surd$	$\surd$		$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$
LACOI-PD	$\surd$	$\surd$		$\surd$			$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
XP	$\surd$	$\surd$		$\surd$		$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$

Capítulo 4

Método de Pesquisa

Após uma extensa revisão teórica, percebeu-se que o problema apresentado é muito atual e ainda não foi abordado sob a mesma perspectiva. Desta forma, esta pesquisa caracteriza-se como um estudo predominantemente exploratório. Do ponto de vista de (Yin 2005), uma pesquisa exploratória tem enfoque principal em desenvolver, esclarecer e alterar idéias e conceitos, com vistas à formulação de novas teorias, modelos e hipóteses pesquisáveis em estudos futuros. Na maioria das vezes, o tema principal é genérico, tornando-se necessário seu esclarecimento e delimitação, exigindo uma consistente revisão literária, debates com especialistas entre outros procedimentos.

Neste estudo, foram utilizados ambos métodos quantitativos e qualitativos. Os métodos quantitativos tiveram como finalidade tornar os resultados do estudo de caso mais confiáveis. E os métodos qualitativos forneceram informações adicionais que auxiliaram na análise dos dados quantitativos.

4.1 Projeto de Pesquisa

A pesquisa exploratória muitas vezes constitui-se na primeira etapa de uma investigação mais ampla, que é o caso deste estudo. Por esta razão o método de pesquisa utilizado é o estudo de caso conforme proposto por (Yin 2001). O estudo de caso desta pesquisa possui uma unidade de análise única a qual encontra-se inserida no contexto GSD. De acordo com (Yin 2001), o estudo de caso único é um projeto adequado em diversas circunstâncias.

Observou-se um estudo de caso em um ambiente de desenvolvimento global de software. Este estudo de caso foi baseado em estudos de caso realizados sobre o mesmo projeto, *Global Studio Project*, porém em suas edições anteriores. A Figura 4.1 ilustra as fases que compuseram esta pesquisa.

A coleta de evidências do estudo de caso utilizou três fontes de evidências:

- Observação Participante
- Análises dos gráficos das Redes Sociais (SNA)

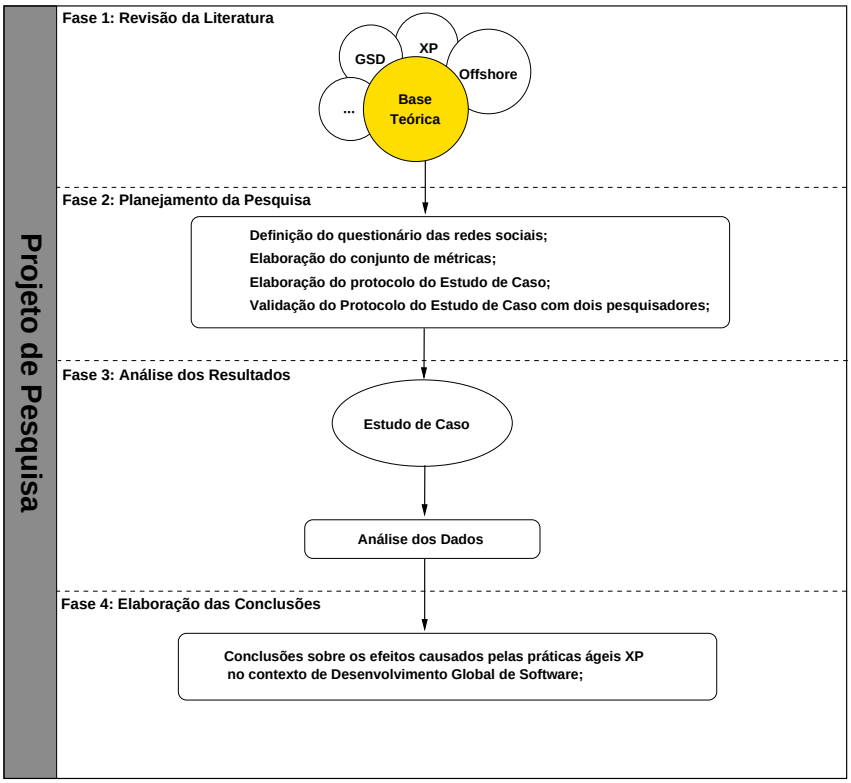

Figura 4.1: Projeto de Pesquisa.

- Roteiro de Entrevistas

Na fase 1 foi realizado um extenso estudo da base teórica com relação as principais características de desenvolvimento global de software. Através deste estudo foram identificados os principais desafios atualmente enfrentados na área da engenharia global de software (GSE).

Na fase 2 da pesquisa, ocorreu a definição do questionário das redes sociais e a definição do conjunto de métricas utilizadas para avaliar a produtividade do time de desenvolvimento observado no estudo de caso. Nós decidimos utilizar métricas para a geração de mais dados quantitativos, a fim de reforçar as conclusões desta pesquisa.

Durante a terceira fase, ocorreu a condução do estudo de caso da pesquisa. A medida que o projeto do estudo de caso se desenvolvia ocorriam as coletas dos dados através dos questionários *on-line* e através de reuniões semanais entre o pesquisador e os membros da equipe. Uma característica marcante desta coleta de dados, foi a utilização da fonte de evidências observação participante proposta por (Yin 2001), isto porque o pesquisador deste estudo também teve participação ativa nas atividades do estudo de caso, tendo este desempenhando um papel de facilitador da metodologia ágil XP durante o projeto do estudo de caso. Segundo (Yin 2005), a técnica de coleta de dados observação participante fornece determinadas oportunidades incomuns em um estudo de caso. Dentre estas oportunidades a mais interessante relaciona-se a possibilidade de se obter permissão para participar de eventos ou grupos que são de certa maneira, inacessíveis a

uma investigação científica. Outra oportunidade interessante apontada por (Yin 2005), é a capacidade de se perceber a realidade a partir do ponto de vista de alguém que esteja diretamente envolvido com o estudo de caso, e não de um ponto de vista de um observador externo. Muitas pessoas argumentam que tal perspectiva tem valor inestimável quando seu resultado reproduz um retrato acurado do estudo de Caso.

Apesar dos dados qualitativos e quantitativos coletados durante o estudo de caso, devem-se ter claras as limitações deste tipo de pesquisa, principalmente no que se refere ao número de estudos de caso e ao restrito contexto no qual a pesquisa foi realizada, determinando uma restrição na generalização dos resultados obtidos e limitando as conclusões desta pesquisa.

4.2 Estudo de Caso

De acordo com o projeto de pesquisa, foi conduzido um estudo de caso único, com o objetivo de identificar os efeitos causados nos desafios de GSD quando fossem adotadas práticas da metodologia XP sobre uma equipe de desenvolvimento co-localizada inserida em um contexto GSD. Inicialmente foi elaborado o protocolo do estudo de caso, definindo o objetivo, escopo, unidade de análise, procedimentos, dimensões e finalmente as questões que constituem o instrumento da pesquisa. A unidade de análise deste estudo de caso está descrita na seção abaixo.

4.2.1 Unidade do Estudo de Caso

A unidade de análise do estudo de caso foi a terceira versão do projeto de desenvolvimento de software distribuído *Global Studio Project* (GSPv3.0). As versões anteriores deste mesmo projeto foram apresentadas na seção 3.1. A versão 3.0 do projeto GSP contou com a participação de três universidades localizadas em 2 continentes e um instituto de engenharia de software, o *Siemens Corporate Research* (SCR), localizado nos Estados Unidos.

Capítulo 5

Resultados do Estudo de Caso

Neste capítulo são apresentados os resultados e análises do estudo de caso conduzido durante o projeto GSP. A seção 5.1 apresenta as características do projeto GSP versão 3.0. A seção 5.2 apresenta a metodologia específica do estudo de caso. A seção 5.3 apresenta o instrumento de pesquisa. A seção 5.4 apresenta a análise dos resultados do estudo de caso.

5.1 Características do Projeto GSPv3.0

O projeto GSPv3.0 foi o projeto de desenvolvimento global de software utilizado por este estudo para extrair os dados sobre a aplicabilidade das práticas de desenvolvimento ágeis XP no contexto de GSD. O projeto contou com a participação de três universidades localizadas em 2 continentes e um instituto de pesquisas em engenharia de software, o *Siemens Corporate Research* (SCR) localizado nos EUA. Toda a gerência do projeto foi realizada por um time central alocado no SCR. A responsabilidade dos times remotos era implementar os componentes distribuídos definidos pela arquitetura definida pelo time central. Alguns dos componentes já estavam implementados, sendo apenas customizados para integrar com os novos componentes.

O time central seguiu o processo *LACOI-PD* descrito na seção 3.3. De acordo com este processo, cada time era totalmente responsável pela codificação de um determinado componente. Para que os componentes pudessem ser integrados com outros componentes, cada equipe remota precisou codificar o que se denominou de programa adaptador. Este programa tinha a função de comunicar os componentes uns aos outros. A Tabela 5.1 mostra quais componentes foram desenvolvidos por cada uma das equipes remotas.

5.1.1 O Componente MOSQUITO

Conforme dito acima, cada time remoto tinha sob sua responsabilidade a implementação de um componente. No caso da equipe italiana, o componente implementado foi o com-

Tabela 5.1: Definição das responsabilidades dos componentes distribuídos no time global do GSPv3.0.

País	**Time**	**Responsabilidades**
EUA	SCR	- Gerência e coordenação global do projeto.
Itália	*L'Aquila*	- Implementação do componente Mosquito.
Brasil	UFRJ	- Implementação do programa adaptador Tangram.
	PUCRS	- Implementação do programa adaptador CQN. - Implementação do componente CQN. - Testes de Integração.
	Chemtech	- Implementação do Componente Tangram.

ponente *Web-Service* batizado com o nome MOSQUITO pela equipe italiana. Devido ao fato de sua função ser essencial para todo o processo de tradução dos diagramas UML para modelos de avaliação e desempenho, o MOSQUITO era considerado o principal componente dentro da arquitetura proposta.

Basicamente o funcionamento do componente MOSQUITO era realizar a leitura de uma entrada de arquivos no formato XML, os quais eram a representação textual de dois diagramas UML, mais especificamente um diagrama de seqüência e outro de componentes, ambos anotados com parâmetros de avaliação e desempenho informados pelo usuário. A partir desta entrada de dados o componente MOSQUITO processava e devolvia um arquivo de saída no formato PMIF (Smith & Llado. 2004).

5.1.2 O Componente TANGRAM

A equipe da universidade UFRJ ficou com a responsabilidade de implementar um programa adaptador que convertesse o arquivo de saída gerado pelo componente MOSQUITO, para o formato de entrada do componente de avaliação e desempenho desenvolvido pela universidade carioca chamado TANGRAM. Para que ele pudesse ser utilizado pelos outros componentes foi necessário desenvolver um programa adaptador que tinha a função de percorrer cada linha do arquivo PMIF, traduzindo-o para um novo arquivo no formato compreendido pelo TANGRAM.

Além da escrita do adaptador, a equipe da UFRJ desenvolveu através de uma empresa terceirizada um componente *Web-Service*. Este componente atendia as chamadas via Internet para a execução do adaptador construído para o TANGRAM. Basicamente o trabalho do Web-Service era de atender as requisições dos usuários recebendo como

parâmetros de entrada o arquivo PMIF e um endereço de e-mail para onde os resultados da simulação do modelo informado deveriam ser retornados. Estas informações eram então repassadas pelo Web-Service para o adaptador, que por sua vez traduzia a entrada de dados e algo compreendido pelo TANGRAM, que por sua vez realizava a simulação do modelo de avaliação e desempenho informado, retornando os resultados para o programa cliente, o cliente então disparava um e-mail anexando os resultados da simulação para o endereço informado nos parâmetros de entrada.

5.1.3 O Componente CQN

O objetivo do projeto do estudo de caso realizado pela equipe da universidade PUCRS foi o de implementar o componente CQN. CQN é uma ferramenta de avaliação e desempenho que a partir de um conjunto de parâmetros de entrada calcula um conjunto de métricas de avaliação e desempenho, tal ferramenta foi desenvolvida pelo grupo de pesquisa PEG (Brenner 2001). O componente desenvolvido pela equipe desenvolvimento da PUCRS no estudo de caso era composto por três partes: a primeira era um programa cliente responsável por realizar chamadas remotas através da *Internet* para um serviço, a segunda parte era um programa chamado de adaptador que basicamente processava um arquivo de entrada no formato PMIF e o convertia em um arquivo de saída no formato de entrada da ferramenta CQN. A terceira parte do componente era o serviço de Internet, *WebService*, que encapsulava o programa adaptador e o disponibilizava na Internet para atender as requisições feitas pelo cliente. A Figura 5.1 ilustra o funcionamento do componente CQN quando o cliente faz uma requisição.

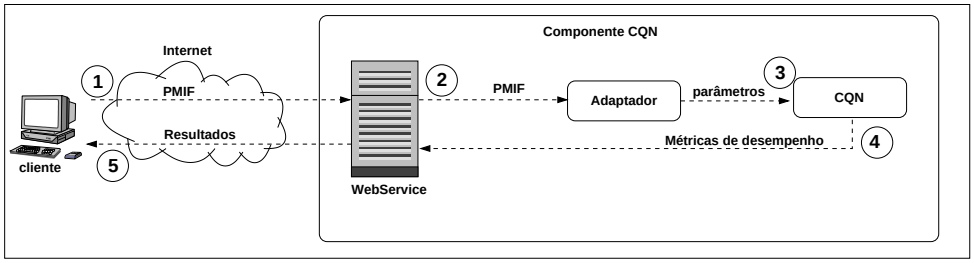

Figura 5.1: Seqüência das operações executadas pelo componente CQN quando o cliente faz uma requisição.

5.1.4 O Teste de Integração

O planejamento e a execução do teste de integração contou com uma forte participação da equipe remota da universidade PUCRS. O planejamento do teste de integração teve grande parte de sua definição realizada pela equipe brasileira em parceria com o time central situado nos EUA. Isto ocorreu pelo fato de que integrantes da equipe brasileira

viajaram para o time central nos EUA e permaneceram lá durante algumas semanas a fim de reforçar o entendimento das responsabilidades de cada time no projeto.

Logo após a definição da estratégia do teste de integração ser finalizada, o líder de testes da equipe PUCRS iniciou a elaboração dos cenários de testes. Um pré-requisito encontrado pelo time de testes, foi a necessidade em se utilizar um plug-in da ferramenta IDE Java Eclipse para a modelagem dos diagramas UML utilizados nos cenários de testes. O plug-in utilizado para isto foi o OMONDO, o qual precisou ser estudado pela equipe. Todos os diagramas UML dos casos de testes, foram construídos usando o plug-in OMONDO devido ao suporte de integração que o componente distribuído MOSQUITO já oferecia para este plug-in.

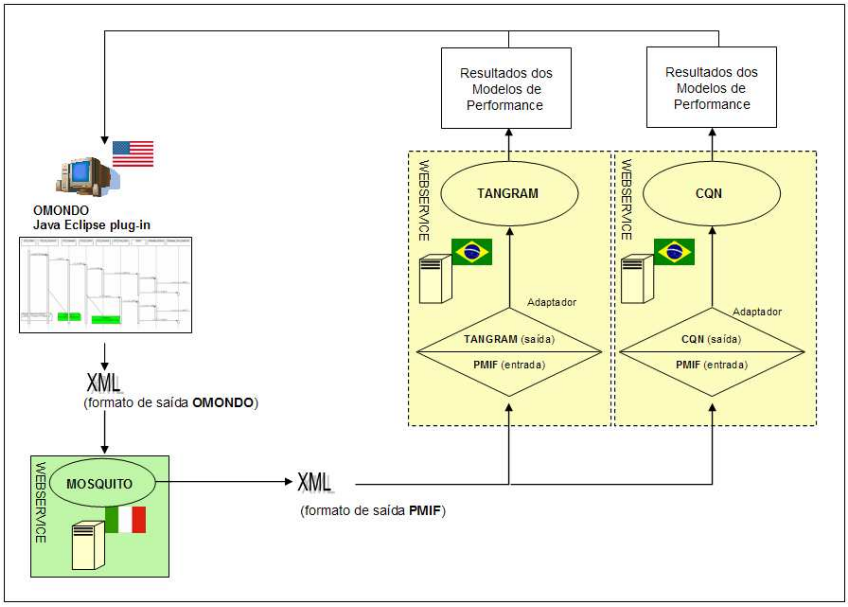

Figura 5.2: Arquitetura global da aplicação distribuída do projeto GSPv3.0.

O objetivo do teste de integração era verificar a existência de eventuais problemas na comunicação entre os componentes distribuídos definidos na arquitetura global ilustrada na Figura 5.2. O objetivo do time central ao definir tal arquitetura foi o de implementar uma aplicação capaz de gerar modelos de avaliação e desempenho a partir de diagramas UML usando diversos métodos de avaliação e desempenho de sistemas.

Devido ao fato de que cada equipe precisou implementar um programa adaptador que permitisse que seu componente compreendesse o arquivo PMIF, foram necessárias várias execuções dos testes até que a comunicação entre os componentes estivesse funcionando corretamente. Isto porque, o componente MOSQUITO estava executando em um servidor localizado na universidade *L'Aquila* na Itália, enquanto que o *Web-Service* e o adaptador do TANGRAM estavam localizados em servidores diferentes na universidade UFRJ no Brasil.

Visando validar a integração de todos os componentes distribuídos, os casos de

testes foram construídos utilizando a estratégia descrita na seqüência de passos abaixo:

Passo 1: Através de uma máquina cliente o testador elaborava os diagramas UML utilizando para isto o plug-in do Eclipse OMONDO.

Passo 2: Uma vez tendo finalizado o diagrama de seqüência e o diagrama de componentes o testador invocava através de um plug-in do Eclipse uma chamada remota para o componente MOSQUITO passando como parâmetros ambos diagramas UML.

Passo 3: O componente MOSQUITO após processar a requisição do testador retornava como resposta um arquivo no formato PMIF.

Passo 4: O testador por sua vez ao receber o arquivo PMIF resultante realizava a segunda chamada remota de método para um dos componentes de avaliação e desempenho disponíveis.

Passo 5: Caso o testador decidisse enviar o arquivo PMIF para o componente TANGRAM, o mesmo realizava uma chamada remota para o Web-Service desenvolvido especificamente para o TANGRAM, utilizando novamente um plug-in desenvolvido para o Eclipse o qual era encarregado de enviar os parâmetros para o Web-Service. A resposta do Web-Service eram os resultados da simulação do modelo de performance os quais eram finalmente enviados para o endereço de e-mail do testador.

Passo 6: Caso o testador optasse em enviar o arquivo PMIF para o componente CQN, o mesmo realizava uma chamada remota para o Web-Service desenvolvido especificamente para o CQN, através de um plug-in desenvolvido para o Eclipse, o qual encarregava-se de enviar os parâmetros para o Web-Service. A resposta do Web-Service eram os resultados da simulação do modelo de performance os quais eram finalmente enviados para o endereço de e-mail do testador.

Os passos descritos acima foram repetidos exaustivamente durante a execução dos testes de integração, até que todos os problemas de comunicação entre os componentes fossem resolvidos.

5.1.5 Execução das Práticas Ágeis no Projeto GSPv3.0

Para coordenar as atividades do projeto GSP de acordo com a metodologia XP, utilizou-se a prática *Planning Game*. Foi adotada a ferramenta *XPlanner* (sourceforge.net 2006) para controlar as iterações, requisitos, cronograma e as atividades do projeto. Foram realizadas ao longo de todo o estudo de caso um total de quatro iterações que compreenderam 4 meses de trabalho. Cada iteração destinava-se a atender um determinado conjunto requisitos do usuário, porém estes eram chamados de histórias do usuário ou *user history* conforme

dita a metodologia ágil. A Figura 5.3 obtida através da ferramenta XPlanner apresenta a seqüência das iterações conduzidas pelo time de desenvolvimento da PUCRS.

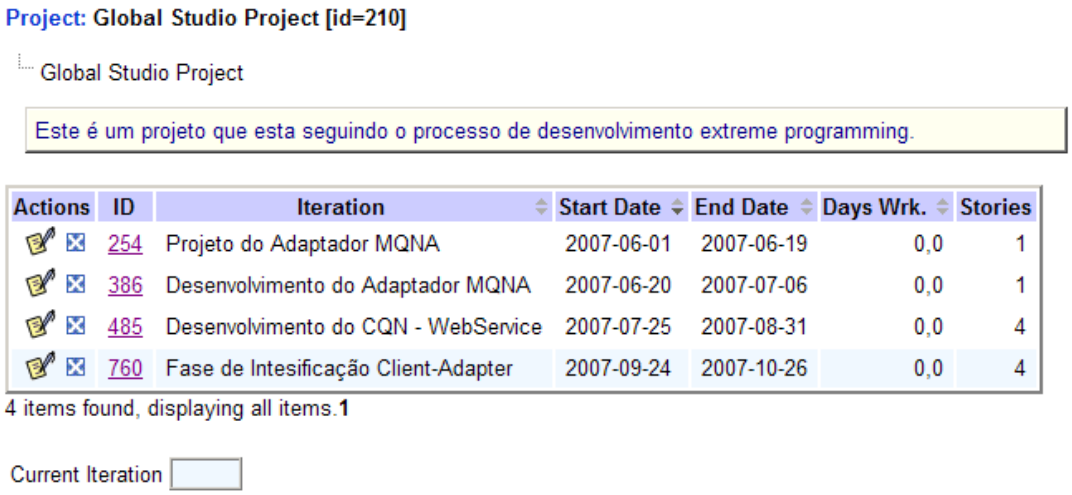

Figura 5.3: Iterações conduzidas no projeto do estudo de caso gerenciadas através da ferramenta *XPlanner*.

Uma vez que o cliente passava sua história para o time de desenvolvimento, ela era traduzida em um conjunto de tarefas, onde cada tarefa era estimada e priorizada. Todas as tarefas foram executadas por duplas de desenvolvedores conforme orienta a prática de desenvolvimento *Pair Programming*.

A equipe de desenvolvimento era formada por quatro estudantes de graduação e dois estudantes de pós-graduação. A Tabela 5.2 apresenta o papel de cada indivíduo dentro do XP. Os dois estudantes de pós-graduação desempenharam os papéis mais importantes em XP, o primeiro papel era o chamado *coach*, que é a pessoa que detém maior experiência dentro da equipe, sendo responsável por guiar o time na elaboração de soluções para questões complexas (Warden et al. 2003). O segundo papel chave definido no XP era o chamado *tracker*, que é a pessoa responsável pelo gerenciamento do cronograma do projeto e pela coleta das métricas.

