UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO JORGE FERREIRA TRECCANI

UM ESTUDO EMPÍRICO SOBRE A APLICAÇÃO DE MÉTODOS ÁGEIS EM ORGANIZAÇÕES BRASILEIRAS

Dissertação de Mestrado

UFPA / ICEN / PPGCC
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM-PARÁ-BRASIL
2012

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO EMPÍRICO SOBRE A APLICAÇÃO DE MÉTODOS ÁGEIS EM ORGANIZAÇÕES BRASILEIRAS

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UFPA para a obtenção do Grau de Mestre em Ciência da Computação

UFPA / ICEN / PPGCC
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM-PARÁ-BRASIL
2012

Dados Internacionais de Catalogação-na-Publicação
(CIP)
Sistema de Bibliotecas da UFPA

TRECCANI, PEDRO JORGE FERREIRA, 1986-
UM ESTUDO EMPÍRICO SOBRE A APLICAÇÃO DE MÉTODOS ÁGEIS EM ORGANIZAÇÕES BRASILEIRAS / PEDRO JORGE FERREIRA TRECCANI. - 2012.

Orientador: Cleidson R. B. de Souza.
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação, Belém, 2012.

1. Engenharia de software. 2. Pesquisa qualitativa. 3. Arquitetura de software. I. Título.

CDD 22. ed. 005.1

AGRADECIMENTOS

À Deus, pela vida. Aos meus pais Raimunda e Girolamo Treccani pela educação, incentivo, carinho, amor incondicional e por todos os valores de vida passados. A minha esposa, Jadielly, pelo apoio, compreensão, amor e companheirismo em todos os dias de alegria e de angústia desta, e de todas as caminhadas. Ao meu orientador, Prof. Dr. Cleidson de Souza, por acreditar no meu potencial, pela confiança depositada e principalmente pelos valiosos ensinamentos repassados com muita paciência, além dos inúmeros conselhos. Aos professores do PPGCC/UFPA, por contribuírem para o meu crescimento acadêmico e profissional. Em especial a profa. Dra. Carla Reis e o prof. Dr. Rodrigo Reis, que se tornaram amigos mais do que apenas professores.

À banca, Profa. Dra. Carla A. Lima Reis e Dr. Alfredo Goldman vel Lejbman, por aceitar o convite para avaliar este trabalho. Aos professores Dr. Sandro Bezerra, Dr. Marco Aurélio Gerosa, Dr. Alfredo Goldman, e aos profissionais Mauricio Aniche, Mauricio de Diana e Márcio Vinicius dos Santos, pelo auxílio no trabalho através da indicação dos contatos das empresas a serem entrevistadas. Agradeço também a todas as empresas estudadas e aos informantes, não apenas por terem colaborado com a pesquisa, mas pela preocupação em contribuir com uma pesquisa acadêmica e o interesse sobre o andamento e resultados do estudo.

Aos amigos, antigos e novos, que fiz no mestrado, na sala de aula, laboratórios e em especial aos membros LINC e do LABES pela companhia, pelos momentos de alegria e lazer, pelo auxilio em discussões e apoio nesta caminhada. Aos amigos da COBRA TECNOLOGIA, que mesmo em momentos alegres ou de dificuldade estão lutando por um mesmo ideal, sem medir esforços, sempre em prol da empresa. Ao coordenador da minha equipe, Daniel Noleto, por compreender o quanto o mestrado significa para mim, dando apoio para esta realização e contribuindo também com minha formação de profissional com sua experiência, se tornando um novo amigo.

“No meio de qualquer dificuldade encontra-se a oportunidade”

Albert Einstein

RESUMO

Organizações que desenvolvem software enfrentam grandes desafios na produção de seus sistemas. Grande parte deste desafio está relacionado à volatilidade de seus requisitos, devido ao crescente dinamismo do mercado. Constantes alterações, adaptações e adições de novos requisitos em sistemas são frequentes no mercado de software, fazendo com que as empresas necessitem de técnicas cada vez melhores para se adequarem a essas mudanças. Nesse contexto, as metodologias ágeis podem auxiliar as organizações de desenvolvimento de software a obter esse dinamismo necessário para acompanhar as frequentes mudanças de requisitos. A adoção de métodos/metodologias ágeis tem se destacado no mercado brasileiro, onde cada vez mais, as metodologias ágeis ganham espaço.

Este trabalho descreve os resultados de um estudo empírico realizado em cinco organizações brasileiras de desenvolvimento de software que utilizam métodos/metodologias ágeis. Neste estudo foram utilizadas entrevistas semiestruturadas e questionários para coleta de dados e informações e métodos da Teoria Fundamentada em Dados para análise, com o objetivo de explorar como as metodologias ágeis são aplicadas por organizações brasileiras. Principalmente com relação à prática da refatoração, que adquire aspectos colaborativos neste contexto, visto que as atividades passam a ser feitas em pares ou grupos, auxiliando assim na difusão do conhecimento sobre a arquitetura do sistema, por exemplo.

O foco principal do trabalho é apresentar como a refatoração é executada e quais as principais mudanças em relação à forma tradicional de realização da mesma. No decorrer do texto serão apresentados os resultados encontrados no estudo, bem como uma discussão sobre os mesmos. Além disso, há uma avaliação dos impactos gerados pela utilização desta nova forma de execução da atividade de refatoração, tanto na forma de trabalho das pessoas, quanto na organização como um todo.

Palavras-chave: Métodos ágeis; Refatoração; Colaboração; Estudo Empírico; Teoria fundamenta em Dados, Pesquisa Qualitativa.

ABSTRACT

Current software development organizations face different challenges. Several of these challenges are related to requirements volatility. For instance, new functionalities are constantly required, as well as changes in the requirements and even with the exclusion of some of them, impacting the entire development process. Given this context, development organizations need to quickly adapt to changes. Agile methods are regarded as an efficient way to help these organizations achieve the necessary dynamism to handle the volatility of the requirements. This can be seen in the software producer market, where agile methods are gaining more and more space.

This work presents the results of an empirical study conducted in five software organizations which using agile methods in their software process. The study was conducted using semi-structured interviews and surveys to collect data and information. Practices from Grounded Theory were used to analyze the collected data, in order to explore how agile methodologies are applied by Brazilian organizations.

Especially with the practice of refactoring, which acquires collaborative aspects in this context, since the activities are made in pairs or groups, thus helps to spread the knowledge about the software architecture of the system among the development team, for example. This paper discusses the results of an empirical study that we conducted in Brazilian organizations that have adopted agile methods. We focus on collaborative refactoring, i.e. refactoring activities that are performed in a collaborative way in these organizations.

Keywords: Agile Methods; Refactoring; Collaboration; Empirical Study; Grounded theory, Qualitative Study.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASD	*Adaptive Software Development*
DSDM	*Dynamic Systems Development Method*
ES	Engenharia de Software
FDD	*Feature Driven Development*
GT	Teoria Fundamentada em Dados (do inglês *Grounded Theory*)
IDE	Ambiente para desenvolvimento integrado (do inglês *Integrated Development Environment*)
MA	Métodos/Metodologias Ágeis
P.O.	*Product Owner*
PPGCC	Programa de Pós-Graduação em Ciência da Computação
ROI	Retorno sobre Investimento (do inglês *Return on Investment*)
UML	*Unified Modeling Language*
XP	*eXtreme Programming*

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A utilização de software ou sistemas computacionais tornou-se parte inerente da sociedade atual. Cada vez mais as atividades do dia a dia são realizadas com o auxílio de computadores, seja para resolver tarefas simples ou complexas. Assim, novas soluções são criadas todos os dias para auxiliar a realização destas tarefas ou automatizar o processo de criação de produtos ou serviços. Para isso é necessário a construção de sistemas que suportem estas demandas, sendo o tempo uma variável primordial para organizações produtoras de software, uma vez que o cliente necessita que a sua solicitação seja atendida o mais rápido possível, proporcionando com isso vantagens estratégicas de mercado. No entanto, essa rapidez no desenvolvimento de software pode causar problemas para as empresas, quando estas negligenciam a qualidade do software desenvolvido.

O Chaos Report (2009) relata que houve uma melhoria na qualidade dos sistemas que estão sendo produzidos atualmente. No entanto, de acordo com este relatório, ainda existem problemas: sistemas são entregues sem funcionalidades importantes para o cliente, as entregas são realizadas fora do prazo estimado, sistemas com defeitos em funcionalidades críticas para o cliente, entre outros problemas. Exatamente por causa destes motivos, as metodologias ágeis ganharam espaço no setor de software, pois baseiam-se em premissas que visam realizar entregas rápidas com ciclos curtos de desenvolvimento, entregando funcionalidades que agreguem valor de negócio ao cliente. As metodologias ágeis, além de proporcionar entregas mais rápidas do sistema ao cliente, têm como foco manter a qualidade do produto gerado, o que proporciona benefícios ao andamento do processo de desenvolvimento, uma vez que o cliente utiliza

e verifica constantemente as funcionalidades do sistema, podendo identificar problemas antecipadamente analisando também a qualidade do produto entregue.

A utilização de metodologias ágeis promove uma série de mudanças nas organizações que desenvolvem software, tanto no âmbito técnico quanto no âmbito social. No âmbito *técnico* as metodologias ágeis têm como foco a adaptação/criação de um processo mais leve, sendo necessária para isso a construção de um ambiente propício a receber e executar tal processo, como por exemplo, a implantação de sistemas para automatização de testes, integração contínua do produto desenvolvido, ambientes de desenvolvimento de software apropriados para refatoração de códigos, dentre outros aspectos. Também pode haver mudanças nas técnicas de documentação do sistema, como exemplo tem-se a alteração de requisitos formais para histórias de usuários ou utilização de metáforas para compreender o ambiente em que o sistema em desenvolvimento será utilizado (Dubinsky e Hazzan, 2003). Muitas destas mudanças técnicas são necessárias para dar agilidade ao desenvolvimento e manter o foco no cliente, verificando o que realmente é preciso e compreendendo, pela visão do cliente, como o sistema deve funcionar.

Já no âmbito *social* as metodologias ágeis têm foco na comunicação efetiva, autogerenciamento das equipes e tomada de decisões compartilhadas, dentre outras características. É necessário também que os colaboradores da organização tenham ou adquiram competências não-técnicas como: bom relacionamento interpessoal, eficiência no trabalho em equipe, respeito, colaboração, compartilhamento de conhecimento, etc. Estes valores, em conjunto com bom conhecimento técnico, são considerados a base para formação de uma boa equipe ágil (Abrahamsson, Salo, *et al.*, 2002) (Hazzan e Dubinsky, 2004).

As metodologias ágeis realizam a coordenação de atividades de forma diferenciada do modelo tradicional. Estes métodos incentivam que as decisões referentes à coordenação das atividades sejam realizadas de forma compartilhada pelos papéis do *Product Owner* (P.O.), Scrum master e equipe de desenvolvimento. As metodologias ágeis também ressaltam a necessidade de envolver estes papéis nas decisões relacionadas às atividades que serão desempenhadas, havendo um incentivo para que todos participem. Corbucci e Goldman (2010) em seu trabalho relatam as semelhanças

entre a forma com que as metodologias ágeis realizam suas atividades e a produção de software livre. A coordenação de atividade em ambos os casos é executada de forma compartilhada, incentivando a participações de diversos papeis na tomada de decisão, sendo este um benefício apontado por ambas às áreas.

As metodologias ágeis também possuem métodos, práticas e técnicas que podem aumentar a satisfação do cliente (Boehm e Turner, 2003), além de possibilitar a produção de um sistema em menor tempo (Anderson, 2003). Elas podem também prover: melhorias no processo de desenvolvimento, respostas rápidas a mudanças de requisitos e incentivos à colaboração e comunicação entre os *stakeholders* (Karlström e Runeson, 2006), (Pikkarainen, Haikara, *et al.*, 2008), (Fowler, 2004).

Estudos como os realizados por Salo (2004), Hazzan e Dubinsky (2003), Sharp e Robinson (2004) evidenciam a necessidade de estudar os efeitos da utilização de metodologias ágeis em ambientes reais. Isto deve ser feito visando identificar os efeitos da aplicação e possíveis problemas das metodologias ágeis, ou seja, quais os pontos positivos e negativos que a aplicação destas metodologias produz em projetos reais. A realização de estudos acerca deste tema poderá identificar: a necessidade de melhoria em determinadas técnicas, comprovação da efetividade das práticas utilizadas, avaliação da adaptação dos funcionários à nova forma de trabalho, dentre outros fatores.

Inúmeros estudos sobre a aplicação de metodologias ágeis foram realizados em organizações produtoras de software, principalmente em países situados na Europa, América do Norte e Ásia, ou seja, a maioria dos trabalhos são realizados em organizações situadas no hemisfério norte (Silva, Kon e Torteli, 2005). Por exemplo, Dybå e Dingsøyr (2008) realizaram uma revisão sistemática [referencia sobre Revisões aqui] averiguando os estudos realizados na área de metodologias ágeis com foco em resultados expostos tanto pela área acadêmica quanto pela indústria. Neste estudo foi identificada a necessidade de realizar estudos empíricos voltados a compreender como as metodologias são aplicadas em contextos reais.

No entanto, as organizações dos países da Europa, América do Norte e Ásia possuem características diferentes em relação aos países do hemisfério sul, tanto em aspectos econômicos, como sociais. Silva, Kon e Torteli (2005) ressaltam a necessidade

de compreender como as metodologias ágeis são aplicadas em organizações com características semelhantes, como por exemplo, organizações da América Latina, em especial brasileiras. Além disso, estudos realizados por Goldman e Katayama (2010) e Corbucci, Goldman e colegas (2011) evidenciam como está o movimento ágil no Brasil, apontando a crescente utilização destas metodologias pelas organizações e evidenciando resultados que corroboram com o estudo de Dybå e Dingsøyr (2008). Com isso, esta dissertação se embasa nestes pressupostos para relatar como as organizações brasileiras estão inseridas no contexto de utilização de metodologias ágeis em seus processos de desenvolvimento.

Para ser mais específico, o trabalho aqui descrito apresenta os resultados de um estudo qualitativo, inicialmente planejado para explorar a aplicação de metodologias ágeis em contextos reais em quinze empresas de desenvolvimento de software brasileiras. No entanto, apenas oito empresas responderam o convite para participarem deste estudo e apenas cinco das empresas solicitadas aceitaram a realização deste estudo. Após a realização de dois ciclos de entrevistas, o trabalho teve como delimitação do escopo de pesquisa a realização da atividade de refatoração. A justificativa para a escolha do tema foco deste trabalho é descrita na subseção a seguir.

1.1 Justificativa

Poppendieck (2003) sugere que a principal razão para solicitações de mudanças no desenvolvimento de software é que o processo de negócio em que o software está atrelado sofre constantes evoluções. Assim, modificações no decorrer do desenvolvimento de um sistema de software são comuns e podem ter impacto direto no código fonte, fazendo com que os desenvolvedores acabem alterando o mesmo de maneira desorganizada, potencialmente tornando a tarefa de alteração e inclusão de novas funcionalidades mais complexa e propensa a erros. Segundo Fowler (1999), isso ocorre devido à execução de modificações no código sem que seja feito um planejamento ou que haja uma compreensão do projeto por completo, muitas vezes visando finalizar as atividades no menor prazo. No entanto, esse comportamento acaba por desestruturar o sistema.

A refatoração é uma alternativa para reestruturar o código, aperfeiçoar o projeto e dar mais qualidade ao produto sem inserir erros no mesmo (Fowler, 1999). Ela é uma forma disciplinada de minimizar o número de falhas, deixando o sistema mais flexível, mais reusável, extensível, com maior manutenibilidade, minimizando o número de dependências entre os artefatos criados. Essa redução de dependências possivelmente facilita a coordenação das atividades de desenvolvimento de software (Parnas, 1972). Em outras palavras, ao se reduzir o acoplamento de um sistema, reduz-se também o esforço de comunicação e coordenação necessário para desenvolvê-lo. A existência de uma boa coordenação e um código fonte estruturado facilita a coordenação das atividades (Parnas, 1972), o que é um fator crítico para que a organização consiga reduzir seu tempo de produção. As metodologias ágeis também enfatizam a necessidade de manter o código fonte estruturado para permitir a paralelização de tarefas de forma controlada, sendo a atividade de refatoração umas das principais formas de possibilitar tal prática, bem como uma das premissas fundamentais propostas pelo movimento ágil (Manifesto for Agile Software Development, 2001).

Como mencionado anteriormente, dentre as premissas das metodologias ágeis está a colaboração frequente entre os membros do time, como, por exemplo, no planejamento conjunto e na programação em pares. Essa prática é comumente apontada como um dos aspectos mais importantes para a implantação de metodologias ágeis e ainda indica um amplo potencial de pesquisa que não tem sido efetivamente explorado, especialmente em organizações brasileiras. Entre as exceções podem-se citar os trabalhos de (Whitworth e Biddle, 2007), (Lindvall, Basili, *et al.*, 2002) e (Dybå e Dingsøyr, 2008) no âmbito global e (Silva, Kon e Torteli, 2005), (Corbucci, Goldman, *et al.*, 2011), (Melo e Kon, 2011), (Melo, Santos Jr, *et al.*, 2010) e (Treccani e de Souza, 2011) no âmbito nacional.

Este trabalho descreve um estudo qualitativo, de caráter exploratório e empírico em cinco organizações brasileiras produtoras de software que utilizam as metodologias ágeis em seu processo de desenvolvimento. Os métodos de codificação da teoria fundamentada em dados (Glaser e Strauss, 1967) foram utilizados em conjunto com as entrevistas semiestruturadas como forma de auxiliar no processo de coleta e análise dos dados. A teoria fundamentada em dados permite compreender aspectos específicos

através da percepção dos entrevistados, não cabendo o julgamento da corretude do que está sendo tratado, apenas é analisado como o fato ocorre e seus derivados na perspectiva do informante.

Strauss e Corbin (2008) definem o princípio da emergência da teoria e conceitos da teoria fundamentada em dados (do inglês, *Grounded Theory* - GT). Eles afirmam que as pesquisas que utilizem tal metodologia tenham seus conceitos e sua teoria desenvolvidas ao longo do processo de pesquisa, não tendo no início da investigação um conceito definido sobre o assunto tratado. Strauss e Corbin (2008) ainda afirmam que os conceitos devem emergir dos dados encontrados, assim como a teoria deve ser derivada da análise e fortemente baseada nos dados encontrados: são eles que irão evidenciar e prover a confiabilidade na teoria proposta. Este princípio também é utilizado para guiar a questão de pesquisa e a evolução do estudo, uma vez que à medida que a análise dos dados é realizada, são descobertos novos conceitos e é identificada a necessidade de aprofundar novos aspectos. Desta forma, as questões de pesquisa evoluem e são derivadas no processo de construção da teoria, fazendo com que o estudo seja delimitado e novas questões de pesquisa sejam criadas no decorrer do estudo.

A partir desta premissa, este estudo teve seu foco de pesquisa delimitado à realização da atividade de refatoração. Esta escolha foi feita após a realização de uma revisão bibliográfica que ocorreu em paralelo a execução da análise dos dados encontrados, na qual foi possível identificar que existem poucos estudos com foco na atividade de refatoração, não havendo nenhum deles voltado a esclarecer como esta atividade é aplicada no contexto real de empresas ágeis *brasileiras*. Um dos artigos encontrados através da realização desta revisão bibliográfica foi a revisão sistemática realizada por Dybå e Dingsøyr (2008) que argumenta sobre a necessidade de compreender como as metodologias ágeis são aplicadas em contextos reais.

Silva, Kon e Torteli (2005) em seu estudo relatam a carência de estudos que visam compreender como empresas aplicam metodologias ágeis em seus processos de desenvolvimento de software no hemisfério sul. Eles ressaltam também a necessidade de realizar estes estudos uma vez que estes países possuem diferenças econômicas,

sociais e culturais se comparados aos países no hemisfério norte podendo resultar em formas diversas de aplicação das práticas sugeridas pelas metodologias ágeis.

Assim, após a realização da atividade de amostragem teórica e da revisão bibliográfica, este estudo tem sua justificativa fundamentada no confronto entre estes dois resultados: a necessidade de compreender a utilização de metodologias ágeis em contextos reais em empresas brasileiras e a pequena amostra de pesquisas que enfocam na atividade de refatoração.

1.2 Questão de Pesquisa

As organizações produtoras de software necessitam de rapidez e qualidade, para atender mais rapidamente as demandas de seus clientes. As metodologias ágeis buscam auxiliar essas organizações a alcançarem esse objetivo, por isso elas têm se destacado no mercado produtor de software, onde cada vez mais ganham espaço (Miller, 2009).

Este trabalho foi iniciado tendo como objetivo responder a seguinte questão de pesquisa: “Como as organizações brasileiras estão aplicando as metodologias ágeis em seu processo de desenvolvimento de software?”. Com a utilização de métodos de codificação da teoria fundamentada em dados existe a necessidade de realizar comparações teóricas entre os dados coletados na fase de análise e amostragem teórica. Esta comparação é realizada a fim de identificar os significados dos fatos e forçar o pesquisador a examinar até mesmo suposições básicas de seu ponto de vista através da visão do entrevistado. Este fator auxilia o pesquisador a reduzir o viés facilitando a compreensão e a identificação de fatos que devem ser melhor explorados. Esta necessidade está intimamente ligada ao princípio de emergência da teoria fundamentada em dados, onde à medida que a análise dos dados é realizada são reveladas ou aprimoradas as questões de pesquisa e o foco nos fenômenos estudados.

Com isso, após as primeiras iterações do estudo e a realização da atividade de amostragem teórica da teoria fundamentada em dados, percebeu-se que a prática da refatoração se torna mais colaborativa no contexto ágil. Tradicionalmente (Fowler, 2004) esta prática é realizada de forma individual e visa evitar que os desenvolvedores acabem alterando de maneira desorganizada o código fonte deixando a tarefa de modificação e inclusão de novas funcionalidades mais complexa e propensa a erros,

muitas vezes motivados em cumprir alguma meta, como por exemplo, prazos irreais de desenvolvimento.

Devido a importância da refatoração nas metodologias ágeis, esta prática foi selecionada como objeto de estudo. Dessa forma, foram geradas duas novas questões de pesquisa, mais específicas, a fim de delimitar o escopo deste trabalho, são elas: “Como as atividades de refatoração são realizadas em organizações de desenvolvimento de software que utilizam metodologias ágeis?” e “Quais as vantagens e desvantagens na execução da atividade de refatoração de forma colaborativa?”. São estas duas perguntas de pesquisa que são respondidas nesta dissertação.

1.3 Objetivos

O objetivo desta dissertação é realizar um estudo qualitativo e exploratório em algumas empresas de desenvolvimento de software brasileiras, que utilizam métodos/metodologias ágeis, com intuito de analisar e compreender as metodologias ágeis, principalmente em relação à atividade de refatoração. Como este trabalho possui um caráter exploratório primeiramente foram definidas as questões de pesquisa mais amplas visando compreender como as organizações aplicavam as metodologias ágeis em seu processo de software. Este objetivo principal está subdividido nos seguintes objetivos específicos:

- Descobrir como as empresas aplicam as metodologias ágeis em seus processos;
- Identificar as mudanças nos aspectos sociais, organizacionais e técnicos que a utilização de metodologias ágeis provoca;
- Identificar quais os problemas enfrentados pelas organizações e seus colaboradores em aplicar as metodologias ágeis;
- Contribuir com a evolução do estado da arte em metodologias ágeis através da análise de como as organizações brasileiras estão utilizando estas metodologias.

Após a realização de algumas etapas de coleta e análise de dados, este trabalho teve seu foco ligeiramente modificado, uma vez que observou-se que a atividade de

refatoração é realizada de maneira colaborativa, algo não identificado na literatura. Após esta delimitação do tema, os objetivos foram redefinidos e aprimorados para melhor compreender como a atividade de refatoração colaborativa era realizada, gerando os seguintes objetivos específicos:

- Compreender como as atividades vinculadas a refatoração, que se torna colaborativa neste contexto, são realizadas, avaliando seus impactos e benefícios visando aspectos sociais e técnicos desta nova abordagem;
- Descrever como a atividade de refatoração colaborativa é realizada nos projetos, identificando as dificuldades de execução, benefícios e características desta atividade;
- Compreender e descrever as atividades que apoiam a refatoração colaborativa, antes e após a sua execução, verificando impactos e benefícios de sua execução; e
- Finalmente, contribuir com a evolução do estado da arte em metodologias ágeis através da análise e descrição de como a atividade de refatoração colaborativa é realizada pelas organizações brasileiras.

1.4 Organização do trabalho

Esta dissertação está organizada em seis capítulos e dois apêndices. O capítulo 1 descreve a justificativa para realização deste trabalho, a questão de pesquisa abordada por este estudo, bem como os objetivos gerais e específicos relacionados a esta dissertação.

O capítulo 2 apresenta a teoria em que este trabalho está fundamentando, aprofundando nas metodologias ágeis utilizadas pelas empresas estudadas. São apresentados também os trabalhos correlatos a este.

No capítulo 3 são apresentadas as metodologias de pesquisa utilizadas para guiar este estudo. Ele apresenta os conceitos acerca da técnica de entrevistas, utilizada na fase de coleta de dados e alguns fundamentos da teoria fundamentada em dados (Glaser e Strauss, 1967) (Strauss e Corbin, 2008), metodologia esta que foi utilizada na etapa de análise dos dados.

O capítulo 4 descreve como o estudo foi realizado, evidenciando como foi realizada a etapa de coleta dos dados e a utilização de métodos de codificação da teoria fundamentada em dados para análise dos dados.

O capítulo 5 descreve os resultados encontrados em categorias, bem como as citações retiradas das entrevistas que embasam os resultados deste trabalho.

O capítulo 6 contempla uma discussão sobre os resultados obtidos neste estudo e relatados no capítulo 5, realizando análises e comparações com os trabalhos correlatos identificados no capítulo 2 e com a revisão bibliográfica realizada.

Por fim, o capítulo 7 aborda as considerações finais, contendo as principais contribuições, limitações e trabalhos futuros deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada a fundamentação teórica utilizada neste trabalho. Devido ao grande volume de material encontrado na revisão da literatura optou-se por descrever aqui apenas as metodologias e práticas ágeis utilizadas pelas organizações onde este estudo foi realizado. Ressaltando características como o ciclo de vida das principais metodologias, principais práticas utilizadas no processo de desenvolvimento das organizações. Buscou-se abordar também acerca das técnicas de compartilhamento de informação, dentre outros aspectos de cada metodologia ágil descrita. Uma segunda parte deste capítulo abordada os trabalhos relacionados a este, realizando uma breve descrição de cada trabalho evidenciando a interseção dos mesmos com este.

2.1 Metodologias Ágeis

Organizações de desenvolvimento de software geralmente enfrentam grandes desafios na produção de seus sistemas. Grande parte disso está relacionada à volatilidade de seus requisitos (Beck, 2000), (Boehm e Turner, 2005), (Pikkarainen, Haikara, *et al.*, 2008). O mercado de software possui um dinamismo em sua forma de trabalho, uma vez que é necessário modificar e/ou aprimorar suas técnicas, métodos ou processo de trabalho constantemente em busca de vantagens comercias em um mercado altamente competitivo.

Este dinamismo é refletido diretamente no software, que deve atender as mudanças solicitadas no decorrer do processo de trabalho, uma vez que o mesmo está em constante criação/ manutenção para atender as necessidades dos clientes (Cavamura Júnior, 2008). Este dinamismo é notado no estudo de Poppendieck e Poppendieck

(2003), onde é relatado que a causa principal das mudanças na produção de um software está intimamente ligada à evolução do processo de negócio, uma vez que o software que está sendo produzido deve acompanhar a evolução do mesmo. Como consequência deste dinamismo, novas funcionalidades são solicitadas constantemente, bem como mudanças parciais no produto e até mesmo a exclusão de funcionalidades, causando impacto imediato no sistema produzido.

Com isso, as organizações produtoras de software têm a necessidade de se adaptar rapidamente a mudanças, mantendo a qualidade do produto, agilidade no desenvolvimento e entrega, além de constantes inovações. Neste contexto as metodologias ágeis podem auxiliar essas organizações a obter o dinamismo necessário para acompanhar as mudanças de requisitos mais rapidamente. Essa realidade tem se destacado no mercado de software, onde cada vez mais, as metodologias ágeis ganham espaço (Miller, 2009).

Estas metodologias recomendam uma comunicação mais intensa entre fornecedores e clientes, de forma que as mudanças possam ser realizadas em tempo de desenvolvimento. Uma das premissas do movimento ágil é que as organizações produtoras estejam dispostas a realizar mudanças no sistema de acordo com a necessidade do cliente, pois estas mudanças são encaradas como parte natural do processo de desenvolvimento (Beck, 2000) (Lindvall, Basili, *et al.*, 2002) (Schwaber e Beedle, 2001).

Como exemplos de metodologias ágeis se têm: Adaptive Software Development (ASD) (Highsmith, 1997) e (Highsmith, 1999); Dynamic Systems Development Method (DSDM) (Stapleton, 1997); eXtreme Programming (XP) (Beck, 2000); Família Crystal (Cockburn, 2000) e (Cockburn, 2004); Feature Driven Development (FDD) (Coad, Luca e Lefebvre, 1999) e (Palmer e Felsing, 2002); LEAN software Development (Poppendieck e Poppendieck, 2003) e Scrum (Schwaber, 1995) e (Schwaber e Beedle, 2001). Porém neste estudo será descrita a metodologia *Scrum* e algumas práticas de outras metodologias, especialmente do XP, visto que estas são a base do processo utilizado por todas as empresas estudas por este trabalho.

2.1.1 Princípios Ágeis

No ano de 2001, dezessete pesquisadores, com diversos papéis na área de desenvolvimento de software, se reuniram para discutir alternativas à utilização de processos burocráticos, metodologias e práticas tradicionais de desenvolvimento e gerência de projetos. A partir desta reunião foi proposto o "Manifesto para Desenvolvimento Ágil de Software" (Manifesto for Agile Software Development, 2001) que é um conjunto de regras, princípios e valores a serem considerados para criação de novas metodologias, consideradas mais leves, menos burocráticas e mais ágeis.

Segundo Silva (2009) os métodos ágeis (metodologias baseadas no manifesto ágil) são métodos leves, propostos como alternativas as metodologias de desenvolvimento de sistemas tradicionais, consideradas pesadas, sistemáticas ou dirigidas a plano. Estas metodologias são baseadas em quatro valores descritos pelo manifesto (Manifesto for Agile Software Development, 2001):

- **Indivíduos e Interações** são mais importantes que processos e ferramentas;
- **Software funcionando** é mais importante que documentação completa e detalhada;
- **Colaboração com o** cliente é mais importante que negociação de contratos; e
- **Adaptação a mudanças** é mais importante que seguir um plano.

Com base nestes valores o manifesto ágil propõe 12 princípios para as metodologias ágeis, são eles (Manifesto for Agile Software Development, 2001):

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.

5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9. Atenção contínua a excelência técnica e bom design aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajusta e otimiza seu comportamento de acordo.

É necessário observar que a utilização de metodologias ágeis não exclui a utilização de processos, ferramentas ou técnicas, atividades de documentação e nem atividades voltadas ao planejamento ou negociação de contratos. As metodologias ágeis ressaltam os valores relacionados ao indivíduo e suas interações, ao sistema produzido com qualidade e rapidez, à colaboração entre o cliente e a equipe produtora do sistema e à adaptação e respostas rápidas a solicitações de mudanças (Silva, 2009).

2.1.2 Scrum

A metodologia Ágil *Scrum* foi criada em 1995 por Ken Schwaber e Jeff Sutherland a partir de estudos nas áreas de controle de processos industriais e teorias de gestão de processo e projetos do modelo de produção da Toyota (Nonaka e Takeuchi, 1986). Com o tempo, Mike Beedle em conjunto com Ken Schwaber, aprimoraram os conceitos sobre *Scrum* (Schwaber e Beedle, 2001). Esta metodologia, além de se basear nos princípios ágeis, dá ênfase nos fatores de gerenciamento de projetos e gerenciamento e controle do processo (Abrahamsson, Salo, *et al.*, 2002), (Sutherland e Schwaber, 2007), (Rooijen, 2006), utiliza uma abordagem iterativa incremental, com foco em pessoas e

em seu desenvolvimento. É comumente utilizada em projetos com grade volatilidade nos requisitos e quando a interação com cliente é possível (Zanatta e Vilain, 2005).

O *Scrum* possui um processo de desenvolvimento simples, voltado à organização das equipes e atividades, a fim de aumentar a produtividade e a qualidade do produto desenvolvido. É considerada uma metodologia leve, no contexto de desenvolvimento, onde é permitido que a equipe, auto gerenciável, escolha a forma mais adequada de desenvolver suas atividades, bem como a quantidade de tempo e atividades por ciclo de desenvolvimento (Schwaber, 1995).

Esta metodologia ágil é normalmente recomendada quando os projetos possuem as seguintes características:

- Equipes pequenas, sendo possível escalar as equipes, de forma que existam vários times *Scrum* controlados por uma pessoa ou equipe, técnica esta conhecida como *Scrum of Scrums* (Scrum de Scrums, em português). O *Scrum* de *Scrums* pode ser usado em grandes projetos (Jeffery, Bannerman e Hossain, 2011), bem como em projetos distribuídos (Seiyoung e Hwan-Seung, 2010), (Sutherland, Schoonheim, *et al.*, 2008) (Sutherland, Schoonheim e Rijk, 2009), sendo uma área com grande foco de pesquisa atualmente;
- Requisitos pouco conhecidos ou poucos estáveis, normalmente são requisitos que podem sofrer bastantes modificações durante o processo de desenvolvimento;
- Iterações curtas, no máximo 4 semanas de desenvolvimento para entrega de parte de um sistema, com funcionalidades que possuam um valor agregado a atividade do cliente, ou seja, as entregas dos sistemas devem poder ser quebradas em funcionalidades úteis para o cliente (Schwaber e Beedle, 2001).

Ainda segundo Schwaber e Beedle (2001) o *Scrum* parte da premissa de que o desenvolvimento de software é uma atividade complexa e imprevisível para realizar todas as estimativas e planejamentos no início do projeto. Por isso o *Scrum* possui atividades de monitoramento e *feedback* constantes, normalmente realizadas através de reuniões diárias rápidas com o intuito de identificar e resolver possíveis problemas ou impedimentos nas tarefas de desenvolvimento que estão sendo realizadas.

2.1.2.1 Ciclo de vida

O ciclo de vida da metodologia *Scrum* é dividido em três partes: Pré-Planejamento, o Desenvolvimento e o Pós-Planejamento (Schwaber, 2004), (Highsmith, 2002). Na Fase de Pré-Planejamento são realizadas duas etapas, o Planejamento e a Arquitetura. A etapa de planejamento, representada na Figura 1 (pré-planejamento), é responsável pela criação do *Product Backlog* do sistema. Esta prática visa criar uma lista de requisitos a serem desenvolvidos após a etapa de levantamento realizada com o cliente.

Após a criação do *Product Backlog*, todos os requisitos levantados devem ser priorizados e estimados pela equipe. Nesta etapa também são discutidos diversos fatores como: alocação da equipe, ferramentas para auxílio do projeto e os riscos que envolvem o projeto e cada requisito levantado. Após a etapa de Planejamento é realizada a etapa de arquitetura. Nela são analisados os requisitos de forma objetiva para identificar quais padrões, técnicas e recursos estarão presentes na arquitetura do sistema, bem como qual linguagem será utilizada.

A fase de desenvolvimento, representada na Figura 1 (desenvolvimento), é composta pelas seguintes atividades:

- **Definição do Timebox da *Sprint*:** esta atividade visa definir o tempo de duração de uma iteração de desenvolvimento. Este tempo deve estar entre duas e quatro semanas de desenvolvimento. É nele também, que é realizada qualquer mudança no *timebox* da *Sprint*, não sendo possível alterá-lo durante a execução de uma *Sprint*. Caso seja necessário alterar o tamanho da *timebox* é necessário aguardar a próxima *Sprint* para redefini-lo.
- **Definição do *Sprint Backlog*:** O *Sprint Backlog* é uma lista de requisitos que serão implementados durante uma *Sprint*. No início da *sprint* são definidos quais requisitos serão removidos do *Product Backlog* e quais serão desenvolvidos naquela iteração. Nesta etapa é levado em consideração o tempo da *Sprint*, a complexidade das atividades e sua prioridade. Nesta etapa as atividades são descritas e caso necessário divididas em atividades menores visando paralelização ou diminuição de complexidade.

- **Execução da *Sprint*:** A execução da *Sprint* é onde as atividades de análise, projeto e implementação e teste de cada requisito são realizadas. No decorrer do desenvolvimento são realizadas as integrações necessárias entre as novas funcionalidades criadas na *Sprint* e o sistema. No decorrer da execução da *Sprint* são realizadas reuniões de *feedback* para identificar possíveis problemas e mitigá-los.

Na fase de Pós-Planejamento, representada na Figura 1 (pós-planejamento), são realizadas as atividades de apresentação das novas funcionalidades do sistema para o cliente e reuniões de retrospectiva da *Sprint*, onde são levantados os pontos fracos, fortes e oportunidades de melhoria para as próximas iterações. Nesta etapa também são realizadas as atividades de testes de *release* e a documentação do sistema.

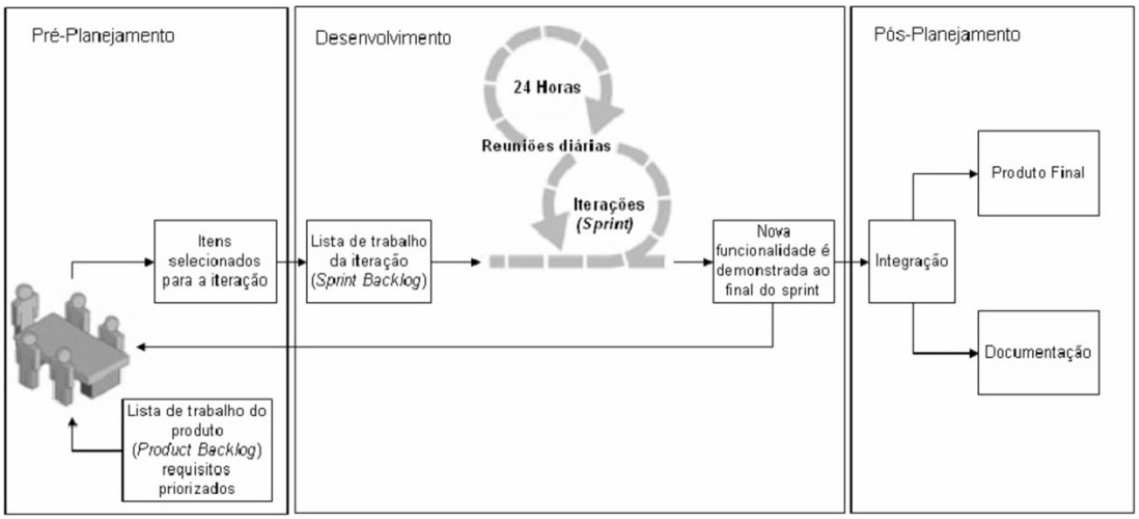

Figura 1 - Fases do Scrum adaptado de (Schwaber, 2004)

2.1.2.2 Papeis

Segundo Schwaber (2004) a metodologia ágil *Scrum* especifica três papéis principais necessários para sua utilização, são eles P.O., *Scrum Master* e o Time. Havendo também a existência de papeis tradicionais, como: os usuários do sistema, o cliente, o administrador ou gestor de negócio, dentre outros papéis.

O P.O. é o papel responsável pela manutenção dos requisitos e pela elaboração e atualização do *Product Backlog*. Sua principal atividade é o gerenciamento deste artefato, incluindo a estimativa e priorização dos itens. Esta atividade deve garantir a

organização, prioridade entendimento e visibilidade dos itens do *Product Backlog* para o time. O papel de P.O. normalmente deve ser representado apenas por uma pessoa para garantir a estabilidade do *Product Backlog*, pode haver um conjunto de pessoas envolvidas nesta priorização, porém a responsabilidade das atividades de manutenção do *Product Backlog* é do P.O., uma vez que isto reflete diretamente no ROI – Retorno sobre investimento (Abrahamsson, 2007), e uma das atividades do P.O. é a manutenção e o aumento do mesmo. Esta atividade é de suma importância para organização visto que o P.O. deve garantir que a construção do produto possua um ganho entre o valor investido e sua rentabilidade.

O Papel de *Scrum Master* é garantir que as cerimônias, práticas e regras do *Scrum* estejam sendo executadas. Ele também é responsável por analisar e adaptar os processos da empresa para tornar o time mais produtivo, incentivar o autogerenciamento da equipe, e a produção do sistema com maior qualidade. O *Scrum master* é responsável também por identificar e remover possíveis impedimentos e problemas que a equipe tenha para realização de suas tarefas.

O papel time é composto por todos os responsáveis pela criação do produto, ou seja, é a equipe que cria o produto. A equipe é responsável por retirar atividades do *Product Backlog* e transformá-las em funcionalidades do sistema em uma determinada *Sprint*. O time deve ser capaz de realizar atividades de análise e programação, controle de qualidade, análise de negócio, arquitetura, projeto de interface de usuário e projeto de banco de dados. O time deve ser auto-organizável, e deve ter capacidade de decidir a forma com que as atividades devem ser realizadas e negociar o tempo e as atividades por *Sprint*, nunca aceitando a diminuição da qualidade do produto para poupar tempo ou custo.

Abrahamsson e amigos (Abrahamsson, Salo, *et al.*, 2002) em seu estudo descrevem mais papéis e responsabilidades para metodologia *Scrum*, são eles: o cliente, responsável por participar das tarefas relacionadas à formação do *Product Backlog*; o Administrador, responsável pelas tomadas de decisões do projeto, e os usuários do sistema, que possuem a responsabilidade de validar junto à cliente o sistema produzido, bem como opinar em melhorias e alterações.

2.1.2.3 Práticas e Atividades

A metodologia ágil *Scrum* possui um conjunto de atividades e cerimônias, voltadas ao planejamento, monitoramento e *feedback* do processo de desenvolvimento. Cada atividade e cerimônia possui uma periodicidade e importância para o projeto, ficando sob-responsabilidade do *Scrum master* a garantia de sua execução. Segundo Abrahamsson e amigos (Abrahamsson, Salo, *et al.*, 2002) o *Scrum* possui sete atividades a serem realizadas durante uma iteração de seu ciclo de vida, são elas: criação/manutenção do *product backlog*, estimativa do esforço das atividades, planejamento da *sprint*, geração do *sprint backlog*, reuniões diárias, reunião de revisão da *sprint* e reunião de retrospectiva da *sprint*.

A atividade de criação/manutenção do *product backlog* é a primeira atividade do fluxo de desenvolvimento. Ela é realizada primeiramente pelo P.O. em conjunto com os *stakeholdes*, usuários final, gerentes e o administrador, onde é definido o documento de visão do projeto. Após obter as informações sobre as necessidades do produto a ser gerado, o P.O. elabora uma lista de requisitos funcionais e não funcionais e os prioriza. Após esta etapa o P.O., em conjunto com o *Scrum master*, realiza a revisão desta lista de requisitos analisando a completude das tarefas, ou seja, se após a execução destas tarefas o projeto estará completo e estará atendendo as necessidades do cliente.

A segunda atividade deste fluxo é a estimativa de esforço das atividades que estão no *product backlog*. Esta atividade é realizada pelo time juntamente com o *Scrum master*. Nela cada requisito é avaliado e estimado na perspectiva de desenvolvimento, atividade base para execução dos próximos passos.

A terceira atividade é a de planejamento da *sprint* feita pelo time, P.O. e *Scrum master*. Nela são definidos os requisitos que serão implementados na *sprint* (montagem do *sprint backlog*) e qual será a meta da *sprint* além de outras variáveis. Esta atividade é dividida em duas etapas, na primeira o time, o P.O. e o *Scrum master*, definem os objetivos e funcionalidades que serão implementadas, esta definição é de comum acordo entre as partes, e fica definido também quais serão os possíveis requisitos que serão implementados na próxima *sprint*. A segunda etapa desta atividade é realizada apenas pelo time e pelo *Scrum master*, onde as atividades são quebradas em itens e tarefas, para

possibilitar a paralelização, facilitar o entendimento e execução das atividades. Nesta fase também é definida uma possível ordem com que as atividades podem ser realizadas.

A quarta etapa do fluxo é a geração da lista de atividades da *sprint*, denominada *sprint backlog*, lista esta formada na segunda etapa da reunião de planejamento da *sprint*. Nesta etapa é comunicado ao P.O. quais serão as funcionalidades implementadas naquela sprint, ou seja, o P.O. deve retirar do *product backlog* estas atividades.

No decorrer da *sprint* são realizadas as reuniões diárias, do inglês, *daily Scrum.* Estas reuniões são realizadas normalmente em pé, pelo time, em conjunto com o *Scrum master*. São reuniões breves onde cada integrante do time deve dizer o que fez no dia anterior, o que irá fazer neste dia e caso houver, qual problema ou impedimento que teve/tem em alguma tarefa. Segundo Schwaber (1995) esta reunião não deve exceder quinze minutos. Esta atividade é de suma importância para identificar possíveis pontos de atraso e dificuldades de evolução do sistema, ficando a cargo do *Scrum master*, sanar estes problemas o mais rápido possível, para não comprometer a meta da *sprint*.

A sexta atividade é a reunião de revisão da *sprint*, envolvendo todos os papeis do *Scrum*. Nela são apresentados os resultados da *sprint,* as metas alcançadas, as funcionalidades implementadas e integradas ao sistema e os artefatos gerados naquela iteração. Esta reunião é realizada ao final de cada *sprint*, ou seja, a reunião acontece no intervalo de no máximo 30 dias, possibilitando que o cliente verifique o produto que está sendo gerado com uma maior frequência. Esta reunião possibilita a obtenção de um *feedback* rápido por parte do cliente sobre o produto gerado, podendo nesse período identificar novos itens para o *backlog*, como melhorias/mudanças no sistema ou novas funcionalidades.

Por fim, é realizada a reunião de retrospectiva da *sprint*. Esta reunião é realizada apenas pelo time e pelo *Scrum master*. Nela são discutidos os pontos fortes e fracos da *sprint*, com o intuito de identificar pontos de melhoria no processo. Nesta atividade o time fica livre para criticar, elogiar e/ou sugerir mudanças no processo. Cada integrante fala sobre os pontos positivos ocorridos na *sprint*, ou pontos negativos e sugere as melhorias sobre estes pontos, os quais são discutidos por toda equipe e pelo *Scrum*

master. O *Scrum master* também atua com um moderador da reunião sendo responsável por representar o interesse do time frente à gerência.

A Figura 2 mostra o fluxo de desenvolvimento do *Scrum* citado acima. Neste fluxo é possível verificar as três fases do *Scrum* e suas respectivas atividades.

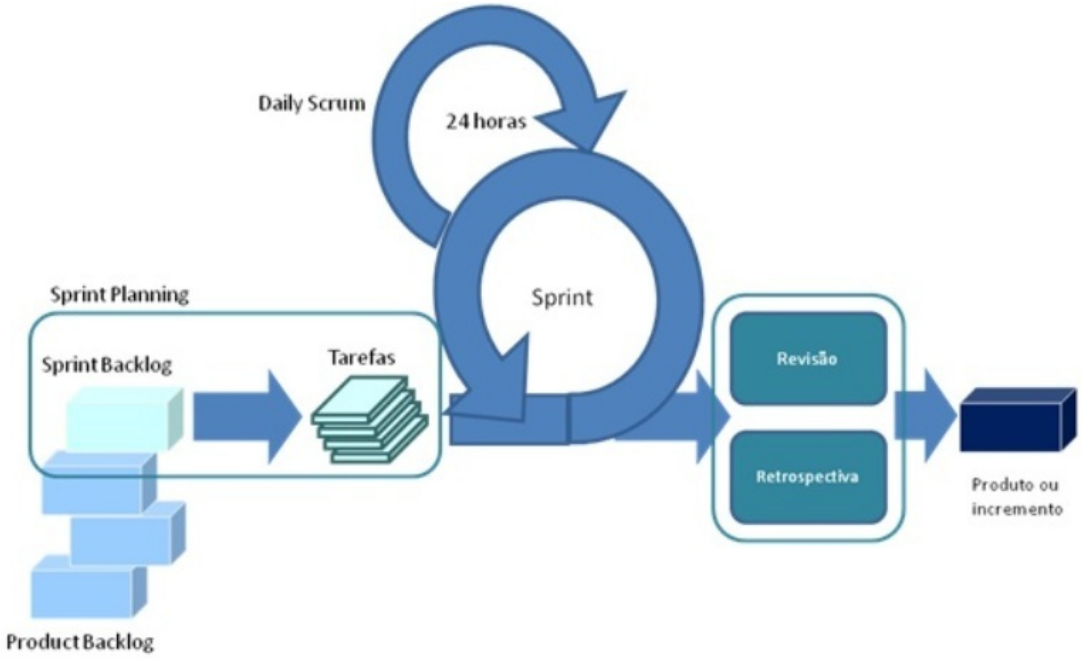

Figura 2 - Fluxo de desenvolvimento do Scrum (adaptado de Schwaber, 2004)

2.1.2.4 Gráficos de controle

A metodologia *Scrum* possui atividades voltadas a gerência de projetos, com isso surge a necessidade de obter índices, gráficos e indicadores do andamento das atividades. Tendo em vista esta necessidade a metodologia *Scrum* propõe a utilização de alguns gráficos de monitoramento. Neste trabalho serão citados dois principais gráficos propostos por Schwaber (Schwaber, 1995) (Schwaber e Beedle, 2001): o gráfico de *product burndown* (Schwaber, 2004) e o gráfico de *sprint burndown* (Schwaber, 2004). Estes gráficos medem indicadores do processo de desenvolvimento do produto no decorrer do tempo de criação.

O gráfico de *product burndown* utiliza os indicadores de tempo decorrido do projeto, quantidade de atividades restantes para conclusão do projeto, quantidade de atividades realizadas do projeto por *sprint* e uma linha de tendência do projeto, que utiliza a regressão linear. Este gráfico verifica a proporção do trabalho que deve ser feito, pelo trabalho já realizado a cada *sprint*, com intuito de indicar quão rápido o time está consumindo o *product backlog* e uma projeção de determinada entrega do produto ou parte dele. A Figura 3 é um exemplo do gráfico de *product burndown*.

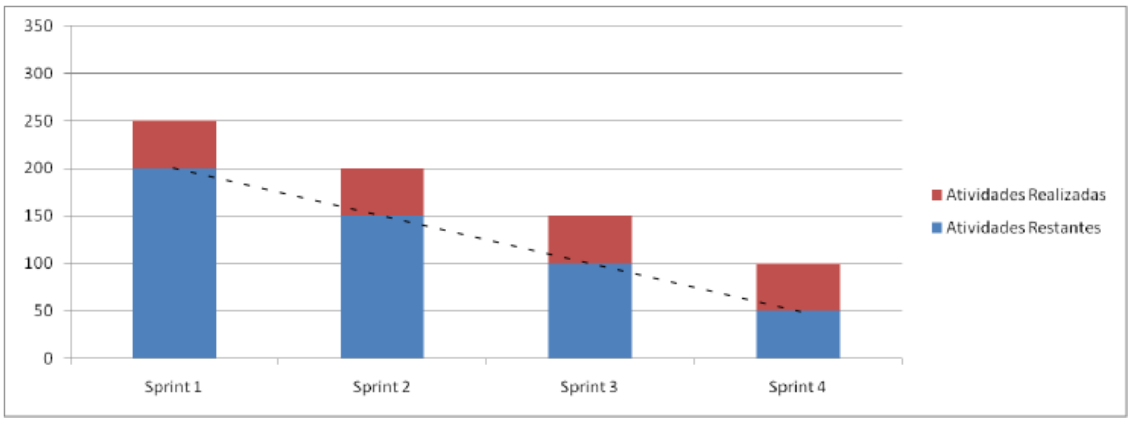

Figura 3 - Gráfico Product Burndown

O gráfico *sprint burndown* utiliza os seguinte indicadores: *timebox* da *sprint*, total de atividades ou pontos estimados para *sprint* e o total de atividades concluídas por dia da *sprint*. Este gráfico verifica o andamento da *sprint* diariamente, indicando a velocidade do time em executar as tarefas e o progresso das atividades, confrontando os resultados planejados inicialmente com os resultados obtidos. A Figura 4 é um exemplo do gráfico de *sprint burndown*.

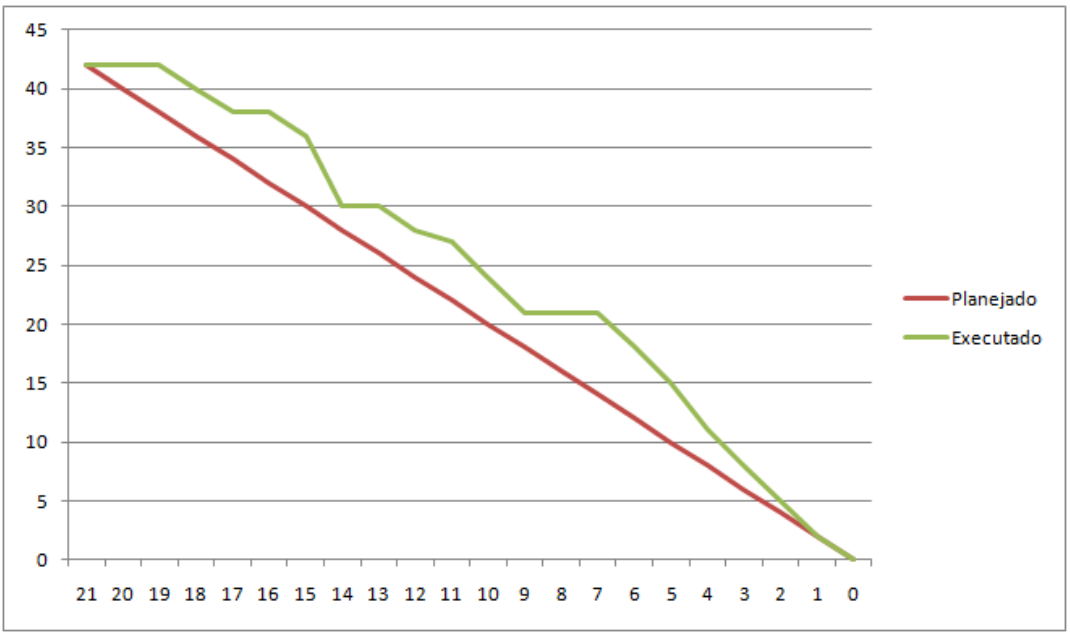

Figura 4 - Gráfico Sprint Burndown

2.1.3 Valores e Práticas Ágeis em Geral

Embora as metodologias ágeis apresentem diferenças entre si elas compartilham alguns valores e práticas, considerados pontos fortes da implantação das mesmas. As próximas subseções apresentam alguns exemplos disso.

2.1.3.1 Código coletivo

O sentimento de código coletivo ou código compartilhado entre os envolvidos na equipe de desenvolvimento é fortemente sugerido pelas metodologias ágeis. Este sentimento diz respeito a não haver "um dono" do código fonte, ou seja, todos os envolvidos no desenvolvimento são autores do código fonte, sendo de responsabilidade de todos aprimorá-lo e mantê-lo (Beck, 2000). Com isso é necessário que a equipe tenha a liberdade para melhorar e refatorar parte do código fonte quando for identificado, prezando sempre pela qualidade do produto gerado, independente de por quem ele foi feito (Beck, 2004).

Segundo Silva (2007) o modelo de propriedade coletiva do código valoriza a comunicação, a colaboração e incentiva o compartilhamento do conhecimento entre os envolvidos. A utilização desta prática também reduz o risco de apenas a pessoa que criou o código ter o conhecimento da funcionalidade, uma vez que o time adquire uma

visão ampla do sistema, evitando com isso que o conhecimento seja perdido. Esta prática também promove o desenvolvimento com maior agilidade e qualidade uma vez que todos conhecem o funcionamento do código e caso algo de errado seja identificado, é imediatamente corrigido. Silva (2007) também relata que esta prática valoriza a coragem e simplicidade, valores largamente apoiados pela metodologia ágil XP.

2.1.3.2 Programação em Pares

Esta prática, oriunda da metodologia XP, exige que cada funcionalidade seja produzida por um ou mais pares de programadores (Beck, 2000). Esta prática indica que o par de programadores deve utilizar o mesmo computador para desenvolver uma tarefa, de preferência utilizando programadores com conhecimento diverso para que um complemente o outro. Esta prática pode promover os seguintes benefícios: maior comunicação entre os envolvidos, maior produtividade, maior qualidade do produto, disseminação de conhecimento, entre outras (Beck, 2000).

Segundo Beck (2000) a tarefa executada em par é realizada da seguinte forma: primeiramente o par discute e define como será implementada a solução para determinada tarefa. Após a solução definida, um dos programadores assume o papel de "piloto", ou seja, é ele quem irá escrever o código, enquanto isso o outro programador irá auxiliar na criação do código. O segundo programador, verifica possíveis erros que o primeiro programador cometeu e não identificou, sugere modificações na forma em que está sendo executada a solução, tendo um aspecto mais estratégico no desenvolvimento. Após certo tempo de desenvolvimento (realizado de forma constante e com intervalos definidos) os papéis se invertem: quem está escrevendo o código passa a auxiliar e quem está auxiliando passa a escrever o código. Para utilizar esta prática é necessário que os desenvolvedores estejam em harmonia, tenham uma boa comunicação e se empenhem em realizar a implementação da melhor forma possível (Silva, 2007).

Outro fator importante para o bom andamento desta forma de realizar a atividade de codificação é a definição dos pares, sempre havendo um rodízio entre os mesmos a fim de disseminar o conhecimento entre todos. É importante também que os programadores não se sintam reprimidos em realizar críticas construtivas sobre o código, devendo haver confiança entre quem está desenvolvendo e quem está auxiliando. Bassi Filho

(2008) afirma que, com o rodízio entre os pares, a replicação de um conhecimento específico do projeto é compartilhado entre todos os programadores, promovendo um maior conhecimento sobre o sistema e evitando que erros sejam inseridos neste processo.

A utilização da prática de programação em pares está sendo difundida tanto entre as empresas que utilizam as metodologias ágeis como em empresas que não as utilizam. Estudos como os realizados por Sharp e Robinson (2004), Cockburn e Williams (2000), Abrahamsson, Salo, *et al.* (2002) e Silva (2007) comprovam a efetividade desta prática em projetos ágeis e em projetos que usam metodologias tradicionais.

2.1.3.3 Refatoração

A refatoração, uma das premissas fundamentais do movimento ágil (Manifesto for Agile Software Development, 2001), é uma alternativa para reestruturar código já existente, aperfeiçoar o projeto e dar mais qualidade ao produto sem inserir erros no mesmo (Fowler, 1999). Fowler (1999) alega ainda que a perda da estrutura do código tem efeitos diretos na qualidade do mesmo, pois quanto mais difícil a visualização do projeto a partir do código, mais difícil será preservá-lo, mantê-lo e rapidamente ele se desestruturará.

Esta atividade visa alterar o código fonte a fim de reestruturá-lo sem alterar seu comportamento externo, corrigir *bugs* ou não conformidades ou adicionar novas funcionalidades ao sistema (Fowler, 1999), preservando sua estrutura, compreensão e legibilidade. Lippert e Roock (2006) afirmam em seu trabalho que a criação e manutenção do *design* da aplicação é uma tarefa difícil, uma vez que alterações no sistema são realizadas constantemente, na maioria das vezes, sem um planejamento arquitetural prévio, devido o tempo para realização da tarefa. Alterações no código fonte são realizadas sem o conhecimento amplo, ou seja, sem saber os efeitos que esta modificação irá provocar no sistema e seu comportamento, no que pode acarretar na perda da estrutura do sistema e seu design. Para resolver este problema é sugerida a realização da atividade de refatoração. Esta é uma forma disciplinada de minimizar o número de falhas, deixando o sistema mais flexível, mais reusável, extensível,

compreensível, com maior manutenibilidade, e minimizando o número de dependências entre os artefatos criados (Fowler, 1999).

No trabalho realizado por Fowler (1999) são apresentados 72 padrões de projetos, que são boas práticas para solucionar um determinado problema. Um padrão de projeto é composto pela descrição de um problema a ser resolvido, o desenho da arquitetura e solução sugerida, quando esta abordagem deve ser utilizada e as vantagens e desvantagens sobre a utilização desta abordagem. A atividade de refatoração utiliza a aplicação de padrões de projetos para realizar as devidas alterações na arquitetura do sistema sem afetar o comportamento externo do mesmo, uma vez que estas são soluções testadas e com efetividade comprovada, sendo formalizadas e se tornando um padrão para solução de determinado problema. Gamma, Helm, *et al.* (2000)e Stroggylos e Spinellis (2007) também relatam a importância da utilização de padrões de projetos na atividade de refatoração, evidenciando que a mesma identifica o problema e necessidade de alterar o código fonte, já os padrões de projeto definem uma abordagem testada e padronizada para solucionar o mesmo.

Devido à necessidade das metodologias ágeis em encarar as modificações do sistema como algo comum e inerente ao seu processo de desenvolvimento, é necessário que a arquitetura do sistema esteja organizada e extensível para receber as mudanças e novas funcionalidades, sendo isto provido principalmente pela atividade de refatoração. A prática da refatoração também é executada e relatada como incentivo ao princípio de *design* simples e na manutenção da arquitetura da metodologia ágil XP (Beck, 2000). Sendo relatada também na metodologia *Scrum* (Schwaber, 2004) entre outras.

2.1.3.4 Ênfase na fase de Testes de Software

Diversas metodologias ágeis apontam a tarefa de testes de software como fundamental para o processo de desenvolvimento afirmando que a realização desta é essencial para validação do funcionamento do sistema e garantir a qualidade do produto gerado (Beck e Fowler, 2001) (Beck, 2004). A metodologia ágil XP descreve a atividade de testes como parte intrínseca do processo de desenvolvimento, havendo a necessidade de participação dos desenvolvedores e dos clientes na definição dos cenários de testes (Beck, 2000), antes mesmo da etapa de desenvolvimento das

funcionalidades. A partir destas informações é possível construir testes de unidade automatizados para todos os componentes do sistema. Além da ênfase dada à realização de testes de unidade automatizados é evidenciando também a criação de testes de aceitação (testes de funcionalidade) realizados em conjunto com cliente.

Devido ao fato de que nas metodologias ágeis as entregas do sistema para o cliente são mais frequentes, é necessário haver uma cobertura de testes, tanto de unidades quanto de aceitação, para todo o sistema a fim de identificar os *bugs* inseridos na aplicação na construção de novas funcionalidades. Para manter a qualidade na geração destas entregas frequentes é necessário que a cada construção do sistema todos os testes sejam realizados, garantindo com isso que nenhum *bug* foi inserido. Beck (2004) sugere que os testes sejam construídos antes da construção da funcionalidade, a fim de garantir que a cada componente construído tenha realmente o comportamento desejado. Beck (2004) também afirma que a cada integração de nova funcionalidade ao sistema e a cada construção do sistema os testes de unidade e teste de aceitação devem ser executados, com isso sendo possível identificar possíveis problemas previamente.

Sato (2007) evidencia os benefícios com a construção de testes automatizados para o processo de desenvolvimento e garantia da qualidade do produto, sendo eles: ênfase na construção de testes voltados a aspectos no desenvolvimento do sistema; preocupação em identificar problemas como acoplamento, coesão e generalização desnecessária já na etapa de construção de testes de unidade; confiança no código escrito, uma vez que os testes de unidade irão verificar se o sistema continua tendo o mesmo comportamento após a inclusão de novas funcionalidades ou modificações da mesma; identificar previamente os comportamentos indesejáveis para o sistema.

Além destes benefícios Beck (2000) relata que a realização de testes automatizados aumenta a confiança dos programadores e do cliente em possuir mais de uma forma de garantir que o sistema está funcionando de fato, ressaltando a importância de realizar esta atividade em conjunto com o cliente.

2.2 Trabalhos Correlatos

Nesta subseção serão apresentados alguns estudos na área de aplicação de metodologias ágeis em organizações produtoras de software. A revisão sistemática

realizada por (Dybå e Dingsøyr, 2008) evidencia uma grande produção científica da academia na área de metodologias ágeis. Porém, ela evidencia poucos trabalhos sobre os efeitos da implantação de metodologias ágeis em organizações produtoras de software. Nesta seção serão apresentados alguns estudos encontrados na revisão sistemática realizada por Dybå e Dingsøyr (2008) a fim de ilustrar a diversidade de abordagens, metodologias de pesquisas, enfoques e resultados encontrados, correlacionados ao tema desta dissertação.

Um destes trabalhos é o estudo de Sharp e Robinson (2004) foi realizado um estudo etnográfico sobre práticas oriundas da metodologia XP. Este trabalho é relevante na área de estudos qualitativos em metodologias ágeis e aponta como a metodologia XP e suas práticas se enquadram nas metodologias ágeis, ressaltando aspectos culturais, de desenvolvimento, de times e descrevendo a comunidade que utiliza a metodologia ágil XP.

Sharp e Robinson (2004) descrevem um estudo etnográfico em uma empresa de desenvolvimento com bastante experiência na metodologia XP, onde foram analisadas as 12 práticas propostas pela metodologia XP. Um dos autores do estudo conduziu a observação e participou do dia a dia da equipe para melhor compreender como as práticas eram realizadas na perspectiva da equipe estudada. Foram realizadas notas de campos, gravações de áudio de discussões e reuniões, fotografias do ambiente físico e cópias de alguns artefatos relacionados às práticas do XP.

Como resultado deste trabalho se tem uma rica descrição do ambiente físico de trabalho que a equipe estudada utilizava, de como eram realizadas as atividades, práticas e cerimônias do XP pela equipe, pela perspectiva das pessoas estudadas, bem como pelas suas impressões durante a observação. Além disso, há uma discussão sobre como as práticas adotadas do XP beneficiavam o desenvolvimento do sistema, ressaltando aspectos culturais, sociais e comportamentais da equipe. O trabalho relatou também evidências empíricas sobre a aplicação das 12 práticas apontadas por (Beck, 2000).

Em um outro estudo, Silva, Kon e Torteli (2005), apresentam um dos estudos pioneiros na área de aplicação da metodologia ágil XP em organizações brasileiras .Neste trabalho são apresentados relatos de experiência sobre a aplicação do seu estudo

de caso em uma *startup* brasileira onde foi introduzida a metodologia ágil XP evidenciando fatores econômicos e culturais que impactam na introdução desta metodologia. Este artigo relata que a empresa estudada realizou treinamentos em práticas de desenvolvimento de software, nas tecnologias que seriam utilizadas no projeto e na utilização da metodologia ágil XP. Esta empresa teve também um *coach* para treinamento e auxílio na utilização da metodologia ágil. O estudo de caso trás como um de seus resultados: Os benefícios da utilização de práticas do XP no desenvolvimento do sistema, como por exemplo, vantagens da programação em par, disseminação do conhecimento, reunião de retrospectiva, design simples da arquitetura, refatoração, integração contínua, entre outras.

Martin, Biddle e Noble (2009a) relatam em seu trabalho um estudo qualitativo em onze empresas, que teve como objetivo a identificação de oito práticas com foco no cliente (atividades que devem ser realizadas em conjunto com o cliente). Neste estudo foram realizadas 66 entrevistas semiestruturadas com a utilização de métodos da teoria fundamentada em dados para análise da aplicação de práticas do XP em empresas de diversos países do hemisfério norte. Os projetos estudados variam de tamanho e de domínio de negócio e foram classificados e agrupados conforme suas características. Em conjunto com a realização das entrevistas foram realizadas observações presenciais nas empresas estudadas, a fim de complementar e comprovar os resultados encontrados a partir da utilização da teoria fundamentada em dados (Strauss e Corbin, 2008). Além do relato das oito práticas encontradas no estudo, foram descritas as relações entre a utilização destas práticas aliadas às práticas propostas por Beck (2004) e os benefícios obtidos na produção dos sistemas.

Utilizando o mesmo estudo qualitativo Martin, Biddle e Noble (2009b) identificam e descrevem sete tipos de papéis e responsabilidades voltados aos *stakeholders*, denominados de "time cliente" no estudo. Dentre os papéis e responsabilidades encontrados no estudo se pode citar:

- "*Geek Interpreter*", responsável por melhorar a comunicação e colaboração entre os programadores em relação às regras de negócio a serem desenvolvidas;

- "*Negotiator*", responsável por intermediar interesses entre os usuários finais e os *stakeholders*, sendo um papel tradicional citado em metodologias tradicionais (DeMarco, 1979);

Melo e Ferreira (2010) relatam um estudo de caso sobre a implantação da metodologia ágil XP em uma organização de grande porte no cenário brasileiro. Este estudo se baseia em quatro questões de pesquisa, que são: "Métodos/Metodologias Ágeis (MAs) aceleram o aprendizado de novas tecnologias, conceitos e padrões?", "MAs aumentam a qualidade do código do sistema?", "MAs aumentam a produtividade do time?", "MAs aumentam a satisfação do cliente?". Como forma de coleta de dados foram realizadas: observações, entrevistas, questionários online, e análise de bases de dados organizacionais. Como resultado deste trabalho é exposto um relato sobre como a organização implantou a metodologia XP, os projetos pilotos para análise de viabilidade e uma descrição sobre os efeitos gerados na organização com a introdução do XP. Um exemplo deste efeito é a melhoria na qualidade do código fonte gerado, uma vezes que o mesmo tiveram melhoria nas seguintes áreas: Aderência a regras de análise estáticas, cobertura de código por testes de unidades, dentre outras melhorias.

Em Melo e amigos (2010) é apresentado um estudo de caso que visa identificar fatores associados ao estímulo do aprendizado em times ágeis. Neste estudo foram aplicados métodos de pesquisa quantitativos e qualitativos. A análise temática foi utilizada como metodologia qualitativa no estudo e resultados e testes estatísticos (cálculo da mediana, moda e testes não paramétricos, como, correlação de *spearman*, regressão linear) para a etapa de análise quantitativa dos dados. Como resultado deste trabalho se têm os fatores que influenciam no aprendizado em times ágeis; como a aplicação do XP trouxe benefícios a curva de aprendizado do time; evidências de que a utilização da prática "*learning-by-doing*" melhorou no aprendizado, dentre outros resultados.

3 METODOLOGIA

Neste capítulo será descrita a fundamentação teórica acerca da metodologia de pesquisa utilizada neste trabalho. Será abordada a metodologia de entrevista utilizada para coleta de dados e alguns métodos da teoria fundamentada em dados que foram utilizados para análise dos mesmos, bem como a descrição de como foram aplicadas tais metodologias. Também será apresentada uma descrição de cada empresa estudada, bem como suas características e de seus colaboradores, seu ambiente físico, entre outros, de forma similar ao estudo realizado por Sharp e Robinson (2004).

3.1 Entrevistas

A realização de entrevistas proporciona a coleta de informações sobre um determinado tema. Uma entrevista pode ser feita em uma conversa a dois ou em grupo, dependendo da iniciativa do entrevistador. As entrevistas visam fornecer informações pertinentes a um objeto de pesquisa (Minayo, 1993) (Minayo, 1996). Entrevistas podem ser feitas por contato face a face, por telefone, pela internet, dentre outros meios.

Segundo DeWalt (2002), existem vários tipos de entrevistas, que podem ser classificados de acordo com dois aspectos: o grau de controle dos pesquisadores e informantes e o grau de uniformidade dos estímulos (questões) apresentados aos informantes. Em relação ao grau de controle, é possível exemplificar da seguinte forma. Em um caso em que ocorre a observação do participante, onde o pesquisador apenas observa e não participa da conversa, o grau de controle é na sua maior parte do informante. Em outro caso, onde o participante responde um questionário com questões previamente formuladas, o controle é maior do pesquisador, pois o informante irá seguir

o roteiro pré-definido e responder somente aspectos que estão dentro do universo de questões que o pesquisador definiu.

Em relação ao grau de uniformidade dos estímulos (questões) apresentados, quando se necessita tratar de assuntos ou temas diferentes de acordo com o participante, ou não se pretende realizar da mesma forma as perguntas a todos os informantes, estes são considerados estimulados de diferentes maneiras ou estímulo pouco uniforme. Já quando se utiliza as mesmas perguntas para todos os participantes, estes são submetidos a estímulos idênticos.