Sempre que necessário, os próprios desenvolvedores atualizavam a ferramenta de gerenciamento de projeto, neste projeto foi a ferramenta *XPlanner*, a fim de alterar ou adicionar tarefas que estes julgassem necessárias para a finalização de uma *User History*. As dúvidas referentes aos requisitos eram esclarecidas diretamente com o cliente, conforme proposto pela metodologia XP, todos os membros do time tinham a liberdade de entrar em contato com o cliente para resolver problemas de interpretação geradas por requisitos mal escritos ou ambíguos.

Uma das constatações desta pesquisa foi a corroboração com a teoria de que a flexibilidade de XP traz benefícios para a auto-estima da equipe, a abertura concedida aos programadores para contatar o cliente, alterar requisitos, adicionar tarefas nas iterações,

Tabela 5.2: Estrutura da equipe ágil *Extreme Programming* do estudo de caso.

Indivíduo	**Papel**
RU	*tracker*
FF	*coach*
LM	Desenvolvedor
VT	Desenvolvedor
VW	Desenvolvedor
EM	Desenvolvedor

fez com que os indivíduos desenvolvessem rapidamente um forte sentimento de comprometimento com os objetivos do projeto. A medida que o tempo passava, os indivíduos percebiam cada vez mais claramente quais os efeitos de suas ações para o andamento do projeto, fossem elas boas ou ruins. Isto ocorreu devido a constante troca de *feedback* entre os próprios membros da equipe e em alguns casos com o cliente.

5.2 Metodologia do Estudo de Caso

O estudo de caso desta pesquisa foi organizado em 3 (três) etapas: planejamento, execução e análise dos resultados. A Figura 5.4 ilustra as etapas do estudo de caso com as suas respectivas atividades.

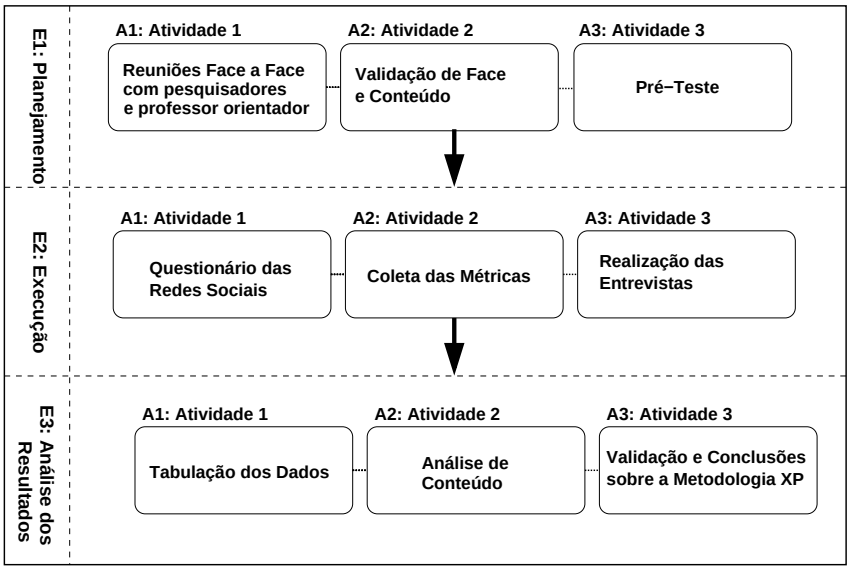

Figura 5.4: Etapas do Estudo de Caso.

A etapa 1 (E1) caracterizou-se como o planejamento do estudo de caso, foram elaborados o protocolo do estudo de caso, o roteiro das entrevistas e o conjunto de métricas e a implementação do questionário das redes sociais (SNA). Para isto foram executadas três atividades. Na primeira atividade (A1), ocorreram as reuniões face a face

com pesquisadores da área de engenharia de software nos Estados Unidos para a elaboração de uma ferramenta de coleta de dados das redes sociais. Foram realizadas também as tele-conferências com o pesquisador sênior e o professor orientador para definição das métricas, levantamento das questões do protocolo do estudo de caso e a estruturação do roteiro das entrevistas. Na segunda atividade (A2), ocorreu a validação de face e conteúdo do protocolo do estudo de caso por 2 pesquisadores seniores. Na terceira e última atividade desta etapa (A3), ocorreu a realização do pré-teste do questionário do protocolo do estudo de caso. Os entrevistados durante o pré-teste não tiveram suas respostas consideradas nesta pesquisa durante a análise dos resultados. O pré-teste serviu apenas para observar o quanto estavam claras as questões do questionário e se as respostas refletiam o que estava sendo perguntado. O pré-teste serviu para aprimorar e ajustar o questionário do protocolo do estudo de caso.

A etapa 2 (E2), foi a etapa mais longa do estudo de caso. Nela desenvolveram-se três atividades de coleta de dados. A atividade 1 (A1) foi o preenchimento do questionário das redes sociais a cada duas semanas, a atividade 2 (A2) foi a coleta das métricas de produtividade nas reuniões semanais entre o pesquisador e a equipe de desenvolvimento PUCRS e por último na atividade 3 (A3), realizaram-se as entrevistas usando o questionário do protocolo do estudo de caso visando obter a percepção dos membros do time da PUCRS sobre a eficiência do processo de desenvolvimento de software adotado durante as atividades do estudo de caso com relação aos desafios de GSD.

Finalmente na etapa 3 (E3), aconteceram as três atividades de análise dos resultados coletados durante o estudo de caso. Na atividade 1 (A1), foi realizada a tabulação dos dados quantitativos coletados através das redes sociais e a partir das métricas, também se tabulou os dados qualitativos retirados das respostas do questionário do protocolo do estudo de caso. Na atividade 2 (A2), ocorreu a consolidação e a triangularização dos resultados e algumas comparações entre a versão 3.0 e a versão 2.0 do projeto GSP para os valores coletados para as métricas. Na atividade 3 (A3), formularam-se as lições aprendidas sobre a utilização da metodologia ágil *XP* em projetos GSD, confrontando as vantagens e desvantagens observadas na prática com as experiência relatadas na teoria pesquisada.

5.3 Instrumento da Pesquisa

O instrumento da pesquisa foi composto por três mecanismos de coleta de dados. A Figura 5.5 ilustra esquematicamente as partes que constituíram o instrumento desta pesquisa.

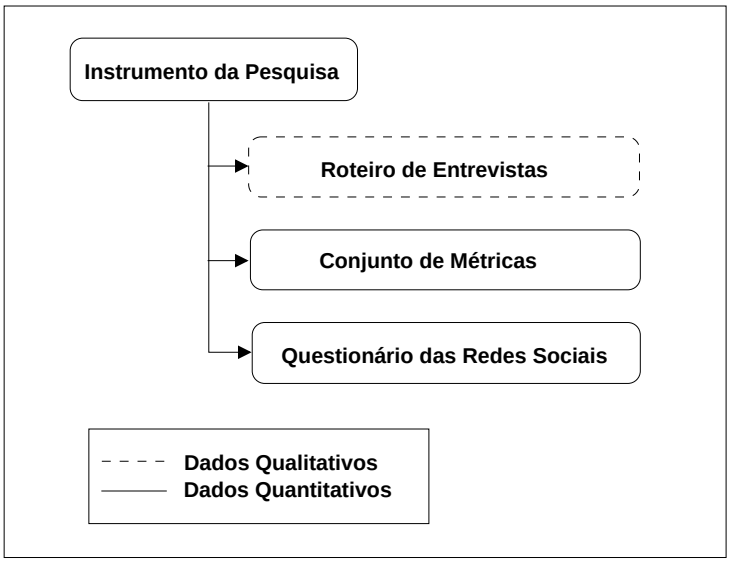

Figura 5.5: Composição do instrumento da pesquisa.

5.3.1 Questionário das Redes Sociais

O questionário das redes sociais coletou dados sobre a comunicação e a colaboração entre os times distribuídos. Todos os membros das equipes distribuídas respondiam o questionário a cada duas semanas. Os dados coletados por este questionário representam as interações entre as equipes. O acesso as respostas do questionário das redes sociais foi autorizado pelo instituto de engenharia de software *Siemens Corporate Research* para utilização neste estudo.

5.3.2 Conjunto de Métricas

Foi definido um conjunto de métricas para avaliar a produtividade da equipe de desenvolvimento observada no estudo de caso e também validar a metodologia *Extreme Programming* utilizada pela equipe. A decisão de incluir métricas no instrumento foi sustentada pela necessidade de uma coleta de dados quantitativos que tornassem as conclusões desta pesquisa mais confiáveis.

As métricas foram organizadas em três grupos. Cada grupo de métricas era coletado em uma fase específica do projeto. A Figura 5.6 ilustra as fases do projeto do estudo de caso em uma ordem cronológica e o respectivo grupo de métricas instrumentado em cada fase. A descrição detalhada das métricas bem como de seus grupos encontra-se no Apêndice C.

5.3.3 Roteiro de Entrevistas

Foi elaborado um questionário composto por cinco dimensões (Apêndice A). O objetivo do questionário foi o de identificar quais desafios de GSD são mais ou menos afetados pela

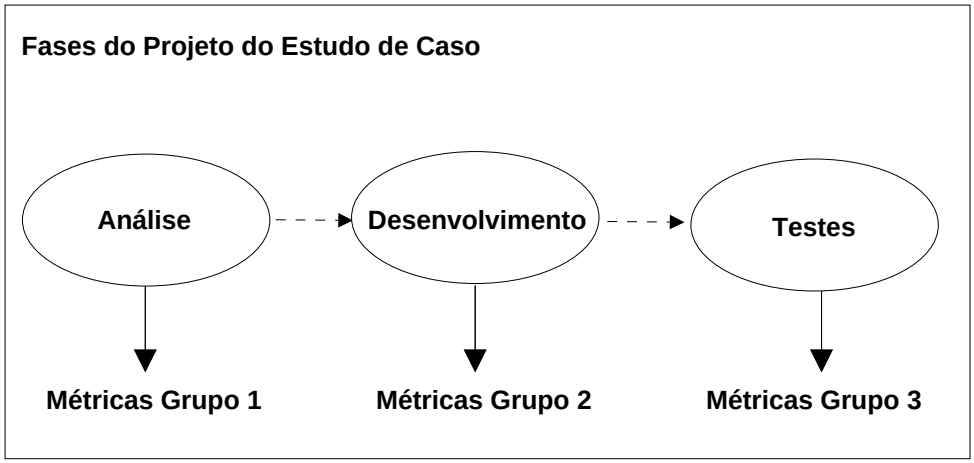

Figura 5.6: Fases do projeto do estudo de caso e o seu respectivo grupo de métricas.

adoção da metodologia ágil *Extreme Programming*. O instrumento passou por uma validação de face e conteúdo realizada por dois pesquisadores. Com base nas recomendações passadas pelos pesquisadores, o instrumento do estudo de caso foi ajustado de modo a atender o objetivo da pesquisa.

O questionário passou por um pré-teste com alunos do curso de mestrado da universidade PUCRS. Uma vez que a versão final do instrumento foi finalizada iniciaram-se as entrevistas. Os entrevistados foram convidados a responder um conjunto de questões que relacionavam as práticas ágeis XP com os desafios enfrentados em GSD. As questões enfocaram nas conseqüências da utilização de tais práticas ágeis em projeto de desenvolvimento distribuído de software. Após a conclusão das entrevistas, iniciou-se a análise dos resultados coletados através do questionário das redes sociais, métricas e entrevistas, relacionando os dados obtidos com a teoria pesquisada.

Caracterização dos Respondentes

Foram entrevistadas 7 pessoas que participaram da equipe da universidade PUCRS. O tempo médio de resposta gasto por pessoa foi de 16 minutos. Dos sete desenvolvedores entrevistados, quatro deles participaram apenas da versão 3.0 do projeto GSP, dois participaram da versão GSP 2.0 e um desenvolvedor que participou de ambas versões do projeto GSP. Os entrevistados possuem, no mínimo, 1 ano de experiência na área de informática e, no máximo de 10 anos. O tempo médio de experiência dos respondentes é de 4 anos. A média de idade dos entrevistados é de 25 anos, sendo a mínima de 18 e a idade máxima de 32 anos. Comparando o tempo de experiência dos membros da equipe de desenvolvimento da versão 2.0 com a versão 3.0, identificou-se uma grande diferença entre as duas equipes. A Tabela 5.3 apresenta a distribuição do tempo de experiência dos entrevistados nas duas versões do projeto:

Com relação ao nível escolar, os respondentes possuem no mínimo o curso de

Tabela 5.3: Tempo de trabalho dos participantes do projeto GSP.

	Número de Pessoas	
Tempo de Experiência	**versão 2.0**	**versão 3.0**
0-2 anos	0	2
2-4 anos	3	1
6-10 anos	0	2

Tabela 5.4: Análise dos dados demográficos dos indivíduos que participaram do projeto GSP nas versões 2.0 e 3.0.

	Versão 2.0	**Versão 3.0**
Idade Mínima:	21	18
Idade Máxima:	28	32
Escolaridade Mínima:	Ensino Médio	Ensino Médio
Escolaridade Máxima:	Ensino Superior	Ensino Superior
Conhecimento sobre GSD	10% conhece 90% não conhece	28% conhece 72% não conhece
Conhecimento sobre o Processo	0% conhece 100% não conhece	43% conhece 57% não conhece

ensino médio completo e no máximo o curso superior completo. A predominância foi dos respondentes com nível médio completo (58%). Dos respondentes da equipe de desenvolvimento da versão 2.0, 100% informaram desconhecer o processo de desenvolvimento *Extended Workbench* (seção 3.1) contra 57% dos respondentes da versão 3.0, que informaram desconhecer o processo ágil *Extreme Programming*. A Tabela 5.4 apresenta as análises demográficas separadas por versões do projeto GSP.

5.4 Análise dos Dados

Segundo Yin (2005), a análise dos dados é a atividade de examinar, categorizar, classificar, testar e recombinar as evidências. Nesta pesquisa, a análise dos dados foi realizada através da análise de conteúdo. Uma das contribuições deste estudo está na identificação de quais os impactos causados pelas práticas XP quando aplicadas em uma equipe de desenvolvimento inserida em um contexto GSD.

5.4.1 Análise dos Resultados das Redes Sociais (SNA)

Com a adoção da metodologia *Extreme Programming* pela equipe da universidade PUCRS no estudo de caso, identificou-se uma intensa comunicação interna e externa da equipe

brasileira durante o projeto GSPv3.0. A análise desta comunicação será apresentada a seguir em uma seqüência de gráficos das redes sociais formadas durante o projeto. Os nodos dos gráficos das redes sociais representam as pessoas envolvidas no projeto global e as linhas ou arestas representam as relações entre os indivíduos.

Os dados dos gráficos foram coletados a partir das respostas de um questionário SNA *on-line*. Para cada pergunta do questionário existiam cinco alternativas, cada alternativa possuía um peso que variava de 1 a 5. Por esta razão as arestas apresentadas nos gráficos das redes sociais possuem espessuras diferentes. Se a alternativa escolhida pelo respondente tivesse um peso maior ou igual a três, a aresta do gráfico era desenhada em negrito. No caso da alternativa escolhida possuir um peso inferior a três, a espessura da aresta desenhada permanecia fina. O preenchimento do questionário SNA era solicitado pelo time central aos times remotos a cada duas semanas completadas de projeto.

A análise realizada nesta pesquisa através das redes sociais visou avaliar a comunicação da equipe da universidade PUCRS tanto internamente entre seus membros, como também a sua comunicação com as demais equipes distribuídas. Para isto foram escolhidas as seguintes questões do questionário SNA:

- *Questão 1: Durante as duas últimas semanas, qual foi a freqüência com que você comunicou com cada pessoa sobre assuntos referentes ao projeto?*
- *Questão 2: Considerando a importância de sua comunicação com esta pessoa nas últimas duas semanas, o quanto foi importante este contato para ter seu trabalho finalizado?*

Conforme descrito anteriormente, foram definidos dois tipos de arestas possíveis para os gráficos das redes sociais: a aresta fina e a aresta em negrito. Desta forma, para os gráficos SNA gerados a partir das respostas da Questão 1, caso a espessura da aresta seja fina, significa que a comunicação entre as pessoas aconteceu algumas vezes ou uma única vez no decorrer das últimas duas semanas. Caso contrário, se a aresta estiver em negrito, significa dizer que a comunicação entre as pessoas aconteceu de forma intensa. Do mesmo modo para os gráficos SNA gerados para a *Questão 2*, caso a aresta seja fina, significa que o nível de importância dos contatos entre as equipes, atribuído pelos respondentes do time global, oscilou de comunicação moderada à levemente importante. E no caso da aresta aparecer em negrito, significa que a comunicação foi considerada pelos membros dos times como fundamental para a conclusão dos trabalhos das equipes.

A Tabela 5.5 identifica para quais times do projeto global pertenciam os nodos apresentados nos gráficos das redes sociais, juntamente com o papel desempenhado por cada indivíduo no projeto.

Comparando os gráficos das redes sociais apresentados nas Figuras 5.7 e 5.8, pode-se constatar com base nas arestas em negrito que ligam os nodos dos membros da PUCRS

Tabela 5.5: Identificação dos nodos apresentados nos gráficos das redes sociais.

País	Empresa/Universidade	Nodo	Papel
EUA	SCR	AA	Gerente de Projeto
		AB	Pesquisador
ITÁLIA	Universidade L'Aquila	VC	Pesquisador
BRASIL	Universidade UFRJ	FD	Líder Técnico
		GF	Desenvolvedor
BRASIL	Universidade PUCRS	EM	Desenvolvedor
		EN	Líder Técnico
		FF	Líder Técnico
		LM	Desenvolvedor
		MC	Desenvolvedor
		PF	Pesquisador
		RU	Líder Testes
		VT	Desenvolvedor

com os nodos dos membros das equipes globais, que no início do projeto do estudo de caso, todo o fluxo de comunicação entre as equipes distribuídas era centralizado em duas pessoas, os nodos PF e RU respectivamente. Atribui-se este fato como uma conseqüência da presença em pessoa do líder de testes da equipe PUCRS, nodo RU, no time central nas duas primeiras semanas do projeto. Esta permanência possibilitou a realização de vários contatos face a face com os membros do time central, o que contribuiu para aumentar a confiança entre os dois times distribuídos, derrubando as barreiras de comunicação entre o time da PUCRS e o time central nos EUA.

Com relação a comunicação interna da equipe PUCRS, as relações formadas entre os nodos EM,EN,LM,MC,PF e RU apontam que todos os membros mantinham uma comunicação muito freqüente com seus colegas co-localizados e também consideravam importante tais contatos para a finalização de suas atividades.

Na análise dos gráficos das redes sociais para a segunda quinzena do projeto, observou-se uma redução na comunicação entre a PUCRS e os demais times. Isto se deve ao fato do coordenador da equipe ter retornado ao Brasil. A comunicação neste período do projeto caracterizou-se por ser intensa apenas dentro das equipes remotas e os contatos entre os times globais acontecerem apenas algumas vezes. Pode-se identificar a partir do gráfico ilustrado na Figura 5.9 este novo cenário, a freqüente comunicação interna do time central se expressa através das relações entre os nodos AA, AB e FD, todos eles conectados por arestas em negrito. Da mesma forma acontecia internamente com a equipe da PUCRS, onde a comunicação também se apresentava bastante intensa, cenário representado pelas arestas em negrito que ligam os nodos EN, EM, LM, MC, PF

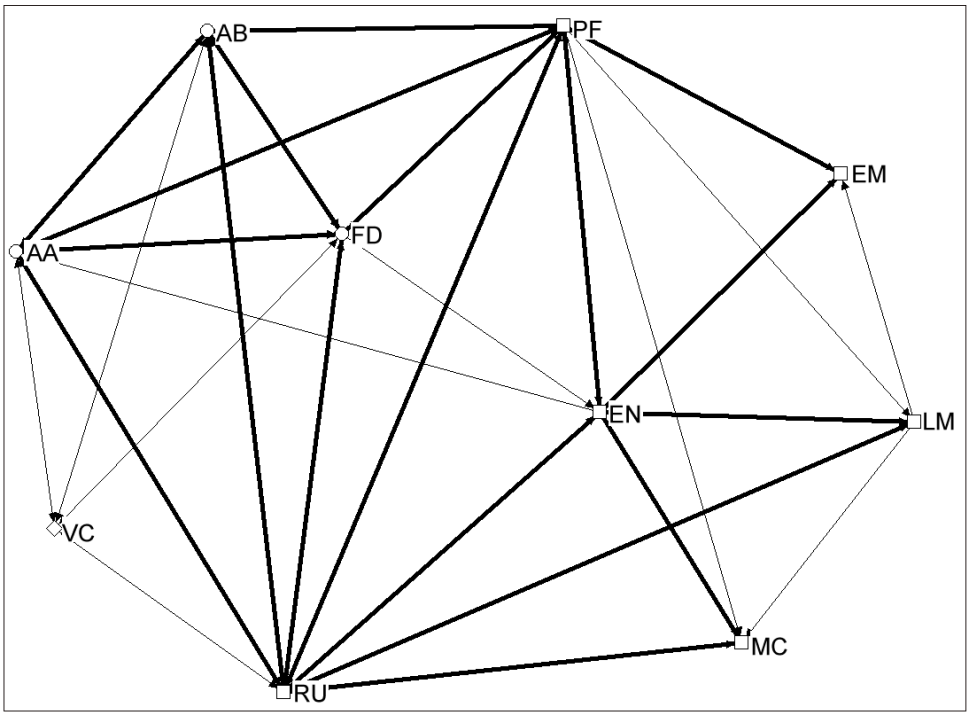

Figura 5.7: Redes sociais formadas a partir das respostas da Questão 1 na primeira quinzena do GSPv3.0.

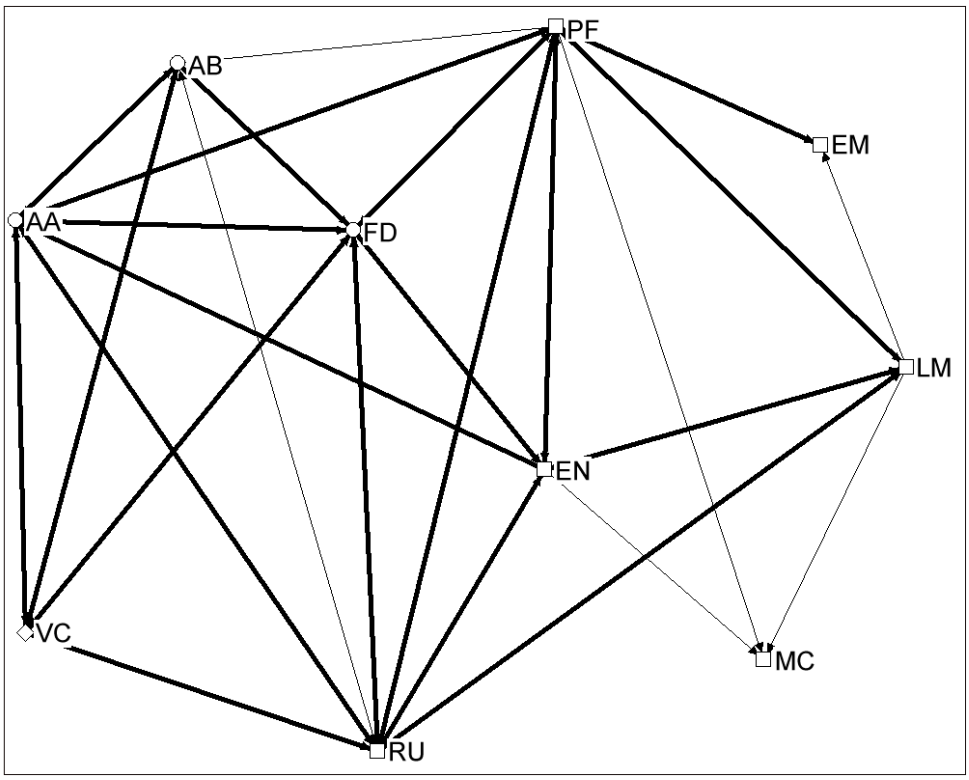

Figura 5.8: Redes sociais formadas a partir das respostas da Questão 2 na primeira quinzena do GSPv3.0.

e RU conforme ilustra a Figura 5.9.

Mesmo com a queda na comunicação entre os times distribuídos, a percepção das equipes quanto a importância da comunicação entre os times continuava bastante sólida. Fato observado através da Figura 5.10, onde várias arestas em negrito ligam diferentes membros das equipes distribuídas.

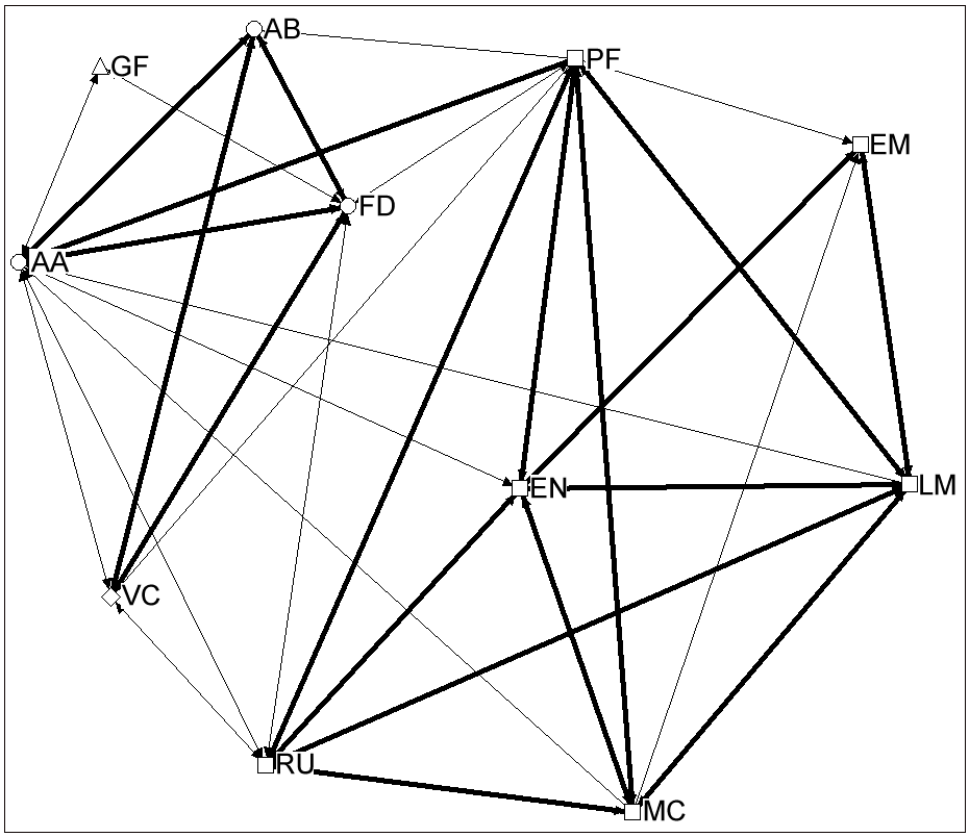

Figura 5.9: Redes sociais formadas a partir das respostas da Questão 1 na segunda quinzena do GSPv3.0.

Na análise das redes sociais para as questões 1 e 2 na terceira quinzena do projeto, percebe-se que o volume de comunicação entre os times remotos voltou a aumentar. Mesmo com as distâncias geográficas e todos os desafios inerentes à GSD, o nível de comunicação do projeto nestas duas semanas atingiu o seu ápice. A Figura 5.11 ilustra o gráfico das redes sociais formadas quase que completamente por arestas em negrito, o que demonstra que o nível de comunicação tanto internamente quanto externamente nas equipes distribuídas foi muito freqüente. Atribui-se esta considerável elevação ao início da etapa de testes do projeto global, conciliado com o início das atividades de desenvolvimento na equipe PUCRS.

Identificou-se também que a comunicação interna da equipe PUCRS atingiu o seu nível máximo possível, isto é, todos os membros da equipe brasileira se comunicavam com muita freqüência. O aumento na comunicação do time PUCRS pode ser explicado pelo efeito que a prática *Pair Programming* causou no comportamento dos membros do

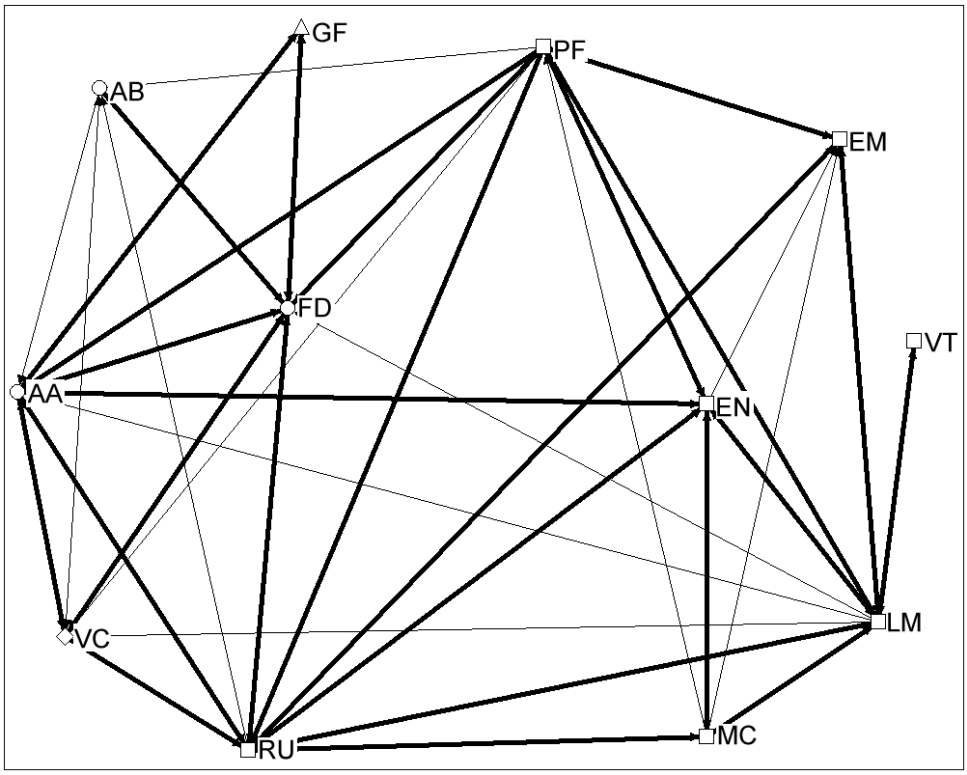

Figura 5.10: Redes sociais formadas a partir das respostas da Questão 2 na segunda quinzena do GSPv3.0.

time. O efeito causado por esta prática ágil está fielmente registrado pelo gráfico SNA apresentado na Figura 5.12, através das relações entre os nodos LM com MC e EM com VT. Tais nodos representam os indivíduos da equipe PUCRS que formaram as duplas de programação durante a terceira quinzena do projeto global.

Ainda com relação a terceira semana do projeto global do estudo de caso, pôde-se constatar através dos gráficos das redes sociais, que alguns desenvolvedores da equipe PUCRS começaram a realizar seus primeiros contatos individuais com os membros das equipes distribuídas. Desta forma, além da comunicação mantida pelo coordenador e pelo líder técnico, os desenvolvedores representados pelos nodos LM e MC apresentados na Figura 5.11, contribuíram para tornar a terceira quinzena, o período do projeto que apresentou os melhores resultados obtidos na comunicação entre as equipes distribuídas registrados através dos gráficos das redes sociais. Acreditamos que este fato atribuiu-se a sensação de liberdade que a *Extreme Programming* concedia aos membros do time, em buscar solucionar suas dúvidas e questionamentos diretamente com o cliente, representado neste projeto pelo time central. Os contatos foram realizados através de *chat* eletrônico, lista de e-mail e fórum de discussões. Os reflexos desta comunicação foram percebidos na aceitação e na motivação demonstrada pelo time em relação a adoção da *Extreme Programming*, conforme constatado nas respostas do questionário do protocolo do estudo de caso.

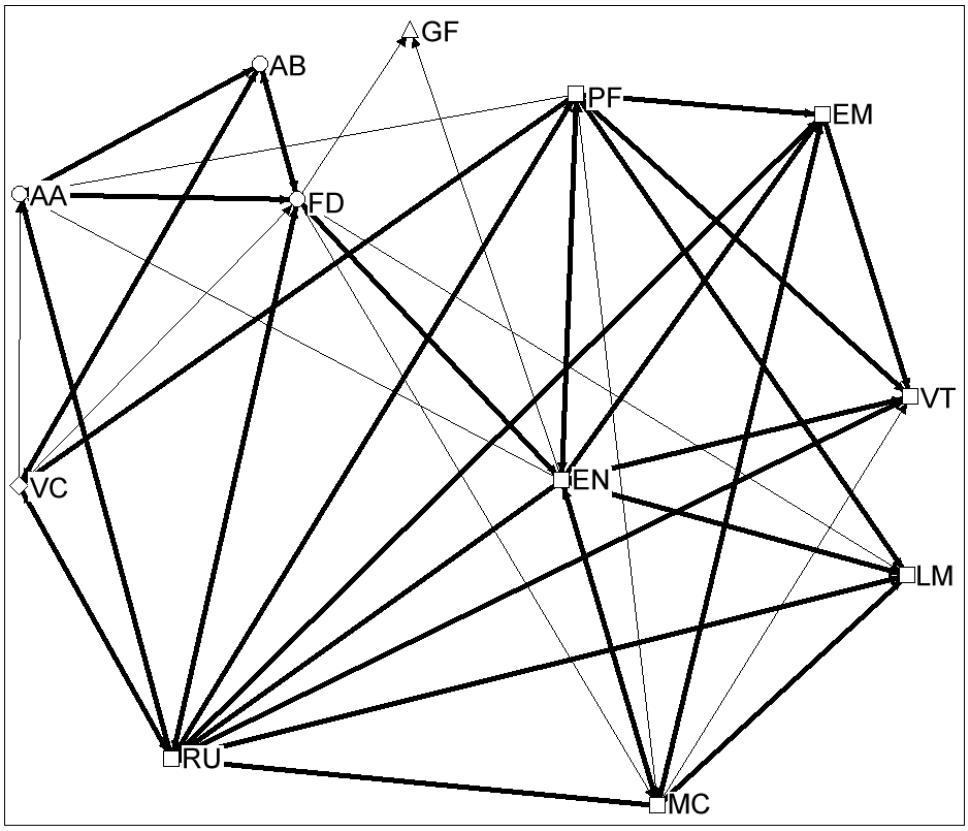

Figura 5.11: Redes sociais formadas a partir das respostas da Questão 1 na terceira quinzena do GSPv3.0.

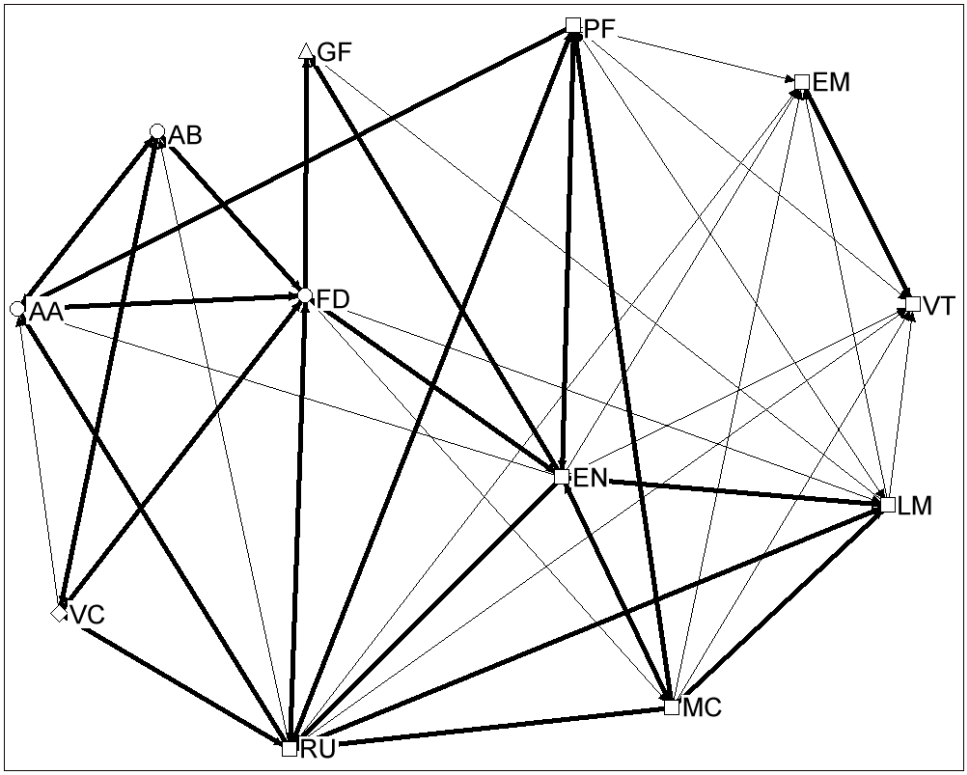

Figura 5.12: Redes sociais formadas a partir das respostas da Questão 2 na terceira quinzena do GSPv3.0.

As redes sociais formadas na quarta quinzena do projeto mostram uma drástica queda na comunicação do time local PUCRS, tanto internamente como externamente com as demais equipes distribuídas. Após um período muito intenso de comunicação, a equipe da PUCRS enfrentou um grave problema com o seu líder técnico. Por motivos fora do escopo desta pesquisa, o líder técnico acabou se ausentando durante toda a quarta quinzena. O maior problema foi que o líder técnico não havia comunicado ninguém sobre sua longa e não planejada ausência. O impacto deste desfalque no restante da equipe pode ser claramente identificado através do gráfico SNA ilustrado na Figura 5.13. O nodo que representa o líder técnico (nodo EN), ficou absolutamente isolado, sem nenhuma ligação com os demais nodos. Com a quebra dos canais de comunicação que eram sustentados pelo líder técnico o nível de comunicação da equipe caiu pela metade. O contato da equipe com os demais times distribuídos só não foi mais afetado porque um desenvolvedor junior, teve uma notável evolução na sua participação no projeto, aliviando em parte os efeitos negativos causados pela saída prematura do membro mais experiente do grupo. Diversos atrasos nas atividades de desenvolvimento da equipe PUCRS foram originados em função deste problema. Os reflexos deste acontecimento acabaram afetando o cronograma do projeto local e conseqüentemente atrasando as etapas de coleta e análise de dados desta pesquisa.

Apesar de todos os problemas citados acima, respeitando uma tendência já observada na quinzena anterior para as respostas da Questão 2, os times globais mantiveram a mesma confiança com relação a importância dos contatos realizados com a equipe brasileira mesmo após estes acontecimentos, conforme ilustra a Figura 5.14. O dado novo identificado nesta quarta quinzena referente a Questão 2, foi que o time central e as equipes globais passaram a comunicar mais freqüentemente com o desenvolvedor junior. As relações entre os nodos AA, FD e GF com o nodo LM ilustradas na Figura 5.14, demonstram claramente a grande importância atribuída pelos times globais para os contatos realizados entre as partes nestas duas semanas de projeto.

Finalmente na última coleta de dados obtida para as redes sociais, identificou-se o retorno da intensa comunicação entre o time da PUCRS com os demais times remotos. A Figura 5.15 ilustra a equipe da PUCRS voltando a se comunicar internamente de forma muito intensa após a substituição do líder técnico por uma nova pessoa, representado no gráfico pelo nodo FF. A entrada deste novo líder técnico, fez a equipe da PUCRS retomar o ritmo forte de trabalho, recuperando sua motivação e auto-estima.

Em relação ao nível de importância da comunicação entre os times distribuídos, a Figura 5.16, ilustra que a tendência apresentada em todas as quinzenas do projeto se manteve constante, isto é, os membros das equipes remotas consideravam o trabalho da equipe PUCRS vital para a conclusão bem sucedida do projeto global.

A análise das redes sociais provou ser uma poderosa ferramenta de pesquisa, pois ela conseguiu retratar com muita fidelidade todos os acontecimentos do estudo de caso,

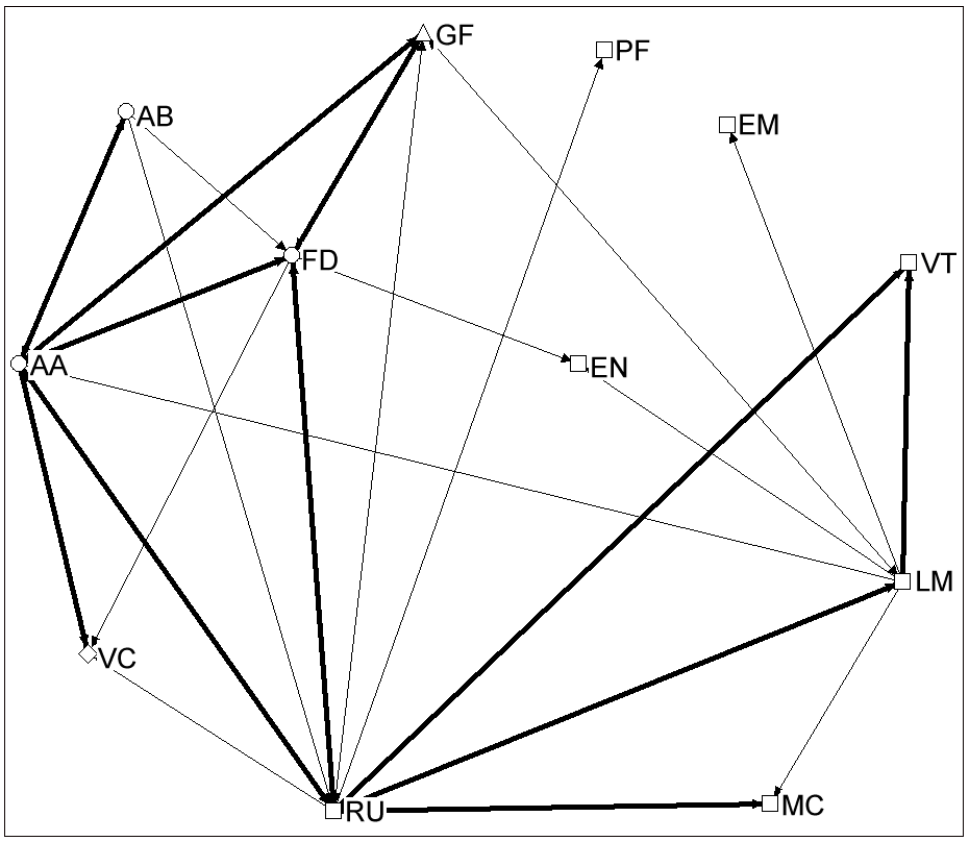

Figura 5.13: Redes sociais formadas a partir das respostas da Questão 1 na quarta quinzena do GSPv3.0.

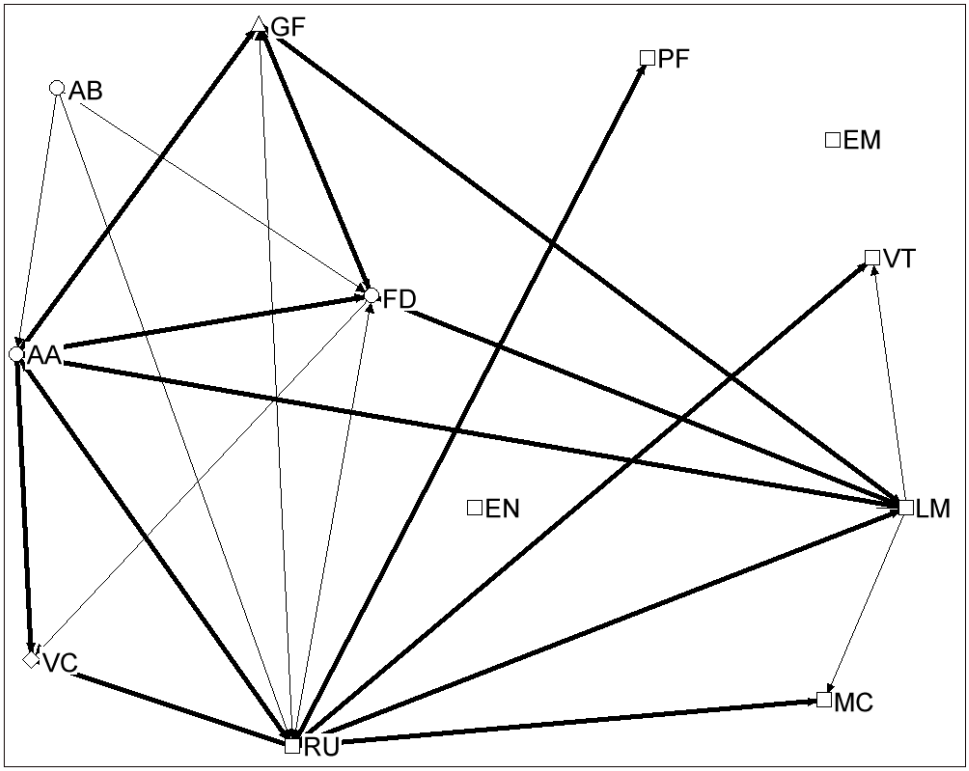

Figura 5.14: Redes sociais formadas a partir das respostas da Questão 2 na quarta quinzena do GSPv3.0.