A maioria dos pesquisadores qualitativos utilizam os tipos de entrevistas que possuam, de forma balanceada, os aspectos citados acima. A Figura 5 ilustra os tipos de entrevista mais comuns, e seu relacionamento com cada aspecto. É possível notar que quanto menos estruturadas forem as questões aplicadas, o grau de controle do informante é maior, e a uniformidade do estímulo é menor. Por outro lado, quanto mais estruturadas forem as questões, menor é o grau de controle do informante e maior é o a uniformidade do estímulo.

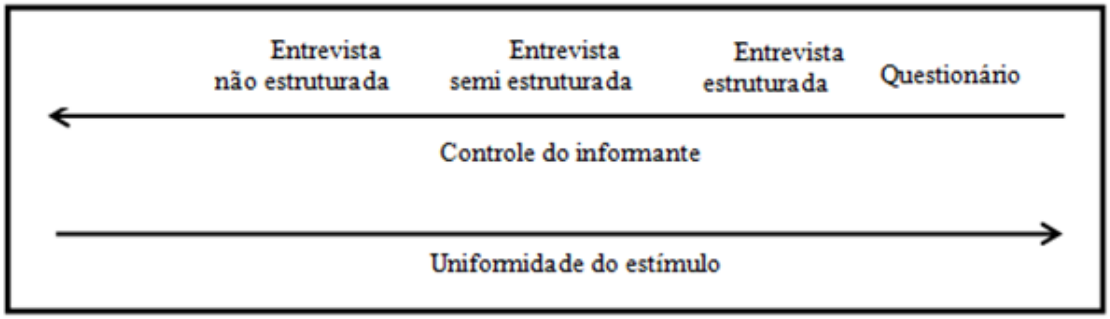

Figura 6 - Tipos de entrevistas em relação aos aspectos (Adaptado de DeWalt (2002)).

Quanto aos tipos de entrevistas descritos na figura acima, tem-se: entrevista não estruturada, entrevista semiestruturada, entrevista estruturada, e questionário. Os tópicos abaixo descrevem com mais detalhes cada tipo de entrevista:

- **Entrevista informal:** o pesquisador segue sob a liderança dos participantes, mas faz perguntas ocasionais para focar no tema ou para esclarecer pontos importantes. Neste caso, o pesquisador não precisa direcionar a pesquisa,

mas apenas seguir os tópicos discorridos pelos entrevistados durante o fluxo natural da conversa;

- **Entrevista não estruturada:** o pesquisador tem normalmente um guia com os tópicos a serem discutidos na entrevista, mas não há um fluxo definido para as perguntas. O pesquisador pode exercer um pequeno nível de controle para direcionar o foco do assunto quando for possível;
- **Entrevista semiestruturada:** o pesquisador possui um guia de entrevista com uma lista de sugestões de questões e dicas que seguem mais ou menos a mesma ordem para cada entrevista. Este roteiro possibilita ao informante, expor seus pensamentos, experiências, visões ou ponto de vista, sobre determinado assunto (Triviños, 1987). Dessa forma, o pesquisador busca garantir que todos os tópicos necessários serão abordados, no entanto com a flexibilidade de poder incluir ou retirar questões de acordo com a necessidade de cada entrevista;
- **Entrevista estruturada:** o pesquisador possui uma lista de questões pré-definidas, tal que sejam feitas as mesmas perguntas para todos os entrevistados, seguindo uma mesma ordem de questões;
- **Questionário:** o pesquisador elabora um conjunto de questões com alternativas, para que o entrevistado indique qual a mais correta no seu ponto de vista. As alternativas podem variar de acordo com uma escala determinada ou possuir respostas específicas sobre determinado assunto;

A distinção entre os tipos de entrevistas está relacionada ainda com os tipos de perguntas que o entrevistador pode fazer: questões abertas e questões fechadas. As questões abertas requerem uma opinião (um discurso) por parte do entrevistado a fim de esclarecer a questão aplicada. As questões fechadas possuem um número pré-definido de respostas a partir das quais o informante deve fazer uma ou mais escolhas.

Dependendo do objetivo que se deseja atingir em um estudo, deve-se selecionar um ou mais métodos de coleta de dados. Por exemplo, etnografias geralmente usam entrevistas semiestruturadas, não estruturadas ou informais com questões abertas, visto que neste caso, se deseja explorar algum fenômeno pouco conhecido. Já na elaboração de questionários, geralmente são utilizadas entrevistas estruturadas com questões

fechadas, pois neste caso o objetivo é obter informações padronizadas sobre um domínio conhecido (DeWalt e DeWalt, 2002).

Além da seleção do tipo de entrevista mais adequado aos objetivos desejados, é necessário conhecer também as técnicas e procedimentos para se planejar e realizar boas entrevistas. Weiss (1994) apresenta boas práticas acerca da condução de entrevistas, explicando e exemplificando as fases de planejamento, transcrição e análise das mesmas. DeWalt (2002) também descreve técnicas que auxiliam na condução de entrevistas, como por exemplo: evitar perguntas tendenciosas, nas quais pode se sugerir uma resposta em específico; fornecer feedback para o entrevistado de forma que ele possa confirmar o entendimento da resposta; dentre outros.

Os conceitos e práticas descritos nas seções anteriores serviram de base para a realização da coleta de dados deste trabalho. O tipo de entrevista utilizado na pesquisa foi semiestruturada com questões abertas, visto que o objetivo da pesquisa era explorar fenômenos pouco conhecidos acerca de um determinado tema.

3.2 A Teoria fundamentada em Dados

O método selecionado para auxiliar a realização da análise qualitativa foi a Teoria Fundamentada em Dados (do inglês, *Grounded Theory* - GT) (Glaser e Strauss, 1967). Glaser e Strauss (1967) são os precursores da teoria fundamentada em dados, sendo esta uma obra clássica e pioneira na área. A teoria fundamentada em dados tem por objetivo gerar, validar ou elaborar uma teoria por meio de coleta e análise realizadas de forma sistemática sobre um determinado fenômeno social a ser estudado (Bandeira-de-Mello e Cunha, 2006). A GT é uma teoria derivada de dados, onde os mesmos são sistematicamente resumidos e analisados utilizando um processo de pesquisa analítico (Strauss e Corbin, 2008).

É necessário relatar que na GT existe uma estrita relação entre coleta de dados, análise e criação da teoria. Muitas vezes não existe uma teoria preconcebida pelo pesquisador na fase inicial do trabalho (Strauss e Corbin, 2008). Segundo Sandelowski (2002) e Patton (1990), mesmo quando a teoria é baseada em dados e possui conceitos bem descritos e fundamentados, é necessário que o pesquisador possua algumas habilidades, a saber:

- Tenha criatividade;
- Fique aberto a possibilidades múltiplas, explorando várias possibilidades antes de escolher uma; e
- Finalmente, utilizar formas usuais na descrição de seus pensamentos e análises dos dados para que outros pesquisadores consigam evoluir ou identificar novas perspectivas para a teoria gerada.

Estas habilidades são de suma importância para que a pesquisa tenha um resultado completo, uma vez que o entrevistador deve compreender a melhor forma de explorar e documentar as informações extraídas nas entrevistas. Neste estudo foram realizadas discussões sobre os pontos de coletas mais importantes, avaliando suas possibilidades e interpretando os dados coletados. Estas atividades guiaram a pesquisa, bem como os ciclos da coleta e análise de dados propostos pela teoria fundamentada em dados.

No processo de pesquisa, existe também a necessidade de realizar, de forma intercalada, coleta e análise dos dados, tal que as hipóteses são formuladas, avaliadas e delimitadas através da análise dos dados, e existe uma reformulação das questões e da próxima etapa de coleta de dados de acordo com os resultados desta análise de dados. Esta forma cíclica implica em que o pesquisador, após a definição das questões iniciais de pesquisa, encontre possíveis aspectos que merecem ser investigados em mais detalhes na próxima etapa de coleta de dados. Também é necessário que o pesquisador faça a reformulação de seu guia de pesquisa, o que possibilitará a obtenção de esclarecimentos sobre o fenômeno estudado, se necessário, indicando a necessidade de mais uma iteração no ciclo de pesquisa. Segundo (Strauss e Corbin, 2008) um estudo utilizando a teoria fundamentada em dados deve ser divido em cinco etapas, sendo algumas etapas realizadas várias vezes, de forma cíclica. Cada uma delas é descrita nas próximas subseções.

3.2.1 Definição do escopo de pesquisa

Na primeira etapa do estudo o pesquisador deve definir claramente o escopo da pesquisa, identificando quais são as questões pertinentes ao estudo, em qual contexto o mesmo pode ser aplicado, qual o fenômeno deve ser estudado, entre outros. Nesta etapa deve-se escolher um problema a ser estudado, identificar a questão de pesquisa e

descrever esta questão sucintamente. É necessário também avaliar objetivamente se a realização do estudo é viável, possível de ser pesquisada e se uma abordagem qualitativa deverá e poderá ser aplicada. Como uma forma de complementar esta análise, o pesquisador deve procurar fontes de pesquisa na literatura técnica e não técnica auxiliando a esclarecer possíveis dúvidas sobre o assunto.

A definição da questão de pesquisa é de suma importância, pois é ela que determina as principais características da pesquisa. Através da questão de pesquisa é possível decidir o método de pesquisa mais viável a ser utilizado, qual será o foco do estudo, em qual contexto ele poderá ser aplicado, além de evidenciar até onde a pesquisa será realizada, ou seja, que parte do problema o pesquisador irá investigar mais profundamente. As respostas destas perguntas possibilitam diminuir o tamanho do problema para algo viável de ser explorado em um determinado intervalo de tempo.

A questão de pesquisa deste trabalho foi apresentada na seção 1.2.

3.2.2 Definição do método de coleta de dados

Após a definição do escopo do estudo, é necessário determinar qual o método de coleta de dados a ser utilizado no mesmo. Nesta etapa deve-se analisar qual método de pesquisa é mais coerente para a pesquisa a ser conduzida, verificando a viabilidade de aplicação da coleta de dados e a abrangência da mesma. Como mencionado na seção 3.1, neste trabalho foi utilizado como método de coleta de dados a entrevista semiestruturada com questões abertas. Juntas, as atividades de definição da questão de pesquisa e do método de coleta de dados formam a primeira etapa da teoria fundamentada em dados (Bandeira-de-Mello e Cunha, 2006).

3.2.3 Coleta dos dados

Para iniciar a etapa de coleta de dados o pesquisador deve ter como base as etapas anteriores de definição da questão de pesquisa e do método de coleta dos dados. A partir destas definições o pesquisador terá a base necessária para iniciar sua pesquisa, sabendo qual o método será utilizado, como será utilizado e qual será o escopo da pesquisa. Com essas características da pesquisa em mente, o pesquisador pode iniciar o planejamento da efetiva coleta de dados.

Neste caso, o planejamento será descrito baseado no método de coleta definido para este estudo, as entrevistas semiestruturadas. O pesquisador inicia o planejamento das entrevistas elaborando seu *guia de entrevistas*, este guia contém um conjunto de perguntas iniciais que auxiliam o entrevistador a conduzir as entrevistas e dar início aos principais assuntos que desejam ser aprofundados. Após a criação do guia de entrevista, o pesquisador deve escolher como as entrevistas serão registradas. Por exemplo, normalmente as entrevistas são gravadas com vídeo e áudio ou somente áudio. Após a realização das entrevistas o pesquisador deve transcrever as gravações para uma forma textual, o que dará início ao processo de codificação. Também podem ser consideradas anotações feitas durante as entrevistas.

3.2.4 Início da análise e Codificação

Segundo Strauss e Corbin (2008) o início da análise dos dados é realizado com a atividade de análise através de exame microscópico dos dados, também chamada de análise "linha a linha". É nela que o pesquisador inicia sua ideia de categorização, ou seja, criação de seus possíveis códigos, sendo o ponto inicial do processo analítico. Nesta etapa devem ser realizados os procedimentos da análise crítica dos dados, considerando todas as implicações e desdobramentos de suas categorias.

O pesquisador deve evitar "tomar partido" ou "assumir uma posição" perante os dados, uma vez que estes procedimentos devem permitir a extração da informação evitando o viés na informação. Mesmo que a informação extraída dos dados seja de conhecimento do pesquisador, o mesmo deve questionar o informante sobre o fato, de forma que seja possível compreender a informação com a mesma perspectiva do entrevistado. Esta prática evita que o pesquisador influencie, com seu conhecimento, as respostas do informante.

Segundo (Strauss e Corbin, 2008) o pesquisador que realiza a análise microscópica dos dados necessita, além de foco nos procedimentos, de outros fatores, tais como:

I. Realizar análises definindo e classificando os fatos/fenômenos encontrados em categorias. Nesta análise é importante realizar comparações entre os dados, para evidenciar a variação dos mesmos, categorizando todos seus aspectos. Junto com a criação e classificação das categorias emergem seus

relacionamentos, indicando as similaridades ou diferenças entre os aspectos estudados. Os conceitos e classificações dão origem às categorias, que irão se desenvolver conforme a variação das propriedades e dimensões dos dados relacionados a elas, parte esta fundamental para a construção da teoria;

II. Examinar quais as suposições sobre determinados dados o pesquisador deve aceitar como corretas. Também se devem realizar comparações entre as suposições geradas pela pesquisa, confrontando umas com as outras, esclarecendo também as suposições em entrevistas, para confirmar a veracidade das suposições.

Juntamente com a execução da atividade de análise microscópica dos dados é necessário realizar a atividade de ordenamento conceitual. O ordenamento conceitual é a atividade responsável por organizar e classificar os dados em categorias discretas (Strauss e Corbin, 2008) baseado em suas propriedades e dimensões. As propriedades são características gerais e especificas do código gerado, ou seja, são elas que definem e dão significado as categorias. Já as dimensões se referem à variação das propriedades, melhor dizendo, são uma medida escalar e variável que possibilita medir determinado fator, por exemplo, a frequência de um fato.

Este tipo de organização permite ao pesquisador o entendimento do fenômeno a ser estudado conforme um esquema classificatório definido, ou seja, ao surgir os itens (dados da pesquisa) o pesquisador classifica e categoriza os mesmos conforme seus atributos. Com isto, ele pode realizar agregações ou comparações entre os itens, uma vez já classificados. Nesta etapa, o pesquisador se baseia nos atributos que cada item possui e em suas descrições previamente realizadas para criação de suas categorias. Nesta etapa os dados são ordenados segundo passos ou estágios já conhecidos pelo pesquisador, porém há uma necessidade de explicar como, quando, onde, por que, estes aspectos ocorrem (Strauss e Corbin, 2008). A partir da realização das atividades de análise microscópica dos dados e do ordenamento conceitual o pesquisador já possui insumos suficientes para iniciar a etapa de codificação. A partir da análise linha a linha e do ordenamento conceitual a etapa de codificação é iniciada.

A codificação é um procedimento dinâmico que auxilia a manter um rigor no processo de análise dos dados. Porém, ele é essencialmente mutável, isto é, não há uma

necessidade de seguir este processo de forma dogmática, e sim, usá-lo de forma criativa e flexível para possibilitar ao pesquisador usar sua criatividade conforme julgar apropriado.

No processo de codificação os dados são divididos, conceituados e suas relações são estabelecidas (Strauss e Corbin, 2008). É importante notar que a teoria fundamentada em dados tem um processo iterativo, onde são realizadas intercaladamente: coletas e análise dos dados e aprimoramento da teoria. O processo de coleta e análise de dados é realizado repetidas vezes com o intuito de refinar e aprimorar a teoria proposta. Segundo Glaser (1978) o processo de codificação é divido em três subetapas: codificação aberta, codificação axial e codificação seletiva.

3.2.4.1 Codificação Aberta

A codificação aberta é a primeira subetapa da fase de codificação. Esta subetapa têm como propósito dar um nome, ou código[1], a um fenômeno de interesse do pesquisador, como forma de abstrair um evento, ação ou interpretação de um significado. Cada código gerado pode ser um (ou parte de um) conceito, sendo uma forma de rotular um fenômeno ocorrido. O objetivo da criação de códigos é nomear um fenômeno para que o pesquisador possa agrupar fenômenos similares sob um tópico abstrato (código) lhe permitindo também inferir características e/ou significados sobre estes fatos.

A codificação aberta é parte do ato de conceituar, pois é nela que o pesquisador divide os dados, agrupando os que possuem características similares, dando um novo nome, um novo código, comum a todos os fatos similares. A codificação aberta também possui como objetivo identificar propriedades e dimensões de cada código gerado. Na codificação aberta também são criadas subcategorias para os códigos. Esta é uma forma de agrupar as propriedades de determinado fenômeno que possuam uma variação específica. As subcategorias são utilizadas para representar uma maior especialização das categorias (códigos) criadas.

[1] Um código é uma representação abstrata de um determinado fator ou característica. Normalmente, um código é formado por uma palavra, ou um conjunto delas, responsável por identificar, ressaltar ou categorizar determinada característica notada pelo estudo. Um código pode representar uma subcategoria do estudo, enquanto um conjunto de códigos, ou abstração de um conjunto, representa uma categoria.

A tarefa de codificação é realizada com base na análise microscópica dos dados. Nesta tarefa o pesquisador deve fazer uma análise detalhada linha a linha dos dados transcritos. Esta análise auxilia o pesquisador a criar rapidamente suas categorias, evidenciando as propriedade e dimensões de cada uma delas. Neste tipo de análise o pesquisador deve tentar responder perguntas do tipo "o que esta acontecendo neste momento?", "porque está ocorrendo isso?", "o que levou este fato a acontecer?" As respostas destas perguntas o auxiliam a criar seus memorando[2] (memos), que são complementos importantes para a análise.

Em resumo, a etapa de codificação aberta dos dados visa permitir a definição de categorias (códigos) e subcategorias relativas ao estudo, identificando a ocorrência dessas categorias e subcategorias, através da análise microscópica dos dados, do exame linha a linha dos dados coletados e sobre as anotações de memorandos que emergiram do processo de coleta e análise dos dados (Triviños, 1987).

3.2.4.2 Codificação Axial

A codificação axial é a segunda subetapa da codificação e tem como dado de entrada a codificação aberta. É nela que é realizado o processo de relacionar e agrupar as categorias e subcategorias criadas na subetapa anterior e desenvolver as categorias geradas. Nesta subetapa é realizada a análise e a implicação deste agrupamento, levando em consideração as propriedades e dimensões de cada categoria. A codificação axial é responsável pelo reagrupamento das categorias que foram dividas anteriormente, uma vez que é necessário saber como as mesmas se relacionam. Este reagrupamento gera explicações mais precisas e completas sobre os fenômenos, seus relacionamentos, semelhanças e diferenças.

---

[2] Segundo Strauss e Corbin (2008) memorandos são registros escritos durante a análise que podem variar em tipo e formato. Os memorandos são compostos pela data em que foram escritos, os códigos, documentos ou número de linhas a qual estão relacionados, uma identificação na qual seja possível compreender qual o assunto tratado por ele e o registro da análise realizada. Estes memorandos são extremamente úteis nas fases de codificação, pois servem como guia das categorias no estudo. Por exemplo, eles possibilitam a identificação de relacionamento entre códigos, a descrição de ideias ou dos motivos para atribuição de determinado código, dentre outros benefícios.

Vale ressaltar que não é apenas com a leitura do texto que os relacionamentos emergem. O pesquisador também deve levar em consideração a abstração dos fatos, o conhecimento sobre o assunto durante as coletas de dados e principalmente as propriedades e dimensões dos códigos gerados. Segundo Strauss (1987), a codificação axial envolve algumas tarefas, que são: (I) Organizar as propriedades de uma categoria e suas dimensões; (II) Identificar a variedade de condições, ações ou consequências associadas a um fenômeno; (III) Relacionar uma categoria à sua subcategoria por meio de declarações que denotem como elas se relacionam uma com as outras; e (IV) Procurar nos dados pistas que denotem como as principais categorias podem estar relacionadas umas às outras.

Outro fator importante é que o pesquisador que realiza a codificação axial deve obter explicações e entender como o fenômeno se relaciona e não apenas suas consequências e condições. Como os fenômenos possuem relacionamentos complexos, definir apenas relações de causa e efeito não é o suficiente para denotar seus relacionamentos (Strauss, 1987). Para obter detalhamento e explicações completas sobre os relacionamentos entre as categorias, ou entre categorias e subcategorias, é necessário analisar o contexto no qual o fenômeno se aplica, normalmente descrito nos memos do pesquisador. É possível também que o pesquisador utilize mini estruturas de análise e diagramas teóricos para guiar a criação dos relacionamentos entre os códigos.

Em resumo, a subetapa de codificação axial é responsável por desenvolver as categorias e reagrupar as categorias através da criação de relacionamentos entre elas. Estes relacionamentos permitem ao pesquisador entender como os fenômenos estão ligados entre si.

3.2.4.3 Codificação Seletiva

A codificação seletiva é a terceira (e última) subetapa da codificação e tem como entrada os resultados da codificação aberta e da codificação axial. Ela é responsável por utilizar as categorias e seus relacionamentos para formar um esquema teórico com os resultados da pesquisa em forma de teoria refinando e integrando as categorias geradas. Nesta etapa os dados, categorias e seus relacionamentos são transformados em teoria.

O primeiro passo da codificação seletiva é a identificação da categoria central da pesquisa. Para definir a categoria central Strauss e Corbin (2008) determinam seis critérios:

- A categoria deve ser central, ou seja, todas as outras categorias importantes podem ser relacionadas a ela;
- Deve aparecer frequentemente nos dados, isso significa que em todos os casos, ou quase todos, há indicadores apontando para este conceito;
- À medida que o conceito é refinado analiticamente por meio de integração com outros conceitos, a teoria ganha mais profundidade e mais poder explanatório; e
- Finalmente, o conceito consegue explicar variações e também o ponto principal dos dados, ou seja, quando as condições variam, a explicação ainda é valida, embora a forma na qual um fenômeno seja expresso possa parecer um pouco diferente. O pesquisador deve ser capaz de explicar casos contraditórios ou alternativos em termos dessa ideia central.

Com a categoria central definida, o pesquisador inicia o processo de integração dos conceitos e a geração da teoria. Para auxiliar a integração, o pesquisador tem várias técnicas à disposição, uma delas é a organização de memorandos, que foi utilizada nesse trabalho. Segundo Strauss e Corbin (2008) os memorandos são como armazéns de ideias. Eles contêm os indícios para a integração e as propriedades de cada conceito ao longo de suas dimensões. Nesta etapa os memorandos devem ser organizados e classificados através de suas categorias. Para realizar esta organização em categorias o pesquisador deve dar nomes que descrevam os conceitos para os memorandos baseados nas propriedades e dimensões de cada categoria descrita nos memorandos. Após esta nomeação, o pesquisador poderá fazer comparações entre os memorandos possibilitando identificar suas características, semelhanças e diferenças, facilitando a organização dos mesmos. Esta categorização permite aprimorar o esquema teórico dominante, com isso refinando a teoria também. Na etapa posterior o pesquisador verifica a consistência interna e falhas na lógica, complementa as categorias mal desenvolvidas, elimina excessos e valida o esquema teórico.

A identificação da categoria central, bem como o processo de integração dos conceitos, é realizada no decorrer da pesquisa, onde existe uma interação cíclica entre coleta e análise dos dados. Esta identificação é iniciada após as primeiras interações da pesquisa e apenas é concluída na redação final da pesquisa. Uma vez que o esquema teórico seja identificado, o pesquisador está apto a refinar a teoria, eliminando excessos e complementando algumas categorias mal descritas utilizando a amostragem teórica adicional.

Em resumo, a etapa de codificação seletiva realiza o refinamento dos códigos. Nessa etapa, o processo de codificação é integrado e refinado provendo a questão central, a qual todos os códigos estão relacionados. Esta questão central sumariza e relaciona todos os códigos com o fenômeno, indicando suas implicações, causas, efeitos e suas características, gerando um esquema teórico com os resultados da pesquisa em forma de teoria.

As etapas de codificação compreendem a segunda parte da utilização da teoria fundamentada em dados. Entretanto, existe um outro aspecto importante na utilização desta teoria, a amostragem. Este aspecto será descrito na próxima seção.

3.2.5 Amostragem

Após a etapa de codificação ser realizada por completo, o pesquisador pode realizar a etapa de amostragem teórica para identificar quais as possíveis fontes de informações para esclarecer ou confirmar determinada categorização feita nas etapas anteriores. A amostragem teórica também é responsável por orientar as novas etapas de coletas de dados, com a reformulação de questões analíticas a partir da realização de comparações entre as informações já obtidas, sendo aconselhável a realização de testes pilotos após a criação do guia. Na amostragem teórica é definido o local ou grupo que irá participar da próxima coleta de dados; quais os tipos de dados a serem utilizados na próxima coleta; o número de locais a serem estudados nesta interação, o número de pessoas a serem estudadas, os recursos disponíveis, as metas de pesquisa a serem respondidas, dentre outros aspectos. Em resumo, a amostragem teórica visa identificar novas fontes de informações que possuam comprovações, ou refutações, sobre a completude da teoria.

É na amostragem teórica que os guias de entrevistas ou guias observacionais são criados ou reformulados para obter uma maior fluidez no processo de coleta de dados. Esta etapa é de suma importância, pois avalia se o pesquisador está gerando categorias corretas sobre o fenômeno estudado.

Em resumo pode-se dizer que após a realização dos ciclos de coleta e análise de dados, é executada a etapa de amostragem teórica, onde são realizadas comparações entre a base de dados já codificada e os resultados de novas coletas de dados, para avaliar a consistência da teoria a ser gerada. Nesta etapa são analisados os pontos a serem melhor explorados nas próximas coletas, auxiliando o pesquisador a reformular o guia de coleta para obter confirmações e esclarecimentos sobre o estudo.

3.2.6 Redação da teoria

Segundo Strauss e Corbin (2008), antes de realizar a última etapa do estudo, o pesquisador deve ter em mente que a atividade de teorização visa desenvolver uma teoria através da concepção, indução e formulação de ideias (conceitos) em um esquema lógico sistemático e explanatório, contendo conceitos e suas relações sobre determinado aspecto. Ainda segundo Strauss e Corbin (2008), quando se é explorada completamente uma ideia, levando em consideração diferentes perspectivas e analisando suas implicações, é possível transformar essa ideia em teoria. A geração dessa última é realizada através da transformação sistemática do resultado de várias análises sobre um determinado aspecto a partir dos dados existentes.

A elaboração da redação da teoria e auditagem do processo de criação consistem na última etapa da teoria fundamentada em dados. Nesta etapa, o pesquisador analisa novamente todas suas anotações, dados e informações de pesquisa para verificar se a teoria, gerada pelo processo iterativo de pesquisa, está concisa, válida e descrita. Após esta análise, o pesquisador pode registrar a redação contendo a teoria e a pesquisa realizada para alcançá-la.

4 A UTILIZAÇÃO DE MÉTODOS DA TEORIA FUNDAMENTADA EM DADOS NO ESTUDO SOBRE MÉTODOS ÁGEIS

O objetivo deste capítulo é descrever como as técnicas da teoria fundamentada em dados foram utilizadas neste trabalho, apresentando como as coletas de dados foram realizadas, descrevendo o ambiente físico e os projetos realizados pelas empresas estudadas, a codificação e análise dos dados e por fim ilustrando os mapas de códigos da GT produzidos.

4.1 Visão Geral

O início da pesquisa descrita nesta dissertação se deu com a realização de uma revisão da literatura sobre metodologias ágeis, onde foram identificados potenciais ramos de pesquisa relacionados ao tema. Um desses ramos está relacionado às mudanças causadas nas atividades da engenharia de software após a aplicação das práticas ágeis em empresas brasileiras, principalmente em relação ao aumento de atividades realizadas de maneira colaborativa. Este aspecto foi notado durante a revisão da literatura, pois várias atividades que normalmente são realizadas individualmente, quando descritas nas metodologias ágeis passam a ser recomendadas a serem feitas de maneira colaborativa. Visando aprofundar o conhecimento acerca desse contexto o tema foi selecionado como foco e objetivo de pesquisa deste trabalho.

Com a delimitação do objetivo de pesquisa foi realizado um estudo para verificar qual metodologia seria mais adequada para realizar a formulação de uma teoria acerca do tema proposto. Feito isso, foi identificado que uma metodologia qualitativa seria mais apropriada para conduzir a pesquisa, haja vista a necessidade de se entender um fenômeno específico em profundidade, além do caráter exploratório do estudo. Conforme descrito anteriormente, o método selecionado para auxiliar a realização da análise qualitativa foi a Teoria Fundamentada em Dados (Glaser e Strauss, 1967). Esse método foi selecionado, pois possui uma análise rigorosa e fiel dos dados coletados, além de se mostrar viável para aplicação no cenário de pesquisa do estudo e seus resultados inferem uma teoria.

Com o objetivo e escopo de pesquisa definidos iniciou-se então o processo de aplicação de métodos de codificação da teoria fundamenta em dados (GT). As atividades definidas para a etapa inicial o de planejamento foram: (1) Definir qual o método de coleta de dados é mais apropriado; (2) Definir quais as características das empresas a serem estudadas; (3) Definir quais os papéis desempenhados pelas pessoas a serem entrevistadas; e (4) Definir o guia de questões. Cada um destes aspectos é discutido a seguir.

A primeira tarefa executada foi a de definir o método de coleta dos dados. Os critérios utilizados para definição do método mais adequado foram: a viabilidade de aplicação na coleta; adequação ao tipo de pesquisa; e abrangência da técnica de coleta para o estudo. Levaram-se em consideração também as seguintes necessidades: compreender um fenômeno social do ponto de vista do respondente, permitindo a ele, externalizar seu ponto de vista sob o aspecto abordado; utilização de um método flexível que admita a adaptação e exploração dos aspectos pesquisados, permitindo obter um maior esclarecimento sobre o que está sendo abordado. Com base nos critérios descritos acima e nas necessidades de pesquisa, o método considerado mais adequado a esta pesquisa foi a entrevista semiestruturada com questões abertas (DeWalt e DeWalt, 2002).

A segunda tarefa executada foi a definição do perfil das empresas que seriam estudadas. Nesta fase foram identificadas as características básicas para seleção das empresas e quais seriam, efetivamente, as possíveis empresas a serem estudadas. Com

base nos estudos iniciais, foram definidos os seguintes critérios para definir as empresas a serem estudadas: elas deveriam utilizar métodos, metodologias ou práticas ágeis na produção de seus softwares; possuir pelo menos um projeto na fase de desenvolvimento; ter pelo menos um projeto concluído utilizando métodos, metodologias ou práticas ágeis; ter implantado métodos, metodologias ou práticas ágeis por pelo menos seis meses; e, finalmente, possuir no mínimo 10 colaboradores. Como neste estudo havia a necessidade de compreender como as empresas aplicam metodologias ágeis no Brasil, os critérios definidos buscaram abranger o maior número de empresas possíveis.

A terceira tarefa executada foi a definição dos papéis a serem entrevistados. Os seguintes critérios foram definidos para a seleção dos entrevistados:

- Exercer, pelo menos um, dos seguintes papéis: desenvolvedor, engenheiro de software, analistas de testes, analistas de requisitos, líder de equipe, arquiteto, *Scrum master*, P.O. ou gerente;
- Possuir, no mínimo, um ano de experiência em desenvolvimento de software; e
- Finalmente, possuir, pelo menos, seis meses de experiência utilizando praticas ágeis.

A quarta tarefa executada foi a definição do guia de entrevistas. Nesta fase foram definidas as perguntas que iriam compor o guia de entrevistas semiestruturadas. Nesta fase, foram definidos os seguintes aspectos gerais a serem abordados pela pesquisa: implantação e execução de métodos, metodologias ou práticas ágeis; requisitos com práticas ágeis; refatoração com práticas ágeis; vantagens e desvantagens desta adoção; e dificuldades da utilização de práticas ou métodos/metodologias ágeis.

Nesta etapa da pesquisa se teve uma grande atenção em manter o guia o mais abrangente possível com perguntas sobre várias áreas da aplicação das metodologias ágeis, como análise e projeto, requisitos, cerimônias das metodologias, o impacto de implantação, dentre outros. No decorrer do estudo, com uma melhor compreensão sobre os dados codificados foi possível identificar os assuntos mais relevantes, podendo assim aprofundar o conhecimento sobre esses assuntos com alterações no guia de entrevista.

Para auxiliar na reconstrução do guia de entrevista foi utilizada a técnica 5W2H [(Bamford e Greatbanks, 2005) apud (Nadae, Oliverira e Oliveira, 2009)] (do inglês: *What, Who, When, Where, Why, How Much/Many, How*). Segundo Bamford e Greatbanks (2005) apud Nadae, Oliverira e Oliveira (2009) a utilização desta técnica auxilia o pesquisador a obter grande detalhamento sobre o fenômeno investigado, uma vez que se questiona sobre diversos aspectos.

Ainda segundo Nadae, Oliveira e Oliveira (2009) apud Bamford e Greatbanks, (2005) a técnica do 5W2H sugere que cada assunto estudado seja avaliado a partir de sete perguntas, que se iniciam pelos 5W. A primeira pergunta refere-se ao que deve ser feito em determinada tarefa (*What*). A segunda diz respeito a quem são/serão os responsáveis por executar determinada tarefa (*Who*). A terceira responde a quando uma tarefa foi ou deve ser executada (*When*), ou seja, qual é o prazo para execução de determina tarefa. O quarto questionamento tem seu foco em onde a tarefa foi ou vai ser executada (*Where*). Já o quinto questionamento foca no porquê de executar determinada tarefa, quais serão seus benefícios e motivações para sua execução. Por fim os 2H,que se referem a qual será o custo para execução da tarefa (*How Much*), onde este custo pode ser em valores monetários, tempo gasto para a execução da tarefa ou frequência. Por fim, o sétimo questionamento (*How*) diz respeito a como a tarefa foi ou será executada, contendo os detalhamentos de sua execução.

Alguns dos aspectos da técnica não foram aplicados em todas as questões, pois estavam fora de contexto em relação aos interesses da pesquisa aqui descrita (por exemplo, o custo das atividades). No entanto, a utilização da mesma possibilitou uma melhor organização e exploração dos aspectos estudados, sendo utilizada a cada nova versão do guia de entrevista.

A primeira versão do roteiro de entrevistas (ver APÊNDICE A – guia de entrevista) proposto para este estudo foi composto por 43 questões abertas que serviram como base inicial para as entrevistas. Com o roteiro em mãos, foram realizados testes piloto (ver APÊNDICE C – DESCRIÇÃO DOS TESTES PILOTOS) para analisar a clareza das questões, aderência e viabilidade do guia. No entanto, em alguns casos foi necessária a adição de outras questões, com o intuito de obter maiores esclarecimentos sobre um determinado assunto, gerando assim uma segunda versão do guia. Com isso foi possível

iniciar o processo de coleta de dados. As etapas de coleta e análise de dados foram divididas em pré-refatoração, refatoração e pós-refatoração para facilitar a apresentação dos resultados. Isto é, esta é uma divisão analítica conceitual usada apenas como forma de organizar as idéias, é uma forma de descrever e apresentar as questões aos entrevistados. Entretanto, deve-se observar que nenhum deles tinha o conhecimento desta divisão conceitual ou mesmo sobre a forma que o guia e os resultados iniciais estavam estruturados. Isto é necessário para evitar influenciar o entrevistado em suas respostas, minimizando o viés da pesquisa e as opiniões do entrevistador sobre o assunto abordado.

As próximas subseções descreveram como foi realizada cada uma das etapas de coleta e análise de dados, bem como a aplicação dos métodos de codificação da GT.

4.2 Execução das coletas de dados.

Neste trabalho foram estudadas cinco empresas produtoras de software, dividas em quatro etapas de coleta de dados. Estas empresas estão situadas nos estados do Pará (PA), Pernambuco (PE) e São Paulo (SP). Inicialmente, os informantes foram selecionados segundo uma amostragem representativa de cada equipe das empresas estudadas. Entretanto, em algumas empresas as entrevistas não puderam ser realizadas com esta amostra pré-definida, uma vez que, por diversos fatores, os entrevistados de interesse não estavam disponíveis para entrevista. A fim de contornar este problema, bem como a dificuldade de acesso as empresas, foi definida uma nova amostra baseada na disponibilidade dos entrevistados.

Vale ressaltar que as empresas deste estudo se intitulam ágeis, ou afirmam que utilizam metodologias ágeis em seu processo, não cabendo, nesta pesquisa, realizar avaliações, confirmações ou investigações sobre a efetiva utilização de metodologias ágeis por partes destas empresas. Em outras palavras, o objetivo desta pesquisa não era "julgar" as empresas estudadas. Muito pelo contrário, um dos objetivos desta pesquisa é justamente compreender como, as empresas que se intitulam ágeis, realizam suas atividades e quais os benefícios, dificuldades e impactos que a utilização de tais práticas ágeis têm no processo de desenvolvimento de software.