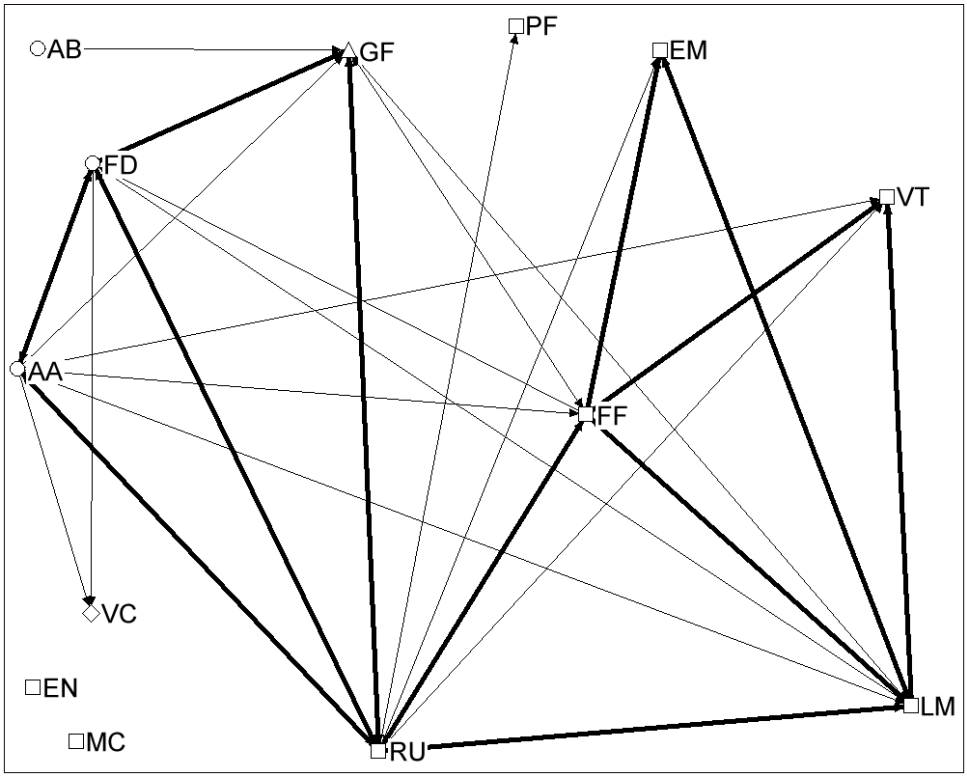

Figura 5.15: Redes sociais formadas a partir das respostas da Questão 1 na quinta quinzena do GSPv3.0.

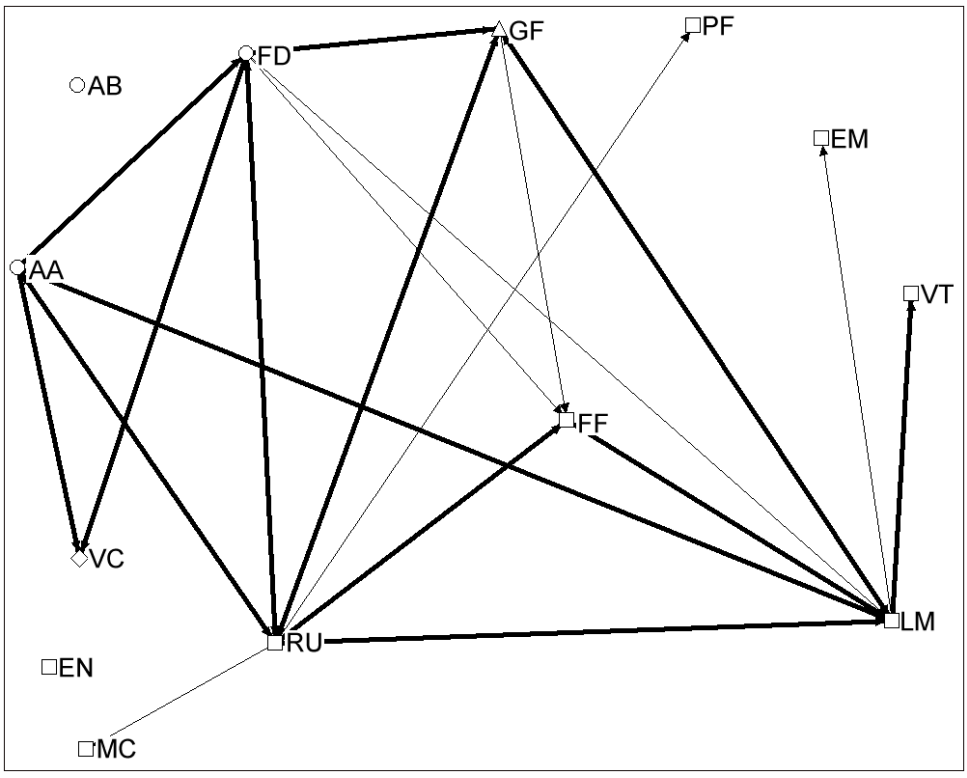

Figura 5.16: Redes sociais formadas a partir das respostas da Questão 2 na quinta quinzena do GSPv3.0.

permitindo a investigação dos efeitos da metodologia ágil XP, conduzida na equipe da PUCRS, causados na comunicação entre os times distribuídos. Os resultados obtidos pelas redes sociais são conclusivos porque a equipe brasileira foi a que concentrou a maior parte de toda da comunicação gerada durante a execução do projeto global. O maior benefício identificado por esta pesquisa através dos resultados dos gráficos das redes sociais sobre o emprego da metodologia ágil XP, foram as contribuições que as práticas ágeis trouxeram para a comunicação e a colaboração entre as equipes distribuídas.

5.4.2 Entrevistas do Protocolo do Estudo de Caso

Os dados do questionário do protocolo do estudo de caso (Apêndice A) foram triangulados com as métricas coletadas nas duas versões do projeto para obter maior confiabilidade nos resultados. A dimensão 1 do questionário, relativa aos dados demográficos, teve como objetivo identificar os respondentes quanto a sua formação acadêmica e profissional. As dimensões 2, 3 e 4 visaram identificar quais os efeitos produzidos nos desafios de GSD em função do emprego de algumas práticas ágeis sobre um projeto de desenvolvimento global de software analisado pelo estudo de caso. A dimensão 5 visou obter dados sobre a percepção dos entrevistados com relação a experiência de trabalhar com um processo de desenvolvimento que segue os princípios da metodologia ágil.

Os desenvolvedores da versão 2.0 do projeto GSP também foram entrevistados. Não se espera concluir que a metodologia XP seja mais eficiente que o processo utilizado na versão GSPv2.0, pois existiram inúmeras diferenças entre os contextos de ambas as equipes. Seria necessário subtrair diversas variáveis para que pudéssemos inferir que a metodologia ágil XP é melhor ou pior que o processo *Extended Workbench*. Entretanto espera-se a partir da comparação das respostas do questionário do protocolo do estudo de caso e dos relatos obtidos nas entrevistas com os membros das duas equipes, exprimir a percepção dos entrevistados sobre a usabilidade das práticas ágeis nos projetos globalmente distribuídos.

A seguir seguem as análises das respostas do questionário para as dimensões 2, 3, 4 e 5. As constatações para os resultados para as médias da segunda dimensão levaram em conta apenas as médias obtidas pelos respondentes da versão 3.0. No parágrafo final antes da análise da terceira dimensão, é apresentada uma breve explanação sobre a percepção dos respondentes da versão 2.0 sobre a segunda dimensão.

Dimensão 2

No conjunto de questões em escala Likert da segunda dimensão, foi solicitado que os entrevistados avaliassem a prática *Pair Programming* em relação aos principais problemas enfrentados nos projetos de desenvolvimento global de software. A síntese das respostas para a segunda dimensão está apresentada na Tabela 5.6.

Tabela 5.6: Resultado das respostas para a segunda dimensão do estudo de caso.

Questão	**Média**	
	v2.0	**v3.0**
7. A *Pair Programming* contribuiu para o entendimento dos requisitos solicitados pelo time central?	3,0	4,2
8. A prática *Pair Programming* intensifica a comunicação entre os membros do time?	3,3	5,0
9. Nas conversas informais entre uma atividade e outra houveram momentos onde você e seu colega puderam conversar sobre assuntos não relacionados ao projeto?	5,0	5,0
10. A rotação dos componentes das duplas facilitou a disseminação do conhecimento sobre o código entre os membros do time?	2,6	4,2
11. No momento em que explicava a lógica de seu código, você costumava descobrir erros que não havia percebido antes?	3,0	3,0
12. Você se sentiu responsável pelo sucesso do trabalho de seu colega?	3,6	4,4
13. O trabalho aos pares facilitou a correção dos defeitos detectados no código?	3,6	4,2
14. A prática Pair Programming desenvolve a confiança entre os integrantes da dupla?	3,0	3,8
15. A Pair Programming o fez sentir mais auto-confiante para solucionar as tarefas mais complexas enfrentadas pela dupla no projeto?	3.3	3.6
16. Qual sua motivação com relação a Pair Programming no início do projeto?	2,0	3,4
17. Você contribuiu com idéias ou experimentos que ajudassem a dupla à encontrar alternativas para a solução dos problemas inerentes a implementação?	3,0	4,8
18. Você concorda que as atividades realizadas em duplas teriam levado muito mais tempo para serem completadas se tivessem sido feitas por apenas uma pessoa?	2,0	4,0
19. Sua contribuição teria sido muito mais valiosa para o projeto se você tivesse trabalhado individualmente nas atividades?	3,0	1,8
20. Como você avalia o grau de comprometimento de seus parceiros de dupla durante a condução das atividades do projeto?	2,6	3,8
21. O seu relacionamento pessoal com os parceiros de dupla durante o projeto sempre manteve-se franco e amistoso?	4,3	4,4
22. Você sentia-se à vontade para discordar de idéias do seu parceiro quando julgava necessário?	3,3	5,0
23. As atividades realizadas pelas duplas foram finalizadas em muito menos tempo do que se tivessem sido feitas individualmente?	2,0	4,2
24. Qual sua motivação com relação a Pair Programming após o projeto?	2,0	5,0
25. Você acredita que na Pair Programming o resultado obtido pelo conjunto é superior à soma dos resultados individuais? Caso afirmativo, quais motivos o fizeram pensar desta forma?	2,3	5,0

Os melhores resultados para as médias na segunda dimensão para a versão 3.0 foram obtidos nas questões 8, 9, 22, 24 e 25 todas elas com a nota máxima 5. Tais resultados indicam um alinhamento entre os respondentes sobre a eficiência da prática *Pair Programming* na intensificação da comunicação do time o que corrobora a teoria de diversos autores (Warden et al. 2003, Beck & Andres 2004, Marchesi et al. 2002, Williams & Kessler 2002). O resultado da questão 17 reforça a questão 8, demonstrando que o envolvimento entre os parceiros das duplas era muito intenso o que colaborou para desenvolver um ambiente de colaboração bastante positivo dentro da equipe para a busca de soluções dos problemas complexos enfrentados no projeto. As questões 8 e 25 demonstram claramente a satisfação geral do grupo com a prática de programação aos pares e o modo como ela foi conduzida no projeto do estudo de caso.

As questões 14 e 15 com as respectivas médias 3,8 e 3,6 demonstram uma leve tendência positiva na percepção dos entrevistados, de que a programação aos pares aumenta a confiança dentro da equipe. Por outro lado, não há consenso quanto ao benefício da rotação das duplas, questão 10, comparando-se separadamente a média das respostas dos membros juniores com a média das respostas dos membros mais experientes (3,0 e 5,0 respectivamente), pois a percepção dos membros mais novos manteve-se neutra enquanto que os membros mais experientes foram unânimes ao considerar a rotação das duplas fundamental para a disseminação do conhecimento dentro do time. Apesar de não esperados, os resultados obtidos nas respostas dos membros mais novos podem ser compreendidos pelo fato destes terem, por diversas vezes, sido expostos à problemas complexos não compatíveis com suas habilidades técnicas, diferentemente dos membros mais experientes da equipe.

Dentre as questões da segunda dimensão a que obteve a maior variação entre os respondentes foi a questão 11. Este resultado demonstra que o benefício da programação em duplas não está necessariamente em ter uma segunda pessoa apenas para revisar o código que alguém está escrevendo, mas muito mais que isto, o verdadeiro benefício desta prática esta na combinação de duas pessoas que juntas cooperam para encontrar a solução de um problema.

Ao avaliar se a prática *Pair Programming* intensificava a colaboração e o espírito de equipe, as maiores médias obtidas ocorreram nas questões 12, 17, 20 e 21 (4,4, 4,8, 3,8 e 4,4 respectivamente). Tais resultados demonstraram que o trabalho em duplas ajudou o time a manter-se motivado e comprometido com os objetivos do projeto global.

As questões 16 e 24 utilizadas para avaliar a percepção do time quanto à programação aos pares antes e após o término do projeto obtiveram os valores para as médias de 3,4 e 5,0 respectivamente. Observou-se uma variação positiva de 32% na percepção dos respondentes quanto ao nível de motivação antes e depois do projeto com relação à prática de *Pair Programming*. O resultado da média para a questão 19 reforça a disposição dos respondentes em realizar as tarefas do projeto ao lado de um colega.

Ao avaliar o tempo de desenvolvimento gasto nas atividades, as médias das questões 18 e 23 (4,0 e 4,2 respectivamente) apontam um provável ganho de produtividade obtido pela programação aos pares, pelo fato de que na percepção dos entrevistados o trabalho realizado em duplas teoricamente é mais rápido do que o trabalho feito por uma só pessoa. Entretanto, ainda são necessários mais dados para poder avaliar com maior precisão o efeito do uso da prática *Pair Programming* no esforço total da equipe e no tempo de desenvolvimento. Com base nas médias das respostas das questões 7 e 13 (4,2 e 4,2 respectivamente), na percepção dos entrevistados, o trabalho em dupla facilitava o entendimento dos requisitos e a correção dos defeitos.

De modo geral, grande parte das médias desta dimensão obtiveram médias altas para a versão 3.0, o que demonstra que a percepção dos respondentes é satisfatória quanto a contribuição da prática *Pair Programming* para agilizar a implementação das atividades do projeto do estudo de caso.

Os resultados obtidos para as médias na versão 2.0 do projeto, demonstraram uma total desaprovação da prática *Pair Programming*. Praticamente todas as questões referentes a avaliação da prática obtiveram respostas baixas. A percepção dos respondentes da versão 2.0 foi exatamente o oposto da percepção dos respondentes da versão 3.0. Em entrevistas realizadas com os participantes da versão 2.0 para esclarecer o motivo das respostas, um dos respondentes da versão afirmou que uma das razões pela qual a *Pair Programming* não ter tido sucesso na versão 2.0 foi o fato desta ter sido empregada tardiamente no projeto, isto é, muito tempo depois que já haviam iniciados os trabalhos de desenvolvimento. Diversos outros problemas foram apontados por outros respondentes da versão 2.0, mas que vão além do escopo e objetivo desta pesquisa. Considerando que muitos dos problemas enfrentados pela equipe na versão 2.0 não terem ocorrido na versão 3.0 e a adoção da *Extreme Programming* ter acontecido de forma gradual desde muito antes do início dos trabalhos de programação, por meio de palestras e tarefas educativas com o objetivo de dar os primeiros passos no processo, é que se pode inferir a razão da enorme discrepância entre as percepções observadas nas duas equipes.

Dimensão 3

O conjunto de questões em escala Likert da terceira dimensão visava avaliar a percepção dos respondentes quanto á prática do processo XP conhecida como *Planning Game*. Apesar dos respondentes da versão 2.0 não terem utilizado tal prática, o processo *Extended Workbench* daquela versão também utilizou a abordagem de iterações para o desenvolvimento das atividades, o que permitiu aplicar as perguntas desta dimensão para todos os respondentes.

Estruturar o projeto através de um conjunto de iterações é uma das características de projetos que seguem os princípios da metodologia ágil (Augustine 2006). A consoli-

Tabela 5.7: Resultado das respostas para a terceira dimensão do estudo de caso.

Questão	**Média**	
	v2.0	**v3.0**
26. Na sua opinião a organização do desenvolvimento através de iterações agilizou o trabalho da equipe?	3,6	3,2
27. A tradução dos requisitos em um conjunto de tarefas menores colaborou para acelerar a implementação do código?	4,0	4,0
28. Você concorda que a utilização do processo XP contribuiu para estabelecer estimativas mais precisas para o esforço requerido por cada tarefa?	2,0	2,4

dação dos resultados para ambas as equipes é apresentada na Tabela 5.7. Ao avaliar se as iterações agilizavam o trabalho da equipe (questão 26) a média mostrou-se ligeiramente favorável mas com divergências entre os respondentes da versão 3.0. Comparando com as respostas dos respondentes da versão 2.0 percebeu-se que também não houve consenso entre os respondentes da versão anterior. Apesar de não apresentarem valores conclusivos, estes resultados não comprometem a aplicabilidade da prática *Planning Game*.

Na questão 27, média de 4,0, em ambas versões do projeto, a percepção dos respondentes aponta a contribuição do processo ágil XP em auxiliar os times de desenvolvimento em compreender os requisitos do usuário através da tradução destes em um conjunto de tarefas de programação. Por fim, a questão 28 com uma média inferior a 2,5 demonstra que na percepção dos respondentes o processo ágil não é determinante para se obter estimativas mais precisas.

Dimensão 4

A quarta dimensão do questionário visava avaliar a percepção dos respondentes quanto á contribuição das práticas *Collective Ownership* e *Coding Standards* sobre os desafios de GSD vivenciados no projeto do estudo de caso e também na versão 2.0 do projeto. A consolidação dos resultados é apresentada na Tabela 5.8.

A quarta dimensão foi a que apresentou os valores mais baixos para as médias em todo o questionário. Os resultados mais baixos foram observados nas questões 30, 31 (2,2, 2,8 respectivamente). Estas questões demonstram que os respondentes concordam que se alguma modificação no código fosse solicitada, por exemplo, correção de defeito ou escrita de uma nova funcionalidade, os mesmos concordam que não teriam condições para executar tal tarefa e também têm dúvidas se que alguém no time seria capaz.

Apesar de não ter sido um resultado esperado pelo menos para as médias dos respondentes da versão 3.0 a qual conduziu o processo XP no estudo de caso, tais valores podem ser explicados pela possível interpretação da pergunta por parte dos respondentes, que possivelmente teriam entendido que a alteração no código fonte seria realizada individualmente. Se isto for verdade, tais respostas reforçam a tendência positiva dos re-

Tabela 5.8: Resultado das respostas para a quarta dimensão do estudo de caso.

Questão	**Média**	
	v2.0	**v3.0**
29. Você tinha conhecimento sobre todas as partes do código produzido por sua equipe?	3,3	3,2
30. Se lhe fosse solicitado corrigir um defeito encontrado no código implementado por um outra dupla, você não teria maiores problemas para resolver a questão?	2,6	2,2
31. Se uma nova mudança fosse solicitada, você acredita que qualquer membro da equipe teria condições de alterar a aplicação, independente de qual parte do código fonte?	1,6	2,8
32. Você contribuiu para manter os códigos entregues por sua dupla bem documentados?	2.3	3,6
33. Os códigos entregues por sua dupla seguiram os padrões de código definidos no projeto?	3,3	4,2

spondentes apresentada na segunda dimensão, onde claramente existe uma preferência pelo trabalho através da programação aos pares. Talvez isto explique a razão dos valores das médias terem sido tão baixos, uma vez que segundo (Beck & Andres 2004) qualquer desenvolvedor deveria ser capaz de alterar qualquer parte do código.

Por outro lado, as melhores médias para a quarta dimensão foram observadas nas questões 29, 32 e 33 (3,2, 3,6 e 4,2). Estas três questões questionavam sobre o conhecimento que o respondente tinha sobre o código produzido por sua equipe e se tal código respeitava os padrões de código definidos no projeto.

Dimensão 5

A quinta dimensão do questionário visava avaliar a percepção dos respondentes quanto ao processo desenvolvimento utilizado nas versões 2.0 e 3.0 do projeto GSP. A consolidação dos resultados é apresentada na Tabela 5.9.

Os melhores resultados nesta dimensão foram obtidos nas respostas dos respondentes da versão 3.0 para as questões 40, 41 e 42 com médias igual a 4,4, 4,6 e 4,5 respectivamente. As médias altas obtidas nestas três questões demonstram um consenso entre os respondentes da versão 3.0 na percepção de que o processo contribui para a intensificar a comunicação entre o time local e as demais equipes distribuídas. Tal comunicação é de fundamental importância para projetos de desenvolvimento global (Paasivaara 2003).

Resultados satisfatórios foram observados nas questões 37 e 38, referentes ao grau de liberdade que o time da versão 3.0 teve para discordar e propor melhorias na forma como era conduzido o processo. Traçando uma comparação com os resultados para as mesmas questões 37 e 38, porém considerando apenas as médias dos respondentes da versão 2.0. Constata-se que não existiram grandes diferenças entre as duas versões do projeto, o que é bastante positivo, pois baseado nas médias de ambas as equipes, os respondentes parecem

Tabela 5.9: Resultado das respostas para a quinta dimensão do estudo de caso.

Questão	**Média**	
	v2.0	**v3.0**
34. Você conhecia todas as pessoas envolvidas no projeto global?	2,3	2,6
35. Você tinha ciência do papel de cada time no projeto?	3,0	3,4
36. Na sua opinião o processo XP trouxe benefícios para a colaboração entre o time local e os times globais?	3,3	3,6
37. Você tinha liberdade para propor melhorias ao processo?	4,3	4,0
38. Você tinha liberdade para contestar o processo?	3,3	4,2
40. Você tinha autonomia para contatar qualquer colega dos times globais sempre que necessário?	3,6	4,4
41. A existência de ferramentas de colaboração tais como wiki, Skype, MSN e GFORGE auxiliaram a agilizar a comunicação entre as duplas e as equipes distribuídas?	4,6	4,6
42. Considerando o seu grau de envolvimento no projeto, você avaliaria sua participação na condução do processo como satisfatória?	3,3	4,5

ter demonstrado um leve comprometimento com a forma de aplicar o processo, criticando quando era necessário e também propondo melhorias.

5.4.3 Resultados das Métricas do Estudo de Caso

Foi realizada uma rigorosa comparação entre duas equipes de desenvolvimento da universidade PUCRS para um mesmo projeto de desenvolvimento global de software. Foi utilizado como fonte de evidências os dados coletados durante o projeto para um conjunto de métricas que avaliaram a produtividade de cada time. O confronto entre os resultados das duas equipes para o mesmo conjunto de métricas teve como objetivo, investigar quais os efeitos causados na utilização ou não de um processo de desenvolvimento sobre os desafios de GSD, apresentados na seção 2.2.

Para isto, foram comparadas duas equipes, a Equipe B , a qual adotou um processo formal de desenvolvimento baseado nos princípios do processo *XP* e uma segunda equipe, a Equipe A, que por sua vez decidiu não seguir nenhum processo de desenvolvimento. Baseando-se nas informações extraídas através da comparação das duas equipes, foi possível estabelecer uma série de inferências buscando-se identificar que vantagens e desvantagens podem ser constatadas quando se adota um processo ágil de desenvolvimento em uma equipe co-localizada inserida em um projeto GSD. A Figura 5.17 ilustra o paralelo feito entre as duas equipes sob contextos análogos de GSD.

Os dados para as métricas na versão 2.0 do projeto foram coletados junto a ferramenta de colaboração wiki disponibilizada pelo SCR para fins de pesquisa. Tais dados

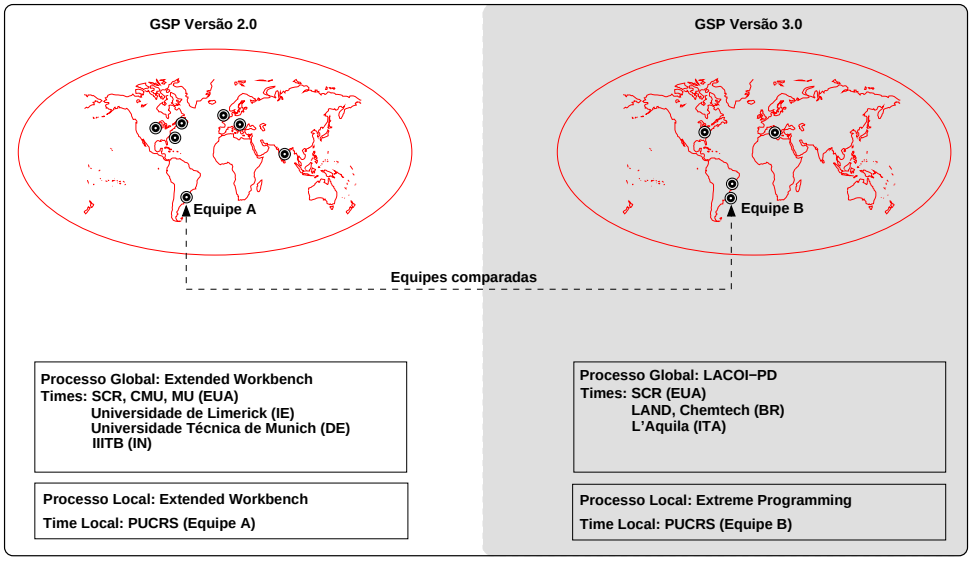

Figura 5.17: Comparação das equipes de desenvolvimento da universidade PUCRS nas versões 2.0 e 3.0 do projeto GSP.

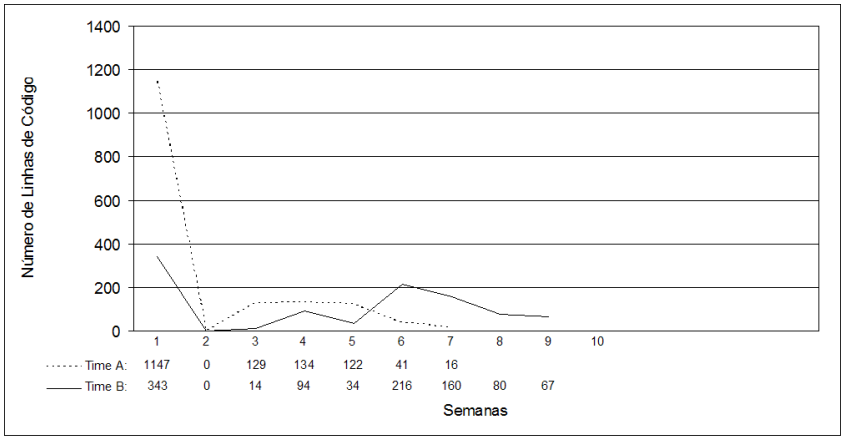

Figura 5.18: Comparação entre o número de linhas de código produzidas pelas equipes.

eram pertencentes a uma iteração isolada do projeto conduzida durante a versão 2.0, a qual teve duração de sete semanas. A iteração da versão 3.0, usada na comparação com a versão 2.0, teve a duração de nove semanas. As métricas aplicadas sobre os trabalhos desenvolvidos pela Equipe A na versão 2.0, foram exatamente as mesmas utilizadas para avaliar a Equipe B da versão 3.0, o que torna as conclusões desta pesquisa mais confiáveis. O enfoque desta comparação de processos foi o de exprimir indícios que pudessem revelar quais foram os desafios de GSD que mais foram afetados pelas conseqüências da utilização de práticas de desenvolvimento *Extreme Programming* em projetos de desenvolvimento distribuído de software. A comparação dos resultados obtidos para a métrica de linhas de código produzidas por semana pelas equipes A e B está ilustrada no gráfico da Figura 5.18. Uma análise dos totais para cada métrica foi realizada nesta pesquisa a fim de comparar os resultados obtidos na versão 3.0 contra os resultados para o mesmo conjunto de métricas

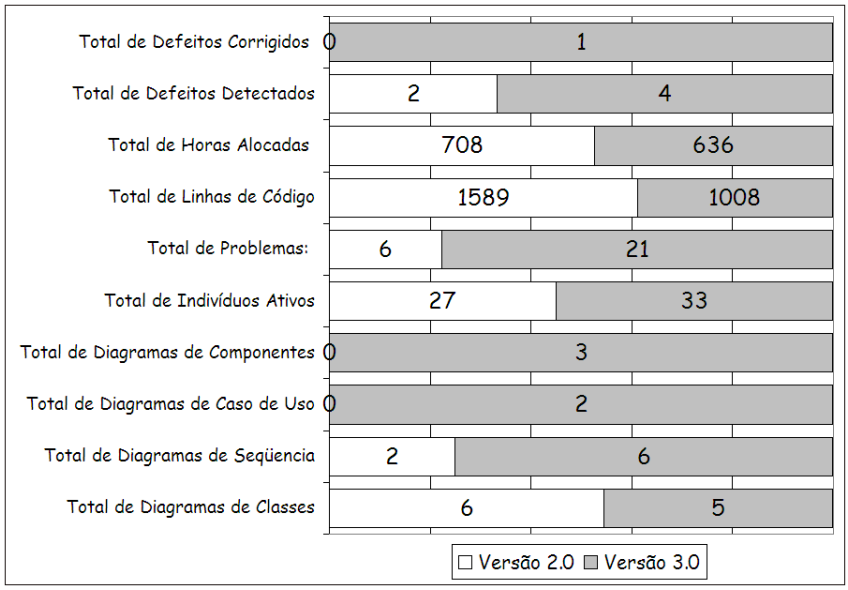

Figura 5.19: Comparação dos resultados das versões GSPv2.0 e GSPv3.0 para o mesmo conjunto de métricas.

na versão 2.0. O resultado final para as métricas foi totalizado sendo apresentado no gráfico da Figura 5.19.

Capítulo 6

Impactos das práticas ágeis XP sobre os desafios de GSD

Neste capítulo são apresentadas as análises dos efeitos observados com relação a aplicação de práticas do processo ágil *Extreme Programming* sobre o projeto de desenvolvimento global conduzido no estudo de caso. Segundo (Warden et al. 2003), a aplicação de um pequeno sub conjunto de práticas XP pode ser uma boa alternativa para as primeiras experiências com o processo em uma equipe, mas podem gerar situações e problemas não previsíveis.

6.1 Lições sobre o processo XP no contexto de GSD

Foram identificadas um conjunto de lições aprendidas a respeito das vantagens e desvantagens em se utilizar o processo XP em projetos de desenvolvimento global de software. As lições também mostram quais problemas foram enfrentados pela equipe quando o processo foi posto em prática. As lições aprendidas nesta seção descrevem quais os efeitos observados na utilização das práticas do processo ágil *Extreme Programming* sobre o projeto de desenvolvimento distribuído de software conduzido no estudo de caso.

As conclusões sobre os impactos causados pelas práticas nos desafios de GSD, foram obtidas com base nos resultados das análises das respostas do questionário das redes sociais, entrevistas e através dos resultados coletados para as métricas de produtividade.

Como resultado desta pesquisa nós identificamos um conjunto de lições aprendidas que apontam as vantagens e desvantagens das práticas ágeis XP quando aplicadas sobre projetos de desenvolvimento global de software. As lições também apresentam quais foram os novos desafios enfrentados pela equipe na adoção das práticas XP.

As lições foram obtidas com base nas observações e análises realizadas a partir dos gráficos das redes sociais. Os dados coletados nas entrevistas com os desenvolvedores serviram para tornar as conclusões da pesquisa mais confiáveis. As lições aprendidas

descobertas através desta pesquisa estão descritas abaixo:

Lição #1: O uso de uma ferramenta de gerenciamento de projeto especialmente construída para suportar a Extreme Programming foi essencial para facilitar a adoção da metodologia ágil na equipe de desenvolvimento.

De acordo com (Damian et al. 2006), em GSD a visibilidade do processo pode-se tornar um problema quando o time que gerencia o projeto não se encontra presente para acompanhar em primeira mão o progresso das atividades. Sem um contato face a face, torna-se praticamente impossível recuperar um projeto no vermelho de volta ao verde. Uma vez que o time responsável pelo gerenciamento não se está presente fisicamente para guiar o projeto, se torna muito difícil motivar os membros do time remoto a dar a volta por cima e recuperar o projeto. Em GSD, e existência de uma base de conhecimento capaz de documentar todos os passos do projeto pode auxiliar no controle dos trabalhos e proporcionar uma grande visibilidade do andamento do projeto.

Corroborando a teoria os resultados obtidos no estudo de caso demonstraram claramente que o uso de uma ferramenta de gerenciamento de projetos fortemente alinhada com a metodologia de desenvolvimento, traz grandes vantagens para a coordenação e o controle das atividades de projeto. Ela também torna mais fácil todo o trabalho de adoção dos princípios da metodologia ágil XP dentro do projeto. A ferramenta utilizada no estudo de caso com as características descritas foi o *XPlanner*. No *XPlanner* eram armazenados diversos tipos de informações a respeito do projeto tais como as iterações realizadas, histórias de usuários, tarefas de desenvolvimento e o apontamento das horas de trabalho despendidas na realização das tarefas. O uso desta ferramenta de controle e gerência permitiu uma grande visibilidade sobre o progresso dos trabalhos do time, assegurando acima de tudo que os princípios da metodologia XP estavam sendo seguidos. A Figura 6.1 exemplifica uma das utilidades da ferramenta *XPlanner*, neste caso o apontamento das horas dos desenvolvedores quando estes codificavam seguindo a prática de desenvolvimento *Pair Programming*.

Lição #2: Para assegurar que o time esteja seguindo corretamente a metodologia XP, é necessário que uma pessoa com bons conhecimentos sobre XP esteja sempre disponível para esclarecer dúvidas e questões da equipe de desenvolvimento.

Um dos problemas apontados pelos membros da equipe que participaram da versão anterior do projeto, GSP versão 2.0, foi que o principal motivo do time não ter conseguido seguir o processo baseado nas práticas da metodologia XP, foi o fato da não existência de uma pessoa dentro da equipe que tivesse a função de suportar o time na condução do processo ágil. Conforme apontado pelos autores (Marchesi et al. 2002, Ebert et al. 2001),

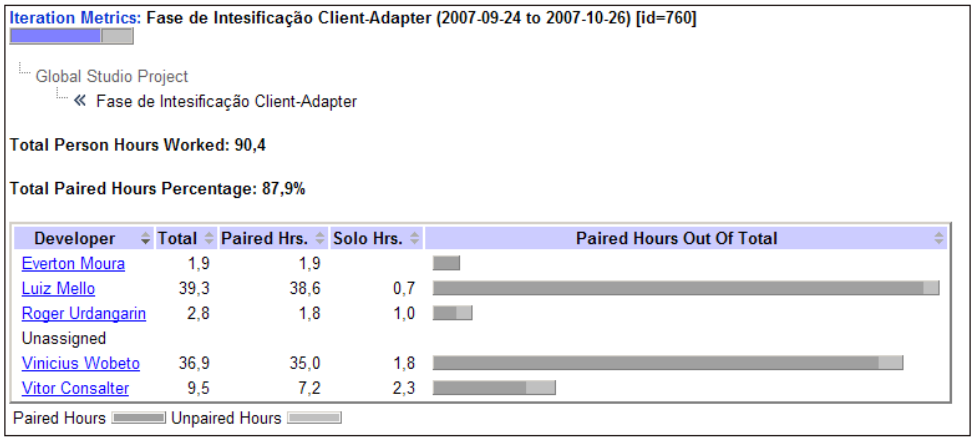

Figura 6.1: Amostra do apontamento de horas feito na ferramenta *XPlanner* pelas duplas de programação durante o estudo de caso.

é necessário que alguém com mais experiência no time e que também conheça a metodologia ágil esteja sempre disponível para auxiliar os desenvolvedores mais novos na realização dos trabalhos. No estudo de caso da pesquisa nós definimos que este papel de mentor ou *coach* do processo ágil, seria uma atribuição do pesquisador que atuava diretamente com a equipe de desenvolvimento que adotou a metodologia.

Lição #3: As práticas ágeis promovem uma comunicação mais direta e freqüente entre os membros das equipes distribuídas.

Os resultados desta pesquisa corroboraram a teoria de que a comunicação é o desafio em GSD que mais se beneficia com a adoção de um processo de desenvolvimento de software que siga os princípios da metodologia XP (Damian et al. 2006, Xiaohu et al. 2004, Ramesh et al. 2006, Paasivaara & Lassenius 2004). Foi observado no estudo de caso que a liberdade de expressão promovida pelo processo ágil fez os membros da equipe sentirem-se mais a vontade em estabelecer contatos com as demais equipes distribuídas. No projeto do estudo de caso da pesquisa, nós observamos que o uso das práticas ágeis da *Extreme Programming* pelo time de desenvolvimento da PUCRS, ajudou a equipe a desenvolver uma intensa comunicação com os demais times distribuídos, a qual perdurou durante todo o projeto, além de manter uma forte comunicação interna entre os membros da própria equipe. A Figura 6.2 ilustra a intensa comunicação interna e externa através das linhas em negrito nos nodos EN, PF e RU, representando a equipe brasileira com os nodos AA, FD, VC representando os membros das demais equipes distribuídas.

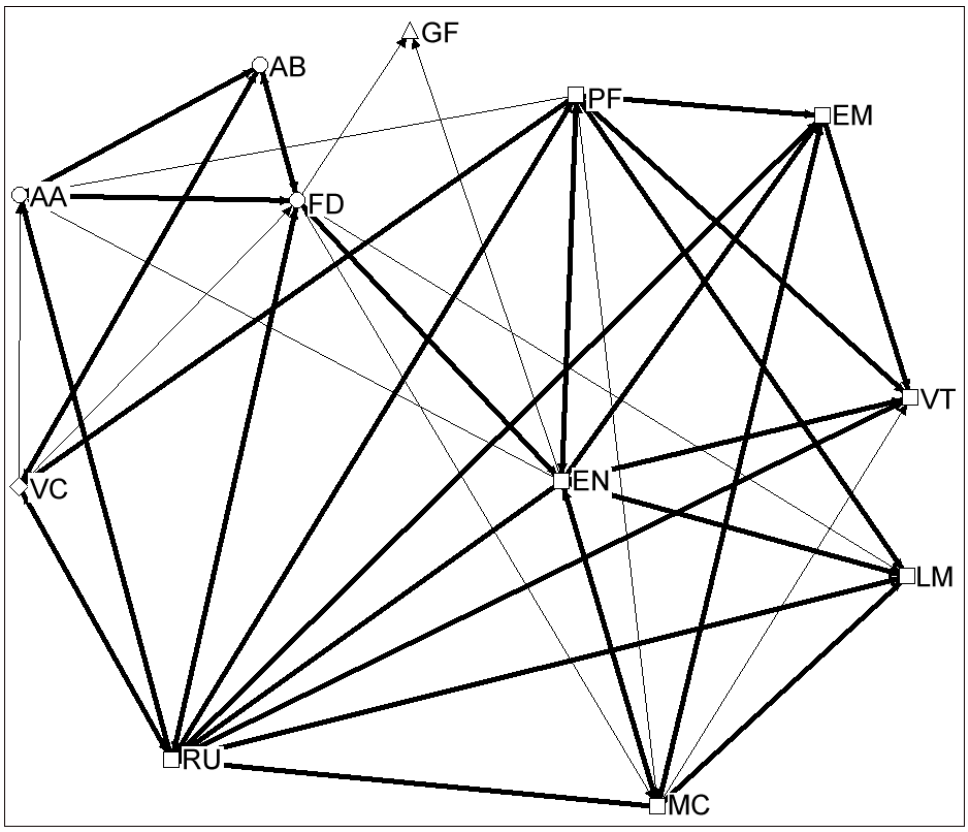

Figura 6.2: Intensa comunicação entre as equipes distribuídas durante o projeto GSPv3.0.

Lição #4: A presença de membros de mesma nacionalidade nas equipes distribuídas contribuiu para o fortalecimento do espírito de equipe entre as equipes.

Nós identificamos que uma das principais razões para a intensa comunicação informal observada durante o estudo de caso, ocorreu devido a presença de pessoas de mesma nacionalidade nos times distribuídos. Tal fato minimizou a barreira cultural do idioma e alavancou o espírito de equipe entre os times, o que trouxe grandes benefícios para o esclarecimento de dúvidas e na resolução de questões complexas. A Figura 6.3 ilustra a intensa comunicação entre o time remoto, nodos PF e RU respectivamente com os membros do time central representados pelos nodos AA e FD, todos estes brasileiros.

Lição #5: A metodologia ágil XP aumenta a confiança entre os membros da equipe em se comunicar abertamente com os times distribuídos contribuindo para evitar a formação de gargalos na comunicação.

A comunicação do projeto não estava centralizada em uma única pessoa a qual seria a representante do time no projeto global, ao contrário, todo os membros da equipe tinham liberdade para se comunicar com qualquer pessoa em qualquer time. Esta comunicação franca e informal entre os times colaborou para fortalecer a confiança mútua dos

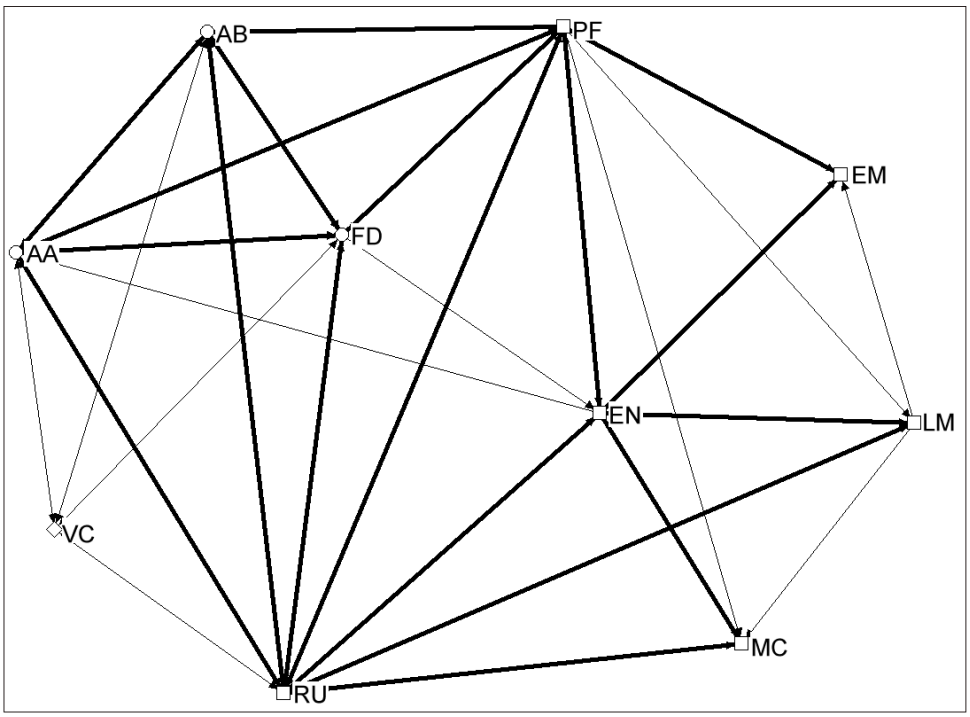

Figura 6.3: Forte comunicação estabelecida entre os compatriotas que pertenciam à equipes diferentes no projeto GSPv3.0.

times em seus parceiros remotos. Na Figura 6.4 as linhas em negrito apresentam o nível de importância atribuído pelas equipes distribuídas a comunicação estabelecida com os membros do time de desenvolvimento que seguiu a metodologia XP.

Lição #6: O uso da prática *Pair Programming* contribui para disseminar o conhecimento de negócio dentro da equipe.

Todas as atividades de desenvolvimento realizadas durante o estudo de caso foram executadas com os estudantes trabalhando aos pares. A lição aprendida identificada no estudo de caso foi resultado das observações das duplas de programação. Constatou-se após a ocorrência de sucessivas rotações nas duplas, que os conhecimentos de lógica de programação, de negócio estavam se disseminando muito rapidamente dentro da equipe em função destas freqüentes movimentações. Tal efeito também foi confirmado pela percepção dos respondentes do questionário do protocolo do estudo de caso. Os resultados observados no estudo de caso corroboraram a teoria de que a vantagem em rotacionar as duplas é que os programadores aprendam mais sobre o produto quando têm a oportunidade de trabalhar com pessoas diferentes dentro da equipe (Williams & Kessler 2002, Beck & Andres 2004).