Por questões de sigilo os nomes das empresas serão preservados e identificados apenas como empresa I, empresa II, empresa III, empresa IV e empresa V e os informantes como Informante I, Informante II, e assim por diante. Cada empresa possui o número do informante iniciado em I, para melhor descrição. Por questões de sigilos algumas informações foram retiradas, como por exemplo, nome de ferramentas ou informações que foram explicitamente solicitadas como sigilosas. A caracterização das empresas será realizada de forma semelhante à descrita no trabalho de Sharp e Robinson (2004), incluindo as características físicas das empresas para melhor caracterização do ambiente de trabalho.

Todas as empresas utilizam a metodologia ágil *Scrum* (Schwaber e Beedle, 2001) como base de seu processo de desenvolvimento, incorporando também práticas de outras metodologias ágeis como: XP (Extreme Programming) (Beck, 2000) e FDD (Feature Driven Development) (Coad, Luca e Lefebvre, 1999). Elas utilizam também gráficos como o *burndown chart* (Schwaber, 2004) e *Gantt Chart* (PMI, 2003), dentre outras técnicas, que auxiliam no controle de suas atividades.

4.2.1 A Primeira coleta de dados.

Na primeira etapa da coleta de dados foram realizadas oito entrevistas na empresa I, entre os dias 25/11/2009 a 18/12/2009. Foram entrevistados quatro desenvolvedores, dois *Scrum master*, sendo um deles arquiteto dos projetos e dois analistas de teste/designer. Os colaboradores possuíam as seguintes características: pouca experiência profissional, jornada de trabalho de quatro horas diárias, conhecimento mediano sobre a linguagem de programação utilizada e pouco conhecimento sobre metodologias ágeis na fase inicial do projeto. Todas as entrevistas tiveram o áudio gravado e foram posteriormente transcritas para análise.

A empresa I possui um ambiente físico composto por três ambientes. O primeiro é uma sala de reunião grande com uma mesa central, um quadro branco e ao canto desta sala tem um forno micro-ondas, um espaço para alimento, com água e café. O segundo ambiente é um grande sala, com quatro grandes bancadas, onde ficam os colaboradores. As bancadas são divididas por equipes, normalmente compostas por seis colaboradores, contendo um espaço livre para cada colaborador, com material de escritório e um

computador. À frente de cada bancada existe um quadro KANBAN (Ikonen, Kettunen, *et al.*, 2010) e o gráfico de *burndown* (Schwaber, 2004) da *sprint*. Neste ambiente também existem um espaço com água e café, e uma mesa para reuniões rápidas. O terceiro espaço é utilizado para algumas práticas ágeis, como as reuniões diárias (*Daily Meeting*), Dojo de programação, entre outras atividades e possui um grande quadro branco.

Esta organização utiliza as metodologias ágeis há mais de um ano e meio, e possui em média 20 colaboradores. Existe na empresa um alto índice de rotatividade dos colaboradores, visto que ela é uma fábrica de software instalada em uma universidade, utilizando os próprios alunos como colaboradores.

Os desenvolvedores estão divididos em três equipes de desenvolvimento, onde cada equipe está alocada a um projeto. Esta organização tem foco no desenvolvimento de sistemas de pequeno e médio porte por encomenda, atuando em diversos seguimentos (saúde, serviços, educação, dentre outros). Os sistemas são desenvolvidos para as plataformas web e desktop utilizando as linguagens de programação Java e .NET, com o seguinte parque tecnológico: controle de versão (SVN) e protótipos (ferramenta proprietária Y, cujo nome não pode ser mencionado aqui).

4.2.2 Segunda coleta de dados.

Na segunda etapa da coleta de dados foram realizadas quatorze entrevistas, em duas empresas, entre os dias 04/01/2010 a 15/01/2010. Foram entrevistados três desenvolvedores, dois analistas de requisitos, um analista de testes, um *Scrum master* que é também arquiteto de software e um P.O. na primeira empresa. Na segunda empresa foram entrevistados: três desenvolvedores, um analista de requisitos e também P.O., um *Scrum master* e um analista de testes.

Os colaboradores das duas empresas possuíam as seguintes características: experiência profissional média, jornada de trabalho de 40 horas semanais, alto conhecimento sobre a linguagem de programação utilizada, sendo que em uma das empresas os colaboradores tinham pouco conhecimento sobre metodologias ágeis na fase inicial do projeto, e na outra o nível de conhecimento era médio. Todas as

entrevistas tiveram o áudio gravado e foram transcritas para análise. A empresa II possui aproximadamente 35 funcionários.

A empresa II possui um ambiente físico composto por dois blocos, cada bloco com dois andares. No andar inferior do bloco A existe uma grande sala para treinamento e capacitação dos funcionários. No andar superior estão lotadas as equipes de suporte operacional e a equipe de manutenção dos sistemas produzidos, esta equipe não utiliza metodologias ágeis na manutenção do sistema, fator este que excluiu esta equipe do estudo. Na parte inferior do bloco B, existe uma copa e uma sala de descanso para os funcionários. Na parte superior do bloco existem duas salas, a primeira é uma sala de reuniões, normalmente utilizada para a realização de reuniões de retrospectiva e *Sprint Review* e a segunda onde está lotada a equipe de desenvolvimento que utiliza a metodologia ágil e se enquadrou nas características requeridas para o estudo.

A sala da equipe de desenvolvimento é divida em três partes, a primeira área é reservada a alimentação, onde estão disponíveis água e café e uma pequena mesa utilizada para conversas rápidas; a segunda área contem uma mesa de trabalho para quatro pessoas, onde ficam o gerente e os arquitetos de soluções da empresa; a terceira área é destinada aos desenvolvedores, analistas de teste e requisitos, coordenador das equipes/P.O./*Scrum master*. Nesta área existe, para cada equipe, um quadro KANBAN (Ikonen, Kettunen, *et al.*, 2010), um gráfico de *burndown* da Sprint (Schwaber, 2004), um espaço com as próximas tarefas a serem realizadas após o termino da *sprint* ou atividades de estabilização do sistema e um gráfico geral de andamento do projeto.

A empresa II atua na produção de sistemas para área da saúde. Ela possui um sistema legado produzido com a linguagem Delphi que se encontra na fase de manutenção e produz também novos módulos para esse sistema. Este sistema é cliente-servidor, como módulos na plataforma web e módulos para plataforma *desktop*. A equipe de desenvolvimento alvo deste estudo produz os novos módulos do sistema conforme a demanda dos clientes e produz a nova versão do sistema na plataforma JAVA utilizando somente para a plataforma WEB. A empresa possui 11 colaboradores na área de desenvolvimento de novos módulos e da nova versão do sistema e 12 colaboradores na área de suporte e manutenção do sistema legado. A empresa possui o seguinte parque tecnológico: controle de versão (CVS), geração de builds automatizadas

(HUDSON), gerenciamento de requisições (*issue tracker* – ferramenta proprietária Z, cujo nome não pode ser mencionado aqui por questões de confidencialidade).

A empresa III possui um ambiente físico composto por um bloco de dois andares dividido em várias salas. Na parte superior existe um refeitório, uma sala de descanso e uma sala de relaxamento com massagista. A parte inferior é dividida em salas, agrupando os colaboradores por área de atuação. Vale informar que neste estudo foram analisadas apenas as salas onde havia atuação da equipe de desenvolvimento de sistemas, sendo uma sala de desenvolvimento e uma sala de gerencia. Existe ainda uma sala destinada ao gerente de desenvolvimento, uma sala de reuniões e uma sala destinada à área de desenvolvimento. A sala de desenvolvimento é divida em três partes, uma para os desenvolvedores, outra para os analistas de requisitos e outra para cinco analistas de teste, um administrador de dados, um designer e um líder de equipe. Esta sala possui também, uma área com café e água, um espaço aberto onde é realizada a reunião diária, e uma mesa para reuniões rápidas. Esta sala é dividida em bancadas e a frente da bancada da equipe de desenvolvimento existe um quadro KANBAN (Ikonen, Kettunen, *et al.*, 2010). A equipe de desenvolvimento é composta por 25 pessoas, dividida em três equipes de desenvolvedores, uma equipe de requisitos e uma equipe de teste. A equipe de desenvolvimento é composta por dois analistas de requisitos, um analista de teste, cinco desenvolvedores e um *Scrum master*/P.O. (comum a todas as equipes). A empresa III possui o seguinte parque tecnológico: controle de versão (SVN), geração de builds automatizadas (HUDSON), integração contínua (HUDSON), protótipos (Ferramenta proprietária Y), gerenciamento de requisições (*issue tracker* – *Apache Jackrabbit*), gerenciamento de mudanças (Ferramenta proprietária W) e *suite* de testes e validações (Junit e Selenium).

4.2.3 Terceira coleta de dados.

Na terceira etapa da coleta de dados foram realizadas seis entrevistas, divididas da seguinte forma: duas entrevistas na empresa IV, três entrevistas na empresa I (sendo um novo entrevistado e duas entrevistas de esclarecimento) e uma entrevista de esclarecimento na empresa III. Estas entrevistas foram realizadas entre os dias 19/03/2010 a 30/03/2010. Nesta nova etapa de coleta foram entrevistados os seguintes papéis: quatro desenvolvedores, um *Scrum master* e um P.O. que também é gerente.

Os colaboradores da empresa IV possuem experiência profissional média e alta, jornada de trabalho de 40 horas semanais, médio e alto conhecimento sobre a linguagem de programação utilizada e conhecimento médio sobre metodologias ágeis. Todas as entrevistas tiveram o áudio gravado e foram transcritas para análise. É necessário informar que foi necessário realizar algumas entrevistas de esclarecimento devido à descoberta de novos fatores/fenômenos/códigos decorrentes de outras entrevistas, os quais foram aprofundados em algumas ocasiões.

A empresa IV é um órgão público do estado e possui diversos sistemas, tanto na fase de manutenção quanto na fase de desenvolvimento. Nesta empresa foi escolhido projeto de desenvolvimento de um sistema de gerenciamento de processos interno de grande porte. Este sistema controla tanto os processos deste órgão quanto de todas as suas filiadas dispostas em vários municípios deste estado. Este sistema está sendo desenvolvido por 26 colaboradores com a linguagem de programação Java para plataforma WEB. A empresa IV solicitou que as informações sobre o seu parque tecnológico não fossem relatadas.

O ambiente de trabalho é divido em três salas equipadas com quadro branco, bancada para os computadores e uma mesa de reunião. Próxima a estas salas existe uma sala de reuniões equipada com projetores, quadro branco e *flipchart*. Os colaboradores estão divididos em três grupos, um gerente, quatro administradores de dados, vinte e um analistas de sistemas, sendo um deles *Scrum master*, divididos em quarto subequipes. A equipe de analistas utiliza o quadro KANBAN (Ikonen, Kettunen, *et al.*, 2010) e o gráfico de *burndown* (Schwaber, 2004) para o gerenciamento de suas tarefas e da *sprint*.

4.2.4 Quarta coleta de dados.

Na quarta etapa da coleta de dados foram realizadas sete entrevistas, divididas da seguinte forma: duas entrevistas na empresa V, duas de esclarecimento com informantes da empresa II, duas de esclarecimento com informantes da empresa III e uma de esclarecimento com um informante da empresa IV. Também foi disponibilizado o guia de entrevista por e-mail para três membros da empresa V que não podiam realizar a entrevista por outro meio. Esta coleta ocorreu entre os dias 29/09/2010 a 07/02/2010. Os

entrevistados possuem os seguintes perfis: líder técnico, P.O., desenvolvedores, analistas de requisitos e *Scrum master*.

Os colaboradores da empresa V possuíam as seguintes características: experiência profissional entre média e alta, jornada de trabalho de 40 horas semanais, alto conhecimento sobre a linguagem de programação utilizada e alto conhecimento sobre metodologias ágeis. As entrevistas foram realizadas através de vídeo conferência e tiveram seu áudio transcrito para análise. Em relação aos guias enviados por email, os mesmos foram respondidos por escrito e sistematizados em forma similar as transcrições das entrevistas para facilitar a análise.

A empresa V é uma empresa privada do ramo de desenvolvimento de sistemas e hospedagem de sites. O foco do desenvolvimento de sistemas está ligado a um sistema de gestão empresarial para web e infraestrutura em *Cloud Computing*. A empresa tem mais de 10 anos de mercado e cinco anos de experiência com metodologias ágeis, possui por volta de 600 funcionários (100 a 150 funcionários em TI) e desenvolve com as linguagens Java e Ruby, para web e *cloud computing*. A equipe estudada é composta por 13 funcionários entre engenheiros de software, administradores de banco de dados e redes. O sistema desenvolvido pela equipe visa ser multiplataforma e possui integração com sistemas na plataforma Windows e Linux.

O ambiente físico da empresa V é composto por um prédio grande com vários andares. No andar mais alto fica o departamento de tecnologia, administração e a diretoria. O prédio lembra um grande galpão onde nas laterais existem salas de reuniões e no centro um espaço completamente aberto apenas com ilhas de mesas espalhadas, não há paredes neste local. As mesas da equipe de tecnologia são grandes com espaço para dois ou três desenvolvedores à frente de um mesmo computador e um espaço ao lado para conversas e pequenas discussões. Esta empresa possui o seguinte parque tecnológico: controle de versão (SVN), integração contínua (HUDSON), *suíte* de testes (Selenium e JUnit) e validações integradas com geração de builds automatizadas (HUDSON), gerenciamento de requisições (*issue tracker*) e mudanças e ferramentas de validação e análise de código e *coding style* (check style, find Bug, etc).

Na subseção 4.3 será apresentada a etapa de codificação das entrevistas bem como os códigos gerados e seus relacionamentos.

A Tabela 1 apresenta um resumo contendo as empresas estudadas, o número de entrevistas realizadas por empresa, o número de entrevistados por empresa e as fases de coleta que a empresa participou, incluindo questionários enviados por e-mail.

Tabela 1 - Entrevistas Realizadas por Etapa

Empresa	Nº de entrevistas	Nº de entrevistados	Etapa da Coleta
Empresa I	11	9	I e III
Empresa II	10	8	II e IV
Empresa III	9	6	II, III e IV
Empresa IV	3	2	III
Empresa V	5	5	IV

4.3 Codificação e análise das entrevistas

Após a etapa de coleta e transcrição dos dados foi iniciada a etapa de codificação das entrevistas. Para auxiliar nesta etapa foram utilizadas duas ferramentas para análise qualitativa: a ferramenta MaxQDA (1995) e a ferramenta Atlas.ti (2010). Esta etapa teve início com a atividade de codificação aberta dos dados, onde os dados transcritos foram importados para a ferramenta MaxQDA e codificados. Esses códigos foram revisados e ajustados quando necessário. Após a etapa de codificação aberta foi realizada a codificação axial e seletiva dos dados para obter as categorias centrais do estudo e seus relacionamentos. É importante ressaltar que *as atividades de codificação e análise dos dados ocorreram ao final de cada etapa de coleta*. A ferramenta Atlas.ti (2010) foi utilizada para geração dos gráficos dos códigos gerados neste estudo.

No decorrer do estudo foram geradas três versões principais do mapa de códigos para auxiliar na identificação de resultados para a pesquisa. Ao final de cada etapa de codificação dos dados e/ou reanálise das transcrições ou códigos, novas versões do mapa de códigos foram geradas. Cada nova versão representava um refinamento dos códigos encontrados na etapa anterior, bem como surgiam novos conceitos, grupos de códigos e relacionamentos. As três versões que serão apresentadas a seguir são as que melhor representam a evolução do estudo, do guia de entrevista e dos códigos e seus relacionamentos.

A primeira versão do mapa contem 18 códigos (vide APÊNDICE B – MAPAS DE CÓDIGOS DA GT) relacionados a diversas áreas da engenharia de software: análise de requisitos, projeto, codificação, testes e manutenção (Pressman, 2005), (Sommerville, 2006). Alguns dos resultados deste mapa são: Metodologias/Práticas ágeis, características da refatoração, pontos fortes e oportunidades de melhoria em diversas áreas da engenharia de software, etc.

A segunda versão do mapa aprimora os códigos da primeira versão e adiciona novos códigos e relacionamentos, formando um total de 64 códigos nesta etapa (vide APÊNDICE B – MAPAS DE CÓDIGOS DA GT). Dentre os códigos encontrados/aprimorados nesta etapa estão: métodos/metodologias ágeis; refatoração; protocolos (cerimônias) da metodologia ágil; e aplicação de boas práticas como, *sprint review*, *sprint plan*, requisitos, testes, refatoração colaborativa, entre outros.

A terceira versão do mapa refina ainda mais os resultados da pesquisa e evidencia melhor os relacionamentos entre os códigos, como, por exemplo, as atividades que são pré-condições ou diminuem ou intensificam algum resultado esperado. Este mapa possui 117 códigos (vide APÊNDICE B – MAPAS DE CÓDIGOS DA GT). Dentre os códigos encontrados pode-se citar: as atividades de pré refatoração, pós refatoração, diminuição de ilhas de conhecimento, mudanças de requisitos formais para *user stories*, etc.

Em paralelo com a codificação, reformulação do guia e novas entrevistas, foi realizada uma nova revisão da bibliográfica com o objetivo de comparar os dados obtidos na pesquisa com os trabalhos encontrados, a fim de delimitar um escopo

específico do trabalho. A partir dessa análise foi possível identificar a carência de pesquisa sobre a área de refatoração em metodologias ágeis. Essas atividades caracterizam a etapa de amostragem teórica da GT.

Após a análise de alguns trabalhos relacionados ao tema, como por exemplo a revisão sistemática realizada por Dybå e Dingsøyr (2008), foi possível identificar que haviam poucos estudos relacionados à atividade de refatoração em organizações que utilizam metodologias ágeis no âmbito internacional e nenhum estudo no âmbito nacional. Isto claramente indicava uma oportunidade de pesquisa e proporcionava uma melhor delimitação de escopo do trabalho, uma vez que foram encontrados resultados em diversas áreas da Engenharia de Software.

Com a execução da atividade de amostragem teórica (ver seção 3.2.5) e as motivações citadas acima, este estudo tem como foco de pesquisa aprofundar sua investigação a atividade de refatoração, quando executada em projetos que utilizam metodologias ágeis. É importante ressaltar que a delimitação do escopo da pesquisa ocorreu em paralelo com as etapas de coleta, por isso foi necessário adaptar o guia de entrevista alguma vezes para melhor explorar os novos resultados com foco na atividade de refatoração. Abaixo serão apresentados somente os resultados e códigos relacionados ao foco principal descrito neste trabalho: a refatoração. Conforme mencionado anteriormente, os códigos de outros aspectos podem ser encontrados no APÊNDICE B – MAPAS DE CÓDIGOS DA GT.

A Figura 7 ilustra a primeira versão do mapa de códigos, onde foram encontrados 12 códigos relacionados ao tema da refatoração como por exemplo: aplicação da prática da refatoração, dificuldades encontradas para aplicação desta prática, pontos positivos da aplicação da prática de refatoração, etc.

A Figura 8 ilustra a segunda versão do mapa, onde foram encontrados 25 códigos relacionados incluindo: as características da refatoração, testes e validações, planejamento da refatoração, etc.

A terceira e última versão do mapa de códigos possui 81 códigos divididos em quatro visões: a Figura 9 ilustra a visão geral; a Figura 10 ilustra a visão referente às atividades de pré refatoração; a Figura 11 ilustra a visão referente às atividades de

execução da refatoração; e, finalmente, a Figura 12 ilustra a visão referente às atividades de pós refatoração.

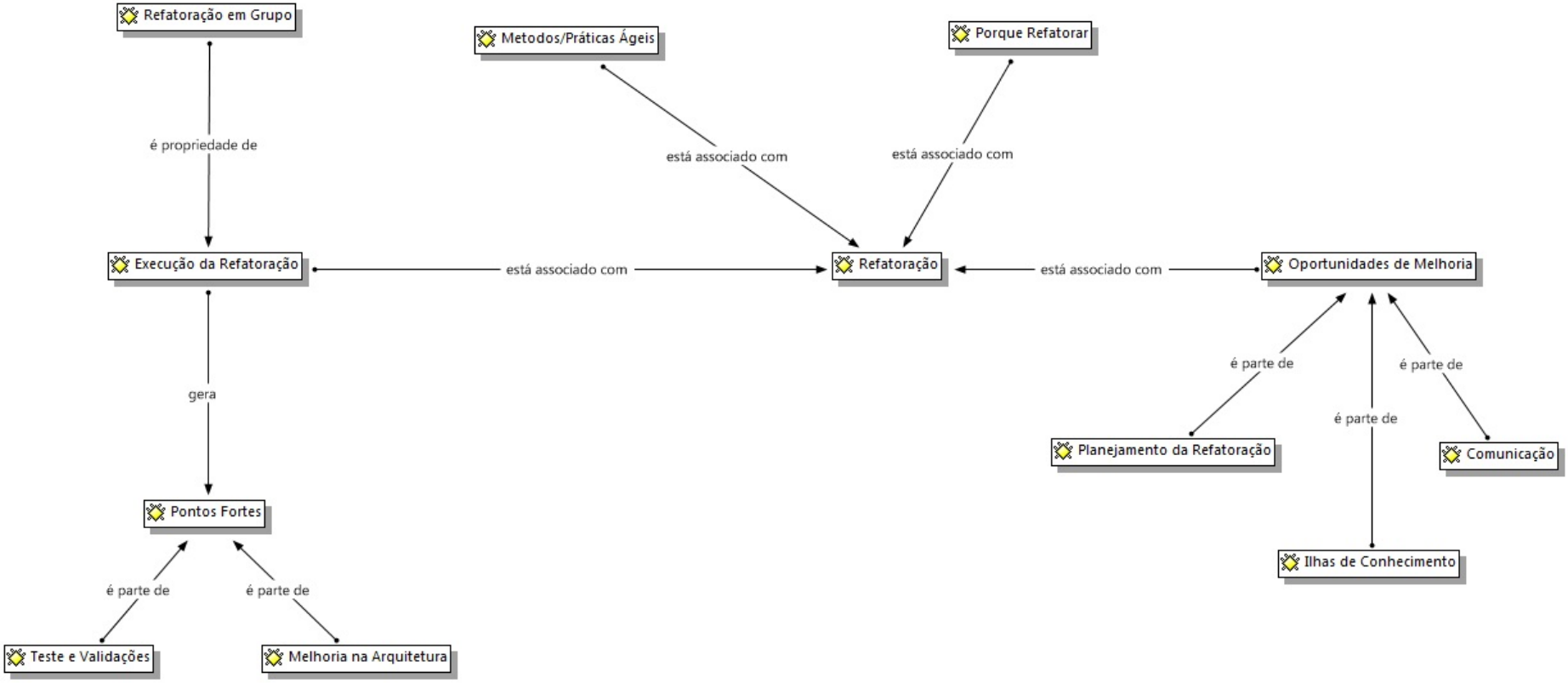

Figura 7– 1ª Versão do Mapa de Códigos (Resultados da Refatoração)

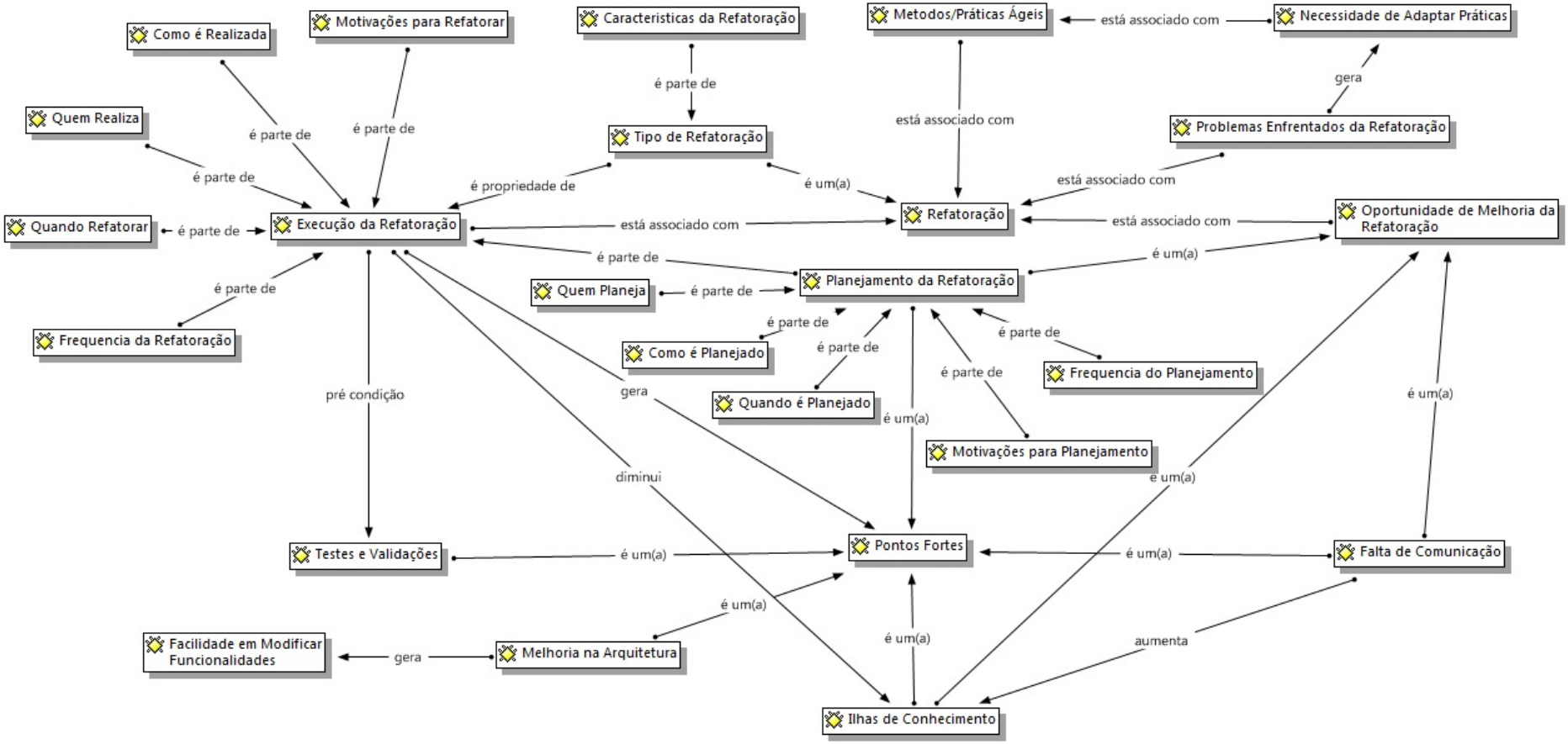

Figura 8 – 2ª Versão do Mapa de Códigos (Resultados da Refatoração)

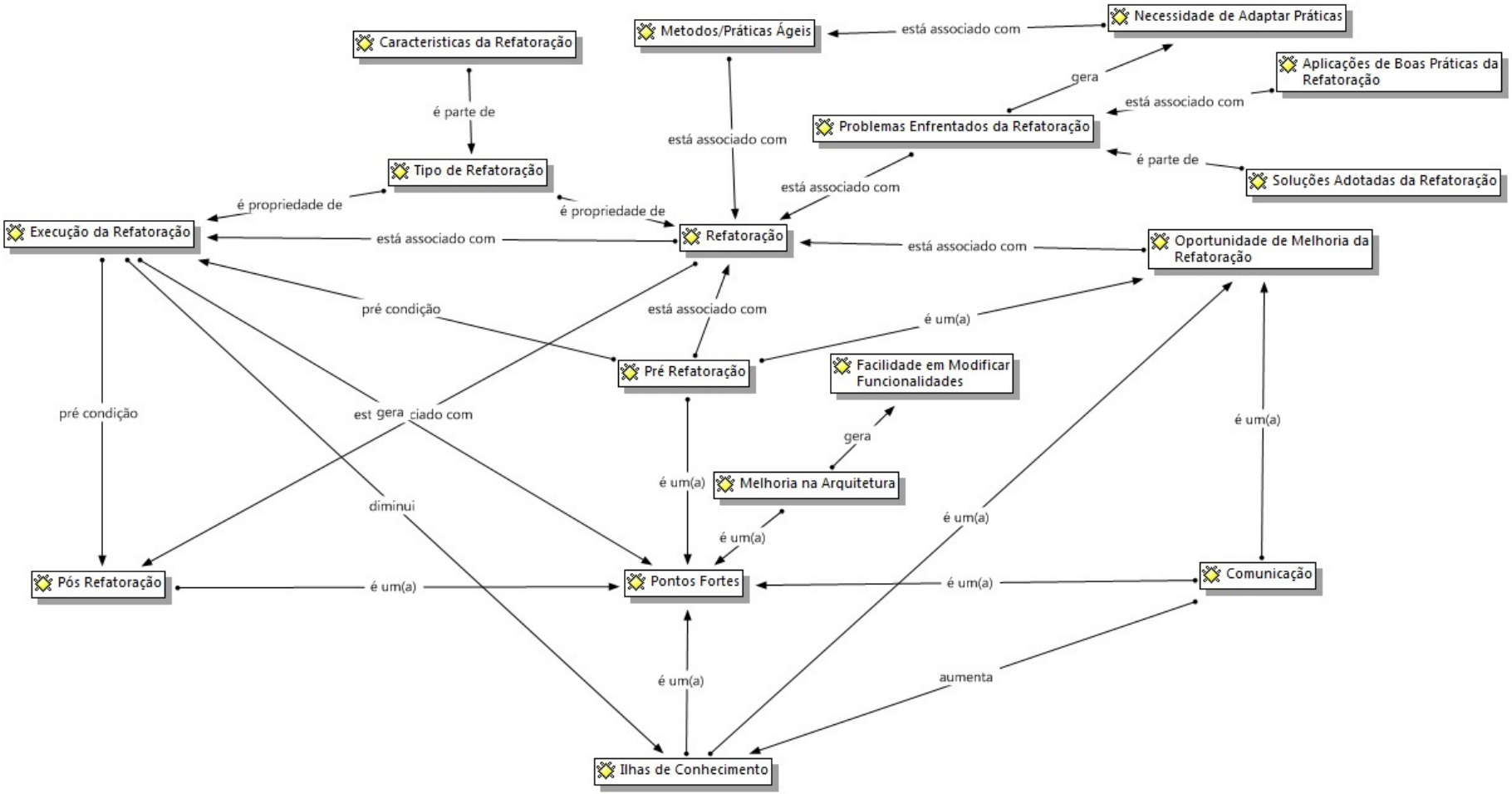

Figura 9 – 3ª Versão do Mapa de Códigos (Resultados da Refatoração): Visão Geral

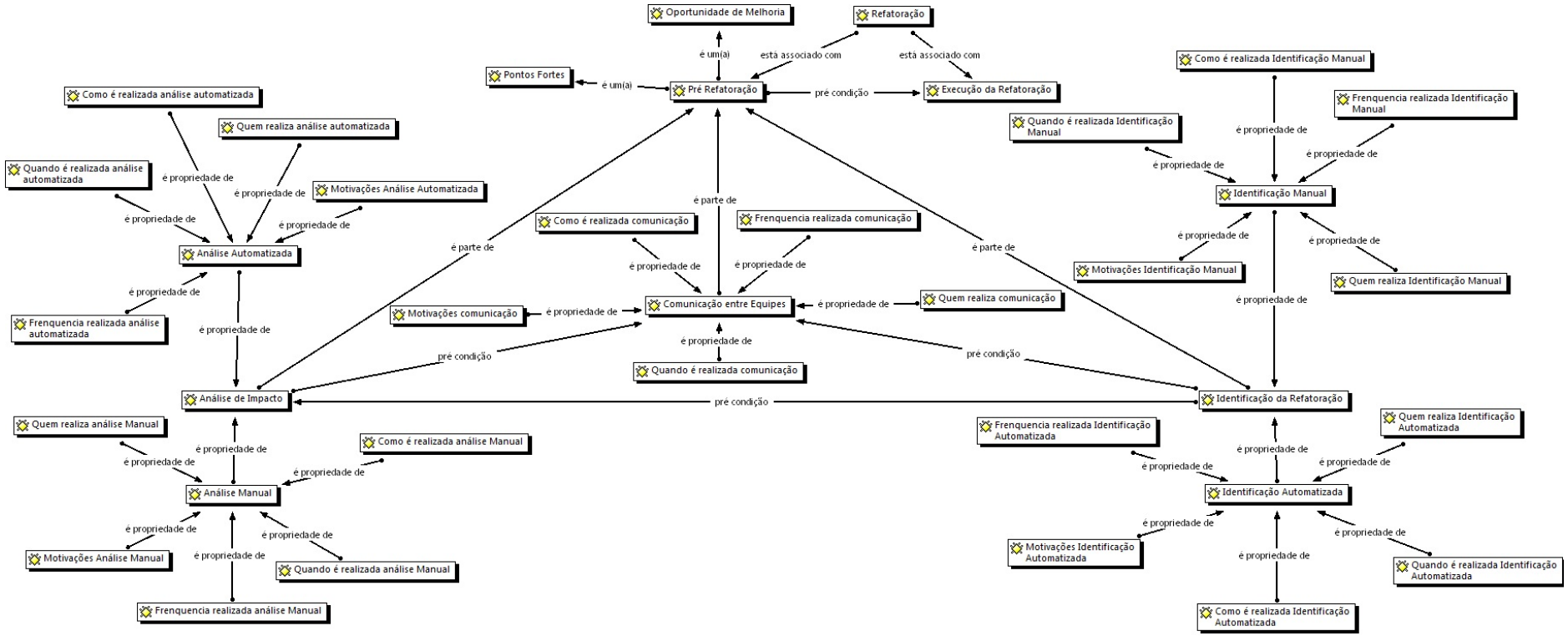

Figura 10 – 3ª Versão do Mapa de Códigos (Resultados da Refatoração): Visão Pré Refatoração

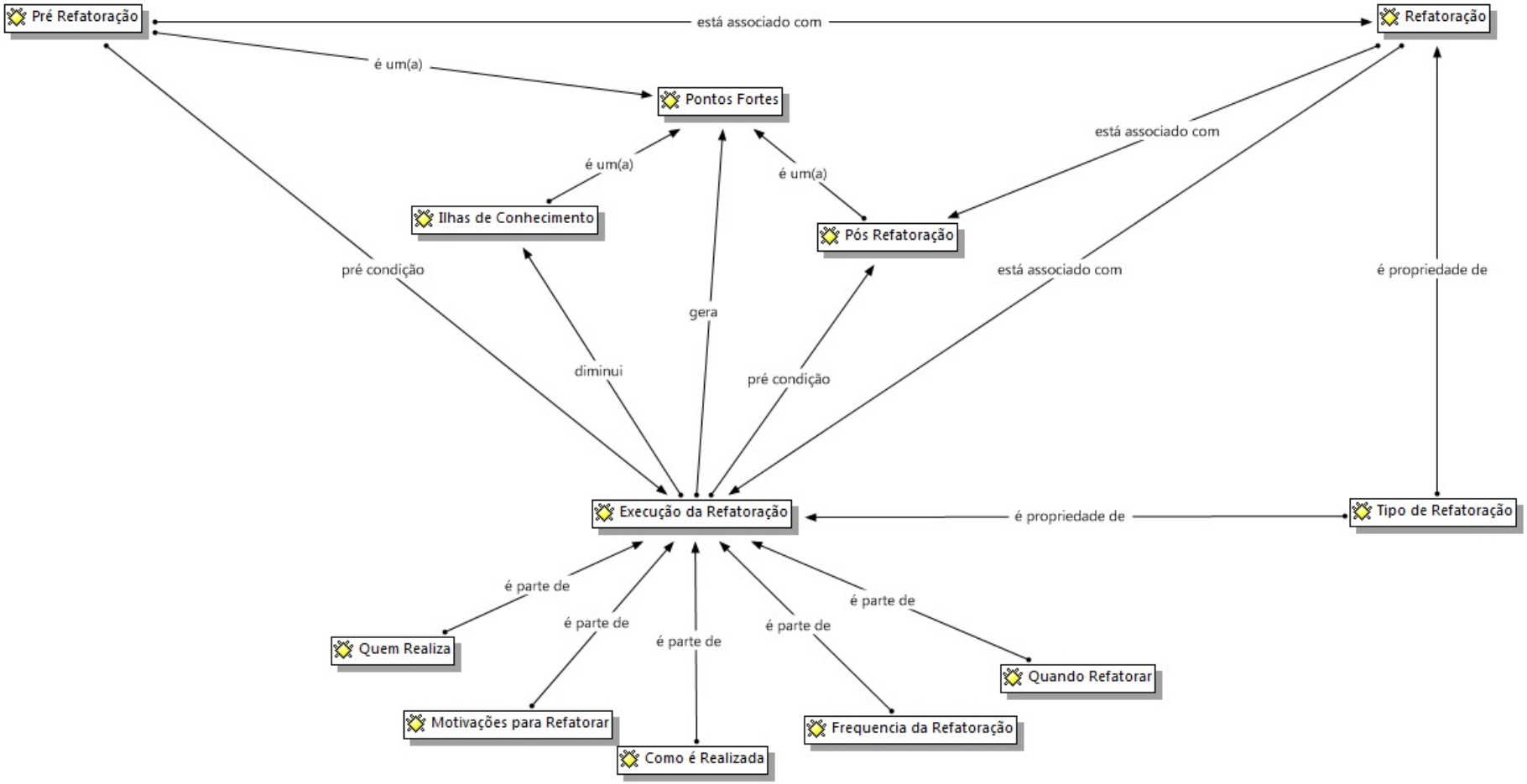

Figura 11 – 3ª Versão do Mapa de Códigos (Resultados da Refatoração): Visão Execução da Refatoração

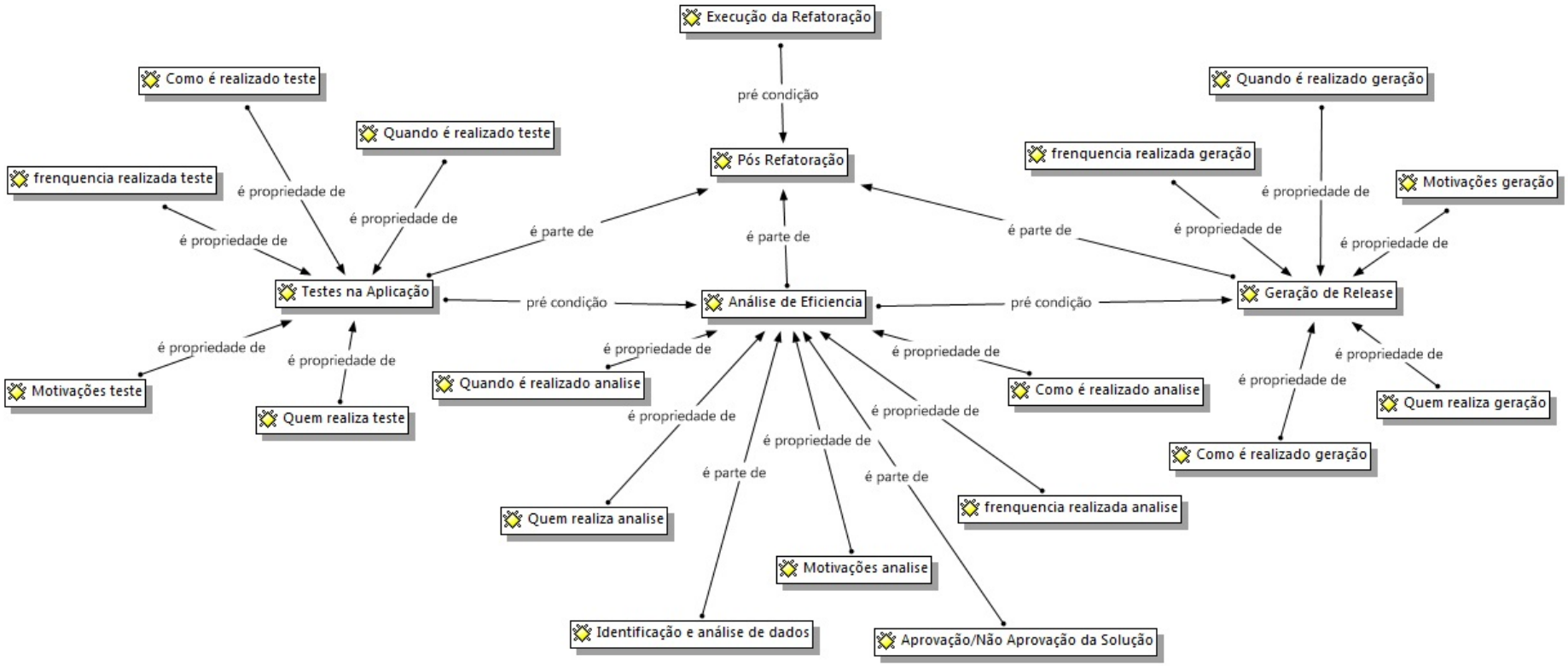

Figura 12 – 3ª Versão do Mapa de Códigos (Resultados da Refatoração): Visão Pós Refatoração

5 RESULTADOS ENCONTRADOS

Após a realização da coleta de dados e da utilização de métodos de codificação da GT para auxiliar na análise, foi obtido um entendimento mais preciso sobre a aplicação das metodologias ágeis nas empresas estudadas. Com isso, foi possível identificar pontos interessantes relativos à adoção de uma abordagem mais colaborativa na atividade de refatoração. Como citado na subseção 4.3 o estudo teve foco na atividade de refatoração, devido esta atividade adquirir um aspecto mais colaborativo e tal foco ainda foi pouco explorado na literatura (Dybå e Dingsøyr, 2008) (Lindvall, Basili, *et al.*, 2002).

Vale ressaltar que foram encontradas duas visões sobre a forma com que a atividade de refatoração é realizada na revisão sistemática realizada neste estudo, que é descrita com maiores detalhes na seção 6.2 deste trabalho. A primeira diz respeito à atividade de refatoração com planejamento prévio voltada à modificações arquiteturais ou de grande complexidade no sistema, modificações que envolvem vários módulos ou o núcleo (*core*) do sistema (Lippert, 2004) (Lippert e Roock, 2006). Já a segunda visão é relacionada à atividade de refatoração com alterações cotidianas realizadas em uma escala menor (Bravo e Goldman, 2010), quando um desenvolvedor, para iniciar sua atividade, necessita realizar pequenas refatorações para deixar o código fonte mais extensível, por exemplo.

No caso deste estudo, a visão de refatoração adotada é baseada no ponto de vista dos entrevistados que expuseram a refatoração como um procedimento de qualidade, voltado à melhoria de código, ou seja, quando a tarefa de refatoração é planejada previamente, onde ocorrerão alterações em módulos complexos ou modificação da

arquitetura geral do sistema. Em outras palavras, a visão adotada neste estudo está relacionada à primeira opção relatada na revisão sistemática.

O estudo dividiu a atividade de refatoração colaborativa em três visões. A primeira visão é atribuída às atividades que antecedem a refatoração, denominada de pré refatoração. Esta visão compreende as atividades executadas antes da atividade de refatoração, como por exemplo, as atividades de planejamento, identificação da necessidade de refatorar e análise de impacto. A segunda visão diz respeito à execução das atividades de refatoração, ou seja, de que forma esta atividade é realizada, por quais papéis, entre outros aspectos. Por fim, a terceira visão está relacionada com as atividades realizadas após a atividade de execução da refatoração, contemplando, por exemplo, as atividades de teste e resolução de conflitos. Esta divisão em visões é apenas para melhor exemplificar e organizar os resultados encontrados neste estudo.

Nesta seção serão apresentadas com detalhes as atividades realizadas em cada uma destas etapas, bem como a motivação para execução das atividades, como estas são realizadas, com que frequência elas são realizadas em um determinado projeto e quais são os pontos positivos e negativos de sua execução segundo os entrevistados.

5.1 Pré Refatoração

Nesta subseção serão apresentados os resultados do estudo relacionados às atividades que antecedem a execução da refatoração. Os resultados estão divididos em dois conjuntos de atividades: O planejamento da refatoração e a comunicação entre os módulos e as equipes sobre os impactos da refatoração.

5.1.1 Planejamento da Refatoração

A partir da análise dos dados coletados nas entrevistas e na análise da bibliografia especializada, foi possível identificar um conjunto de tarefas executadas na etapa de planejamento da refatoração. Este conjunto de tarefas é realizado de forma similar entre as organizações estudadas, exceto na empresa I[3], variando apenas na frequência com

[3]Deve-se levar em consideração que a empresa I possui características diversas das outras empresas estudadas, uma vez que a mesma conta com mão de obra rotativa, baixa experiência de seus

que ocorrem e na forma de execução: automatizada ou manual. Primeiramente é realizada a tarefa de identificação da necessidade de refatorar. A empresa I não realiza as tarefas de planejamento da refatoração segundo o informante 1 conforme o trecho abaixo retirado de sua entrevista.

> *A única atividade que nós realizamos deste tipo [Planejamento] é de avisar os outros programadores que vamos alterar alguma parte do código, para evitar problemas com alterações do mesmo código ao mesmo tempo.*

No entanto, em entrevista com o *Scrum master* (ver seção 2.1.2.2) da empresa I foi notado um grande interesse em introduzir a prática de planejamento da refatoração, conforme o trecho abaixo:

> *Por enquanto nós não temos nenhuma atividade formal de planejamento da refatoração, visando identificar quais os impactos de uma alteração do código [...] Em reunião de retrospectiva já foi mencionado a introdução desta prática [Planejamento da refatoração], já que uma vez houve um grande problema por não planejar direito essas mudanças e foi necessário voltar atrás com a modificação, então nós já estamos pensando em uma forma de fazer este planejamento sim.*

Nas empresas II, III e IV a atividade de planejamento da refatoração é realizada. Segundo o informante 3 da empresa II esta tarefa é realizada de forma manual ao final de cada *sprint* (ver seção 2.1.2), sendo realizada através de inspeção de código fonte conforme o trecho abaixo retirado de sua entrevista:

> *A primeira tarefa é identificar onde é preciso refatorar, normalmente ao final de cada sprint os desenvolvedores têm a tarefa de inspecionar o código produzido naquela sprint [...] Cada desenvolvedor pega o código de outro e faz a inspeção, anota onde precisa refatorar para depois discutir se vai ser feito numa revisão técnica.*

Esta prática também é realizada de forma similar nas empresas III e IV. Os desenvolvedores primeiro fazem a inspeção do código fonte do sistema em busca de oportunidades de melhorias e pontos que necessitam ser refatorados. Após a etapa de inspeção, os desenvolvedores realizam uma reunião chamada de "revisão técnica" ou "revisão de código", sendo realizada de forma similar pelas empresas II, III e IV. Nesta reunião são mostrados os códigos que devem ser refatorados, bem como o motivo para realizar a sugerida refatoração é explicado e a possível solução a ser executada posteriormente. Ao final desta reunião é gerada uma lista de atividades de refatoração

---

colaboradores e está instalada (encubada) em uma universidade, conforme descrito na seção 4.2.1 desta dissertação.

que serão posteriormente executadas. Abaixo um trecho retirado do informante 1 da empresa IV:

> *Depois de inspecionar o código nós vamos para uma reunião para avaliar se realmente é necessário fazer aquela mudança [...] algumas das refatorações são 'só perfumarias' e podem não ser feitas, mas também têm aquelas refatorações que precisam ser feitas para manter a qualidade, as que vão pra lista [...] nós fazemos uma lista de coisas para fazer, que são as refatorações que devem ser feitas no sistema [...] depois de gerar a lista inicial com as refatorações nós vamos definir como iremos solucionar elas.*

A atividade de definição da solução é realizada a partir da lista gerada anteriormente, onde cada refatoração é analisada a fim de identificar possíveis soluções para resolver o problema indicado. Estas possíveis soluções são avaliadas pelos desenvolvedores, arquiteto da equipe, e em alguns casos o arquiteto do sistema participa, para identificar qual seria a solução mais adequada ao problema. Neste momento, todos os envolvidos fazem considerações sobre funcionamento do módulo, e sobre quais seriam as modificações a serem realizadas na arquitetura, gerando com isso mais insumo na lista de tarefas. O trecho abaixo retirado do informante 4 da empresa II evidencia como está tarefa é realizada:

> *Depois que a lista inicial foi gerada, nós analisamos item a item da lista para identificar como vamos proceder naquela alteração. [...] dessa reunião participamos desenvolvedores, arquiteto do módulo, e algumas vezes o arquiteto de sistema [...] todo mundo na reunião tem voz, pode dar sugestões e comentários e também criticar a solução dos outros. [...] no final da reunião nós anotamos a solução na lista de tarefas e geramos o novo design da arquitetura do sistema.*

Vale ressaltar que nas empresas III e IV a atividade de identificação não ocorre com a mesma frequência que na empresa II. Segundo o informante 1 da empresa IV esta atividade é realizada de forma esporádica, ou quando sobra tempo na *sprint*. Ocorrendo de forma semelhante na empresa III. Um aspecto negativo notado sobre esta tarefa é que, algumas vezes, ela deixa de ser executada devido à exigência do cliente pela execução de outro módulo em um prazo reduzido ou por parte da própria empresa, em busca de diminuir o tempo/esforço do projeto. Neste caso, a tarefa de planejamento da refatoração não é realizada conforme trecho abaixo do informante 1 da empresa IV.

> *Nem sempre nós temos tempo para realizar a atividade de planejamento, porque o cliente já quer outras funcionalidades, daí temos que ir logo fazendo coisas novas. Normalmente, quando temos tempo livre nós fazemos todo esse protocolo, mas não é de forma "religiosa" esse planejamento não.*

Os desenvolvedores de todas as empresas têm conhecimento que estas atividades são importantes para garantir a qualidade do produto gerado. Os informantes 1 e 2 da

empresa IV afirmam que a equipe está ciente do problema e se sente incomodada em não realizar o planejamento, pois sabem que isso vai impactar diretamente na qualidade do sistema, conforme trecho abaixo.

> *Toda vez que estamos programados para fazer o planejamento da refatoração e por algum motivo não fazemos, nós {Desenvolvedores} avisamos que isso pode inserir muitos bugs na aplicação e não nos comprometemos mais com a qualidade, já que não estão deixando a gente fazer do jeito certo [...] e muita gente não fica satisfeita quanto ocorre esse tipo de situação, porque depois sempre somos cobrados para manter o mínimo possível no índice de bugs e retrabalho, mas sem planejamento fica difícil fazer isso.*

Os desenvolvedores também ressaltam que já são conhecidos os problemas quando não são realizadas as atividades de planejamento. Segundo o informante 6 da empresa III quando esta atividade de planejamento não é executada ocorre algum tipo de problema conforme o trecho abaixo.

> *Normalmente quando não realizamos o planejamento da refatoração nós encontramos problemas na hora da execução. Tipo: o código que temos que refatorar está muito pulverizado em várias classes e o programador deve juntar em uma classe só, é fácil esquecer parte do código em alguma classe, daí temos que refazer esta parte que faltou depois, quando encontramos de novo [...].*

As empresas II e III realizam uma atividade de análise de impacto sobre o código que deve ser refatorado ao final da reunião de revisão técnica. Esta análise de impacto é realizada pelos programadores e o arquiteto de software ou por um programador mais experiente. Somente a partir desta análise que os desenvolvedores iniciam o processo de refatoração, conforme o trecho abaixo retirado do informante 1 da empresa III:

> *Ao final de cada revisão técnica, depois de ter gerado a lista de refatorações que devem ser feitas, nos realizamos uma análise de onde devemos modificar e quais os impactos da modificação, vendo as dependências entre o código fonte, tanto no mesmo módulo, como entre outros módulos, para evitar a criação de um bug.*

Ainda na entrevista com o informante 1 da empresa III pode-se perceber que a execução da atividade de análise de impacto em código fonte é uma tarefa importante para equipe de desenvolvimento, uma vez que a equipe tem a percepção de que ao executar esta tarefa é possível evitar problemas futuros de integração do sistema. Sendo possível também verificar os problemas quando esta tarefa não era executada, conforme o trecho abaixo:

> *Antes {sem realizar a atividade de análise de impacto} nós refatoravamos o código e muitas vezes os módulos que utilizam informações do módulo que foi modificado paravam*

> *de funcionar[4], justamente porque nós mudávamos a assinatura de um método, ou seu resultado [...] Quando não ficávamos atentos a essa mudança, principalmente quando o modulo é feito por outra equipe, temos problemas depois da refatoração, por isso que é muito útil fazer essa análise de impacto.*

A empresa I não realiza a atividade de identificação da necessidade de refatorar nem a análise de impacto em código fonte: a identificação é feita de forma *ad hoc*, o que pode gerar um impacto negativo na aplicação da refatoração como visto no trecho abaixo. Segundo o informante 7 da empresa I, os desenvolvedores, ao iniciar uma tarefa, realizam uma análise do código onde a nova tarefa irá ser integrada de tal maneira que uma possível necessidade de refatoração pode ser identificada ou os desenvolvedores fazem análise em determinados marcos de entrega de *release*. Porém, esta atividade ainda não é uma prática comum na empresa, conforme o trecho abaixo:

> *Quando nós vamos começar a fazer uma tarefa, o primeiro passo é saber onde ela vai se encaixar no sistema, então nós começamos a ver o código onde vamos ligar nosso método e fazemos uma análise, se o código estiver precisando de mudanças, ou não estivermos entendendo o que ele faz, nós procuramos quem fez a funcionalidade para entender e fazemos uma anotação para lembrar que temos que modificar o código na próxima sprint [...] esse é o código que precisamos refatorar [...].*

Uma das equipes estudadas da empresa V realiza a identificação da necessidade de refatorar e a análise de impacto de forma automatizada através de uma ferramenta denominada de ferramenta A, por questões de sigilo de informação, a pedido da empresa. Outra equipe realiza a identificação e análise de impacto via inspeção de código, similar a empresa II, inclusive em termos de execução, forma e frequência de realização. Esta ferramenta é utilizada para identificar a necessidade de refatoração bem como os possíveis artefatos que serão impactados com esta mudança conforme o trecho abaixo retirado da entrevista com o informante 1 da empresa V.

> *Aqui na empresa nós usamos a ferramenta A, ela faz uma leitura no código fonte verificando se precisa ou não de refatoração. Se for necessário podemos pedir para a ferramenta verificar todo o código fonte de determinado repositório em busca de bugs, códigos duplicados ou possíveis códigos que precisem de refatoração [...] Ela [Ferramenta A] também mostra alguns artefatos que possuem ligação direta com o trecho de código que a ferramenta sugere refatorar, tanto a ligação direta por chamada de método e até por herança ou reflexão [...] O bom é que essa ferramenta é acoplada a uma IDE, que permite verificar direto o código que devemos modificar [...]*

[4] Esta situação ocorreu devido o sistema que estava sendo desenvolvido ter sua arquitetura baseada em cliente e servidor e em chamadas remotas de métodos entre projeto diferentes via reflexão computacional (Forman e Forman, 2004). Estas características dificultam a identificação do erro, sendo que as IDEs de auxilio ao desenvolvimento não são capazes de alertar sobre tal problema.

De forma similar, as duas equipes da empresa V executam as atividades de identificação da necessidade de refatorar e a análise de impacto do código fonte durante a execução de cada *sprint*. Esta a atividade é planejada em cada iteração de desenvolvimento, antes da realização dos testes finais para apresentação da *sprint review* (ver seção 2.1.2.1). Vale ressaltar que esta atividade é parte inerente do processo de desenvolvimento, conforme trecho abaixo, retirado da entrevista com o informante 2 da empresa V:

> *No planejamento das tarefas da nossa sprint nós já colocamos as tarefas de inspeção de código e análise de impacto, para que antes de entregar qualquer release para o cliente ou antes da review, o código já seja refatorado.*

5.1.2 Comunicação entre equipes

Outro resultado identificado na etapa de pré refatoração é a atividade de comunicação entre as equipes de desenvolvimento. Esta comunicação é feita sobre os possíveis impactos da refatoração. Esta atividade tem por objetivo comunicar as equipes de desenvolvimento do projeto sobre as modificações a serem feitas em determinados módulos. Ela é importante para informar as equipes de desenvolvimento que determinado módulo irá sofrer modificações internas, podendo com isso realizar análises sobre estas mudanças a fim de prevenir qualquer efeito colateral desta modificação no restante do sistema.

A atividade de comunicação de impactos da refatoração é realizada pelas empresas II, III e V logo após a realização da etapa de planejamento da refatoração. Segundo o *Scrum master* da empresa III esta atividade é muito importante para manter a coordenação entre as equipes quando se trata de um projeto de médio ou grande porte, segundo o trecho abaixo, retirado de sua entrevista:

> *Esta atividade é uma das mais importantes para a coordenação, nós comunicamos qualquer mudança que vai ser feita, para deixar todo mundo ciente e ver na sua equipe se não vai impactar no seu código. [...] {a comunicação} Sempre executada em conjunto com o planejamento das refatorações.*

Os envolvidos na atividade de comunicação são diferentes entre as empresas estudadas. Uma vez que cada empresa possui uma estrutura organizacional própria, com a existência de diferentes cargos, ou diferentes denominações para o mesmo cargo, utilizou-se o mesmo código para representar este fenômeno que descreve a necessidade

de envolver diversas pessoas na atividade de refatoração. Nas empresas II e V esta atividade é realizada entre os seguintes papéis: *Scrum master*, arquiteto de software e líder de equipe. O trecho retirado da entrevista com o informante 3 da empresa II exemplifica o fato comentado acima.

> *Nessa reunião participam o arquiteto para entender onde vai ser alterado e o porquê desta alteração, o líder de equipe, para explicar os motivos da mudança e o Scrum master que normalmente é o mediador da reunião.[...] O líder da equipe, normalmente o nosso programador sênior, que tem mais experiência, normalmente é pra ele que perguntamos sobre as duvidas técnicas.*

Em algumas ocasiões, parte da equipe de desenvolvimento pode participar desta reunião, porém isto acontece de forma esporádica, apenas quando há a necessidade de uma explicação mais detalhada sobre as possíveis alterações ou quando o líder técnico não está presente.

Vale ressaltar que os papéis citados acima e a forma com que é executada esta reunião é equivalente para as empresas II e V. Já na empresa III a atividade de comunicação das alterações é realizada entre o S*crum master*, o arquiteto de software, os líderes de equipe e o gerente de projetos. A presença do gerente de projetos nesta reunião é necessária, pois em algumas situações, esta alteração pode impactar em outros módulos. Neste caso, seria necessário re-estimar algumas atividades ou modificar o cronograma de desenvolvimento. O trecho abaixo, retirado da entrevista com o gerente da empresa III, evidencia o fato citado:

> *Normalmente eu {Gerente de projetos} não participava destas reuniões, deixava a cargo do Scrum master e dos coordenadores {líderes de equipe} a conversa mais ligada ao código fonte. Mas, algumas vezes o Scrum master ou algum coordenador vinha conversar comigo sobre essas alterações, quase sempre colocando alguma tarefa a mais, ou avisando que iria precisar modificar alguma coisa que já foi entregue ao cliente [...] Como algumas vezes é preciso alterar o cronograma de desenvolvimento, comecei a participar destas reuniões para ficar a par do que estava ocorrendo e se realmente era necessário realizar essas modificações.*

Quanto as outras empresas estudadas, a empresa IV afirmou não realizar a atividade de comunicação sobre a refatoração enquanto que a empresa I ao ser questionada sobre a execução desta tarefa informou que existem no máximo duas equipes atuando em um mesmo projeto, não havendo a necessidade de formalizar esta atividade. Para ser mais especifico, segundo o *Scrum master* da empresa I esta tarefa é executada de forma transparente, já que os times são pequenos, estão localizados no mesmo ambiente e

existe apenas um *Scrum master*, logo a informação está centralizada nele que coordena as duas equipes, conforme o trecho abaixo:

> *Aqui não precisa ter essa comunicação porque só temos duas equipes por projeto e um Scrum master para as duas equipes [...] As equipes são pequenas e sentam na mesma bancada, uma de frente para outra, eu sou o Scrum master das duas equipes então nem temos pra quem comunicar [...]*

Na próxima seção serão apresentados os resultados referentes à etapa de execução propriamente dita da refatoração de maneira colaborativa.

5.2 Execução da Refatoração colaborativa

Esta visão está associada à execução propriamente dita da refatoração, ou seja, às atividades relativas à alteração do código fonte. Ela tem como entrada as atividades citadas na etapa de pré refatoração, onde ocorre todo o planejamento e identificação da necessidade de refatorar. A maioria das empresas tem a prática da refatoração prevista em seu processo e uma das empresas realiza esta atividade de forma *ad hoc*.

Para a realização da atividade de refatoração é feita uma reunião, onde é decidido *como* a refatoração irá ocorrer e *onde* as tarefas serão executadas. O artefato de entrada para esta reunião é a lista de atividades de refatoração que devem ser realizadas. Em conjunto, uma explicação sobre a arquitetura do sistema, boas práticas de programação, entre outros assuntos, é realizada visando relembrar as definições realizadas na etapa de planejamento e compartilhar informações entre os envolvidos. O trecho abaixo foi retirado da entrevista com o informante 4 da empresa II que descreve a reunião de execução da refatoração. Tal reunião é realizada de forma equivalente em todas as empresas estudadas.

> *Todos os programadores da equipe são convocados para reunião de refatoração [...] Começamos vendo a lista de modificações da reunião passada {reunião de planejamento}, junto com suas soluções, daí nós pegamos item a item da lista para executar.[...] A reunião começa com o arquiteto explicando o motivo da modificação, para relembrar o que tem que ser feito, onde vai impactar e como irá ficar o design da arquitetura. Um dos programadores vai mostrando o código que será afetado no computador [...] Depois de explicar o problema e a sugestão de melhoria, o programador que está no computador faz as modificações e os outros programadores acompanham dando sugestões na forma como está sendo implementado.*

Esta reunião é realizada com a participação dos desenvolvedores e do arquiteto de software do projeto ou do módulo em questão e em algumas ocasiões o *Scrum master*

participa. É necessário apontar uma peculiaridade na forma com que a empresa V executa esta atividade. Como citado anteriormente, esta empresa possui duas equipes que trabalham de formas distintas: uma das equipes realiza esta atividade de forma similar as demais empresas, utilizando uma reunião técnica para executar a refatoração; enquanto que a outra equipe realiza esta atividade em pares, sem a necessidade da reunião técnica. Segundo um entrevistado desta equipe, esta prática não é realizada através desta reunião por escolha da equipe, uma vez que os desenvolvedores não ficam a vontade ao criticar a solução proposta por outro membro:

> *Essa prática não é utilizada na nossa equipe por criar muitas brigas entre os desenvolvedores. [...] Ninguém gosta de receber criticas do seu código, sabemos que isso é um defeito da equipe, porém como já tivemos muitas brigas decidimos fazer do jeito tradicional mesmo, quando uma pessoa percebe que precisa refatorar parte do código, realiza logo a alteração ... no máximo pergunta para quem fez a funcionalidade para tirar alguma dúvida.*

Devido esta divergência na forma com que as equipes da empresa V executam a atividade de execução da refatoração, apenas será levada em consideração, como fonte de informação para este resultado, a equipe que utiliza a reunião técnica como forma de executar a refatoração.

Um aspecto observado nas entrevistas é que a periodicidade com que a atividade de execução da refatoração é feita é diferente entre as empresas estudadas. Por exemplo, a empresa I realiza esta atividade de forma esporádica, quando é realizada a inspeção de código antes de realizar determinada funcionalidade ou após os marcos de entrega de produto para o cliente, como citado na atividade de planejamento da refatoração, conforme o trecho abaixo retirado da entrevista com o informante 4 da empresa I.

> *Assim que o desenvolvedor percebe que devemos refatorar o código, ele cria uma tarefa no nosso quadro, que vai ser feita no final da Sprint, além de comunicar ao Scrum master [...] É uma tarefa tipo, "Realizar a refatoração nas classes/métodos: X, Y e Z"... que no final da sprint, antes da review e dos testes finais, nós reunimos para fazer essa tarefa em conjunto [...] Certa vez nós estávamos com uma versão pronta, aí nós apresentamos o release para o cliente. Dentro das alterações que eles pediram, a gente aproveitou já pra refatorar o que era preciso.*

As empresas III e IV executam a tarefa de execução da refatoração de forma esporádica, também seguida das atividades da etapa de pré refatoração. Como citado anteriormente, a atividade de pré refatoração é realizada quando os desenvolvedores estão com tempo livre, normalmente entre a execução de uma *sprint* e outra, refletindo

também nas atividades da execução da refatoração, conforme o trecho abaixo o informante 1 da empresa IV.

> *Essa reunião {reunião de execução da refatoração} acontece logo após o nosso planejamento. Toda vez que nós temos tempo livre no projeto, nós focamos na qualidade do código, fazemos a inspeção para identificar as refatorações ou melhorias a serem feitas e depois executamos.*

Já nas empresas II e V esta atividade é realizada ao final de cada *sprint*, sendo executada após a atividade da etapa de pré refatoração. O trecho abaixo foi retirado da entrevista com o informante 2 da empresa II para elucidar esta informação.

> *Nossas tarefas são definidas no sprintplan e sempre procuramos manter o mesmo fluxo das atividades, principalmente para manter nossa boa prática, então nós executamos a reunião de refatoração após as reuniões de planejamento, que sempre são feitas no final de cada sprint.*

De um modo geral, a análise dos dados sugere que as motivações para adaptar a refatoração colaborativa são: a melhoria no nivelamento de conhecimento técnico entre a equipe; a maior compreensão da arquitetura do sistema por todos; disseminação da informação e diminuição das ilhas de conhecimento sobre o sistema. Estas atividades permitem a diminuição do índice de *bugs*. Estas vantagens foram identificadas em todas as empresas estudadas. Abaixo o trecho que foi retirado da entrevista com o informante 2 da empresa V: ele confirma algumas destas vantagens na execução da atividade de refatoração de maneira colaborativa.

> *Os principais resultados que nós tivemos quando começamos a utilizar a refatoração em grupo foi que a equipe começou a ter o mesmo conhecimento técnico. Como conversávamos juntos com toda a equipe, acabava que um sabia mais do que outro sobre, por exemplo, determinada tecnologia que usamos ou sobre um padrão de projeto ... como todos davam sugestões para melhoria do código, todos acabávamos aprendendo algo novo, o que com o tempo fazia com que todos tivessem o mesmo nível de programação, dos mais novos aos mais experientes da equipe. Nossa meta era sempre estar se atualizando em termos de tecnologia e sempre temos a preocupação de repassar a todos esse conhecimento [...] Várias vezes foi apontado que havia ilhas de conhecimento sobre a arquitetura do sistema e suas funcionalidades, depois que começamos a fazer a refatoração em grupo, vimos que essa reclamação diminuiu, já que quando estávamos refatorando, sempre tinha uma explicação sobre a arquitetura e o módulo que estávamos fazendo.*

Outra fonte que evidencia estes benefícios é vista na entrevista com o informante 5 da empresa III:

> *Nós vimos que quando fazíamos a refatoração em grupo diminuíamos as ilhas de conhecimento, que eram muito grandes dentro do projeto. [...] No final da sprint, na retrospectiva, sempre alguém apontava a refatoração em grupo como algo muito positivo,*

> *sempre dizendo que aprendeu algo novo da linguagem de programação ou que aprendeu algo novo da arquitetura do sistema ou uma funcionalidade que ainda não conhecia muito bem.[...] Outra grande vantagem é que diminuiu o número de bugs na aplicação, como todo mundo começa a conhecer melhor o sistema, o índice de bugs tende a diminuir.*

É necessário ressaltar que estes benefícios foram citados em quase todas as entrevistas realizadas, sendo citados pelo menos uma vez em todas as empresas estudadas. Estes benefícios são em parte responsáveis pela motivação em realizar a refatoração de forma mais colaborativa, sendo esta prática uma adaptação na atividade de refatoração, que é difundida normalmente como uma prática individual em sua forma de execução.

Na empresa V identificou-se que a equipe que realizava a refatoração em grupo, com o passar do tempo, fizeram uma variação na forma com que a atividade de refatoração era realizada. Após certo tempo executando a refatoração de forma colaborativa, os envolvidos notaram que o tempo gasto na execução gastava mais tempo do que o time dispunha para realizar a atividade. Então foram realizadas algumas medições para identificar qual sub-atividade utilizava mais tempo e por qual motivo. Após esta análise, foi identificado que a atividade de executar as tarefas de refatoração em conjunto era atividade que mais despendia tempo. A execução já não agregava mais tanto valor à equipe, uma vez que o trabalho criativo, para definir a solução do problema, já havia sido proposto anteriormente e a equipe já possuía o conhecimento técnico equivalente. Não havendo mais ganhos reais na execução desta parte da tarefa em conjunto, a equipe decidiu abandonar a tarefa de executar a refatoração na reunião, porém permaneceram realizando as atividades de explicação, análise e criação da solução para o problema exposto. Abaixo o trecho retirado da entrevista com o informante 2 da empresa V para elucidar o fato.

> *Com o passar do tempo, nós percebemos que esta tarefa estava gastando mais tempo do que nós tínhamos disponível, o que acabava atrapalhando o andamento de outras atividades, então nós decidimos ver qual era o real problema da reunião. A princípio nós pensávamos que as pessoas estavam gastando muito tempo debatendo a solução do problema a ser resolvido ou que estávamos tendo que refatorar muitas coisas. Porém, vimos que a grande demora estava na implementação da solução definida. Então resolvemos tirar essa parte, executar apenas ela individualmente. [...] Nós achávamos que poderíamos ter perdas em retirar a implementação, já que muitas vezes nós acabamos trocando conhecimento nessa execução, mas pelo contrário, na nossa terceira experiência em usar a reunião apenas para mostrar qual era o problema, como o módulo funcionava e definíamos a solução para o problema encontrado, nós notamos que não havíamos perdido nada, uma vez que todo mundo já estava acostumado a implementar o código do jeito*

> *parecido. Então retiramos esta parte da tarefa {a implementação}, ficando só a parte de criação para reunião, o que nós fez ganhar quase 55% a mais de tempo.*

No trecho acima é possível notar que houve uma preocupação por parte da equipe em retirar a atividade de implementação da refatoração sem provocar perdas nos benefícios que as outras atividades da refatoração colaborativa provêm, pois as decisões importantes em relação à refatoração continuam sendo tomadas em grupo.

Na próxima subseção serão apresentados os resultados do estudo relacionado às atividades de pós refatoração.

5.3 Pós Refatoração

Nesta visão estão associadas as atividades que ocorrem após a atividade de refatoração. Elas abrangem as atividades de realização de testes de regressão sobre o sistema, análise de eficácia nas soluções aplicadas e geração de *release* (ver seção 2.1.1). Todas as empresas estudadas executam estas atividades, variando em determinados aspectos, que serão abordados no decorrer do texto.

5.3.1 Testes

A primeira atividade realizada após a execução da refatoração é a de teste do sistema. Esta atividade visa identificar se algum tipo de *bug* foi inserido no sistema durante o processo de refatoração. Todas as empresas estudadas executam esta tarefa com a mesma periodicidade e papéis envolvidos, porém, existe uma variação na forma de execução da mesma.

Esta tarefa é executada sempre após a realização da atividade de refatoração em todas as empresas, tendo como executores desta atividade os seguintes papéis: equipe de desenvolvimento, equipe de testes e *Scrum master*. O trecho baixo, retirado da entrevista com o informante 2 da empresa II, ilustra a afirmação acima sobre os papéis e a periodicidade da execução desta tarefa, sendo comum a todas as empresas estudadas.

> *O nosso padrão é de realizar os testes após a execução da refatoração e antes de fechar qualquer baseline. [...] Esses testes são feitos pela nossa equipe {equipe de desenvolvimento}, com o Scrum master e com a equipe de testes.*

As empresas I e IV realizam esta tarefa de forma similar, onde os testes são realizados de forma manual ou apenas com testes de unidade. Segundo o informante 3

da empresa I, após a execução da refatoração são realizados os testes unitários do sistema, para identificar se algumas das funcionalidades do sistema estão sendo executadas de forma errada ou deixaram de funcionar. Após este teste de unidade, são realizados os testes de caixa preta e de aceitação, para verificar se o sistema mantém o comportamento desejável. Abaixo o trecho retirado da entrevista que evidencia esta afirmação.