Para a rotação das duplas, a lógica utilizada para rotacionar os membros foi baseada na teoria apresentada por (Williams & Kessler 2002), o qual sugere que a cada nova atividade, uma nova rotação de ser feita nas duplas. Sendo assim, os desenvolvedores

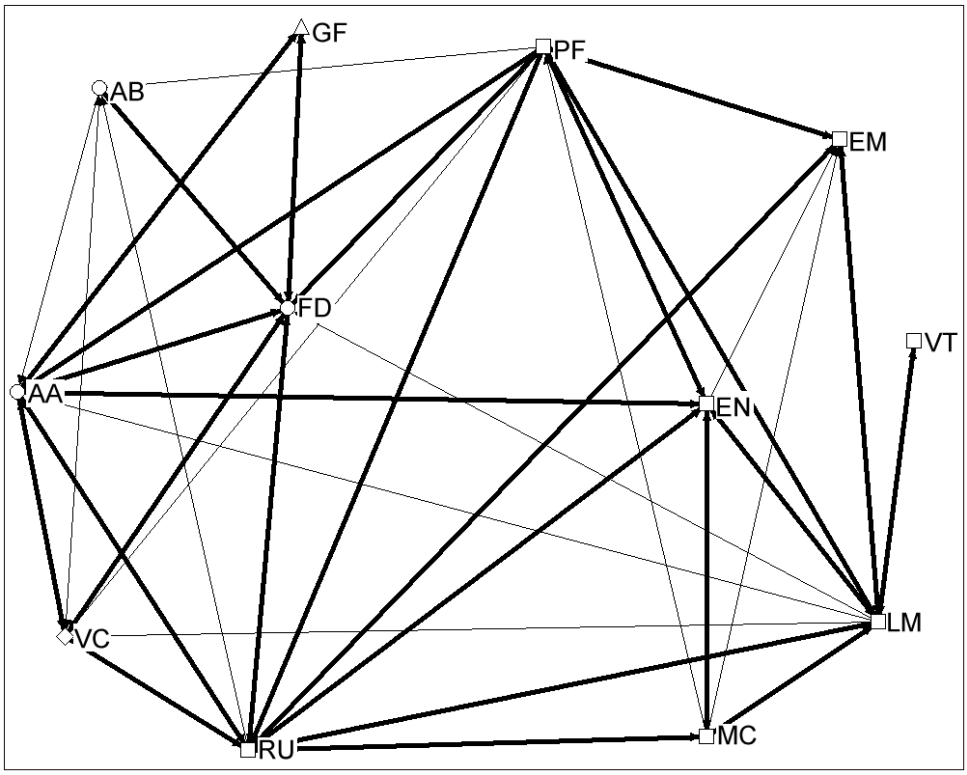

Figura 6.4: Grau de importância atribuído a comunicação entre os times distribuídos.

que atuaram como *navigator* em uma tarefa que finalizava passavam a atuar como *driver* em uma nova atividade que se iniciava, conseqüentemente àqueles que haviam atuado como *driver*, passavam a ser o *navigator* da nova dupla formada após a rotação. Esta estratégia de rotação das duplas permitiu que o conhecimento sobre os programas fosse se disseminando naturalmente a medida que sucessivas rotações aconteciam. O esquema de rotação das duplas está representado graficamente na Figura 6.5.

Lição #7: A experiência técnica da equipe de desenvolvimento tem repercussão direta na qualidade da condução das práticas da metodologia Extreme Programming.

Conforme os estudos de (Warden et al. 2003), a metodologia XP requer extrema disciplina e não é adequado para ser aplicado em equipes com pouca experiência em programação. Nós constatamos em nosso estudo de caso que duplas formadas por programadores juniores não atingem resultados de produtividade satisfatórios. Independente da complexidade das tarefas atribuídas à estas duplas, observamos sucessivos atrasos de entrega, falta de compreensão do requisito constante engajamento de um membro mais experiente da equipe para finalizar as atividades passadas para a dupla. A Figura 6.6 ilustra o caso de uma dupla formada por desenvolvedores juniores que se isolaram das demais equipes do projeto, por considerarem mais importante a comunicação entre ela do que com o restante do time global. Identificamos com este cenário uma clara distorção na forma de conduzir

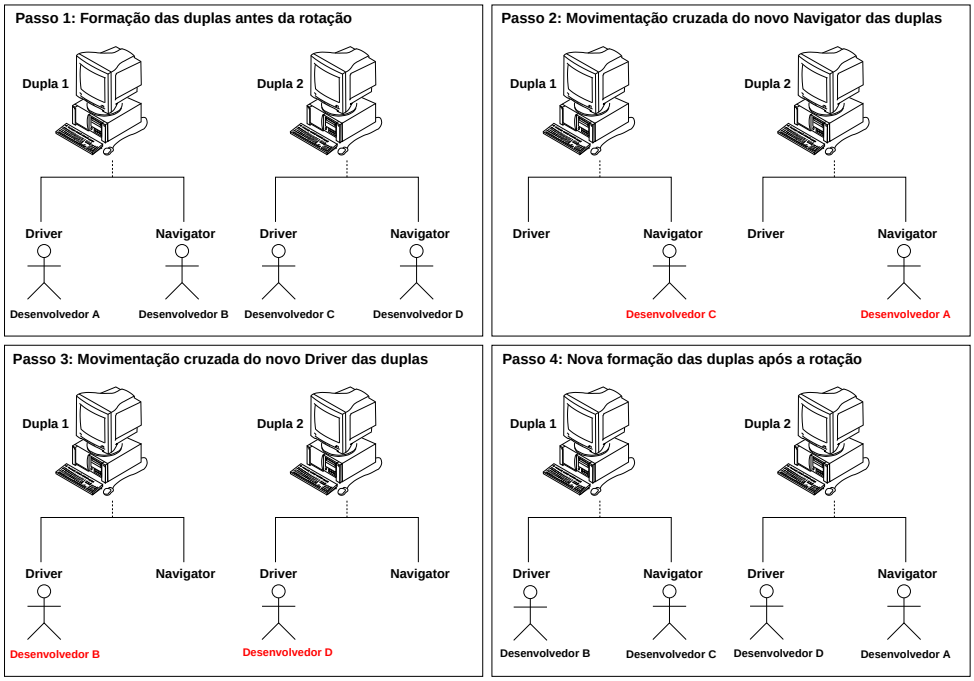

Figura 6.5: Estratégia de rotação das duplas de desenvolvedores adotada no estudo de caso.

a prática *Pair Programming*, completamente alheia aos princípios da metodologia em questão. Tal comportamento dos desenvolvedores deixa bem claro o risco que se corre na condução das práticas ágeis XP se conduzidas com indivíduos pouco experientes.

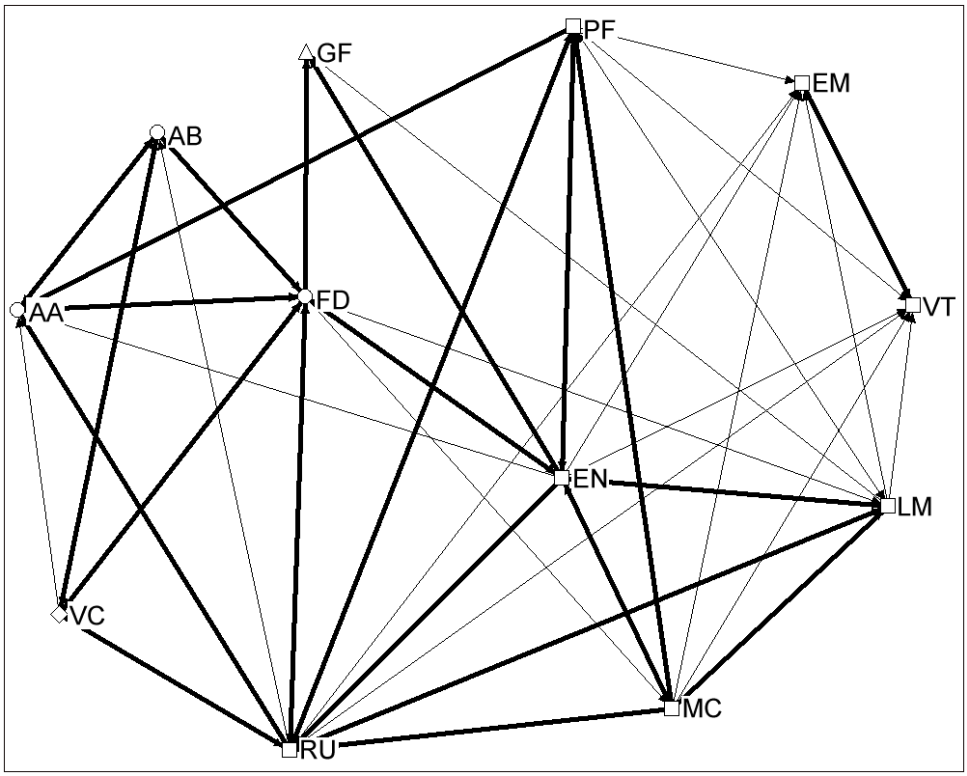

Figura 6.6: Isolamento da comunicação em uma das duplas com o resto do time global.

Lição #8: Para adotar a metodologia ágil XP sobre uma equipe de desenvolvimento é fundamental que sejam realizadas algumas seções prévias de estudo para introduzir os princípios básicos da metodologia e apresentar como as práticas se inter-relacionam.

Identificamos também que determinadas práticas da metodologia ágil XP não são triviais de serem aplicadas sobre equipes com pouca experiência em programação. Em nossos estudos descobrimos que a prática ágil *Test-Driven Development*(TDD) é muita complexa de ser aplicada sobre uma equipe de alunos de início de curso de graduação. Conforme relatado nos estudos de (Grossman et al. 2004, Paasivaara & Lassenius 2006), é importante realizar treinamentos sobre as práticas XP para facilitar o entendimento e a assimilação da cultura ágil pelos membros da equipe.

Nós observamos no estudo de caso que mesmo realizando sessões de estudos sobre as práticas XP que seriam adotadas no estudo de caso, uma delas o TDD, não conseguiu ter sua idéia e objetivo entendida pelos desenvolvedores mais novos da equipe. Além do pouco tempo disponível para a realização do estudo de caso e com o grande tempo que estava sendo despendido tentando aplicar a prática decidimos abortar sua adoção. Segundo os estudos de (Erdogmus et al. 2005), o TDD está longe de ser uma prática simples de incorporar em um time de desenvolvimento e é considerada contraproducente e difícil de aprender pelos mais céticos. Nossa opinião é que devido ao curto tempo para desenvolver as tarefas e a pouca experiência do time impossibilitaram a adoção da prática logo de início dos trabalhos.

Lição #9: Ausências prolongadas de indivíduos com responsabilidades chave dentro da metodologia XP afetam negativamente os níveis de comunicação da equipe.

Observamos no estudo de caso que para manter a equipe de desenvolvimento sempre motivada e produtiva é fortemente recomendado que os indivíduos que estejam atuando nos papéis de *coach* e tracker estarem sempre disponíveis para auxiliar o time de desenvolvimento na correta condução do processo e na resolução dos problemas de comunicação entre as equipes distribuídas. Nós observamos no estudo de caso em uma das quinzenas do projeto graves problemas de desmotivação do time, decréscimo acentuado nos níveis da comunicação entre os times, ameaça da perda total de interesse pelo projeto por parte dos membros da equipe quando repentinamente o *coach* da equipe por motivos pessoais ausentou-se por duas semanas seguidas do projeto sem aviso prévio.

O resultado do desfalque do *coach* na equipe pode ser percebido através das relações entre os nodos dos gráficos das redes sociais ilustrado na Figura 6.7. O nodo EN que representa o *coach* ficou absolutamente isolado, sem nenhuma ligação forte com os demais nodos. Com a quebra dos canais de comunicação que eram sustentados pelo

coach a comunicação da equipe despencou drasticamente. A Figura 6.7 ilustra o severo impacto negativo na comunicação causado pela prolongada ausência do *coach*, que teve efeitos drásticos tanto na comunicação interna e principalmente na comunicação externa com os times distribuídos.

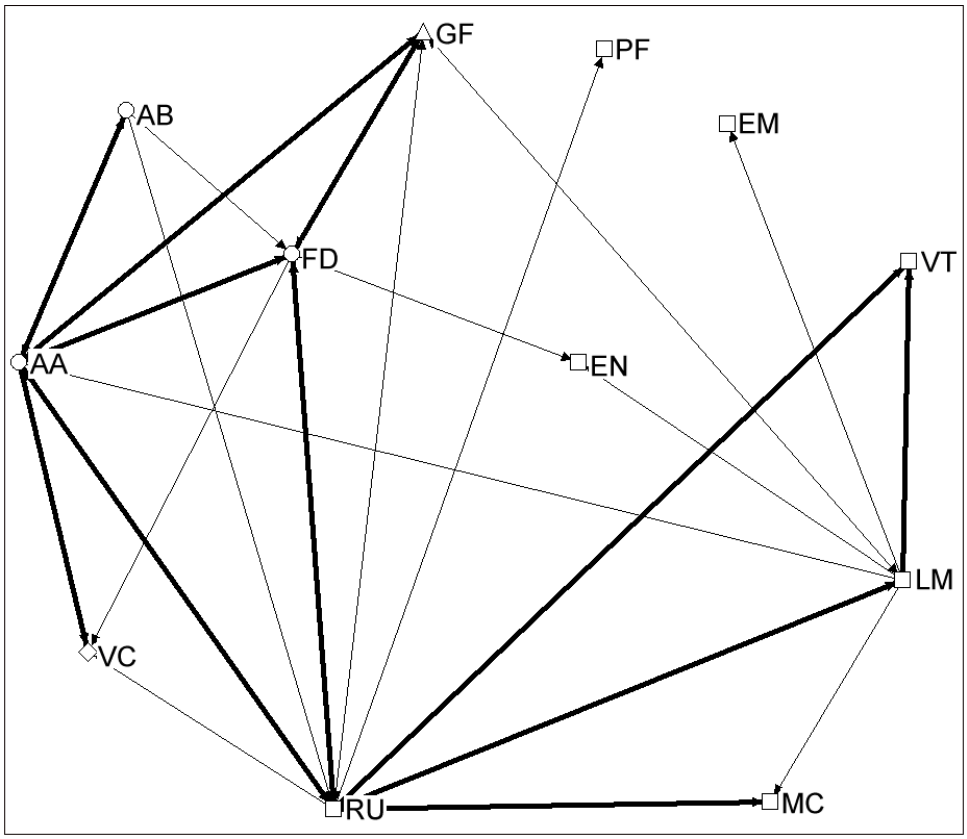

Figura 6.7: Acentuada diminuição no nível de comunicação entre a equipe ágil XP com as equipes distribuídas.

Lição #10: Desfalques de membros que atuem em papéis chave junto a metodologia XP, precisam ser imediatamente repostos para evitar impactos demasiadamente negativos para a equipe.

Nós observamos após substituir o *coach* do time por uma nova pessoa que o nível motivacional da equipe subiu rapidamente, bem como houve um aumento significativo da autoconfiança dos membros da equipe em razão das atitudes positivas e confiantes do novo *coach* da equipe. Os desenvolvedores passaram a acreditar mais em si próprios e juntamente com um melhor discernimento no uso da prática *Pair Programming*, o time todo passou a solucionar problemas complexos que nem mesmo os próprios desenvolvedores imaginavam serem capazes de resolver. Toda esta mudança de comportamento da equipe aconteceu devido a motivação, segurança e o sólido domínio técnico do novo *coach*. Em um relato obtido nas entrevistas, um membro do time afirmou que a principal razão

que levara sua dupla a finalizar uma atividade bastante complexa, foi a motivação e o apoio técnico prestado pelo *coach*. O gráfico exibido pela Figura 6.8 ilustra a reconstrução de todos os canais de comunicação internos e externos da equipe após a entrada do novo *coach* no time de desenvolvimento. O efeito desta mudança pode ser percebido durante as últimas semanas do projeto, onde a equipe desenvolvimento apresentou os melhores resultados de produtividade durante todo o projeto.

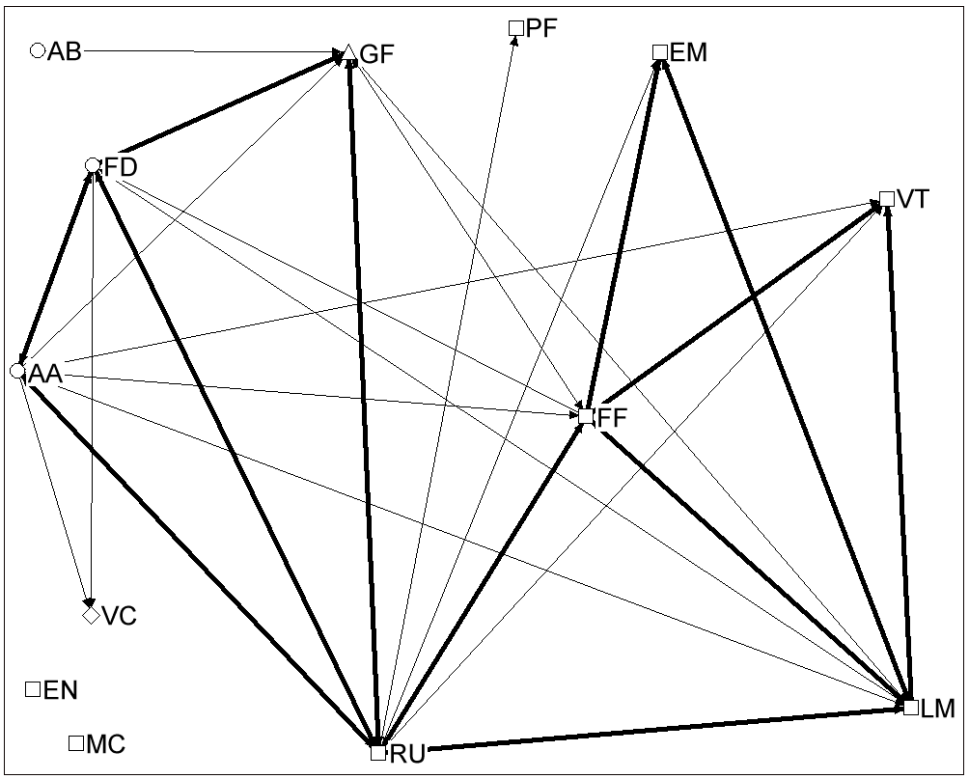

Figura 6.8: Reconstrução dos canais de comunicação após a entrada do novo coach na equipe XP.

Lição #11: O uso de ferramentas de colaboração melhora a disseminação do conhecimento entre as equipes.

Ferramentas de colaboração tais como wiki, funcionam muito bem em projeto de desenvolvimento distribuído porque são muito simples de usar, podem ser acessadas de qualquer *browser* de Internet e são ainda muito simples de configurar. Qualquer tipo de informação comum deve ser colocada no *wiki*, tais como, requisitos do cliente, diagramas UML que expliquem uma solução, instruções para integração de aplicações e notas sobre os progressos do time, em suma, qualquer tipo de informação que necessite ser escrita para referências futuras pelas equipes distribuídas (Fowler 2006).

No projeto do estudo de caso desta pesquisa a utilização da ferramenta *wiki* mostrou ser muito valiosa para o projeto. O time de desenvolvimento local utilizou o *wiki* para publicar os diagramas UML que serviram de documentação para a arquitetura da

solução desenvolvida para a construção do seu componente. Publicou-se também no *wiki*, os papéis desempenhados por cada um dos membros do time local dentro do processo *Extreme Programming*. A página do time apresentava uma breve descrição sobre o escopo de trabalho do time dentro do projeto global.

Além do *wiki* foi utilizada uma segunda ferramenta de colaboração, o *GFORGE*. Esta ferramenta serviu para que o time central coordenasse todas as atividades dos times distribuídos, nela eram registradas todas as tarefas que cada equipe remota deveria realizar durante uma determinada estimativa de tempo. O *GFORGE* forneceu listas de discussões que serviram muito bem para a resolução de dúvidas entre os times distribuídos. O controle dos defeitos nas aplicações de cada time também eram gerenciados por esta ferramenta.

O estudo realizado permitiu identificar que as ferramentas de colaboração ajudam as equipes distribuídas a entenderem o papel de cada time dentro do projeto e como cada time planejou implementação da sua solução. Tais ferramentas também auxiliaram na redução do tempo da curva de aprendizado de novos membros na equipe que entraram com o projeto em andamento.

Lição #12: Contatos face a face entre as equipes distribuídas contribuem para consolidar a confiança entre os times distribuídos e fortalecer os canais de comunicação.

No início do projeto do estudo de caso, um dos membros da equipe local viajou para o time central para se inteirar do projeto. A experiência vivida por este indivíduo no time central trouxe vários benefícios para o time local. Um deles foi a abertura dos canais de comunicação entre o time local e o time central. O segundo benefício proveniente da viagem a time central foi um grande reforço na confiança do gerente do time central na equipe local. Os contatos face a face resultaram em uma boa impressão do time central para com o time local. O reflexo deste fato pôde ser percebido em todos os gráficos dos sociogramas da seção 5.4.1 que mostram um forte canal de comunicação entre o membro local, que esteve durante um mês visitando o time central, com os membros das equipes remotas inclusive com o time central, sendo que este canal de comunicação prevaleceu durante todo o projeto.

Este relacionamento é explorado por (Damian 2002), onde um membro do time funcionava como um intermediário entre a equipe remota e a equipe central com o objetivo de sanar eventuais dúvidas sobre o projeto e agilizar o esclarecimento de problemas de comunicação entre as equipes distribuídas.

Lição #13: Utilização de uma lista de e-mails para a troca de informações entre o time local com os times distribuídos.

Segundo (Damian et al. 2006), quando os contatos face a face, comunicação síncrona torna-se inviável, use uma lista de e-mail para aumentar a chance de uma resposta e encorajar um rápido atendimento, útil, e com respostas conclusivas para os e-mails. Uma resposta muito demorada para uma dúvida implica em horas não produtivas do desenvolvedor que se transformar em um caminho crítico para o projeto enquanto se aguarda uma resposta definitiva.

O time local de desenvolvimento utilizou durante o projeto uma lista de e-mails que serviu como um canal de comunicação interno do time bastante importante para que antes de passar uma dúvida para os times externos, a equipe buscava resolver os problemas internamente enviando e-mails para a lista. Quando as questões não eram resolvidas internamente pela falta de informações suficiente os e-mails eram encaminhados para o time remoto que pudesse esclarecer os pontos que não estavam claros para o time local. A utilização da lista de e-mail foi muito útil para a comunicação interna da equipe e para a resolução dos problemas enfrentados durante o projeto. Chegou-se até mesmo a adicionar um membro chave no desenvolvimento que estava sendo realizado no time central na lista, de modo a agilizar a resolução das dúvidas da equipe local.

Lição #14: A *Pair Programming* contribui para a redução dos defeitos e para a resolução de problemas complexos de implementação.

Foram detectados um número muito baixo de defeitos no código produzido pelas duplas de programadores no estudo de caso. Todos os códigos produzidos estavam respeitando os padrões de código estipulados no projeto. Tal constatação corrobora a teoria de vários autores (Marchesi et al. 2002, Williams & Kessler 2002), que afirmam que duas pessoas em uma mesma tarefa passam a produzir códigos de melhor qualidade e com menos defeitos, em razão do pouco tempo a mais que é gasto em comunicação quando se está codificando.