> *Assim que nós terminamos de refatorar, todos os testes unitários são executados para ver se o sistema ainda está se comportando da maneira correta. [...] Depois que os testes unitários estão todos OK, nós começamos a fazer os testes de caixa preta e de aceitação, para ver se nada do que foi alterado teve efeito colateral no sistema e também é uma forma de garantir que os documentos estão de acordo com o que foi implementado.*

Já as empresas II, III e V realizam esta tarefa através da realização de teste de regressão do sistema, teste de integração, testes de caixa preta e aceitação do sistema e testes unitários partindo das mesmas motivações das empresas I e IV para sua execução. Abaixo o trecho retirado da entrevista com o informante 1 da empresa V corrobora a informação acima.

> *Para manter a qualidade do sistema e verificar se algum tipo de anomalia foi inserido no processo de refatoração, nossa equipe realiza primeiramente os testes unitários, para verificar se as funcionalidades continuam sendo executadas de forma correta. [...] Após a execução dos testes unitários são realizados os testes de integração e de regressão do sistema em busca de identificar alguma anomalia ou mau funcionamento no sistema. [...] Após verificar que o sistema executou com sucesso todos os testes nós liberamos uma versão mais estável para a equipe de testes realizar os testes de caixa preta e de aceitação.*

Vale salientar que a empresa II, após cinco meses de aplicação desta prática, deixou de realizar os testes de regressão, testes de integração e testes unitários, uma vez que a mesma identificou o alto custo para execução dos testes citados. A empresa afirma que a produção destes testes despende de um grande tempo de projeto e um alto custo financeiro para manter profissionais com tal especialidade e infraestrutura para suportar a execução das atividades. O trecho abaixo foi retirado de uma entrevista com um informante da empresa II, onde é evidenciado o fato acima.

> *Nós notamos que fazer principalmente os testes de integração, unidade e de regressão estava levando muito tempo e cada vez mais precisávamos de um servidor melhor para aguentar a execução de todos os testes de forma automatizada. O prazo de entrega foi ficando mais apertado e vimos que não dava mais para criar os testes [...] Como o tempo foi passando, nós acabamos deixando de lado e os evangelistas de TDD e testes foram saindo do projeto, já que tinham os maiores salários e o projeto estava cada vez com o orçamento mais comprometido.*

5.3.2 Análise de eficiência

A atividade de análise de eficiência consiste em realizar medições sobre o desempenho do sistema *antes* da realização da atividade da refatoração e *após* a execução desta atividade. Ela tem como objetivo identificar, analisar e verificar o comportamento do sistema após a aplicação da refatoração, sendo realizada antes da refatoração e após a refatoração e a tarefa de teste. Esta atividade é realizada pelas empresas III e V, conforme o trecho abaixo retirado da entrevista com o informante 4 da empresa III, na primeira citação ratificado pelo informante 2 da empresa V na segunda citação:

> *O líder técnico ou o arquiteto inicia a tarefa de verificação de eficiência do sistema executando uma ferramenta de medição na versão imediatamente anterior ao refactoring e outra medição na versão imediatamente após o refactoring para encontrar os dados para medição [...] Depois da identificação dos dados e das variáveis ele executa uma ferramenta que nós desenvolvemos para auxiliar na análise dos dados, ela calcula os indicadores de desempenho conforme a configuração feita e exibe em forma de gráficos ou texto [...] Com isso ele pode avaliar se precisamos alterar algo para melhorar a performance ou não.*

Em ambas as empresas esta atividade é realizada por dois papéis: o líder técnico ou o arquiteto de software. É possível perceber também que esta tarefa é realizada em duas etapas: a etapa de identificação das variáveis, dados de entradas para o calculo e análise da eficiência do sistema; e a etapa de aprovação ou não aprovação da solução proposta. Sendo a primeira etapa subdividas na atividade de planejamento, onde são definidas as variáveis e quais serão os resultados esperados e a etapa de execução onde é realizada a análise do sistema.

A motivação para execução da análise de eficiência do sistema existe, pois o tempo de resposta para determinadas funcionalidades deve ser rápido, uma vez que isto é um requisito obrigatório dos sistemas desenvolvidos pelas empresas. O trecho abaixo retirado da entrevista com o informante 4 da empresa III evidencia este fato, comum a empresa V.

> *Um dos requisitos não funcionais que o cliente mais se preocupa é o tempo de resposta para algumas funcionalidades do sistema. Como o negócio tratado pelo sistema exige essa rapidez, que seja gasto o menor tempo possível no sistema, nós temos uma preocupação especial com este requisito [...] A execução da tarefa de analise é uma garantia que nós temos para mostrar para o cliente que o sistema está com qualidade e de forma eficiente.*

Vale ressaltar que a atividade de análise de eficiência possui a mesma frequência em ambas as empresas citadas, sendo sempre executada após a atividade de testes da aplicação, realizando as coletas antes e após a execução da refatoração. O informante 4 da empresa III em sua entrevista evidencia a frequência com que esta tarefa é executada, vide trecho abaixo.

> *A pré-condição da atividade de analise de eficiência do sistema é a atividade de refatoração, já que é na refatoração que desejamos mostrar ao cliente que o houve uma melhora na qualidade interna do sistema sem alterar o tempo de execução. [...] Sempre depois de fazer as refatorações tem a atividade de analise de eficiência sempre sendo executadas uma após a outra.*

5.3.3 Geração de *release*

A atividade de geração de *release* (ver seção 2.1.2.3) representa a finalização do processo de refatoração, bem como é a ultima atividade da etapa de pós refatoração. Nesta tarefa é realizada a geração de um release estável do sistema, sendo também um marco de projeto em todas as empresas estudadas. De forma similar, as empresas estudadas realizam a geração de um release estável, normalmente que representará parte de uma entrega do sistema ao cliente. O trecho abaixo retirado da entrevista com o informante 2 da empresa II melhor exemplifica o que representa esta atividade é como a mesma é realizada.

> *A última atividade de todo o processo é a geração da release e entrega do sistema para a equipe de homologação, que vai iniciar a entrega real ao cliente e auxiliar em por o sistema em produção. [...] Nessa atividade é formada uma baseline do sistema com a versão mais estável. [...] Essa versão possui todo o sistema anteriormente implementado, as alterações da refatoração para melhoria da qualidade interna do sistema e as possíveis modificações encontradas pelos testes de aceitação. [...]*

Os papéis envolvidos nesta atividade são: o gerente de configuração e o líder técnico do módulo alvo da *sprint*. Em conjunto são indicados quais os artefatos que irão compor a *baseline* do sistema e os arquivos de configuração a fim de gerar a release completa para o ambiente de homologação com o cliente. Em ocasiões particulares, quando o líder técnico não pode auxiliar nesta geração, a mesma é realizada pelo *Scrum master* da equipe. O trecho abaixo melhor evidencia o citado fato, sendo retirado da entrevista com o informante 1 da empresa IV.

> *Normalmente o líder técnico da sprint junto com o gerente de configuração definem quais vão ser os artefatos da baseline passados para a equipe de homologação [...] Às vezes, quando o líder técnico já está envolvido em outra atividade nosso Scrum master auxilia na geração da baseline, mas normalmente quem faz essa atividade é o líder técnico mesmo*

> *[...] depois de definir os artefatos da baseline e as configurações do sistema para o ambiente de homologação, a release é gerada e instalada no ambiente para testes e apresentação para o cliente.*

A frequência que a atividade de refatoração ou de geração de release é realizada e a pré-condição para sua execução variam de forma similar entre as empresas I, II e IV. Todas executam esta tarefa após a atividade de testes na aplicação. Já as empresas III e V realizam após a atividade de análise de eficiência. A primeira citação, retirada da entrevista com o informante 1 da empresa IV, exemplifica o citado fato para as empresas I, II e IV e o segundo trecho, retirado da entrevista com o informante 2 da empresa V, evidencia o citado fato para as empresas III e V.

> *[...] Essa tarefa {Geração de release} acontece logo após a atividade de teste do sistema, nela o gerente de configuração gera todo o sistema para o ambiente de testes e homologação.*

> *[...] A última etapa, de geração do release é realizada após a aprovação da refatoração com os testes de eficiência, o que nos permite gerar a release para a equipe de homologação.*

Vale ressaltar que todas as empresas realizam esta tarefa de forma similar variando apenas a ferramenta para geração da *baseline* e do *release*. Outro fato comum a todas as empresas estudadas é a motivação para execução desta tarefa, que visa gerar uma release estável e com novas funcionalidades que agreguem valor de negócio ao sistema produzido.

Como citado anteriormente esta é a ultima atividade da visão da pós refatoração, o que conclui os resultados para esta última visão. O próximo capítulo contempla as discussões acerca dos resultados deste estudo através da comparação dos mesmos com trabalhos encontrados na literatura.

6 Discussão

Este trabalho tem como objetivo principal estudar como a atividade de refatoração é realizada em organizações que utilizam metodologias ágeis, focando principalmente nas mudanças em relação à forma como a atividade é executada nas metodologias tradicionais. Após a realização de um estudo qualitativo acerca do tema foram apresentados resultados parciais (apresentados no capítulo 5). Neste capítulo esses resultados serão discutidos e confrontados com algumas referências bibliográficas conceituadas da área. Para manter o padrão, organização e visando obter uma melhor compreensão dos resultados encontrados e da discussão, este capítulo será divido da seguinte forma: os subcapítulos representarão as mesmas três etapas do estudo citadas no capítulo 5; cada subcapítulo agrupará um conjunto de resultados relacionados a ele, seguido da discussão e comparação com as referências bibliografias quando possível, apontando pontos positivos e negativos da aplicação de determinadas práticas ou falta de aplicação das mesmas.

6.1 Pré Refatoração

6.1.1 Planejamento da Refatoração

A aplicação desta prática sugere que sua execução proporcionou uma diminuição do retrabalho que era ocasionado pela execução de refatorações que não produziam o efeito desejado, realizadas de forma incorreta ou por conflitos gerados por alterações realizadas em paralelo sobre o mesmo artefato. Outro ponto positivo sugerido pela aplicação desta prática é a disseminação de conhecimento sobre a atividade de inspeção de código entre a equipe de desenvolvimento, que antes da aplicação desta prática não

era realizada em nenhuma das empresas estudadas. No entanto, alguns estudos revelam que a atividade de inspeção de código manual é uma forma precária de identificação e propensa a erros, já que pode variar conforme a experiência do desenvolvedor que a realiza (Campbell e Miller, 2008). É importante apontar que apenas uma das organizações realiza a atividade de identificação da necessidade de refatorar de forma automatizada, o que pode diminuir os prejuízos conforme citado por Campbell e Miller (2008).

Outro problema identificado na etapa de planejamento da refatoração é a ausência da utilização de técnicas para organização da execução das atividades de refatoração, principalmente relacionadas a interdependência entre elas (Liu, Li, *et al.*, 2007), fato este não citado por nenhum entrevistado nas organizações estudadas. Segundo Liu e colegas (2007) realizar várias refatorações sem uma organização (ou ordem) prévia pode ocasionar conflitos entre elas, visto que a execução de uma refatoração pode ocasionar erro em outra, ou reduzir seus benefícios. Este problema também é tratado nos trabalhos de Fowler (2007) e (2004).

Para evitar este problema Liu e colegas (2007) e Mens e Runge (2004) propõem frameworks para organização das tarefas de refatoração. A falta da utilização destas práticas tem impacto na tarefa de execução da refatoração já que pode ocorrer de refatorações serem realizadas na ordem incorreta, gerando retrabalho. Lippert e Roock (2006) afirmam que atividades de refatoração complexas devem ser realizadas a partir de um processo de desenvolvimento evolutivo, sendo necessária a realização de análise e detalhamento de como estas atividades estão organizadas, a fim de servirem como um guia de passo a passo contendo a ordem entre as atividades. Também em seu trabalho são relatadas ferramentas integradas ou não a IDE, para o auxílio desta tarefa, porém a utilização desta prática ou destas ferramentas são relatadas como pouco utilizadas, o que pode ocasionar problemas na refatoração, como: inserção de *bugs*, refatorações sem efeito, entre outros.

Além da disseminação de conhecimento na área de inspeção de código também é sugerido que a atividade de planejamento da refatoração dissemina conhecimento sobre a arquitetura do software que está sendo produzida, sobre aplicações de padrão de projetos (Fowler, 1999) (Gamma, Helm, *et al.*, 2000) (Lippert e Roock, 2006),

linguagem de programação utilizada, entre outros, uma vez que no planejamento das refatorações é realizada a tarefa de análise e definição de como será realizada esta refatoração.

O planejamento proposto por Lippert e Roock (2006) é indicado apenas para refatorações de grande porte, que tenham as seguintes características: o período para execução da refatoração ultrapasse um dia de trabalho, afetam partes centrais da arquitetura do projeto, são compostas por refatorações do tipo segura[5] (*safe refactoring*) e não seguras[6] (do inglês *unsafe refactoring*). Além da atividade de identificação da refatoração, também é proposta a realização da atividade de análise de como a refatoração deve ser executada e caso necessário, deve ser feito o desmembramento em refatorações menores. Beck e Fowler (2001) também sugerem que a execução da atividade de planejamento ocorra da mesma forma citada acima.

A prática de planejamento da refatoração é utilizada por quase todas as empresas estudadas (exceto a empresa 1) e é apontada como responsável por inúmeros benefícios à equipe e ao projeto. Segundo vários entrevistados afirmam, esta prática é realizada quando se têm refatorações de média e alta complexidade, pelas seguintes motivações:

- Proporciona uma melhor integração entre a equipe, uma vez que todos participam da discussão e têm suas soluções comentadas por toda equipe, melhorando o conhecimento de todos;
- Agrega o sentimento de código coletivo, uma vez que toda equipe de desenvolvimento está envolvida no planejamento da arquitetura futura;
- Melhora a arquitetura do sistema, uma vez que cada integrante da equipe possui conhecimentos diversos, que são complementados na etapa de discussão, gerando com isso incrementos na arquitetura;

[5]São refatorações realizadas através de um passo a passo (Também denominadas de refatorações mecânicas (Fowler, 1999), que podem ser executadas por uma IDE), onde sua execução não traz risco ao projeto, não irão inserir erros ou inserir comportamentos adversos no sistema (Lippert e Roock, 2006).

[6] São refatorações que não podem ser realizadas através de um passo a passo (Também denominadas de refatorações não mecânicas (Fowler, 1999), que não podem ser executadas por uma IDE), ou seja, sua execução é variada de acordo com as propriedades da arquitetura do sistema. Este tipo de refatoração podem inserir erros no sistema e trazem riscos inerentes a sua execução.

- Disseminação do conhecimento sobre a arquitetura do sistema e sobre os frameworks utilizados ou a linguagem de programação, dentre outros benefícios.

Vale ressaltar o enfoque nesta atividade como colaborativa e que deve ser realizada pelo time de desenvolvimento visando o compartilhamento do conhecimento entre todos os envolvidos, evidenciando assim uma premissa dos princípios ágeis, como por exemplo, as decisões realizadas de forma compartilhada pelo time. O estudo realizado por Diana, Kon e Gerosa (2010), relata resultados positivos acerca da condução de uma arquitetura definida em grupo e ressalta a importância da definição desta forma em organizações que utilizam metodologias ágeis, corroborando com os resultados encontrados neste estudo.

Lippert e Roock (2006) propõem que a atividade de planejamento seja realizada da seguinte forma: (I) discussão sobre como a refatoração será realizada entre os arquitetos e os desenvolvedores envolvidos na tarefa; (II) divisão de refatorações grandes em refatorações menores entre o time de desenvolvimento; (III) geração de um *flipchart* contendo o diagrama UML da nova arquitetura e em quais módulos ocorrerão as modificações; (IV) exposição deste *flipchart* próximo do ambiente onde a equipe de desenvolvimento trabalhe, para observação durante a execução da refatoração. Nas empresas estudadas a execução destas atividades ocorre da seguinte forma: discussão sobre como a refatoração será realizada, geração de um guia sobre como a refatoração será executada, de forma textual, descrevendo o passo a passo para sua execução da refatoração e disponibilização deste guia em um repositório para posterior consulta pela equipe de desenvolvimento.

Algumas das organizações não utilizam a prática de planejamento da atividade de refatoração, ou utilizam de forma esporádica, o que ocasiona problemas para execução de outras atividades. A falha na execução desta atividade, ou não execução, acarreta problemas que já são conhecidos pelas organizações que não a utiliza, por exemplo: conflitos entre refatorações realizadas ao mesmo tempo e por pessoas diferentes sobre o mesmo código fonte. Porém, dentre as práticas que as organizações gostariam de aplicar, a prática de planejamento é a mais citada.

Outro ponto relevante a ser discutido é que apenas uma das empresas estudada realiza a identificação da necessidade da refatorar de forma automatizada, mesmo havendo um grande esforço da academia e da indústria em produzir ferramentas para apoio à identificação da necessidade de refatorações do tipo mecânicas (Fowler, 1999). Ferramentas[7] como: ClassCycle (ClassCycle, 2003), Code-Smells (Code-Smells, 2002), Dr. Freud (Dr.Freud, 1997), ecplise Metrics Pluging (EclipseMetricsPlugin, 2002), Jdepend (JDepend, 1999), JDepend4Eclipse (JDepend4Eclipse, 2004), Mock Object (MockObject, 2002), PMD (PMD, 2002), RefactorIT[8] (RefactorIT, 2002) (RefactorIT, 2008), SonarJ (SonarJ, 2005), SA4J (Sa4j, 2004), sotoGraph (Sotograph, 2006) e Frameworks publicados em trabalhos acadêmicos como: Martin (1994), Bastide (2006), Bayley e Zhu (2007), Bryton e Brito e Abreu (2008), Cinneide (2000), Drozdz, Kourie (2006), Luecke e Ellis (2007), são exemplos de ferramentas/frameworks para auxílio na identificação da necessidade de refatoração, sendo que poucas delas eram conhecidas pelos entrevistados, e apenas uma das empresas estudadas utilizava duas destas ferramentas/frameworks na fase de planejamento da refatoração. Este fato sugere que a indústria está absorvendo de forma muito lenta as ferramentas relacionadas à identificação da necessidade de refatorar. Este ponto pode ser um potencial foco de pesquisa, que tenha como objetivo descobrir o motivo da baixa adoção desse tipo de ferramenta na indústria.

6.1.2 Comunicação entre Equipes

Os estudos realizados em organizações ágeis normalmente possuem um foco na comunicação entre o consumidor do sistema e a equipe produtora, evidenciando os benefícios que esta interação gera na qualidade do produto e na satisfação do cliente (Beck, 2004). Estudos como os realizados por Fowler (2002) e Takats e Brewer (2005) apontam não apenas os benefícios na comunicação entre cliente e unidade produtora do sistema, mas também a importância da comunicação entre as equipes de desenvolvimento. Fowler (2002) e Takats e Brewer (2005) afirmam que a comunicação

[7]As ferramentas ClassCycle, Eclipse Metrics Plugin, JDepend4Eclipse, Code-Smells, Eclipse Metrics Plugin, PMD, RefactorIT, e sotoGraph estão em constante evolução e continuam sendo desenvolvidas e aprimoradas.

[8] Esta ferramenta teve seu projeto iniciado no ano de 2002 porém em 2008 foi reformulada dando origem ao novo projeto.

intraequipe e com clientes deve ocorrer de forma efetiva, onde todos os envolvidos se sintam a vontade para falar e que ocorra de forma periódica. Nesta subseção serão apresentados os resultados deste estudo que corroboram com estas afirmações, bem como outros resultados voltados a atividade de comunicação.

Na reunião de comunicação o *Scrum master* (ver seção 2.1.2.2) ou arquiteto de software da equipe que irá realizar a refatoração explica sobre as refatorações que serão realizadas e os impactos que as mesmas irão produzir no sistema aos coordenadores/lideres técnicos de outras equipes e para a equipe gerencial do projeto. Foram encontrados os seguintes benefícios proporcionados por esta atividade: identificação de possíveis alterações no cronograma; verificação da necessidade de realocação de pessoas entre equipes ou de reorganização estratégicas de atividades/módulos a serem desenvolvidas; compartilhamento de conhecimento entre equipes sobre mudanças no design da arquitetura do projeto; apoio da equipe gerencial do projeto na realização da atividade de refatoração devido a compreensão sobre os benefícios que esta atividade irá trazer para o projeto.

Os resultados apontados nos estudos realizados por Hazzan e Dubinsky (2006) e Shatil, Hazzan e Dubinsky (2010) corroboram com os resultados encontrados neste estudo. Ambos os estudos indicam os benefícios proporcionados pela atividade de comunicação efetiva entre equipes/membros do projeto, evidenciando que esta atividade é vital em projetos que utilizam métodos/metodologias ágeis, tendo um efeito direto na qualidade do produto gerado e em aspectos sociais das equipes (Hazzan e Dubinsky, 2006).

Vale ressaltar também que a comunicação não deve se restringir apenas a uma equipe e sim ser realizada de forma efetiva entre todos os envolvidos no projeto. Hazzan e Dubinsky (2006) afirmam que a comunicação tem um efeito benéfico em aspectos sociais dos envolvidos, uma vez que ela proporciona um relacionamento de confiança e relações de trabalhos positivas trazendo, com isso, benefícios ao projeto, como maior compreensão do sistema a ser produzido e maior produtividade.

O estudo realizado por Santos e Goldman (2011) evidencia que possíveis problemas apontados nas metodologias ágeis são resolvidos com a atividade de comunicação entre

equipes, bem como pelo compartilhamento de conhecimento. Em seu estudo a comunicação é relatada como a difusora de conhecimento tácito e explícito entre os envolvidos. Problemas, como pouca documentação escrita/formal do sistema, são minimizados com a comunicação e compartilhamento de informação. Mesmo havendo um alto índice de rotatividade entre os funcionários, o conhecimento se mantém na organização, uma vez que está disseminado entre várias pessoas.

Outro ponto importante a ser observado sob os benefícios da comunicação é apontando por Cohn (2005), onde é afirmado que a atividade de comunicação no projeto tem impacto direto no planejamento de atividades. Uma vez que esta atividade aumenta a percepção dos envolvidos, possibilita identificar possíveis indicadores de atraso no projeto, necessidade de alteração do cronograma de atividades, entre outros. Neste estudo também foi identificado que a comunicação efetiva entre os envolvidos do projeto auxilia na compreensão do sistema, possibilita a identificação da necessidade de ajustar as estimativas e da necessidade rever o planejamento de determinadas atividades, corroborando com o estudo realizado por Cohn (2005).

Estudos como Goto, Rosa e de Souza (2008), de Souza, Treccani, et al. (2009), de Souza (2005) e de Souza, Froehlich e Dourish (2005) evidenciam quão crítico é a necessidade de comunicação durante a atividade de refatoração (mudanças estruturais) no desenvolvimento de software livre. O trabalho de Corbucci e Goldman (2010) aponta semelhanças entre equipes que desenvolvem software livre e ”equipes ágeis”, principalmente em questões que envolvem comunicação e colaboração, como decisões compartilhadas, código coletivos, dentre outras. O trabalho de Souza, Treccani, et al. (2009) apresenta evidências de que a comunicação aumenta significativamente durante a etapa de refatoração. Partindo disto, é possível inferir que a comunicação é uma prática bastante relevante para o bom andamento dessa atividade, o que corrobora os resultados encontrados neste estudo.

Os resultados aqui encontrados também vão ao encontro dos trabalhos realizados por Sosa, Eppinger e amigos (2002), Sosa, Eppinger e Rowles (2003) e (2004). Sosa e amigos em seus estudos encontraram uma correlação entre os componentes dependentes de uma arquitetura de software e a frequência de comunicação entre engenheiros de software ligados a estes componentes, havendo a necessidade de maior comunicação

nesse período. Neste estudo foi possível identificar o aumento do número de atividades relacionadas a comunicação e compartilhamento de conhecimento/informação realizadas durante as atividades de refatoração. Os resultados deste trabalho reforçam os resultados encontrados nos trabalhos de Sosa e amigos.

A não realização da atividade de comunicação, tanto relacionada à atividade de planejamento da refatoração quanto a outras tarefas, tem um impacto negativo para a organização, identificado por este estudo, gerando problemas técnicos como: resolução de conflitos quando mais de um desenvolvedor utiliza o mesmo código, essa atividade normalmente é complexa; inserções de *bugs* no sistema; retrabalho; desperdício de tempo; refatorações sem efeitos; má compreensão da arquitetura do sistema; duplicidade de código, entre outros. Isto ocasiona também problemas sociais como: baixa confiança entre equipes, ilhas de conhecimento, problemas interpessoais, ambiente de trabalho pouco produtivo, entre outros.

Outro ponto negativo encontrado é o baixo apoio de práticas de comunicação por alguns envolvidos no processo de desenvolvimento. Este problema foi apontando por alguns colaboradores em diversas empresas, principalmente relatado nas organizações que possuíam um menor tempo de utilização de métodos/metodologias ágeis. Alguns destes colaboradores (considerados pelos entrevistados, contra métodos/metodologias ágeis) apontavam as práticas de comunicação como cansativas, desnecessárias e de pouco ganho, uma vez que era afirmado que era possível desenvolver apenas com a leitura das histórias ou requisitos e que o sistema possui uma arquitetura de fácil entendimento, sendo estas reuniões desnecessárias.

Outro fator importante é que a academia relata estudos e desenvolvimento de ambientes voltados a melhoria da comunicação e *feedback* para organizações que utilizem ou não metodologias ágeis. O trabalho realizado por Oliveira e Goldman (2011) indica características de ambientes voltados a comunicação, colaboração e *feedbacks* para auxilio da melhoria no processo de desenvolvimento em organizações ágeis. Sendo a utilização deste um potencial fator de melhoria no desenvolvimento do produto.

6.2 Execução da Refatoração Colaborativa

Os resultados encontrados neste estudo relacionados à atividade de execução da refatoração apontam uma forma de realizar a atividade diferente da forma habitual. Corriqueiramente, a atividade de refatoração é realizada de forma individual: apenas uma pessoa realiza todos os passos necessários deste procedimento. Estudos como o de Lippert e Roock (2006), Fowler (1999), Beck e Fowler (2001), Fowler (2002), (Gamma, Helm, *et al.*, 2000) relatam a atividade de refatoração como atividades individuais ou realizadas em par, principalmente pelo papel de arquiteto do sistema. Já neste estudo observou-se que a atividade de refatoração pode ser feita de maneira colaborativa, uma vez que todos os envolvidos em determinado módulo a ser refatorado terão participação, tanto no planejamento, quanto na execução da tarefa, incluindo desenvolvedores experientes ou não, arquitetos e engenheiros de software.

A participação destes papéis, em conjunto com a execução colaborativa da atividade de refatoração, auxilia a manter uma das premissas dos métodos ágeis, o sentimento de código coletivo. O fato de que cada integrante da equipe tem o "sentimento" coletivo sobre o código que está sendo produzido, proporciona uma maior preocupação com a qualidade do produto gerado e pode aumentar a vontade de compreender melhor a arquitetura e participar de sua construção. Este "sentimento" coletivo, além de ser um incentivo a manter a qualidade do produto gerado, pode gerar benefícios no aspecto social da equipe, promovendo um bom relacionamento entre os colaboradores, maior comunicação e maior produtividade dos mesmos, como visto nas entrevistas da seção 5.2.

Alguns resultados deste estudo sugerem que a realização da refatoração colaborativa aumenta a produtividade da equipe que desenvolve o sistema devido o maior conhecimento adquirido. A realização da atividade de refatoração colaborativa também proporciona a disseminação de conhecimento sobre a arquitetura do sistema, boas práticas de programação, padrões de projetos, entre outros conhecimentos técnicos, conforme citado na seção 5.2.

Os resultados desta dissertação mostram que a refatoração ocorre de forma similar a um *Dojo*[9] de programação (Bossavit e Gaillot, 2005). Estudos como os realizados por Bossavit e Gaillot (2005), Bravo e Goldman (2010) e Sato, Corbucci e Bravo (2008) relatam a utilização de *dojos* de programação como forma de dissseminação de conhecimento, transferência de experiencia e nivelamento de colaboradores. Fato este também apontando nos resultados deste estudo.

Além disso, os resultados também identificam as atividades de refatoração e a aplicação de métodos/práticas ágeis como possíveis responsáveis pelo aumento de produtividade no projeto, resultado este semelhante ao estudo de Melo e Kon (2011). De fato, o trabalho de Melo e Kon (2011) apresenta indícios de que a utilização de metodologias ágeis tem efeito direto na produtividade da equipe.

Uma vez havendo a disseminação de conhecimento sobre o software sendo desenvolvido, suas funcionalidades e sua arquitetura, as ilhas de conhecimento[10] tendem a diminuir, tendo impacto positivo na redução de *bugs* na aplicação, visto que os desenvolvedores passam a conhecer melhor a aplicação. A diminuição das ilhas de conhecimento e o compartilhamento de informação/conhecimento entre os integrantes proporciona um grande valor ao sistema, uma vez que este conhecimento pode auxiliar no aumento da qualidade de produto e da produtividade dos desenvolvedores (Melo, Santos Jr, *et al.*, 2010), diminuindo a incidência de *bugs* e retrabalho, entre outros benefícios.

Lippert (2004) e Lippert e Roock (2006) em seus trabalhos relatam práticas, atividades e procedimentos utilizados na atividade de refatoração, bem como as características e pontos positivos e negativos para cada item. Nesta subseção é apresentado parte dos itens relatados por Lippert e Roock (entre outros estudos), bem como uma comparação com os resultados/características encontrados neste estudo e a sugestão de incremento de novas práticas.

[9]É uma atividade que reúne um grupo de programadores para resolver um determinado problema. Nesta reunião todos expõem suas ideias e definem, de forma compartilhada, a melhor forma de solucionar tal desafio. Esta atividade é baseada em uma metodologia própria e em trabalho em grupo.

[10]Ilhas de conhecimentos ocorrem quando apenas um integrante do time, ou uma pequena parte, possua informações ou conhecimentos sobre determinado assunto. Esta segregação de conhecimento em pessoas especificas causam um efeito chamado ilhas de conhecimento.

Lippert e Roock (2006) afirmam que a atividade de execução da refatoração deve ser realizada acompanhada de um guia que descreve as refatorações que devem ser feitas. Esse guia engloba quais atividades devem ser feitas, qual a ordem de execução das atividades, quais os impactos entre as atividades e quais os resultados esperados após a refatoração. Neste estudo foi possível identificar que apenas parte destas informações são descritas no guia informal gerado pelas empresas estudadas, a saber: as refatorações que devem ser realizadas; quais os impactos gerados no código fonte em geral, não em outra atividade de refatoração; e os resultados e benefícios que a refatoração deve prover. A falta de informações como: análise de impacto em outras atividades de refatoração e ordem de execução da refatoração indicam possíveis melhorias a serem implantadas pelas empresas estudadas. Além disso, a falta dessas informações pode causar alguns problemas, como: refatorações com baixo ou sem efeito, inserção de *bugs*, conflitos entre refatorações, entre outros problemas.

Por outro lado, as organizações estudadas acrescentaram outra informação não sugerida por Lippert (2004) e Lippert e Roock (2006): *a forma com que a refatoração deve ser aplicada*. Esta informação auxilia na execução da atividade de refatoração, uma vez que a solução para o problema já foi definida anteriormente. Com o *design* da nova arquitetura contendo as refatorações planejadas, os desenvolvedores podem manter o foco de sua atenção na melhor forma de implementar tal solução. Assim, um dos resultados desta dissertação é a sugestão de que esta informação sobre a forma com que a refatoração deve ser aplicada seja incorporada ao guia proposto por Lippert e Roock (2006), já que a mesma proporciona os benefícios citados.

Outro aspecto apontado como uma boa prática para atividade de refatoração no trabalho de Lippert (2004) e Lippert e Roock (2006) é a análise e categorização das refatorações conforme o grau de risco de sua aplicação. Esta análise é importante, pois alerta quais são as refatorações de alto risco, que devem ser realizadas com mais cautela e cuidado, uma vez que as mesmas podem afetar o sistema de forma transversal, ocasionando problemas em varias áreas do mesmo. Uma observação importante desta dissertação é que esta prática não é realizada por nenhuma das empresas estudadas, sendo este um ponto de melhoria ao processo de desenvolvimento das empresas.

Outra prática relatada por Lippert (2004), Lippert e Roock (2006) e Beck e Fowler (2001) é a divisão de refatorações grandes ou complexas em refatorações menores. Segundo estes autores, uma refatoração é considerada grande ou complexa quando seu tempo de execução excede um dia de trabalho. Os resultados deste estudo corroboram com os resultados sobre a divisão de refatorações complexas em refatorações menores, porém foi possível notar que as organizações não utilizam o fator "tempo de execução" para avaliar as refatorações. Segundo Lippert (2004) e Lippert e Roock (2006) não considerar o tempo de execução da refatoração evidencia uma falha na etapa de planejamento uma vez que a identificação do tempo de execução de uma refatoração é um fator essencial para evitar problemas. Este é um aspecto que, novamente, pode sugerir melhorias aos processos das empresas estudadas.

Lippert (2004) e Lippert e Roock (2006) comentam também que, em relação à divisão de refatorações complexas em refatorações menores, existe a possibilidade de paralelizar a realização das mesmas por vários desenvolvedores executando de forma individual. No entanto, este estudo identificou que as organizações estudadas não são adeptas desta prática, uma vez que a mesma gera a necessidade de realizar o tratamento de conflitos complexos, sendo esta uma atividade considerada desnecessária e que deve ser evitada. Para isso, as organizações sugerem que a equipe realize a refatoração de forma colaborativa. No entanto, uma das organizações estudadas optou por deixar de executar a refatoração colaborativa (embora continuem fazendo o planejamento e definindo as soluções em conjunto), visto que, depois de algum tempo essa atividade deixou de "agregar valor" aos seus empregados, pois os desenvolvedores já possuíam um nível de conhecimento similar entre eles e também porque a atividade gastava muito tempo. Com isso, é possível constatar que em times onde haja desenvolvedores com pouco conhecimento, talvez seja interessante realizar a execução da refatoração de maneira colaborativa, mas para times com mais experiência a atividade pode ser feita individualmente sem grandes perdas.

A definição de marcos para visualização de resultados intermediários na atividade de refatoração é outra prática relatada por Lippert e Roock (2006). Estes marcos auxiliam os desenvolvedores a verificar se as alterações estão provendo os benefícios esperados, além de servirem como pontos de controle, caso ocorra alguma alteração

incorreta ou que gere problemas e também como metas para identificar o andamento da atividade. Esta prática foi identificada apenas nas organizações II, III e V sendo utilizada de forma parcial, pois as empresas apenas realizam as definições de metas para identificar o andamento da atividade, normalmente durante a segunda etapa de planejamento da *sprint*.

A regularidade de execução da atividade de refatoração evita a desestruturação do código, a perda de legibilidade, o aumento do acoplamento, o aumento do número de dependências entre artefatos, entre outros problemas (Fowler, 1999). Lippert (2004) e Lippert e Roock (2006) relatam a importância em executar refatorações no sistema com frequência, evitando os problemas citados acima. Embora algumas das organizações estudadas possuam a tarefa de refatoração no seu processo de desenvolvimento, realizando-a de forma frequente e controlada em marcos pré-definidos, outras organizações realizam a atividade esporadicamente causando problemas, já identificados pelas empresas, na arquitetura do sistema conforme discutido na seção 5.2. É sugerido que esta prática seja incorporada por todas as organizações a fim de manter o código e a arquitetura do sistema estruturados, de fácil entendimento e extensível.

Outra prática relatada por Lippert e Roock (2006) é a realização das seguintes tarefas: análise de impacto das refatorações no sistema e entre as próprias refatorações, além de prototipação e estratégias para reconhecimento de falhas provocadas pela refatoração antecipadamente. Hoffman (2003) cita em seu trabalho algumas ferramentas para o auxilio a identificação de possíveis falhas provocadas pela refatoração. Não foi observada a realização desta prática em nenhuma das empresas estudadas, assim, sugere-se que a mesma seja incorporada por todas as organizações.

Um problema evidenciado neste estudo é a baixa utilização de ferramentas que auxiliem na execução da tarefa de refatoração, mesmo havendo um esforço por parte da academia e da indústria em produzir ferramentas para auxílio da atividade de refatoração. No intuito de identificar quais são estas ferramentas, suas funcionalidades e como foram desenvolvidas, foi realizada uma revisão sistemática, ainda não publicada, acerca de quais ferramentas existem para o auxílio da atividade de refatoração. Foi decidido utilizar uma revisão sistemática, com o objetivo de alcançar um grau de rigor científico, além da necessidade de obter resultados mais confiáveis e da necessidade de

uso de uma metodologia rigorosa e passível de auditagem e repetição (Conte, Travassos e Mendes, 2005). Como resultado desta revisão foram identificadas seis ferramentas que fornecem auxílio ao registro das decisões tomadas durante a execução da refatoração, 28 ferramentas/frameworks voltados ao auxilio da execução de refatorações do tipo mecânica e nenhuma ferramenta para apoio a refatoração de forma colaborativa.

Os resultados desta revisão em conjunto com as entrevistas realizadas neste estudo indicam que as organizações estudadas estão absorvendo de forma muito lenta as ferramentas elaboradas. É possível notar também a ausência de ferramentas que apoiem a atividade de refatoração de forma colaborativa. Com isso, percebe-se que é necessária a realização de uma investigação mais ampla sobre quais os motivos que levam para a falta de produção de ferramentas que auxiliem na refatoração colaborativa, identificar o motivo para a baixa aceitação das ferramentas criadas e quais as iniciativas que devem ser tomadas para aumentar o ritmo de absorção destas ferramentas. Estes são importantes trabalhos futuros resultantes dessa dissertação.

Em resumo, com a realização do estudo foram encontrados os seguintes benefícios sobre a atividade de execução da refatoração de forma colaborativa, sendo eles:

- Auxílio em manter o "sentimento" de código coletivo (ver seção 2.1.3.1);
- Possível aumento da produtividade da equipe, devido a maior compreensão do sistema;
- Compartilhamento de experiências entre os envolvidos;
- Compartilhamento de conhecimento/informações sobre a arquitetura do sistema, a linguagem de programação utilizada, padrões arquiteturais, padrões de projetos, boas práticas de programação entre os desenvolvedores;
- Possível diminuição de retrabalho devido à etapa de planejamento e identificação da refatoração ser realizada anteriormente, direcionando todo empenho dos colaboradores em compreender e executar as modificações planejadas;
- Diminuição de ilhas de conhecimento; e
- Finalmente, diminuição dos índices de *bugs* na aplicação.

Neste estudo foi possível identificar que a não execução da atividade de planejamento tem impacto direto na atividade de execução da refatoração, podendo gerar os seguintes problemas: alterações simultâneas no mesmo trecho de código gerando problemas de integração; aumento no tempo de execução da tarefa de implementação da refatoração; esquecimento da realização de refatorações; e maior inserção de *bugs* na aplicação, devido o baixo conhecimento da arquitetura (ver seção 5.1.1).

Alguns colaboradores, de uma organização, evidenciam outros pontos negativos sobre a aplicação da atividade de execução da refatoração, são elas: despende muito tempo, uma vez que são realizadas explicações sobre diversos aspectos técnicos e perda da efetividade do compartilhamento do conhecimento com o passar do tempo. Estes pontos negativos foram percebidos após a utilização desta prática por alguns anos e a realização de medições para identificar as possíveis causas da diminuição da efetividade desta atividade. A medição realizada relatou que a disseminação do conhecimento não possuía mais o mesmo efeito que no início da realização da atividade, uma vez que todo o projeto é desenvolvido apenas por uma equipe com baixa rotatividade de funcionários. Como citado nos trechos retirados de entrevistas na seção 5.2, uma das organizações retirou está prática, devido estes motivos. Vale ressaltar que apenas uma das empresas estudadas evidencia problemas nesta com o passar do tempo, não havendo um consenso entre a opinião relatada por esta empresa com as demais empresas.

6.3 Pós Refatoração

6.3.1 Teste na Aplicação

Nesta subseção serão apresentados os resultados relacionados à atividade de teste da aplicação, que são realizados após a atividade de refatoração. Primeiramente, é necessário relatar que as empresas estudadas realizam esta atividade de formas diferentes, variando no tipo de teste realizado, na forma de execução e no número de colaboradores que realizam os testes. O critério utilizado para agrupar as empresas foi a forma predominante de realização dos testes, que podem ser de forma automatizada ou manual.

O primeiro grupo de empresas realiza teste de aceitação, teste de caixa preta (realizados de forma manual) e teste de unidade. O segundo grupo realiza teste de regressão, teste de carga, teste de integração, além dos testes realizados pelo primeiro grupo, porém de forma automatizada (exceto o teste de aceitação).

A realização de testes automatizados é vista com um fator positivo por todos os entrevistados deste estudo, inclusive nas organizações que realizam de forma manual. Segundo os entrevistados das empresas onde esta prática é realizada de forma manual, existe ou existiu a iniciativa de realizar estes testes de forma automatizada, sendo conhecidos seus benefícios, havendo também a vontade de realizar outros tipos de testes que facilitariam a avaliação de problemas durante a atividade de refatoração, que são os testes de integração, regressão e de carga, sendo este último o maior insumo para atividade de análise de eficiência da refatoração. Lippert e Roock (2006) relatam os benefícios oriundos da atividade de testes automatizados após a realização da atividade de refatoração.

A atividade de refatoração é uma atividade propensa a erros (Fowler, 1999), sendo necessária a realização de testes após sua execução. Devido a esta premissa é de suma importância que o projeto possua uma grande cobertura de testes de unidade de seu código, a fim de realizar de forma automatizada a verificação de que algum *bug* foi inserido (Lippert e Roock, 2006). A forma automatizada é relatada como a mais eficiente para realizar estes testes na pós refatoração, uma vez que a realização de forma manual varia de acordo com a experiência de quem a realiza, sendo mais propensa a erros (Lippert e Roock, 2006).

Algumas das empresas estudadas realizavam a atividade de testes de forma automatizada no passado, porém foi necessário modificar esta abordagem. Segundo seus informantes, manter profissionais que detém o conhecimento para executar ou criar testes automatizados, dentre outros fatores, tem um impacto financeiro no projeto muito alto. Uma vez que o projeto não tinha como suportar estes colaboradores foi necessário modificar a forma com que os testes eram realizados. A realização desta prática é apontada como aspecto negativo por este estudo e pelos próprios informantes da empresa. Problemas relacionados a não utilização desta prática são relatados nos

trabalhos de Sinha e Harrold (2004) e de Delemaro, Maldonado e Jino (2007), corroborando com os resultados deste estudo.

Estudos como os realizados por Hazzan e Dubinsky (2004) e Beck (2004) relatam os papeis relacionados ao desenvolvimento de software utilizando metodologias ágeis. O papel de analista de teste é responsável pela realização de testes na aplicação, inclusive os testes unitários. Shatil, Hazzan e Dubinsky (2010) realizaram estudos na indústria e descobriram que a construção dos testes unitários é realizada, na maioria das vezes, pelo arquiteto de software ou pelos desenvolvedores e a realização dos testes manuais é feita pelos analistas de teste, evidenciando uma mudança de responsabilidade desta tarefa. Este estudo também evidencia o arquiteto do sistema e, principalmente, os desenvolvedores como os responsáveis pela construção dos testes automatizados para o sistema. É evidenciada também a importância do papel de analista de teste em ambos os estudos na construção e execução de teste de caixa preta e testes de aceitação do sistema.

Shatil, Hazzan e Dubinsky (2010) relatam também a importância da comunicação efetiva entre a equipe de desenvolvimento e a equipe de teste. A comunicação aliada à realização de testes automatizados e não automatizados gera maior compreensão sobre os cenários a serem testados, aumentando a cobertura do teste da aplicação. É relevante observar que as empresas estudadas priorizam a comunicação tanto nas atividades de testes quanto nas demais.

No estudo realizado por Di Bernardo, Sales Jr, *et al.* (2011) e Di Bernardo, Castor e Soares (2011) é relatada a importância da realização de testes no sistema como forma de identificar os *bugs* previamente, promovendo melhoria nas relações de confiança entre a organização produtora do sistema e a organização consumidora. A realização de testes automatizados também é relatada como forma de documentação viva do projeto[11], uma vez que as metodologias ágeis evidenciam formas alternativas à escrita de documentação extensa do projeto.

[11]A construção de testes automatizados possui a descrição de como o sistema deve se comportar, informando as características que determinada funcionalidade deve possuir. Esta informação pode ser considerada uma documentação viva, uma vez que a mesma esta disponível a todos os desenvolvedores, mesmo não sendo uma documentação formal do projeto.

Os testes automatizados devem verificar o comportamento e o negócio do sistema que está sendo produzido e não apenas verificar exceções do sistema, inconsistência de dados, infraestrutura ou problemas da linguagem de programação utilizada (Di Bernardo, Sales Jr, *et al.*, 2011). Ainda neste estudo é relatada uma forma semi-automatizada para realização de testes e monitoramento de exceções do sistema.

Segundo alguns entrevistados a prática da comunicação aliada à realização de testes unitários é a forma de documentação mais importante do projeto. É afirmado que a informação sobre o sistema memorizada pelos colaboradores é muito mais utilizada que a documentação do projeto, tendo um grande benefício para o projeto uma vez que toda a equipe possui o conhecimento geral do sistema que está sendo produzido. Como se pode notar no trecho abaixo retirado da entrevista com o informante 2 da empresa V.

> *Depois que cobrimos todos os módulos do sistema com testes unitários começamos a perceber que quando tínhamos uma duvida, sobre determinada funcionalidade, não procuramos a explicação nas estórias ou nos diagramas e sim no próprio teste unitário com suas regras [...] se não achássemos nada nos testes unitários, procuramos algum amigo que soubesse mais sobre aquela funcionalidade, uma vez que sabíamos quem dominava cada mais sobre aquele assunto pelas reuniões diárias.*

Estes relatos corroboram com o trabalho de Di Bernardo, Sales Jr, *et al.* (2011), fortalecendo a necessidade da realização de estudos para identificar e analisar a correlação entre a atividade de comunicação e a atividade de testes como forma de documentação viva para projetos que utilizem metodologias ágeis.

6.3.2 Análise de Eficiência

A atividade de análise de eficiência é realizada de duas formas distintas pelas organizações estudadas. Como descrito na subseção 6.3.1 as empresas serão agrupadas conforme suas características. O primeiro grupo referencia as organizações que realizam esta atividade de forma *ah doc*, não havendo nenhum tipo de guia ou processo para ajudar nesta atividade. Já o segundo grupo é composto por organizações que realizam esta atividade com base em um processo ou com um guia contento um passo a passo a ser realizado nesta atividade.

Lippert e Roock (2006) afirmam que a atividade de análise de eficiência da refatoração é parte da atividade de teste, sendo esta responsável por avaliar se a refatoração teve o efeito esperado na aplicação. Apesar disso, para facilitar a

compreensão do estudo, foi decidido que esta atividade seria apresentada individualmente, visto que a discussão sobre esta atividade é extensa. Lippert e Roock (2006) também ressaltam a necessidade de planejamento desta atividade, a fim de definir métricas para avaliar a eficiência da refatoração, avaliando quantitativamente os resultados mensuráveis.

O primeiro grupo de empresas realiza a atividade de análise de eficiência da refatoração baseando-se nos resultados obtidos pela atividade de teste, não havendo nenhum formalismo durante a realização atividade, já o segundo grupo realiza esta atividade conforme a proposta de Lippert e Roock (2006). Lippert e Roock (2006) propõem que esta atividade deve possuir: um planejamento, definições de métricas e índices que apontem a real melhoria da aplicação após a refatoração. Neste segundo grupo, quando a refatoração não possui tarefas, como melhoria da compreensão e da legibilidade de código, métricas e índices podem ser definidas a fim de analisar se a atividade de refatoração obteve os resultados desejados. Nesta etapa de planejamento são identificados quais os resultados desejáveis para o sistema após a realização da atividade de refatoração. Após esta definição, ferramentas de medição são utilizadas para avaliar a melhoria de desempenho de determinadas funcionalidades do sistema. Estas ferramentas identificam os índices, ou geram gráficos automatizados, necessários para a realização da análise, sendo o arquiteto do sistema ou o líder técnico o responsável por realizar as medições e análise dos resultados.

Em uma das empresas esta análise é realizada através de medições e definições de metas e resultados para a atividade de refatoração. A motivação para esta análise se dá principalmente porque a eficiência é um requisito primordial ao negócio do cliente, havendo a necessidade da realizar constantes testes de desempenho do sistema. Já nas outras equipes esta atividade é importante por manter a qualidade do produto e avaliar a efetividade das tarefas de refatoração.

Vale ressaltar também como pontos positivos da atividade de verificação e análise da refatoração, são: possível diminuição do tempo de execução da tarefa; a necessidade de utilizar poucas pessoas para realizá-la; e a maior confiança nos resultados obtidos. Porém, também é necessário relatar os problemas enfrentados nesta tarefa, que são: necessidade de profissionais com alto conhecimento na área de medição e teste, o que

normalmente é encarado pelas empresas como gasto que pode ser cortado; a falta do papel de analistas de teste na realização da atividade; e a realização desta atividade de forma *ad hoc* por um grupo de empresas.

6.3.3 Geração de Release

A geração de release é uma atividade realizada corriqueiramente antes de uma entrega do sistema, em organizações ágeis, que incentivam a geração de entregas menores, mostrando a evolução e entregando primeiramente funcionalidades mais importantes ao negocio do cliente. A geração de releases ocorre de forma similar a relatada por Abrahamsson, *et al.* (2002) e Coram e Bohner (2005) sendo executada pelo arquiteto de software ou líder técnico junto com o gerente de configurações.

A primeira etapa da atividade é a definição da *baseline* do projeto, a segunda é a geração do *release notes*, a terceira é o empacotamento do projeto e por último vem a implantação do sistema no ambiente de testes de aceitação ou de homologação.

A tarefa de definição da *baseline* é realizada por todas as empresas estudadas de forma automatizada, tendo os seguintes benefícios em sua execução: o *release notes* serve para formalizar o escopo das modificações realizadas no módulo; a atividade é um marco do projeto, tendo em visa que normalmente antecede uma entrega do sistema ao cliente; a *baseline* provê um ponto estável e uma visão dos artefatos de desenvolvimento naquele momento; a *baseline* fornece também uma forma de retroceder a versão do sistema, caso as mudanças sejam consideras instáveis; além disso, a comparação dos conteúdos de dois *baselines* ajuda na depuração e criação de documentação; dentre outros.

Tabela 2 - Mapeamento entre as atividades realizadas pelas empresas e as indicadas nos principais trabalhos relacionados.

Atividades relacionadas à refatoração	Planejamento da execução	Identificação da necessidade de refatorar	Planejamento em grupo	Comunicação e feedback	Refatoração colaborativa	Utilização de guia	Descrição da forma de aplicação	Divisão das refatorações	Definição de Marcos	Testes na aplicação	Análise de eficiência e resultados
Empresas I	Não	Aleatório	Não	Sim	Sim	Não	Não	Sim	Não	Manual	Manual
Empresas II	Lista/Guia	Inspeção	Sim	Sim	Sim	Parcial	Sim	Sim	Parcial	Teste de Unidade	Manual
Empresas III	Lista/Guia	Inspeção	Sim	Sim	Sim	Parcial	Sim	Sim	Sim	Ferramenta	Ferramenta
Empresas IV	Lista/Guia	Inspeção	Sim	Sim	Sim	Parcial	Sim	Sim	Parcial	Teste de Unidade	Manual
Empresas V	Ferramenta	Ferramenta	Sim	Sim	Sim	Parcial	Sim	Sim	Sim	Ferramenta	Ferramenta
Campbell e Miller (2008)	Ferramenta	Ferramenta	Não	Sim	N/A	Não	N/A	Sim	N/A	N/A	N/A
Fowler (2004)	Lista	Inspeção	N/A	Sim	Não	Não	N/A	N/A	N/A	N/A	N/A
Liu (2007)	Ferramenta	Ferramenta	Não	N/A	Não	N/A	N/A	Sim	N/A	N/A	N/A
Mens e Runge (2004)	Framework	Framework	Sim	N/A	Não	Sim	N/A	Sim	N/A	N/A	N/A
Lippert e Roock (2006)	Guia	Ferramenta/ Inspeção	Sim	Sim	Não	Sim	N/A	Sim	Sim	Ferramenta	Ferramenta
Beck e Fowler (2001)	Guia	Inspeção	Sim	Sim	N/A	Não	N/A	Sim	Sim	N/A	N/A
Abrahamsson, et al. (2002)	N/A	N/A	Sim	Sim	N/A	Não	N/A	N/A	N/A	N/A	N/A

7 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um estudo qualitativo e exploratório em cinco empresas brasileiras sobre a utilização de práticas ágeis, sobretudo a refatoração colaborativa, no processo de desenvolvimento de software. Todas as empresas utilizam metodologias ágeis como base de seu processo de desenvolvimento, produzindo sistemas em diversas áreas de negócio. As empresas também variam no tempo de execução e no grau de conhecimento das metodologias ágeis, na linguagem de programação utilizada, entre outras características.

O principal objetivo deste estudo foi compreender como as metodologias ágeis eram aplicadas nas empresas nacionais de desenvolvimento de software, bem como os efeitos da aplicação das mesmas nas atividades de desenvolvimento de software. Para isto foi necessário utilizar uma metodologia de pesquisa qualitativa. Bandeira-de-Mello e Cunha (2006) relatam que quando existe a necessidade de compreender fenômenos sociais em diversos aspectos ou em aspectos pouco explorados é necessária a utilização de uma metodologia qualitativa.

Após algumas etapas do estudo, foi delimitado como foco principal da pesquisa a atividade de refatoração, que adquire um caráter mais colaborativo no contexto das empresas estudadas. Assim, como resultados do estudo descrito nesta dissertação foram identificadas às mudanças que a aplicação destas metodologias causam nas atividades de refatoração, identificando os pontos positivos e negativos, bem como a descrição de quais práticas tiveram a necessidade de adaptações para serem aplicadas. Este trabalho também realiza uma comparação dos resultados encontrados neste estudo com os trabalhos existentes na literatura. Esta análise aponta os pontos convergentes entre este

trabalho e os estudos existentes, os pontos divergentes e possíveis extensões dos trabalhos da área. A Tabela 2, apresentada na página 107, ilustra um mapeamento entre a forma com que as empresas realizam as atividades e o que os trabalhos relatam sobre elas.

Neste capítulo são apresentadas as contribuições oriundas deste estudo, as limitações identificadas e os trabalhos futuros que poderão ser desenvolvidos através dos resultados obtidos nesta dissertação.

7.1 Principais contribuições

A principal contribuição deste trabalho está no incremento do conhecimento acerca da atividade de refatoração em metodologias ágeis. O trabalho apresenta uma descrição rica em detalhes sobre como as organizações estudadas aplicam a refatoração em seus projetos, bem como os benefícios e problemas da utilização desta abordagem.

Para obter esses resultados foi necessária a realização de entrevistas como forma de coleta de dados e a utilização de métodos de codificação da teoria fundamentada em dados para organizar e melhor compreender os aspectos encontrados neste trabalho. A descrição destes métodos bem como as dificuldades encontradas para executá-los, caracteriza outra contribuição desta dissertação, visto que serve como guia para outros autores que pretendam utilizar metodologias similares.

Através deste trabalho foi possível identificar quais atividades precisaram ser adaptadas para que pudessem efetivamente ser realizadas, bem como quais as modificações que estas atividades sofreram para serem adaptadas ao processo de desenvolvimento. Isto permite que outras organizações possam utilizar este conhecimento de forma a aplicá-lo em seus processos diminuindo assim a chance de falha.

Outro resultado obtido através deste trabalho é a identificação da baixa utilização de ferramentas para auxiliar nas atividades de identificação e execução da refatoração por parte das organizações estudadas. Este fato revela que mesmo havendo um grande esforço nos últimos anos em criar ferramentas e workshops para divulgação destas ferramentas, as mesmas não estão sendo utilizadas nas organizações estudadas. Também

não foi possível identificar nenhuma ferramenta que auxiliasse nas atividades de refatoração *colaborativa*, nem na etapa de planejamento colaborativo, nem na atividade de execução. Este resultado evidencia a necessidade da construção de ferramentas visando auxiliar a atividade de refatoração de forma colaborativa. A falta de ferramentas voltadas ao compartilhamento de informação e *feedbacks*, essencial para varias práticas ágeis inclusive a atividade de refatoração, é relatada também no estudo realizado por Oliveira e Goldman (2011). Dessa forma, este trabalho contribui também sugerindo novas áreas de pesquisa, permitindo que novos trabalhos possam ser realizados através da investigação mais detalhada desses problemas, bem como da proposição de abordagens para minimizar estes problemas.

Os resultados deste trabalho apresentam uma extensão ao estudo realizado por Lippert e Roock (2006) uma vez que a forma com que as atividades de planejamento e execução são executadas em grandes refatorações pode também ser utilizada em refatorações de menor porte, provendo os mesmos benefícios. Existe também uma extensão à forma como a atividade de planejamento é realizada, onde este estudo propõe uma abordagem mais voltada à colaboração e à comunicação. A contribuição dessa extensão consiste no fato de que as empresas podem utilizar os resultados de ambos os trabalhos, implantando assim um processo de refatoração mais aprimorado.

Por fim, estas contribuições foram divulgadas na comunidade acadêmica através da publicação de artigos em conferências: (Treccani e de Souza, 2011), (Treccani e de Souza, 2010) e (Treccani e de Souza, 2010).

7.2 Limitações do trabalho

Este trabalho possui limitações relacionadas à sua natureza qualitativa e exploratória. Uma das restrições está relacionada ao fato de não ser possível realizar a generalização dos resultados, sendo os mesmos restritos ao universo de pesquisa (as empresas estudadas), situação natural na realização de pesquisas qualitativas. Neste tipo de estudo são geradas descrições detalhadas da amostra estudada, contendo suas características e fatores primordiais para esclarecimento e compreensão dos aspectos estudados. São levantadas também hipóteses baseadas nos resultados encontrados que devem ser comprovadas.

Outra limitação está relacionada à utilização da entrevista como forma de coleta dados. A utilização desta técnica relata o ponto de vista do entrevistado, não cabendo o julgamento sobre a informação. Este ato pode levar a perdas de conhecimento caso o entrevistado não relate uma informação relevante para entendimento de seu trabalho por não julgar relevante ou ter pouco conhecimento sobre a mesma.

Outro fator limitador está relacionado à baixa aceitação de pesquisas qualitativas em organizações de desenvolvimento de software brasileiras. Inúmeras solicitações para realização de entrevistas em diversas empresas não foram aceitas, fator este que não permitiu a diversificação da amostra do estudo, nem a cobertura representativa das empresas brasileiras que utilizem métodos/metodologias ágeis.

Conforme exposto na seção 4.2 do texto, este estudo não visava avaliar se as empresas participantes da pesquisa possuíam o "grau de agilidade" que era exposto por seus informantes, isto é, se as empresas que se autodenominavam ágeis efetivamente empregavam metodologias ágeis de forma correta. Atualmente, existe um interesse dos praticantes de métodos ágeis em avaliar o "grau de agilidade" de uma empresa[12], assim, uma limitação e possível extensão deste trabalho seria analisar o "grau de agilidade" de cada empresa estudada possibilitando assim uma melhor categorização dos resultados encontrados.

7.3 Trabalhos futuros

Além das limitações descritas na seção anterior, foi possível identificar vários pontos de pesquisa em potencial em outras áreas. Assim, os seguintes estudos são sugeridos como trabalhos futuros:

7.3.1 Condução de estudos qualitativos e quantitativos

- Realizar entrevistas em outras empresas a fim de confrontar os novos resultados encontrados com os resultados encontrados nesta dissertação;

[12] http://www.pmisp.org.br/noticias/o-conceito-de-agilidade-no-gerenciamento-ágil-de-projetos

- Realizar entrevistas com os profissionais que não puderam ser entrevistados nas empresas deste estudo a fim de remover a limitação do trabalho na questão da amostra de pesquisa;
- Realizar uma observação não participativa (ou participativa) ou até mesmo um estudo etnográfico nas empresas estudadas a fim de identificar e enriquecer os resultados obtidos neste estudo provendo um detalhamento maior sobre esta aplicação;
- Realizar estudos de natureza qualitativa para analisar a aplicação e os efeitos das atividades relacionadas aos protocolos ágeis;
- Realizar estudos de natureza qualitativa para analisar a aplicação e os efeitos das atividades relacionadas a teste de software em metodologias ágeis;
- Realizar estudos qualitativos (sobre aspectos sociais e organizacionais) voltados à identificação do relacionamento da atividade de teste e da atividade de comunicação em organizações ágeis como forma alternativa a escrita de documentação extensa e como forma documentação viva do projeto;
- Realizar estudos de natureza qualitativa para analisar a aplicação e os efeitos das atividades relacionadas a requisitos em metodologias ágeis;
- Realizar análises em artefatos, documentos, e possíveis evidências sobre a aplicação de métodos/metodologias ágeis para complementar o estudo;

7.3.2 Condução de estudos quantitativos

- Realizar a triangulação de informações a fim de obter uma validação das hipóteses geradas neste estudo e possível generalização dos resultados. Este trabalho deve buscar novas entrevistas, em outras organizações provendo uma variabilidade nas informações e a realização de estudos quantitativos para validar as hipóteses geradas neste trabalho;
- Realizar um estudo de natureza quantitativa para analisar qual das abordagens proporciona melhores resultados para atividade de planejamento da refatoração: se a proposta por Lippert e Roock (2006) ou a proposta relatada por este estudo;

7.3.3 Avaliação/construção de ferramentas e frameworks

- Realizar estudos de natureza quantitativa e/ou qualitativa para analisar a hipótese de que a indústria está absorvendo de forma muito lenta as ferramentas/frameworks relacionadas à identificação da necessidade de refatorar sugerida neste estudo;
- Realizar estudos de natureza quantitativa para avaliar a hipótese de baixa aceitação de ferramentas para auxílio da refatoração por organizações produtoras de software no Brasil;
- Realizar estudos para identificar requisitos ferramentas que visam auxiliar a atividade de planejamento e comunicação das refatorações;
- Realizar estudos para identificar requisitos para construções de ferramentas para auxílio da atividade de refatoração colaborativa.

7.3.3.1 Sugestão de Ferramenta para Auxilio à Refatoração Colaborativa

Uma possibilidade é a construção de uma ferramenta de auxilio a refatoração colaborativa. Neste caso, sugere-se que esta ferramenta deva ser uma extensão para um IDE já existente, uma vez que é necessária uma infraestrutura de desenvolvimento para que a tarefa de refatoração seja executada, sendo esta abordagem já provida pelas IDEs existentes no mercado atualmente. Esta infraestrutura tange as atividades de apoio à codificação do sistema, compilação e identificação de erros, apoio a realização de testes automatizados (unidade, integração, etc) e apoio a análise de stress, escalabilidade e performance do sistema.

É recomendado que a ferramenta auxilie o compartilhamento de código fonte com alteração simultânea e compartilhada do código, uma vez que a refatoração pode ser realizada por vários indivíduos. É necessário também que a ferramenta possua um chat para que os envolvidos possam conversar sobre como será desenvolvido determinado aspecto do sistema.

Com o objetivo de apoiar as atividades de planejamento e comunicação da refatoração o sistema deve conter também funcionalidades de armazenamento da lista de tarefas (refatorações) que serão realizadas, bem como sua ordem e suas anotações. A ferramenta também deve auxiliar na comunicação assíncrona da equipe que está

realizando a refatoração, permitindo envio de e-mail sobre determinada parte do código ou outra forma de comunicação similar. Além de apoiar a realização de reuniões, com funcionalidades de anotação, quadro virtual, compartilhamento de áudio e vídeo, bem como arquivos, preferencialmente de forma distribuída para que seja possível a colaboração entre pessoas geograficamente distribuídas.

O sistema deve armazenar as decisões que a equipe tenha tomado sobre determinada alteração ou realizar anotações para possíveis consultas sobre o que motivou determinada mudança. É desejável também manter a rastreabilidade das alterações para auxiliar na correção de bugs e guiar possíveis testes após a conclusão das atividades de refatoração, bem como, possuir funcionalidade de compartilhamento de arquivos, diagramas UML, relatórios e planilhas com edição colaborativa para auxiliar nas etapas de pré-refatoração (exemplos: apoio a reuniões e compartilhamento de informações), refatoração (exemplos: utilização de diagramas UML ou busca de informações referentes à refatoração) e pós-refatoração (exemplos: exibição de relatórios de performance para melhoria do código).

Por fim, este trabalho relatou os resultados encontrados em um estudo qualitativo na área de aplicação de metodologias ágeis, em contextos reais de desenvolvimento, realizado em cinco organizações brasileiras. A principal contribuição está relacionada à atividade de refatoração, que neste cenário se torna mais colaborativa. Neste trabalho apresentou-se como esta atividade é realizada e quais os impactos desta atividade em outras atividades do processo de desenvolvimento de software nas organizações estudadas.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHAMSSON, P. Agile Software Development of Mobile Information Systems. **Advanced Information Systems Engineering, Lecture Notes in Computer Science**, 2007.

ABRAHAMSSON, P. et al. **Agile Software development methods Review and Analysis**. [S.l.]: VIT Publications, 2002.

ANDERSON, D. **Agile management for software engineering, applying the theory and constraints for business results.** [S.l.]: Prentice Hall, 2003.

ATLASTI, 2010. ISSN V10.06. Disponivel em: <www.atlasti.com>.

BANDEIRA-DE-MELLO, R.; CUNHA, C. **Grounded Theory:** Pesquisa Qualitativa em Estudos Organizacionais: Paradigmas, Estratégias e Métodos. São Paulo: Editora Saraiva, 2006.

BASSI FILHO, D. **Experiências com desenvolvimento ágil**. São Paulo: Dissertação de Mestrado Universidade de São Paulo, 2008.

BASTIDE, G. A Refactoring-based Tool for Software Component Adaptation. **Conference on Software Maintenance and Reengineering (CSMR)**, 2006.

BAYLEY, I.; ZHU, H. Formalising Design Patterns in Predicate Logic. **Conference on Software Engineering and Formal Methods**, 2007.

BECK, K. **eXtreme Programming Explained:** Embrace Change. 1. ed. [S.l.]: Addison-Wesley, 2000.

BECK, K. **eXtreme Programming Explained:** Embrace Change. 2. ed. [S.l.]: Addison-Wesley, 2004.

BECK, K.; FOWLER, M. **Planning Extreme Programming**. [S.l.]: Addison-Wesley, 2001.

BOEHM, B.; TURNER, D. Using risk to balance agile and plan-driven methods. **Journal IEEE Computer**, 36, n. 6, 2003.

BOEHM, B.; TURNER, D. **Management challenges to implement agile processes in traditional development organizations**. [S.l.]: IEEE Software, 2005.

BOSSAVIT, L.; GAILLOT, E. The Coder's Dojo - A Different Way to Teach and Learn Programming. **International Conference on Agile Software Development - XP2005**, 2005.

BRAVO, M.; GOLDMAN, A. Reinforcing the Learning of Agile Practices Using Coding Dojos. **Agile Processes in Software Engineering and Extreme Programming - XP2010**, 48, 2010.

BRYTON, S.; BRITO E ABREU, F. Modularity-Oriented Refactoring. **Conference on Software Maintenance and Reengineering(CSMR)**, 2008.

CAMPBELL, M.; MILLER, M. Designing refactoring tools for developers. **Workshop on Refactoring Tools**, Nashville, 2008.

CAVAMURA JÚNIOR, L. **AQUA – Atividades de qualidade no contexto ágil**. [S.l.]: Dissertação de Mestrado UFSCar, 2008.

CINNEIDE, M. Automated Refactoring to Introduce Design Patterns. **International Conference on Software Engineering(ICSE)**, 2000.

CLASSCYCLE, 2003. Disponivel em: <classycle.sourceforge.net/index.html>.

COAD, P.; LUCA, J.; LEFEBVRE, E. **Java Modeling Color with UML:** Enterprise Components and process. [S.l.]: Prentice Hall, 1999.

COCKBURN, A. Selecting a project's Methodology. **IEEE SOFTWARE**, p. 64-71, 2000.

COCKBURN, A. **Crystal Clear:** A Human-Powered Methodology for Small Teams. [S.l.]: Addison-Wesley Professional, 2004.

COCKBURN, A.; WILLIAMS, L. The Costs and Benefits of Pair Programming. **International Conference on Extreme Programming and Flexible Process in Software Engineering**, Cagliari, IT, 2000.

CODE-SMELLS, 2002. Disponivel em: <http://c2.com/cgi/wiki?CodeSmell>.

COHN, M. **Agile Estimating and Planning**. 1. ed. [S.l.]: Prentice Hall, 2005.

CONTE, T.; TRAVASSOS, G.; MENDES, E. Revisão Sistemática sobre Processos de Desenvolvimento para Aplicações Web. **Relatório Técnico ESE/PESC - COPPE/UFRJ**, 2005.

CORAM, M.; BOHNER, S. The impact of agile methods on software project management. **International Conference and Workshops on the Engineering of Computer-Based Systems**, 2005. 363 - 370.

CORBUCCI, H. et al. Genesis and Evolution of the Agile Movement in Brazil – A perspective from the Academia and the Industry. **Simpósio Brasileiro de Engenharia de Software (SBES 2011)**, São Paulo, 2011.

CORBUCCI, H.; GOLDMAN, A. Open Source and Agile Methods: Two Worlds Closer than It Seems. **Agile Processes in Software Engineering and Extreme Programming(XP 2010)**, Trondheim, Norway, 48, 1-4 Junho 2010.

DE DIANA, M.; KON, F.; GEROSA, M. Conducting an Architecture Group in a Multi-team Agile Environment. **Workshop Brasileiro de Métodos Ágeis**, Porto Alegre, 1, 2010. 137-149.

DE SOUZA, C. On the Relationship between Software Dependencies and Coordination: Field Studies and Tool Support. **Tese de Doutorado em Department of Informatics, Donald Bren School of Information and Computer Sciences.**, Irvine, CA, 2005.

DE SOUZA, C. et al. On the effects of Refactoring in the Coordination of Software Development Activities. **EUROPEAN CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK**, Vienna, Austria., 2009.

DE SOUZA, C.; FROEHLICH, J.; DOURISH, P. Seeking the source: software source code as a social and technical artifact. **ACM Conference on Group Work**, Sanibel Island, USA, 2005.

DELEMARO, M.; MALDONADO, J.; JINO, M. **Introdução ao Teste de Software**. [S.l.]: Elsevier, 2007.

DEMARCO, T. **Structured Analysis and**. [S.l.]: Prentice-Hall , 1979.

DEWALT, K. M.; DEWALT, B. R. **Participant observation:** a guide forfieldworkers. Walnut Creek: AltaMira Press, 2002.

DI BERNARDO, R. et al. Agile Testing of Exceptional Behavior. **Simposio Brasileiro de Engenharia Software**, 2011.

DI BERNARDO, R.; CASTOR, F.; SOARES, S. Towards Agile Testing of Exceptional Behavior. **Latin-American Symposium on Dependable Computing Workshops**, 2011. 21-24.

DR.FREUD, 1997. Disponivel em: <http://www.freiheit.com/technologies/download>.

DROZDZ, M. et al. Refactoring Tools and Complementary Techniques. **International Conference on Computer Systems and Applications**, 2006.

DUBINSKY, Y.; HAZZAN, O. Using Metaphors in eXtreme Programming Projects. **international conference on Extreme programming and agile processes in software engineering** , 2003.

DYBÅ, T.; DINGSøYR, T. Empirical studies of agile software development: A systematic review. **Inf. Softw. Technol**, 2008.

ECLIPSEMETRICSPLUGIN, 2002. Disponivel em: <http://sourceforge.net/projects/metrics>.

FORMAN, I. R.; FORMAN,. **Java Reflection in Action (In Action series)**. [S.l.]: Manning Publications, 2004.

FOWLER, M. **Refactoring - Improving the Design of Existing Code**. [S.l.]: ADDISON-WESLEY, 1999.

FOWLER, M. Martin Fowler - WikiWikiWeb. **XP Customer Quotes**, 2002. Disponivel em: <http://martinfowler.com/articles/agileStory.html.>.

FOWLER, M. **Patterns of Enterprise Application Architecture**. [S.l.]: Addison-Wesley Professional, v. 1, 2002.

FOWLER, M. Is Design Dead?, 2004. Disponivel em: <http://martinfowler.com/articles/designDead.html>.

FOWLER, M. Mocks Aren't Stubs, 2007. Disponivel em: <http://martinfowler.com/articles/mocksArentStubs.html>.

GAMMA, E. et al. **Design Patterns:** Elements of Reusable Object-Oriented Software. [S.l.]: Bookman, 2000.

GLASER, B. **Theoretical Sensitivity:** Advances in the Methodology of Grounded Theory. Mill Valley: The Sociology Press, 1978.