Esta lição corrobora a teoria de (Layman et al. 2004), o qual constatou que a prática *Pair Programming* facilita a implementação de funcionalidades complexas. No projeto do estudo de caso, observou-se que os desenvolvedores conseguiram implementar módulos bastante complexos, os quais na opinião dos próprios programadores não teriam sido codificados completamente em tempo hábil, se a prática *Pair Programming* não tivesse sido utilizada.

Lição #15: O processo ágil de desenvolvimento não contribuiu para produzir estimativas mais precisas para as atividades de desenvolvimento.

Uma lição aprendida que se identificou através dos resultados do estudo de caso foi de que ao contrário do que se esperava a adoção do processo ágil não contribui para a geração de

estimativas de esforço mais precisas para as atividades do projeto. Não existe um processo de estimativa definido em XP, ficando a critério da equipe definir a forma como gerar suas estimativas. No caso do projeto do estudo de caso, todas as estimativas de esforço foram baseadas na experiência do líder técnico e do pesquisador somadas a uma margem extra de tempo que variava conforme as habilidades técnicas da pessoa que recebesse a atividade.

Lição #16: A análise das redes sociais (SNA) foram fundamentais para a elaboração das lições aprendidas identificadas no estudo de caso.

Nós constatamos que a utilização das redes sociais para a análise dos dados trouxeram uma valiosa contribuição para a confiabilidade das conclusões deste estudo empírico. Os gráficos gerados a partir dos dados do questionário das redes sociais nos permitiram observar com bastante precisão os principais acontecimentos ocorridos na comunicação e na colaboração entre as equipes distribuídas durante o andamento do projeto de desenvolvimento global conduzido no estudo de caso.

Acreditamos que um dos maiores benefícios que a análise das redes sociais trouxe para a pesquisa tenha sido a clara representação da evolução das interações entre os times distribuídos no projeto de desenvolvimento global. Praticamente todos os acontecimentos que foram mais marcantes durante o projeto GSPv3.0, foram fielmente registrados pelos gráficos das redes sociais. Um exemplo disto foi o caso da saída do principal integrante do time da universidade PUCRS em meio ao franco andamento das atividades do projeto GSPv3.0. Tal fato desencadeou uma drástica redução na comunicação entre o time de desenvolvimento PUCRS com as demais equipes remotas do projeto. Posteriormente nas análises dos gráficos das redes sociais, os efeitos deste acontecimento puderam ser facilmente observados através das comparações feitas entre os gráficos das redes sociais conforme ilustrado na Figura 6.9.

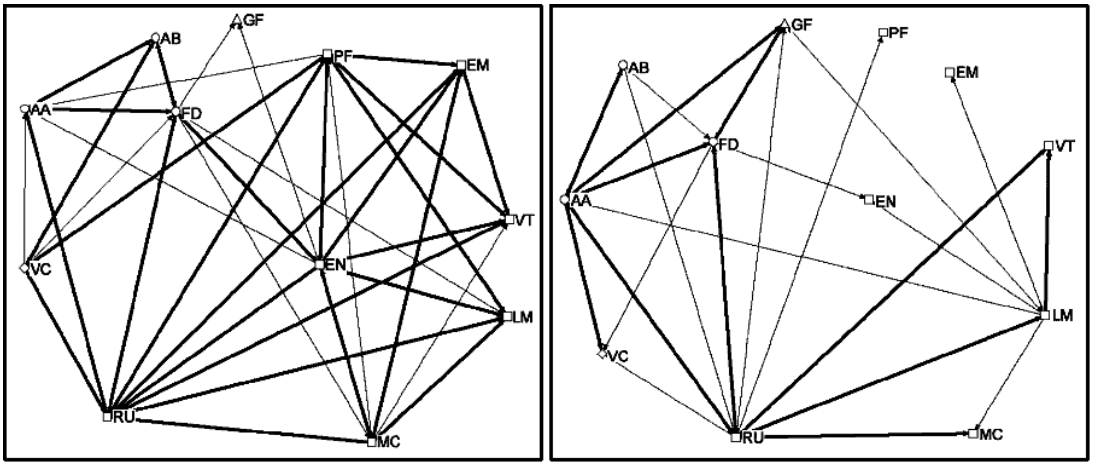

Figura 6.9: Comparação entre os gráficos das redes sociais gerados a partir dos dados da terceira e quarta quinzena do projeto GSPv3.0.

Diversos outros importantes cenários que ocorreram durante o projeto do estudo de caso puderam ser perfeitamente representados pelos gráficos das redes sociais. Isto demonstra o enorme poder de investigação que as redes sociais possuem o que é altamente desejável em estudos empíricos tais como este. Praticamente todas as lições aprendidas foram baseadas principalmente nos dados da análise das redes sociais e complementadas com o cruzamento dos resultados quantitativos com os dados qualitativos provenientes do questionário do protocolo do estudo de caso.

Lição #17: As diferenças na forma como as equipes adotaram as práticas ágeis nas duas edições do projeto GSP, permitiram identificar quais os motivos que inviabilizaram o processo LAGPRO de atingir os resultados positivos esperados.

Nós identificamos através da comparação das respostas do questionário do protocolo do estudo de caso e através de entrevistas com os membros da equipe PUCRS no GSPv2.0, que as principais razões que inviabilizaram o sucesso do processo LAGPRO foram as seguintes:

- Os desenvolvedores não tiveram a oportunidade de trabalhar na mesma sala durante todo o projeto.
- Os desenvolvedores não tiveram em nenhum momento aulas de introdução às práticas ágeis XP e suas inter-relações.
- O time não possuía uma pessoa a quem pudesse recorrer para o esclarecimento de dúvidas sobre o processo.
- Não havia ninguém no time com a responsabilidade de coordenar e monitorar a qualidade dos trabalhos conforme determinava o processo LAGPRO.
- Desenvolvedores que tiveram problemas com o acúmulo de muitos pontos negativos na competição proposta pelo LAGPRO, começaram a negligenciar o processo.
- Os desenvolvedores não mostraram muita motivação em competir com seus colegas co-localizados.
- Os membros da equipe preferiram trabalhar individualmente ao invés de trabalhar aos pares, muitas vezes trabalhando de suas próprias casas.
- As práticas *Pair Programming* e TDD foram utilizadas pela equipe muito tempo após o projeto ter começado.
- Falta de *feedback* tanto da equipe local com o time central, quanto do time central com a equipe PUCRS.

- Diversos problemas fora do escopo do projeto GSPv2.0 acabaram desmotivando o time.

Acreditamos que o processo LAGPRO poderia ter atingido os resultados positivos esperados e conseqüentemente contribuído de forma mais efetiva na redução dos desafios de GSD enfrentados no projeto GSPv2.0 conforme descritos por (Paulish et al. 2005), se durante esta edição do projeto GSP tivesse existido na equipe a figura do *coach*. O *coach* teria tomado as ações necessárias para derrubar as barreiras de comunicação que ocorreram durante o projeto, e faria também, todo o controle de qualidade do processo LAGPRO sobre os trabalhos da equipe.

Capítulo 7

Considerações Finais

A engenharia global de software vem realizando significativos progressos nos últimos anos. Vários estudos vêm acontecendo na tentativa de reduzir os impactos dos desafios enfrentados pelos projetos de desenvolvimento global de software. As metodologias ágeis aparecem neste cenário global com o objetivo de aliviar dos efeitos negativos causados pelas grandes distâncias geográficas. Entretanto, a experimentação desta nova metodologia ainda precisa ser mais explorada em novas pesquisas uma vez que muitos dos paradigmas do desenvolvimento de software são quebrados quando se adota as metodologias ágeis. É preciso realizar mais estudos sobre a aplicação dos métodos ágeis em ambientes de desenvolvimento global de software a fim de se identificar quais vantagens e desvantagens emergem da sua utilização. Existem muitas lacunas sobre que novos desafios as práticas ágeis adicionam à complexidade dos projetos GSD.

Conforme apresentado no capítulo 6, o objetivo geral desta dissertação foi atingido, com uma extensa investigação empírica sobre os efeitos da adoção de práticas ágeis da metodologia *Extreme Programming* sobre um projeto GSD, que resultaram em um conjunto de lições aprendidas. O objetivo específico de aprofundar o conhecimento em engenharia global de software, metodologia ágil XP e tópicos relacionados foi atingido, conforme demonstrado pela base teórica apresentada no capítulo 2. No capítulo 5, foram apresentados os resultados para o estudo de caso aplicado sobre um projeto de desenvolvimento global de software envolvendo a participação de três universidades localizadas em dois continentes e um centro de pesquisa em engenharia de software nos EUA. Ainda no capítulo 5, foram descritas as análises das redes sociais formadas no projeto do estudo de caso entre as equipes distribuídas. O capítulo 6 apresentou as lições aprendidas sobre os efeitos identificados nos problemas enfrentados por GSD na adoção da metodologia ágil XP.

7.1 Contribuições

Nós apresentamos nesta pesquisa um estudo de caso de um time XP inserido em um projeto GSD e quais os impactos e novos desafios identificados nas interações entre um time de desenvolvimento situado no Brasil com duas equipes distribuídas em outros 2 países. A metodologia de desenvolvimento ágil XP necessita de uma freqüente comunicação informal entre o cliente e o desenvolvedor (Damian et al. 2006). Nós observamos que apesar dos desafios de GSD, a comunicação ocorrida durante o projeto entre a equipe ágil XP brasileira com as demais equipes distribuídas manteve-se bastante intensa durante toda a duração do projeto. Nós descobrimos que a presença de membros da mesma nacionalidade nas equipes distribuídas auxiliaram no fortalecimento do espírito de equipe do time global, o que foi fundamental para o sucesso do processo de desenvolvimento baseado nos princípios ágeis adotados pela equipe brasileira, minimizando drasticamente os obstáculos provenientes das diferenças culturais e do idioma.

A pesquisa também demonstrou algumas ações que precisam ser tomadas antes da adoção da metodologia ágil para a sua condução adequada e também apresentou claramente a importância do papel de um *coach* dentro de um time XP. Foram apresentados os impactos causados pela metodologia no projeto GSD que foram basicamente o aumento significativo da comunicação informal entre os times distribuídos, a aumento da transferência de conhecimento entre os membros do time XP e o aumento da confiança dos times distribuídos na importância do trabalho da equipe ágil XP dentro do projeto. Nosso estudo também demonstrou as práticas ágeis XP que funcionaram na equipe de estudantes, e que trouxeram os benefícios esperados descritos na teoria. Por outro lado identificamos também, práticas que não atingiram os resultados esperados em função do nível de conhecimento e maturidade da equipe.

Nosso estudo identificou que a adoção das práticas da metodologia XP, representaram um grande desafio para os programadores menos experientes e uma ótima forma de trabalho para os programadores mais experientes. Nós concluímos também, que as práticas ágeis XP transferem mais confiança para o time de desenvolvimento, encorajando os indivíduos no desenvolvimento das atividades mais complexas, que se tivessem sido executadas individualmente, provavelmente não teriam tido sucesso. O estudo também revelou os diversos benefícios da prática *Pair Programming* para o ambiente de trabalho para a transferência de conhecimento, senso de comprometimento com o projeto e também com relação a colaboração e o espírito de equipe.

Nós demonstramos a partir dos gráficos das redes sociais e através dos dados qualitativos das entrevistas com a equipe, que a adoção da metodologia ágil XP tornou a equipe de desenvolvimento mais confiante, comunicativa e motivada com o projeto. Acreditamos que a principal contribuição deste estudo tenha sido a identificação de um conjunto de lições aprendidas que relatam os impactos da metodologia XP em GSD.

Nossa pesquisa contribuiu com novos dados para os estudos empíricos sobre à adoção das metodologias ágeis no contexto de GSD, auxiliando do preenchimento da lacuna existente sobre este tema na área da engenharia global de software (Paasivaara & Lassenius 2006, Damian et al. 2006, Fowler 2006). Finalmente, este estudo visa contribuir com a prática ao atender uma demanda organizacional crescente por processos mais leves e ágeis que contribuam para aumentar a comunicação, colaboração e a transferência de conhecimento nas equipes envolvidas em projetos GSD.

7.2 Limitações do Estudo

As limitações desta comparação foram a de que os artefatos produzidos por ambas equipes não eram os mesmos e nem ao menos passíveis de comparação. Porém as linguagens de programação utilizadas pelas equipes são equivalentes. A equipe do projeto GSP versão 2.0 utilizou a linguagem de programação C#.Net e a equipe que participou da versão 3.0 do projeto GSP utilizou a linguagem Java. Segundo os estudos de (Kachru & Gehringer 2004), estas duas linguagens são comparáveis pelo fato de serem extremamente similares. O estudo de caso desta pesquisa também não tem como ser replicado pois o projeto utilizado para unidade de análise não cobriu todos os principais problemas apresentados na teoria por projetos GSD. Segundo o autor (Sangwan et al. 2006), os projetos GSD diferem drasticamente entre si sendo este um problema característico deste tipo de projeto. Por esta razão o estudo de caso não pode ser replicado devido as características particulares do contexto do projeto GSP na sua versão 3.0.

Uma outra limitação do estudo foi o fato do time da PUCRS ter iniciado a execução do processo ágil sem a intensa comunicação com o cliente característica nos projetos que executam o processo *Extreme Programming* (Beck & Andres 2004), isto ocorreu por problemas externos a pesquisa que ocasionaram sucessivas perdas de datas no cronograma do projeto global, o que inevitavelmente resultou em um enorme atraso no início das observações do estudo de caso. Tais problemas aconteceram pela resistência do líder técnico em se envolver mais intensamente com o time, o que mais tarde resultou no seu desligamento do projeto. As conseqüências foram que a maior parte dos trabalhos realizados pela equipe de desenvolvimento acabassem sendo realizadas posteriormente a todas as fases do projeto global, onde em teoria a comunicação do time local com os demais times distribuídos teria sido muito mais intensa e valiosa, porém apesar disto, existiu ainda uma considerável troca de informações entre a equipe local com os outros times distribuídos o que garantiu uma boa coleta de dados para seguir adiante com a pesquisa.

Os resultados encontrados por este estudo são baseados em um único estudo de caso, o qual limita conclusões mais generalizáveis sobre os efeitos da metodologia ágil XP

em projetos GSD. Além disto, a equipe de desenvolvimento que executou as práticas ágeis tinha pouca experiência em desenvolvimento de software. Também o fato de termos tido apenas um time, e não todos os times, trabalhando com a metodologia ágil limitou nossas conclusões sobre as vantagens e desvantagens dos métodos ágeis no contexto de GSD. Nós sugerimos para estudos futuros que o estudo de caso seja replicado em um projeto GSD, composto por equipes da academia ou da indústria onde todos os times estejam de fato seguindo a metodologia ágil XP da mesma forma. Uma outra sugestão seria conduzir um estudo comparativo entre o desenvolvimento aos pares contra o desenvolvimento realizado individualmente, medindo a produtividade e observando a transferência de conhecimento dentro das equipes.

Um último fator que exerceu um impacto negativo na condução do estudo de caso foi a alta rotatividade de membros entrando e saindo da equipe de desenvolvimento. Tal fato trouxe efeitos bastante negativos para a condução das práticas da metodologia ágil XP a qual é fortemente focada em colaboração e comunicação, por este motivo cada vez que um membro abandonava o grupo o impacto era bastante forte na motivação do time em iniciar todo o processo de transferência do conhecimento sobre o projeto para o indivíduo que entrava para substituir o que havia saído.

7.3 Trabalhos Futuros

Identificou-se ao longo desta pesquisa potencial para a realização de trabalhos futuros nesta linha de pesquisa da Engenharia de Software Empírica tendo como tema a introdução dos métodos ágeis no desenvolvimento global de software. Como pesquisas futuras sugerem-se:

- Continuidade da pesquisa em um projeto de desenvolvimento global utilizando o restante das práticas da metodologia ágil *Extreme Programming*.
- Realização de pesquisa envolvendo vários projetos de desenvolvimento global de software, onde cada projeto utilizará uma metodologia ágil diferente do outro. As metodologias ágeis que podem ser utilizadas para cada projeto são listadas:
  - *Scrum*
  - *Dynamic Systems Development Method (DSDM)*
  - *Crystal Methods*
  - *Lean Development*
  - *Adpative Software Development*
  - *Feature Driven Development*

- Realização de pesquisa envolvendo organizações em diferentes continentes que estejam utilizando a mesma metodologia ágil em seus projetos, a fim de se comparar os resultados obtidos por cada uma destas organizações e estudar com profundidade os efeitos das diferenças culturais na condução dos métodos ágeis.

Referências

Ambler, S. (2005), 'The agile unified process (aup)', Capturado em http://www.ambysoft.com/unifiedprocess/agileUP.html, Julho de 2006.

Augustine, S. (2006), 'Adotando métodos de entrega Ágil', *Revista PMI* **7**(11), 70–74.

Avritzer, A., Hasling, W. & Paulish, D. (2007), Process investigations for the global studio project, *in* '2nd International Conference on Global Software Engineering', IEEE Computer Society, Germany.

Beck, K. & Andres, C. (2004), *Extreme Programming Explained: Embrace Change*, Cambridge: Addison Wesley Professional, 224p.

Blois, M., Czekster, R. & Webber., T. (2006), Improving productivity of local software development teams in a global software development environment, *in* 'IEEE First International Conference on Global Software Engineering', IEEE Computer Society, Brazil, pp. 253–254.

Borchers, G. (2003), 'The software impacts of cultural factors on multi-cultural software development teams', *IEEE Software* pp. 540–545.

Bradner, E. & Mark, G. (2002), Effects of team size on participation, awareness, and technology choice in geographically distributed teams, *in* '36th Hawaii International Conference on System Sciences', IEEE Computer Society, USA, pp. 150–160.

Brenner, L. (2001), 'Peg'. Capturado em http://www.inf.pucrs.br/peg, Outubro de 2006.

Carmel, E. (1999), *Global Software Teams - Collaborating Across Borders and Time Zones*, New Jersey: Prentice Hall PTR, 269p.

Carmel, E. & Agarwal, R. (2001), 'Tatical aproaches for alleviating distance in global software development', *IEEE Software* **18**(2), 22–29.

Carmel, E. & Agarwal, R. (2002), 'The maturation of offshore sourcing of information technology work', *MIS Quarterly Executive* **1**(2), 65–77.

Carmel, E. & Tija, P. (2005), *Offshoring Information Technology*, Cambridge: Cambridge University Press, 282p.

Clerc, V., Lago, P. & Vliet, H. (2007), Global software development: Are architectural rules the answer?, *in* '2nd International Conference on Global Software Engineering', IEEE Computer Society, Germany.

Damian, D. (2002), Workshop on global software development., *in* 'International Conference on Software Engineering (ICSE'02)', IEEE Computer Society, USA, pp. 19–25.

Damian, D. & Hargreaves, E. J. (2004), Can global software teams learn from military teamwork models?, *in* '3rd International Workshop on Global Software Development (ICSE04)', IEEE Computer Society, UK, pp. 21–23.

Damian, D., Williams, L., Layman, L. & Bures, H. (2006), 'Essential communication practices for extreme programming in a global software development team', *Information & Software Technology* **48**(9), 781–794.

Damian, D., Zowghi, D., Vaidyanathasamy, D. & Pal, L. (2002), An industrial experience in process improvement: an early assessment at the australian center for unisys software, *in* 'International Symposium on Empirical Software Engineering (ISESE02)', IEEE Computer Society, JP, pp. 111–123.

Ebert, C., Parro, C. H., Suttels, R. & Kolarczyk, H. (2001), Improving validation activities in a global software development, *in* '23rd International Conference on Software Engineering', IEEE Computer Society, CA, pp. 545–554.

Erdogmus, H. (2007), 'A primer test-driven development'. Capturado em http://www.esicenter.unisinos.br/, Dezembro de 2007.

Erdogmus, H., Morisio, M. & Torchiano, M. (2005), 'On the effectiveness of the test-first approach to programming.', *Transactions on Software Engineering* **31**, 226–237.

Espinosa, J. A. & Carmel, E. (2003), Modeling coordination costs due to time separation in global software teams, *in* 'International Workshop on Global Software Development', IEEE Computer Society, USA, pp. 230–235.

Espinosa, J. A., Cummings, J. N., Pearce, B. M. & Wilson, J. M. (2002), Research on teams with multiple boundaries., *in* '35th Hawaii International Conference on System Sciences', IEEE Computer Society, USA, pp. 253–263.

Fowler, M. (2006), 'Using an agile software process with offshore development'. Capturado em http://martinfowler.com/articles/agileOffshore.html, Maio de 2006.

Ågerfalk, P., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B. & Conchúi, E. (2005), A framework for considering opportunities and threats in distributed software development, *in* 'International Workshop on Distributed Software Development (DiSD 2005)', Austrian Computer Society, Austria.

Grossman, F., Bergin, J., Leip, D., Merritt, S. & Gotel, O. (2004), One xp experience: introducing agile (xp) software development into a culture that is willing but not ready, *in* '2004 conference of the Centre for Advanced Studies on Collaborative research', CA.

Hazzan, O. & Dubinsky, Y. (2006), 'Can diversity in global software development be enhanced by agile software development?', *Proceedings of the 2006 international workshop on Global software development for the practitioner* pp. 58–61.

Herbsleb, J. D., Mockus, A., Finholt, T. A. & Grinter, R. E. (2000), Distance, dependencies, and delay in a global collaboration, *in* '2000 ACM conference on Computer supported cooperative work (CSCW '00)', ACM, USA, p. pp.209.

Herbsleb, J. D., Mockus, A., Finholt, T. A. & Grinter, R. E. (2001), An empirical study of global software development: Distance and speed, *in* '23rd International Conference on Software Engineering (ICSE'01)', IEEE Computer Society, USA, pp. 81–90.

Herbsleb, J. D. & Moitra., D. (2001), 'Global software development.', *IEEE Software* pp. 16–20.

Holmstrom, H., Conchúir, E., Ågerfalk, P. & Fitzgerald, B. (2006), Global software development challenges: A case study on temporal, geographical and socio-cultural distance, *in* 'International Conference on Global Software Engineering', IEEE Computer Society, BR, pp. 03–11.

Jeffries, R. & Melnik, G. (2007), 'The art of fearless programming'. IEEE SOFTWARE Published by the IEEE Computer Societ.

Kachru, S. & Gehringer, E. (2004), 'A comparison of j2ee and .net as platforms for teaching web services.', *Frontiers in Education, 2004. FIE 2004. 34th Annual* **3**(S3B), 12–17.

Katzy, B., Evaristo, R. & Zigurs, I. (2000), Knowledge management in virtual projects: A research agenda, *in* '33rd Hawaii International Conference on System Sciences', IEEE Computer Society, USA, pp. 150–160.

Khan, N., Currie, W. L., Weerakkody, V. & Desai, B. (2003), Evaluating offshore it outsourcing in india: Supplier and customer scenarios, *in* '36th Hawaii International Conference on System Sciences', IEEE Computer Society, USA, pp. 210–220.

Kishore, R., Rao, H. R., Nam, K., Rajagopalan, S. & Chaudhury, A. (2003), 'A relationship perspective on it outsourcing.', *Communications of the ACM* **46**(12), 43–49.

Layman, L. (2006), Changing students' perceptions: Analysis of the supplementary benefits of collaborative development, *in* '19th Conference on Software Engineering Education and Training', IEEE Computer Society, US, pp. 159–166.

Layman, L., Williams, L. & Cunningham, L. (2004), Exploring extreme programming in context: An industrial case study, *in* 'Proceedings of the Agile Development Conference (ADC'04)', IEEE Computer Society, US, pp. 32–41.

Marchesi, M., Succi, G., Wells, D. & Williams, L. (2002), *Extreme Programming Perspectives*, Michigan: Addison Wesley Professional, 640p.

Marquardt, M. J. & Horvath, L. (2001), *Global Teams: how top multinationals span boundaries and cultures with highspeed teamwork*, Davies-Black Publishing, USA.

Mcconnell, S. (1996), *Rapid Development.*, Redmond: Microsoft Press, 647p.

Mockus, A. & Herbsleb, J. D. (2001), Challenges of global software development., *in* '7th International Software Metrics Symposium (METRICS'01)', IEEE Software, USA, pp. 182–185.

Mullick, N., Houda, E., Paulish, D., Cataldo, M., Herbsleb, J., Sangwan, R. & Bass, L. (2006), Siemens global studio project: Experiences adopting an integrated gsd infrastructure, *in* 'First International Conference on Global Software Engineering', IEEE Computer Society, Brazil, pp. 203–212.

Narayanaswam, R. & Henry, R. (2005), Effects of culture on control mechanisms in offshore outsourced it projects, *in* 'ACM SIGMIS CPR Conference on Computer Personnel Research', ICSE Workshop, USA, pp. 32–39.

Paasivaara, M. (2003), Communication needs, practices and suporting structures in global inter-organizational software development projects, *in* 'International Workshop on Global Software Development', ICSE Workshop, USA, pp. 59–63.

Paasivaara, M. & Lassenius, C. (2004), Using iterative and incremental processes in global software development, *in* 'International Workshop on Global Software Development', ICSE Workshop, UK, pp. 10–16.

Paasivaara, M. & Lassenius, C. (2006), Could global software development benefit from agile methods?, *in* 'IEEE First International Conference on Global Software Engineering', IEEE Computer Society, Brazil, pp. 109–113.

Paulish, D. J., Bass, M. & Herbsleb, J. D. (2005), Global software development at siemens: Experience from nine projects, *in* '27th International Conference on Software Engineering (ICSE'05)', IEEE Computer Society, USA, pp. 524–533.

Prikladnicki, R. & Yamaguti, M. H. (2004), Risk management in global software development: A position paper, *in* '3rd International Workshop on Global Software Development at ICSE', IEEE Computer Society, UK.

Ramesh, B., Cao, L., Mohan, K. & Xu, P. (2006), 'Can distributed software development be agile?', *SPECIAL ISSUE: Flexible and distributed software processes: old petunias in new bowls?* **49**(10), 41–46.

Ramesh, V. & Dennis, A. (2002), The object-oriented team: Lessons for virtual teams from global software development, *in* '35th Hawaii International Conference on System Sciences', IEEE Computer Society, USA, pp. 353–362.

Rockenbach, A. (2003), 'Troubleshoot your way to success in remote team management!', *Project Management Institute* **13**(4), 145–161.

Sa, J. & Maslova, E. (2002), A unified process support framework for global software development, *in* '26th Annual International Computer Software and Applications Conference', IEEE Computer Society, Oxford, England, pp. 1065–1070.

Sahay, S. (2003), 'Global software alliances: the challenge of *"standardization"*', *Scandinavian Journal of Information Systems* **15**(1), 10–17.

Sangwan, R., Paulish, D., Mullick, N. & Bass, M. (2006), *Global Software Development Handbook.*, New York: Auerbach Publications, 253p.

Senge, P. (1998), *A quinta disciplina: arte e prática da organização de aprendizagem*, São Paulo: Best Seller, 443p.

Sharma, R. (2003), Influence of geographic dispersion on control and coordination aproaches for management of software development projects, *in* '9th Americas Conference on Information Systems', Doctoral Consortium, USA, pp. 3388–3393.

Smith, C. & Llado., C. U. (2004), Performance model interchange format (pmif 2.0): Xml definition and implementation, *in* 'First International Conference on the Quantitative Evaluation of Systems', IEEE Computer Society, USA, pp. 38–47.

sourceforge.net (2006), 'Xplanner', Capturado em http://www.xplanner.org/index.html, Outubro de 2007.

Sparrow, E. (2003), *Successful It Outsourcing: From Choosing a Provider to Managing the Project*, Springer Verlag, USA.

Warden, S., Diaz, T. & Chromatic (2003), *Extreme Programming Pocket Guide*, Sebastopol: O'Reilly, 83p.

Whitney, J. O. (1994), *The Trust Factor: Liberating profits and restoring corporate vitality*, R. R. Donnelley & Sons Company, USA.

Williams, L. & Kessler, R. (2002), *Pair Programming Illuminated*, North Carolina: Addison Wesley Professional, 292p.

Xiaohu, Y., Bin, X., Zhijun, H. & Maddineni, S. (2004), 'Extreme programming in global software development', *Electrical and Computer Engineering* **4**(2-5), 1845–1848.

Yin, R. (2001), *Estudo de Caso: Planejamento e Métodos*, São Paulo: Bookman, 205p.

Yin, R. (2005), *Estudo de Caso Planejamento e Métodos*, São Paulo: Bookman, 205p.

Apêndice A

Protocolo para o Estudo de Caso

A.1 Avaliação do Uso das Práticas XP em GSD

A.1.1 Objetivo

Avaliar a aplicabilidade das práticas de desenvolvimento ágil no desenvolvimento global de software, bem como, os benefícios decorrentes de sua utilização em projeto acadêmico conduzido em um ambiente GSD.

A.1.2 Questão de Pesquisa

Avaliar os efeitos na utilização das práticas da metodologia ágil *Extreme Programming* causados nos principais desafios enfrentados pelo desenvolvimento global de software?

A.1.3 Unidade de Estudo

Projeto acadêmico de desenvolvimento de software que seguiu o processo de desenvolvimento *Extreme Programming* em um contexto GSD.

A.1.4 Característica-chave do método de estudo de caso

Este é um roteiro para o desenvolvimento e aplicação de um instrumento de pesquisa estruturado com questões abertas e em escala Likert que se caracteriza com uma pesquisa do tipo transeccional. O objetivo é avaliar o processo de desenvolvimento de software proposto, de acordo com a visão dos envolvidos, bem como a percepção da qualidade dos artefatos gerados.

A.1.5 Organização desse Protocolo

O protocolo será organizado como o segue:

A.1.6 Procedimentos

Levantamento das questões e estruturação do guia para a entrevista	
Participantes:	Roger Urdangarin
Local:	Curso de Pós-Graduação em Ciência da Computação - PUCRS

Reuniões de revisão do guia para a entrevista	
Participantes:	Dr. Alberto Avritzer. Doutor em Ciência da Computação - UCLA (University of California, Los Angeles) - 1990
	Prof. Dr. Paulo Fernandes. Doutor em Informática - Institut National Polytechnique de Grenoble, INPG - 1998
Local:	Pós Graduação da PUCRS
	Via teleconferências (Alberto & Paulo)

Autorização da empresa participante (SCR)	
Participantes:	Dr. Alberto Avritzer
Local:	SCR, Princeton

Formato de Aplicação do Instrumento			
Tipo:	Entrevista semi-estruturada com questões em escala Likert		
Entrevistador:	Roger Gonçalves Urdangarin		
Caso:	Projeto *Global Studio Project*		
Entrevistado:	**Nome**	**Data**	**Local**
Pesquisador	Roger Urdangarin	30/10/2007	PUCRS
Desenvolvedor	Aluno 1	01/11/2007	PUCRS
Desenvolvedor	Aluno 2	01/11/2007	PUCRS
Desenvolvedor	Aluno 3	03/11/2007	PUCRS
Desenvolvedor	Aluno 4	01/11/2007	PUCRS
Desenvolvedor	Aluno 5	07/11/2007	PUCRS
Desenvolvedor	Aluno 6	06/11/2007	PUCRS

Validação de Face e Conteúdo	
Participantes:	Dr. Alberto Avritzer. Doutor em Ciência da Computação - UCLA (University of California, Los Angeles) - 1990 (Especialista, Pesquisador Sênior)
	Prof. Dr. Marcelo Blois - Doutor em Ciência da Computação - PUC-RJ - 2002 (Especialista, Pesquisador Sênior)
	Roger Gonçalves Urdangarin
Local:	PUCRS
	Via teleconferências
Datas:	24/10/2007

Pré-teste	
Participantes:	Kleinner Farias
Local:	PUCRS
Datas:	25/10/2007

Análise dos Dados
Participantes: Roger Gonçalves Urdangarin
Local: Porto Alegre
Datas: 08/10/2007 a 15/11/2007

A.1.7 Relação *Respondentes x Dimensões*

O instrumento desta pesquisa aborda cinco dimensões, conforme listadas no cabeçalho da tabela abaixo. A descrição para cada uma delas, encontra-se na seção A.1.11 deste protocolo.

Respondentes	Dimensões				
	1	2	3	4	5
Desenvolvedor 1	X	X	X	X	X
Desenvolvedor 2	X	X	X	X	X
Desenvolvedor 3	X	X	X	X	X
Desenvolvedor 4	X	X	X	X	X
Desenvolvedor 5	X	X	X	X	X
Desenvolvedor 6	X	X	X	X	X
Pesquisador	X	X	X	X	X

A.1.8 Outros recursos utilizados

Recursos materiais

- Sistema de gerenciamento de e-mails para envio e recebimento das entrevistas.
- Gravador para entrevistas face a face.
- Microcomputador com Windows XP e Microsoft Excel para análise de dados.

A.1.9 Modelo do estudo e Dimensões da Pesquisa

O esquema ilustrado pela Figura A.1 representa o objetivo da pesquisa que é avaliar quais os efeitos causados pela utilização de um conjunto de práticas da metodologia XP sobre um projeto de desenvolvimento global de software. As elipses mais externas representam as práticas ágeis da metodologia *Extreme Programming* e as elipses mais internas são os desafios de GSD.

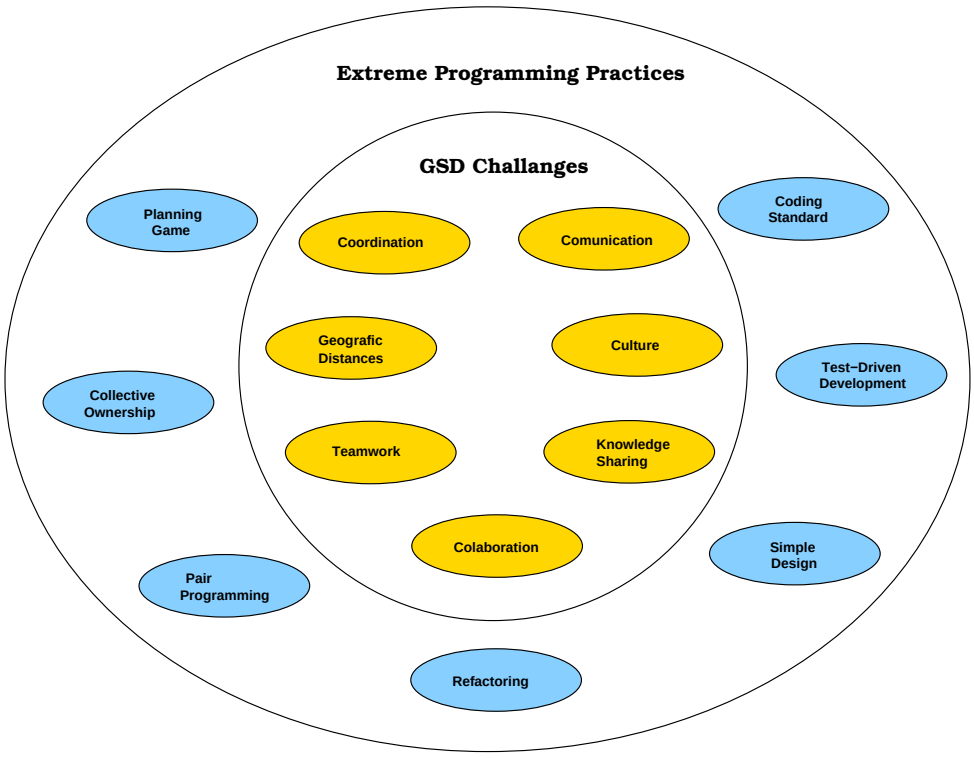

Figura A.1: Relação das dimensões enfocadas no instrumento da pesquisa.

A.1.10 Análise de dados

A análise dos dados será realizada com a técnica de análise de conteúdo e para a tabulação dos dados coletados pretende-se utilizar o módulo estatístico do excel. A coleta dos dados teve como fontes de evidências os dados quantitativos obtidos a partir dos resultados das métricas coletadas durante o estudo de caso, bem como, os dados qualitativos obtidos nas entrevistas e através da observação participante realizada pelo pesquisador. Uma outra importante fonte de evidências foi a coleta de dados quantitativos utilizando um ques-

tionário específico para as redes sociais. A triangularização dos dados coletados proposta por (Yin 2005) permitirá maior confiabilidade nos resultados obtidos.

A.1.11 Roteiro das entrevistas

Baseado em sua mais recente experiência no projeto GSP, responda as questões abaixo assinalando com um "X"a opção mais condizente com o seu grau de concordância com as afirmações propostas.

Dimensão 1 - Dados Demográficos

Indivíduo	1. Qual seu nome?
	2. Qual sua idade?
Escolaridade	3. Qual a sua maior formação escolar? [ ] 1^o Grau [ ] 3^o Grau [ ] Mestrado [ ] 2^o Grau [ ] Especialização [ ] Doutorado
Experiência	4. Tempo de experiência profissional na área de informática? 5. Qual era seu conhecimento sobre desenvolvimento global de software antes do projeto? [ ] Nenhum [ ] Pouco [ ] Já ouviu falar [ ] Conhece [ ] Conhece bem 6. Qual era seu conhecimento sobre o processo de desenvolvimento *XP* antes do projeto? [ ] Nenhum [ ] Pouco [ ] Já ouviu falar [ ] Conhece [ ] Conhece bem

Dimensão 2 - Prática *Pair Programming*

Desafios	**Questões**
Comunicação (Damian et al. 2006) (Xiaohu et al. 2004) (Paasivaara & Lassenius 2006)	7. A *Pair Programming* contribuiu para o entendimento dos requisitos solicitados pelo time central? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 8. A prática *Pair Programming* intensifica a comunicação entre os membros do time? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 9. Nas conversas informais entre uma atividade e outra houveram momentos onde você e seu colega puderam conversar sobre assuntos não relacionados ao projeto? Sim [ ] Não [ ]
Disseminação do Conhecimento (Katzy et al. 2000)	10. A rotação dos componentes das duplas facilitou a disseminação do conhecimento sobre o código entre os membros do time? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 11. No momento em que explicava a lógica de seu código, você costumava descobrir erros que não havia percebido antes? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente
Colaboração (Layman 2006) (Hazzan & Dubinsky 2006)	12. Você se sentiu responsável pelo sucesso do trabalho de seu colega? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 13. O trabalho aos pares facilitou a correção dos defeitos detectados no código? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 14. A prática *Pair Programming* desenvolve a confiança entre os integrantes da dupla? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 15. A *Pair Programming* o fez sentir mais auto-confiante para solucionar as tarefas mais complexas enfrentadas pela dupla no projeto? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 16. Qual sua motivação com relação a *Pair Programming* no início do projeto? Muito baixa [ ] [ ] [ ] [ ] [ ] Muito alta 17. Durante as tarefas, você contribuiu com idéias ou experimentos que ajudassem a dupla à encontrar alternativas para a solução dos problemas inerentes a implementação? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente

Espírito de Equipe	18. Você concorda que as atividades realizadas em duplas teriam levado muito mais tempo para serem completadas se tivessem sido feitas por apenas uma pessoa? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 19. Sua contribuição teria sido muito mais valiosa para o projeto se você tivesse trabalhado individualmente nas atividades? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 20. Como você avalia o grau de comprometimento de seus parceiros de dupla durante a condução das atividades do projeto? Negligente [ ] [ ] [ ] [ ] [ ] Comprometido 21. O seu relacionamento pessoal com os parceiros de dupla durante o projeto sempre se manteve franco e amistoso? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 22. Você sentia-se à vontade para discordar de idéias do seu parceiro quando julgava necessário? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 23. As atividades realizadas pelas duplas foram finalizadas em muito menos tempo do que se tivessem sido feitas individualmente? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 24. Qual sua motivação com relação a *Pair Programming* após o projeto? Muito baixa [ ] [ ] [ ] [ ] [ ] Muito alta 25. Você acredita que na *Pair Programming* o resultado obtido pelo conjunto é superior à soma dos resultados individuais? Caso afirmativo, quais motivos o fizeram pensar desta forma? Sim [ ] Não [ ]

Dimensão 3 - Prática *Planning Game*

Desafios	**Questões**
Coordenação (Fowler 2006) (Damian et al. 2002)	26. Na sua opinião a organização do desenvolvimento através de iterações agilizou o trabalho da equipe? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 27. A tradução dos requisitos em um conjunto de tarefas menores colaborou para acelerar a implementação do código? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 28. Você concorda que a utilização do processo XP contribuiu para estabelecer estimativas mais precisas para o esforço requerido por cada tarefa? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente

Dimensão 4 - Práticas *Collective Ownership* e *Coding Standards*

Desafios	**Questões**
Disseminação do Conhecimento (Katzy et al. 2000)	29. Você tinha conhecimento sobre todas as partes do código produzido por sua equipe? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 30. Se lhe fosse solicitado corrigir um defeito encontrado no código implementado por um outra dupla, você não teria maiores problemas para resolver a questão? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 31. Se uma nova mudança fosse solicitada, você acredita que qualquer membro da equipe teria condições de alterar a aplicação, independente de qual parte do código fonte? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente
Comunicação (Paasivaara & Lassenius 2006)	32. Você contribuiu para manter os códigos entregues por sua dupla bem documentados? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 33. Os códigos entregues por sua dupla seguiram os padrões de código definidos no projeto? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente

Dimensão 5 - Percepção Geral

Desafios	**Questões**
Espírito de Equipe (Ramesh et al. 2006)	34. Você conhecia todas as pessoas envolvidas no projeto global? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 35. Você tinha ciência do papel de cada time no projeto? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente
Colaboração	36. Na sua opinião o processo XP trouxe benefícios para a colaboração entre o time local e os times globais? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente
Cultura (Fowler 2006)	37. Você tinha liberdade para propor melhorias ao processo? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 38. Você tinha liberdade para contestar o processo? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente
Comunicação (Fowler 2006) (Damian et al. 2006) (Xiaohu et al. 2004)	39. Assinale abaixo quais foram os meios de comunicação utilizados no projeto entre as equipes distribuídas: [ ] Teleconferência [ ] Correio Eletrônico [ ] Reunião face a face [ ] Telefone [ ] Chat eletrônico [ ] Outros 40. Você tinha autonomia para contatar qualquer colega dos times globais sempre que necessário? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 41. A existência de ferramentas de colaboração tais como *Wiki*, *Skype*, *MSN* e *GFORGE* auxiliaram a agilizar a comunicação entre as duplas e as equipes distribuídas? Discordo totalmente [ ] [ ] [ ] [ ] [ ] Concordo Totalmente 42. Considerando o seu grau de envolvimento no projeto, você avaliaria sua participação na condução do processo como satisfatória? Muito baixa [ ] [ ] [ ] [ ] [ ] Muito alta

Apêndice B

Métricas utilizadas para o Estudo de Caso

B.1 Métricas de Produtividade

B.1.1 Grupo 1 - Fase de Análise

As métricas deste grupo tiveram o objetivo de capturar informações sobre a quantidade de diagramas UML produzidos semanalmente pela equipe durante a fase de análise do projeto.

- **Número de Diagramas de Caso de Uso:** Quantos diagramas de Caso de Uso foram criados pelo time por semana.
- **Número de Diagramas de Classes:** Quantos diagramas de classe foram criados pelo time por semana.
- **Número de Diagramas de Seqüencia:** Quantos diagramas de Seqüencia foram criados pelo time por semana.
- **Número de Diagramas de Componentes:** Quantos diagramas de Componentes foram criados pelo time por semana.

B.1.2 Grupo 2 - Fase de Desenvolvimento

As métricas deste grupo tiveram como finalidade extrair informações referentes ao esforço despendido para a execução das atividades de codificação.

- **Número de Indivíduos Ativos:** Número de indivíduos da equipe que estiveram participando ativamente das atividades do projeto durante a semana.

- **Número de Horas alocadas por Semana:** Número total de horas de trabalho alocadas na semana para o time. Seu valor pode variar em função do número de membros e em função de feriados.
- **Número de Linhas de Código por Semana:** Número de linhas de código fonte produzidas pelo time por semana.
- **Número de Problemas:** É o número de problemas associados com o código da equipe ao final de cada semana.

B.1.3 Grupo 3 - Fase de Testes

Este grupo de métricas foi utilizado para validar a qualidade dos artefatos entregues pelo time de desenvolvimento. A lógica para a análise dos resultados para este métrica foi a de que, quanto menor o número defeitos encontrados nos artefatos entregues, maior o nível de qualidade apresentado pelo código produzido pela equipe de desenvolvimento. Por outro lado, quanto maior o número de defeitos detectados nos artefatos, conseqüentemente menor seria a qualidade do código da equipe de desenvolvimento.

- **Número de Defeitos Detectados:** Total de defeitos detectados pelo time por semana.
- **Numero de Defeitos Corrigidos por Semana:** Não deverá existir mais de um defeito a cada 1000 linhas de código.

Apêndice C

Publicações

Artigos aceitos em eventos - Atualizado até 31 de Janeiro de 2008:

URDANGARIN, Roger; **Uma Análise de Práticas Agile XP no Desenvolvimento Global de Software.** In: XII Workshop de Teses e Dissertações em Engenharia de Software (SBES 2007), João Pessoa 2007.