GLASER, B.; STRAUSS, A. **The discovery of grounded Theory**. Aldine de Gruyter: [s.n.], 1967.

GOLDMAN, A.; KATAYAMA, E. Retrato da comunidade acadêmica de métodos ágeis no Brasil, 2010, 2010. Disponivel em: <http://www.agilcoop.org.br/files/pesquisadores.de_.metodos.ageis_.pdf>.

GOTO, C.; ROSA, M.; DE SOUZA, C. Um Estudo Exploratório sobre os Efeitos da Refatoração na Coordenação das Atividades de Desenvolvimento de Software Livre. **Workshop de Manutenção de Software Moderna**, Florianópolis, Brasil., 2008. 1-8.

HAZZAN, O.; DUBINSKY, Y. Teaching a Software Development Methodology: The Case of Extreme Programming. **Software Engineering Education and Training**, 2003. 176 - 184.

HAZZAN, O.; DUBINSKY, Y. Roles in Agile Software Development Teams. **Extreme Programming and Agile Processes in Software Engineering**, 2004. 157-165.

HAZZAN, O.; DUBINSKY, Y. Empower Gender Diversity with Agile Software Development. In: TRAUTH, E. M. **Encyclopedia of Gender and Information Technology**. [S.l.]: Idea Group Publishing, v. 2, 2006. p. 249-236.

HIGHSMITH, J. Messy, exciting and anxiety-ridden: Adaptative software development. **American Programmer**, X, n. 1, 1997.

HIGHSMITH, J. **Adaptative software development:** A Collaborative Approach to managing complex systems. [S.l.]: Dorset House, 1999.

HIGHSMITH, J. **Agile Software Development EcoSystems**. 2. ed. [S.l.]: Addison Wesley, 2002. 448 p.

HOFFMAN, M. Automated impact analysis of object-oriented software system. **OOPSLA**, 2003.

IKONEN, M. et al. **Exploring the Sources of Waste in Kanban Software Development Projects**. [S.l.]: SEAA, Euromicro, 2010.

JDEPEND, 1999. Disponivel em: <http://clarkware.com/software?JDepend.html>.

JDEPEND4ECLIPSE, 2004. Disponivel em: <http://andrei.gmxhome.de/jdepend4eclipse>.

JEFFERY, D.; BANNERMAN, P.; HOSSAIN, E. Scrum Practices in Global Software Development: A Research Framework. **Product-Focused Software Process Improvement, Lecture Notes in Computer Science**, 2011. 88-102.

KARLSTRÖM, D.; RUNESON, P. Integrating agile software development into stage-gate managed product development. **Journal Empirical Software Engineering**, v. 11, n. 2, 2006.

LINDVALL, M. et al. Empirical Findings in Agile Methods Source. **XP/Agile Universe**, 2002.

LIPPERT, M. Towards a proper integration of large refactorings in agile software development. **International Conference on Extreme Programming and Agile Processes**, 2004.

LIPPERT, M.; ROOCK, S. **Refactoring in Large Software Projects:** Performing Complex Restructurings Successfully. [S.l.]: Wiley, 2006.

LIU, H. et al. Scheduling of conflicting refactorings to promote quality improvement. **International Conference on Automated Software Engineering**, Atlanta, 2007.

LUECKE, K.; ELLIS, B. SOFTWARE CODE BASE CONVERSIONS. **Digital Avionics Systems Conference(DASC)**, 2007.

MANIFESTO for Agile Software Development. **http:** //agilemanifesto.org/, 2001.

MARTIN, A.; BIDDLE, R.; NOBLE, J. The XP Customer Practices: A Grounded Theory. **Agile Conference**, 2009a.

MARTIN, A.; BIDDLE, R.; NOBLE, J. The XP Customer Team: A Grounded Theory. **Agile Conference**, 2009b.

MARTIN, R. OO Design Quality Metrics – An Analysis of Dependencies. **Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)**, 1994.

MAXQDA2, 1995. ISSN V2. Disponivel em: <http://www.maxqda.com/>.

MELO, C. et al. Um estudo exploratório dos fatores associados ao estímulo do aprendizado em times ágeis na indústria. **Experimental Software Engineering Latin American Workshop**, Goiânia, Go, 2010.

MELO, C.; FERREIRA, G. Adoção de métodos ágeis em uma Instituição Pública de grande porte - um estudo de caso. **Workshop Brasileiro de Métodos Ágeis**, Porto Alegre, RS, 2010.

MELO, C.; KON, F. Empirical evaluation of agile practices impact on team productivity. **International Conference on Agile Software Development - XP11**, 2011.

MENS, T.; RUNGE, G. Detecting Structural Refactoring Conflicts. **SETra 2004 Preliminary Version**, 2004.

MILLER, L. Agile user experience SIG. **CHI** , Boston, USA, 2009.

MINAYO, M. C. D. S. O desafio do conhecimento, São Paulo, 1993.

MINAYO, M. C. D. S. **Pesquisa Social:** Teoria, Método e Criatividade. 6. ed. Petrópoli: Editora Vozes, 1996.

MOCKOBJECT, 2002. Disponivel em: <http://c2.com/cgi/wiki?mockObject>.

NADAE, J.; OLIVERIRA, J. A.; OLIVEIRA, O. J. UM ESTUDO DE CASO SOBRE A ADOÇÃO DOS PROGRAMAS E. **REVISTA CIENTÍFICA ELETRÔNICA DE ADMINISTRAÇÃO**, Sao Paulo, Junho 2009.

NONAKA, I.; TAKEUCHI, H. The new new product development game, p. 137-146, 1986.

OLIVEIRA, R.; GOLDMAN, A. How to build an informative workspace? an experience using data collection and feedback. **International Conference on Agile Software Development**, 2011.

PALMER, S.; FELSING, J. **A Practical Guide to Feature-Driven Development**. [S.l.]: Prentice Hall, 2002.

PARNAS, D. On the Criteria to be used in decomposing systems into modules. **Communications of the ACM**, 1972.

PATTON, M. Q. **Qualitative evaluation and research methods**. 2. ed. [S.l.]: Sage Publications, 1990. 532 p.

PIKKARAINEN, M. et al. The impact of agile practices on communication in software development. **Empirical Software Engineering**, 2008.

PMD, 2002. Disponivel em: <http://pmd.sourceforge.net>.

PMI, P. M. I. **A Guide To The Project Management Body Of Knowledge**. 3ª Edição. ed. [S.l.]: [s.n.], 2003.

POPPENDIECK, T.; POPPENDIECK, M. **Lean Software Development:** Na Agile Toolkit. [S.l.]: Addison-Wesley, 2003.

PRESSMAN, R. **Software Engineering:** a Practitioner's Approach. 6 Edição. ed. [S.l.]: McGraw-hill, 2005.

REFACTORIT, 2002. Disponivel em: <http://www.refactorit.com>.

REFACTORIT, 2008. Disponivel em: <sourceforge.net/projects/refactorit>.

ROOIJEN, H. Agile software development with scrum: Scrum faq by ken schwaber, 2006. http://xps-project.googlecode.com/svn-history/r6/trunk/scrum/scrum.pdf.

SA4J, 2004. Disponivel em: <http://www.alphaworks.ibm.com/tech/sa4j>.

SALO, O. Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies. **Euromicro Conference 2004**, Rennes, France, 2004. 310 - 317.

SANDELOWSKI, M.; BARROSO, J. Reading qualitative studies. International Journal of Qualitative Methods, 2002. http://www.ualberta.ca/~iiqm/backissues/1_1Final/html/sandeleng.html.

SANTOS, V.; GOLDMAN, A. An Approach on Applying Organizational Learning in Agile Software Organizations. **Agile Processes in Software Engineering and Extreme Programming**, 2011.

SATO, D. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. São Paulo: Dissertação de Mestrado Universidade de São Paulo, 2007.

SATO, D.; CORBUCCI, H.; BRAVO, M. Coding Dojo: An Environment for Learning and Sharing Agile Practices. **International Conference on Agile Software Development**, 2008. 459-464.

SCHWABER, K. The Scrum Development Process. **Workshop on BusinessObject Design and Implementation**, 1995. 23.

SCHWABER, K. **Agile Project Management with Scrum**. [S.l.]: Microsoft Press, 2004. 192 p.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**. [S.l.]: Prentice Hall, 2001. 158 p.

SEIYOUNG, L.; HWAN-SEUNG, Y. Distributed agile: project management in a global environment. **Empirical Software Engineering**, 2010.

SHARP, H.; ROBINSON, H. An Ethnographic Study of XP Practice. **Empirical Software Engineering**, p. 353–375, 2004.

SHATIL, A.; HAZZAN, O.; DUBINSKY, Y. Agility in a Large-Scale System Engineering Project: A Case-Study of an Advanced Communication System Project. **International Conference on Software Science, Technology & Engineering**, 2010.

SILVA, A. **Reflexões sobre o ensino de metodologias ágeis na academia, na indústria e no governo**. São Paulo: Dissertação de Mestrado Universidade de São Paulo, 2007.

SILVA, A.; KON, F.; TORTELI, C. XP South of the Equator: An eXPerience Implementing XP in Brazil. **International Conference on Agile Software Development**, p. 10-18, 2005.

SILVA, F. S. **Uso de representação de conhecimento para documentação em metodologias ágeis**. [S.l.]: Dissertação de Mestrado PUCRS, 2009.

SINHA, S.; HARROLD, M. Automated support for development, maintenance, and testing in the presence of implicit control flow. **International Conference on Software Engineering**, 2004. 336-345.

SOMMERVILLE, I. **Software Engineering**. 8ª Edição. ed. [S.l.]: Addison-Wesley, 2006.

SONARJ, 2005. Disponivel em: <http://www.hello2morrow.com>.

SOSA, M. et al. Factors that influence technical communication in distributed product development: an empirical study in the telecommunications industry. **IEEE Transactions on Engineering Management**, 1, 2002. 45 - 58.

SOSA, M.; EPPINGER, S.; ROWLES, C. Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions. **ASME Journal of Mechanical Design**, 2003. 240-252.

SOSA, M.; EPPINGER, S.; ROWLES, C. The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. **Management Science**, Maryland, USA, 50, 2004.

SOTOGRAPH, 2006. Disponivel em: <http://www.sotograph.com>.

STAPLETON, J. **DSDM DYNAMIC SYSTEMS DEVELOPMENT METHOD:** THE METHOD IN PRACTICE. [S.l.]: ADDISON WESLEY, 1997.

STRAUSS, A. L. **Qualitative analysis for social scientists**. [S.l.]: Cambridge University Press, 1987. 319 p.

STRAUSS, A.; CORBIN, J. **Pesquisa Qualitativa:** Técnicas e Procedimentos Para o Desenvolvimento de Teoria Fundamentada. 2ª Edição. ed. São Paulo, SP: Artmed/bookman, 2008.

STROGGYLOS, K.; SPINELLIS, D. Refactoring – Does it improve software quality? **International Workshop on Software Quality - WoSQ'07**, 2007.

SUTHERLAND, J. et al. Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams. **Agile 2008**, 2008. 339-344.

SUTHERLAND, J.; SCHOONHEIM, E.; RIJK, E. Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams. **HICSS '09 - HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES**, 2009.

SUTHERLAND, J.; SCHWABER, K. **The Scrum papers:** nuts, bolts, and origin of an agile process. [S.l.]: [s.n.], 2007. http://scrumfoundation.com/.

TAKATS, A.; BREWER, N. Improving Communication between Customers and Developers. **International Conference on Agile Software Development - Agile 2005**, IEEE Computer Society, 2005.

THE CHAOS Report. **Standish Group**, 2009. Disponivel em: <http://www.standishgroup.com/services.php>.

TRECCANI, P.; DE SOUZA, C. R. B. Um Estudo Exploratório sobre Métodos Ágeis Utilizando Grounded Theory. **Workshop de Teses e Dissertações no Simpósio Brasileiro de Sistemas Colaborativos**, Belo Horizonte, 2010.

TRECCANI, P.; DE SOUZA, C. R. B. Utilização de Metodologias Ágeis no Desenvolvimento de Software: Resultados de um Estudo Empírico. **Experimental Software Engineering Latin American Workshop**, Goiânia, 2010.

TRECCANI, P.; DE SOUZA, C. R. B. Collaborative Refactoring: Results of an Empirical Study Using Grounded Theory. **Iternational Workshop on Groupware - CRIWG**, Paraty, Br, 2011. 73–80.

TRIVIÑOS, A. **Introdução à pesquisa em ciências sociais**. São Paulo: Atlas, 1987.

WEISS, R. S. **Learning from Strangers:** The Art and Method of Qualitative Interview Studies. [S.l.]: The Free Press, 1994.

WHITWORTH, E.; BIDDLE, R. Motivation and Cohesion in Agile Teams. **Conference on Agile Processes in Software Engineering and Extreme Programming**, 2007.

ZANATTA, A. L.; VILAIN, P. Uma análise do método ágil Scrum conforme abordagem nas áreas de processo Gerenciamento e Desenvolvimento de Requisitos do CMMI. **Workshop em Engenharia de Requisitos**, p. 12, 2005.

APÊNDICE A – GUIA DE ENTREVISTA

❖ PRIMEIRA VERSÃO DO GUIA DE ENTREVISTA

- **- Perguntas Gerais**
  1. Por favor, qual seu nome completo?
  2. Há quanto tempo você trabalha na empresa?
  3. Qual o cargo que você ocupa na empresa?
  4. Há quanto tempo você ocupa este cargo?
  5. Quais são as atividades que você realiza como <cargo>?
  6. Fale um pouco sobre o projeto em que você está atualmente envolvido?
  7. Quantas pessoas trabalham com você neste projeto? Qual trabalho que elas realizam?

- **Perguntas sobre o projeto principal. <<aqui tens de dizer para a pessoa enfocar apenas no principal projeto em que a pessoa trabalha>>**
  8. Qual (ais) metodologia ágil este projeto utiliza?
  9. Qual (ais) práticas ágil este projeto utiliza?
  10. Qual o tamanho do projeto?
  11. Como está organizado este projeto?(Em módulos, por etapas, por fases?)
  12. Qual é o estado atual do projeto? (em qual fase se encontra o projeto?)
  13. Este projeto utiliza/utilizou algum sistema de gerencia de configuração?
      a) Qual foi utilizado?

b) Existe alguma forma de controle no repositório?(Locks de usuário. Controle de merge...)

14. Este projeto utiliza/utilizou algum sistema de bug tracking ou issue tracking?
    a) Qual foi utilizado?(Mantis, bugzilla, jira, trackStudio)

- **Perguntas sobre Pré- Refatoração**

15. Foi aplicado algum tipo de refatoração neste projeto?
16. Com que frequência ocorrem as refatorações?
17. O que motivou esta refatoração?
18. É utilizada alguma ferramenta para identificar a necessidade de refatoração?
19. Como foi identificada a necessidade de refatorar?
    a) Quando foi identificada?
    b) Quais os papeis envolvidos nesta identificação?
    c) É necessária a autorização de alguém?
20. Foi realizado algum planejamento as atividades de refatoração?
    a) Quais os papeis envolvidos neste planejamento?
    b) Quando ocorre?
    c) É necessária a autorização de alguém?
21. É utilizada alguma ferramenta para auxiliar o planejamento da refatoração?
    a) Qual (ais) são utilizadas?
22. É realizada alguma análise de impactos nos artefatos sob as mudanças provenientes da refatoração?
23. É realizada alguma analise de impactos sobre as pessoas afetas nas mudanças provenientes da refatoração?

- **Perguntas sobre Refatoração**

24. É utilizada alguma ferramenta para execução da refatoração?
    a) Qual (ais) é (são) utilizada(s)?

b) Como esta tarefa ocorre?

25. Quem realiza a refatoração?
    a) Quais os papeis envolvidos na execução?
26. As refatorações são realizadas em paralelo?
27. Quanto tempo em media é gasto refatorando o código?
28. Como é realizada a refatoração?
29. Você já teve algum problema com o seu próprio código enquanto outra pessoa refatorava códigos no projeto?
    a) Com que frequência ocorre estes problemas?
    b) Porque ocorreu?
    c) Quais os envolvidos na correção?
    d) Qual o esforço envolvido?
    e) Quais as soluções?
    f) Quem avalia as soluções?
    g) Quem realiza as soluções?

- **Perguntas sobre Pós- Refatoração**

30. É realizado algum teste no artefato refatorado? (por exemplo, testes de integração ou de regressão)
    a) Qual (ai)?
    b) Qual o papel o realiza?
31. Você já teve algum problema (Conflito) na integração dos artefatos refatorados?
    a) Qual?
    b) Como são resolvidos?
    c) Quem os resolve?
    d) Caso não tenha ocorrido, você tem conhecimento se isso ocorreu com alguém.
32. O que ocorre quando os problemas não podem ser resolvidos?

❖ SEGUNDA VERSÃO DO GUIA DE ENTREVISTA

- **Perguntas Gerais**

1. Por favor, qual seu nome completo?
2. Qual o nome da empresa?
3. Como é o espaço físico da empresa?
4. Quais os cargos e perfis que trabalham na empresa? Quantos funcionários?
5. Há quanto tempo você trabalha na empresa?
6. Qual o cargo que você ocupa na empresa?
7. Há quanto tempo você ocupa este cargo?
8. Quais são as atividades que você realiza como <cargo>?
9. Qual (ais) metodologia ágil a empresa utiliza?
10. Qual (ais) a(s) prática(s) geral (is) que a empresa utiliza?Ou como é o processo da empresa?
11. Qual (ais) prática(s) ágil (ágeis) a empresa utiliza?
12. Fale um pouco sobre o projeto em que você está atualmente envolvido?
13. Quantas pessoas trabalham com você neste projeto? Qual trabalho elas realizam?

- **Perguntas sobre o projeto principal. << focar apenas no principal projeto em que você trabalha>>**

14. Qual (ais) metodologia ágil este projeto utiliza?
15. Como foi a implantação desta metodologia?
16. Houve problemas na implantação? Quais? Como foram resolvidos?
17. Qual (ais) prática(s) ágil é (são) utilizada(s) neste projeto?
18. Existem dificuldades em aplicar tais práticas? Em quais práticas ocorrem estas dificuldades? Como foram resolvidas?
19. Qual o tamanho do projeto?
20. Como está organizado este projeto?(Em módulos, por etapas, por fases?)
21. Qual é o estado atual do projeto? (em qual fase se encontra o projeto?)
22. Este projeto utiliza/utilizou algum sistema de gerencia de configuração?

a) Qual foi utilizado?
b) Existe alguma forma de controle no repositório?(Locks de usuário. Controle de merge...)
c) No que estes sistemas lhe ajudam?

23. Este projeto utiliza/utilizou algum sistema de bug tracking ou issue tracking?
    a) Qual foi utilizado?(Mantis, bugzilla, jira, trackstudio)
    b) No que estes sistemas lhe ajudam?

- **Perguntas sobre Pré- Refatoração**

24. Foi aplicado algum tipo de refatoração neste projeto?
25. Com que frequência ocorre as refatorações?
26. O que motivou esta refatoração?
27. Como foi identificada a necessidade de refatorar?
    d) Quando foi identificada?
    e) Quais os papéis envolvidos nesta identificação?
    f) É necessária a autorização de alguém?
28. É utilizada alguma ferramenta para identificar a necessidade de refatoração?
29. Foi realizado algum planejamento nas atividades de refatoração?
    d) Quais os papéis envolvidos neste planejamento?
    e) Quando ocorre?
    f) É necessária a autorização de alguém?
30. É utilizada alguma ferramenta para auxiliar no planejamento da refatoração?
    a) Qual (ais) é (são) utilizada(s)?
31. Já ocorreu algum problema no planejamento da refatoração? Como foi solucionado?
32. É realizada alguma análise de impactos nos artefatos sob as mudanças provenientes da refatoração?
33. É realizada alguma analise de impactos sobre as pessoas afetas nas mudanças provenientes da refatoração?

- **Perguntas sobre Refatoração**

34. Quem realiza a refatoração?
    a) Quais os papéis envolvidos na execução?
35. Como é realizada a refatoração?
36. É utilizada alguma ferramenta para execução da refatoração?
    a) Qual (ais) é (são) utilizada(s)?
    b) Como esta tarefa ocorre?
37. Existiu algum problema na utilização desta ferramenta?
38. As refatorações são realizadas em paralelo?
39. Quanto tempo em media é gasto refatorando o código?
40. Você já teve algum problema com o seu próprio código enquanto outra pessoa refatorava códigos no projeto?
    a) Com que frequência ocorre estes problemas?
    b) Porque ocorreu?
    c) Quais os envolvidos na correção?
    d) Qual o esforço envolvido?
    e) Quais as soluções?
    f) Quem avalia as soluções?
    g) Quem realiza as soluções?

- **Perguntas sobre Pós- Refatoração**

41. O que acontece depois da refatoração?
42. É realizado algum teste no artefato refatorado? (por exemplo, testes de integração ou de regressão)
    a) Qual (ai)?
    b) Qual o papel o realiza?
43. Você já teve algum problema (Conflito) na integração dos artefatos refatorados?
    a) Qual?
    b) Como são resolvidos?

c) Quem os resolve?
d) Caso não tenha ocorrido, você tem conhecimento se isso ocorreu com alguém.

44. Ocorreu algum problema que não pode ser resolvidos? O que foi feito?

- **Perguntas sobre Requisitos**

45. Como é realizada a coleta de requisitos? Quem faz?
46. Existe algum documento de requisitos? Como ele é elaborado?
47. Como é formado o Product Backlog da empresa? São escritas estórias?
48. Como é realizada a priorização desses requisitos?
49. Existe o Product Owner ou algum papel semelhante na empresa?
50. É utilizado o burndown Chart?

- **Perguntas sobre Comunicação**

51. Quais tipos de reuniões são realizadas?Sprint plan 1e 2? Stand-Up Meeting?
52. É realizado o Sprint Plan ou Release Plan?
53. É realizado reuniões de retrospectiva?
    a) Quando ocorre?
    b) Com que frequência?
    c) Quem participa?
54. São planejadas as reuniões de retrospectiva? Como? Quem realiza?
55. Fale um pouco sobre como é esta reunião?
56. Como são coletadas as lições aprendidas?

57. Na sua perspectiva, alguma pratica deveria ser removida do processo?
58. Quais práticas ágeis vocês gostariam de aplicar, mas não conseguem?
59. Existe alguma pretensão em aplicar outras práticas no futuro? Quais?

❖ TERCEIRA VERSÃO DO GUIA DE ENTREVISTA

- **Perguntas Gerais**

1. Por favor, qual seu nome completo?
2. Qual o nome da empresa?
3. Como é o espaço físico da empresa?
4. Quais os cargos e perfis que trabalham na empresa? Quantos funcionários?
5. Há quanto tempo você trabalha na empresa?
6. Qual o cargo que você ocupa na empresa?
7. Há quanto tempo você ocupa este cargo?
8. Quais são as atividades que você realiza como <cargo>?
9. Em linhas gerais como é o processo da empresa?
10. Qual (ais) metodologia ágil a empresa utiliza?
11. Qual (ais) prática(s) ágil (ágeis) a empresa utiliza?
12. Fale um pouco sobre o projeto em que você está atualmente envolvido?
13. Quantas pessoas trabalham com você neste projeto? Qual trabalho elas realizam?

- **Perguntas sobre o projeto principal. << focar apenas no principal projeto em que você trabalha>>**

14. Como foi a implantação desta metodologia?
15. Houve problemas na implantação? Quais? Como foram resolvidos?
16. Qual (ais) prática(s) ágil é (são) utilizada(s) neste projeto?
17. Existem dificuldades em aplicar tais práticas? Em quais práticas ocorrem estas dificuldades? Como foram resolvidas?
18. Qual o tamanho do projeto?
19. Como está organizado este projeto?(Em módulos, por etapas, por fases?)
20. Qual é o estado atual do projeto? (em qual fase se encontra o projeto?)

- **Perguntas sobre Pré- Refatoração**

21. Foi aplicado algum tipo de refatoração neste projeto?
22. Com que frequência ocorre as refatorações?
23. O que motivou esta refatoração?
24. Como foi identificada a necessidade de refatorar?
    g) Quando foi identificada?
    h) Quais os papéis envolvidos nesta identificação?
    i) É necessária a autorização de alguém?
25. É utilizada alguma ferramenta para identificar a necessidade de refatoração?Porque não é utilizado?
26. É realizado algum planejamento nas atividades de refatoração?
    g) Quais os papéis envolvidos neste planejamento?
    h) Quando ocorre?
    i) É necessária a autorização de alguém?
27. É utilizada alguma ferramenta para auxiliar no planejamento da refatoração?
    b) Qual (ais) é (são) utilizada(s)?
28. Já ocorreu algum problema no planejamento da refatoração? Como foi solucionado?
29. É realizada alguma análise de impactos nos artefatos sob as mudanças provenientes da refatoração?
30. É realizada alguma análise de impactos sobre as pessoas afetas nas mudanças provenientes da refatoração?

- **Perguntas sobre Refatoração**

31. Quem realiza a refatoração?
    b) Quais os papéis envolvidos na execução?
32. A refatoração é realizada de forma individual ou em grupo?
    a) Quando em grupo? Por quê?
    b) Quando é individual? Por quê?
    c) Quais as vantagens de ser em grupo?

d) Quais as vantagens de ser individual?

33. Como é realizada a refatoração?
34. É utilizada alguma ferramenta para execução da refatoração? Porque não é utilizada?
    c) Qual (ais) é (são) utilizada(s)?
    d) Como esta tarefa ocorre?
    e) Em que caso é utilizada?
35. Existiu algum problema na utilização desta ferramenta?
36. Você julga necessária uma ferramenta para auxiliar o desenvolvedor na execução da refatoração?
    a) Em quais casos?
37. As refatorações são realizadas em paralelo?
38. Quanto tempo em media é gasto refatorando o código?
39. Você já teve algum problema com o seu próprio código enquanto outra pessoa refatorava códigos no projeto?
    h) Com que frequência ocorre estes problemas?
    i) Porque ocorreu?
    j) Quais os envolvidos na correção?
    k) Qual o esforço envolvido?
    l) Quais as soluções?
    m) Quem avalia as soluções?
    n) Quem realiza as soluções?
40. Sobre a refatoração colaborativa:
    a) Porque você acha que ela é importante?
    b) Quais as vantagens de utilizá-la?
    c) Quais as desvantagens de utilizá-la?
    d) Em que situação utilizá-la?
    e) Quando ela não é utilizada?
    f) Você identificou algum problema neste tipo de refatoração? Qual? Quando ocorreu?
    g) As decisões sobre a arquitetura tomadas na execução da refatoração

são registras de alguma forma? Qual?

h) Você julga necessária uma ferramenta para auxiliar a refatoração de forma colaborativa?

- **Perguntas sobre Pós- Refatoração**

41. O que acontece depois da refatoração?
42. É realizado algum teste no artefato refatorado? (por exemplo, testes de integração ou de regressão)
    c) Qual (ai)?
    d) Qual o papel o realiza?
43. Você já teve algum problema (Conflito) na integração dos artefatos refatorados?
    e) Qual?
    f) Como são resolvidos?
    g) Quem os resolve?
    h) Caso não tenha ocorrido, você tem conhecimento se isso ocorreu com alguém.
44. Ocorreu algum problema que não pode ser resolvidos? O que foi feito?

- **Perguntas sobre Requisitos**

45. Como é realizada a coleta de requisitos? Quem faz?
46. Existe algum documento de requisitos? Como ele é elaborado?
47. Como é formado o Product Backlog da empresa? São escritas estórias?
48. Como é realizada a priorização desses requisitos?
49. Existe o P.O. ou algum papel semelhante na empresa?
50. É utilizado algum mapa de indicadores como burndown chart, kanban, ou algo do gênero?

- **Perguntas sobre Comunicação**

51. Quais tipos de reuniões são realizadas? Sprint plan 1e 2? Stand-Up

Meeting?

52. É realizado o Sprint Plan ou Release Plan?
53. É realizado reuniões de retrospectiva?
    a) Quando ocorre?
    b) Com que frequência?
    c) Quem participa?
54. São planejadas as reuniões de retrospectiva? Como? Quem realiza?
55. Fale um pouco sobre como são estas reuniões?
56. Como são coletadas as lições aprendidas?

- **Perguntas finais**

57. Na sua perspectiva, alguma pratica deveria ser removida do processo?
58. Quais práticas ágeis vocês gostariam de aplicar, mas não conseguem?
59. Existe alguma pretensão em aplicar outras práticas no futuro? Quais?
60. Você acha que a refatoração em grupo deve ser apoiada por uma ferramenta?
61. Na sua perspectiva, em quais aspectos a refatoração em grupo pode melhorar?
62. A seu ver, quais são os principais benefícios que a refatoração colaborativa trás para equipe? E para o projeto?
63. A seu ver, quais são os pontos negativos que a refatoração colaborativa trás para equipe? E para o projeto?

APÊNDICE B – MAPAS DE CÓDIGOS DA GT

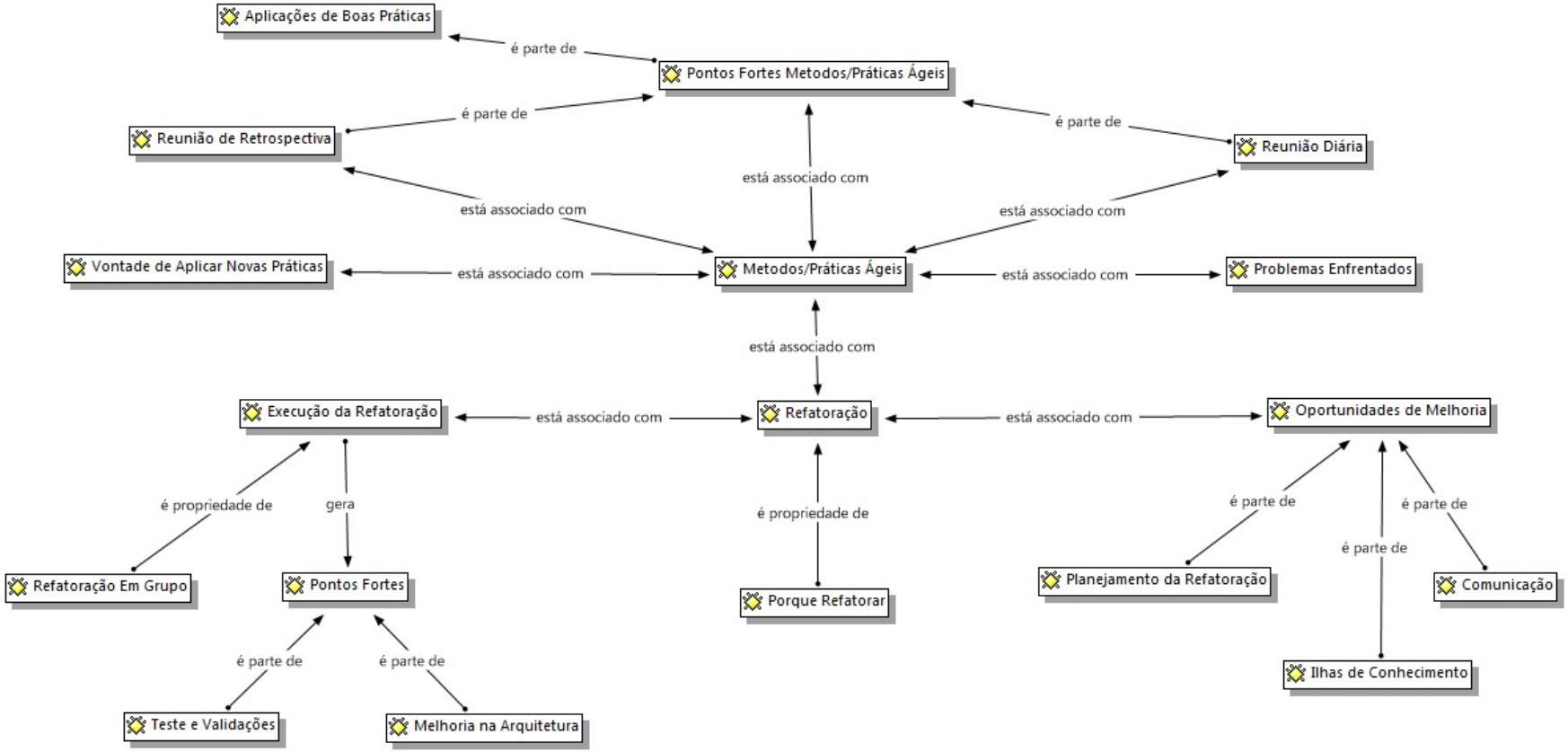

Figura 13 - 1ª versão do mapa de códigos da GT (Todos os Resultados)

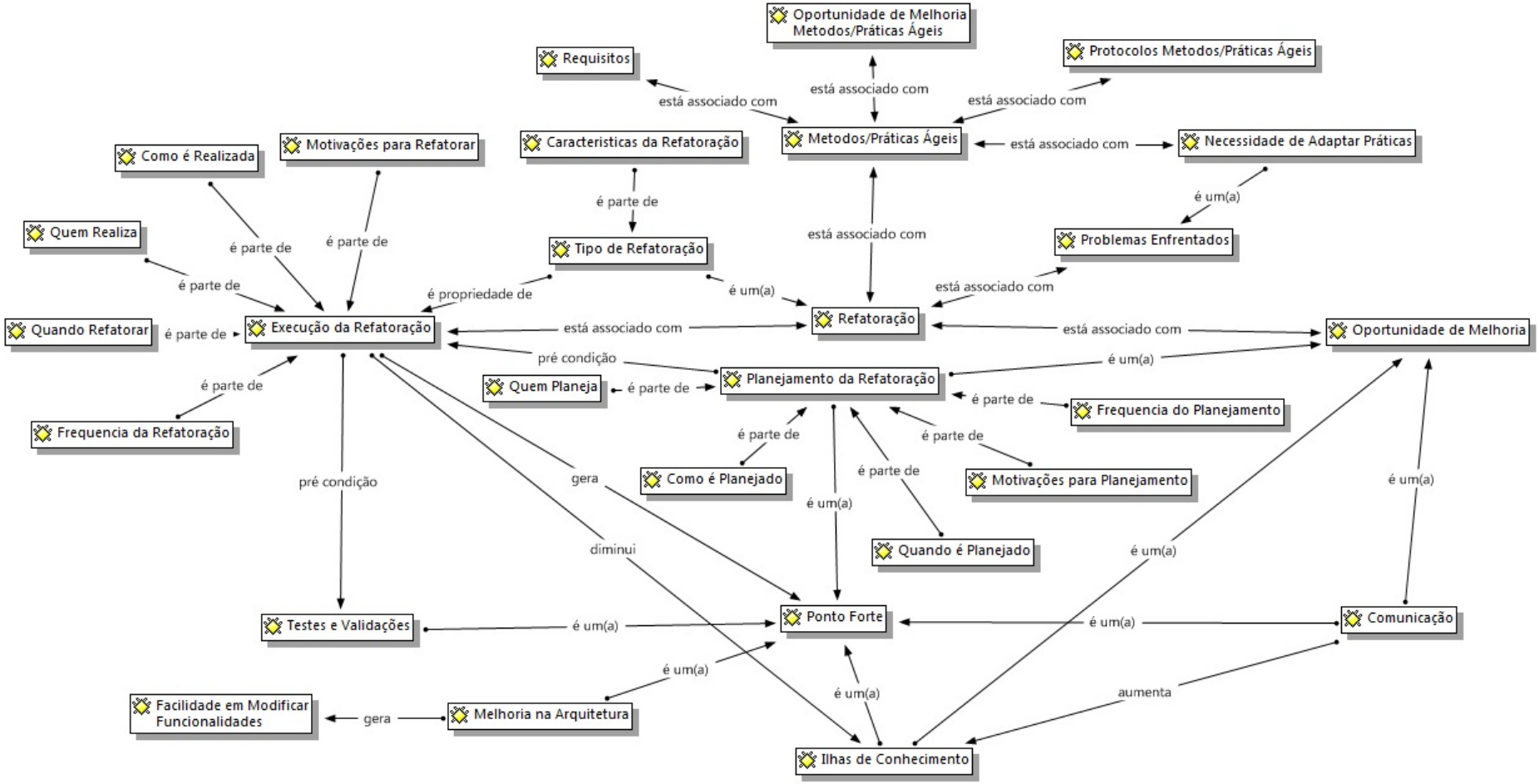

Figura 14 - 2ª versão do mapa de códigos da GT (Todos os Resultados): Visão Geral

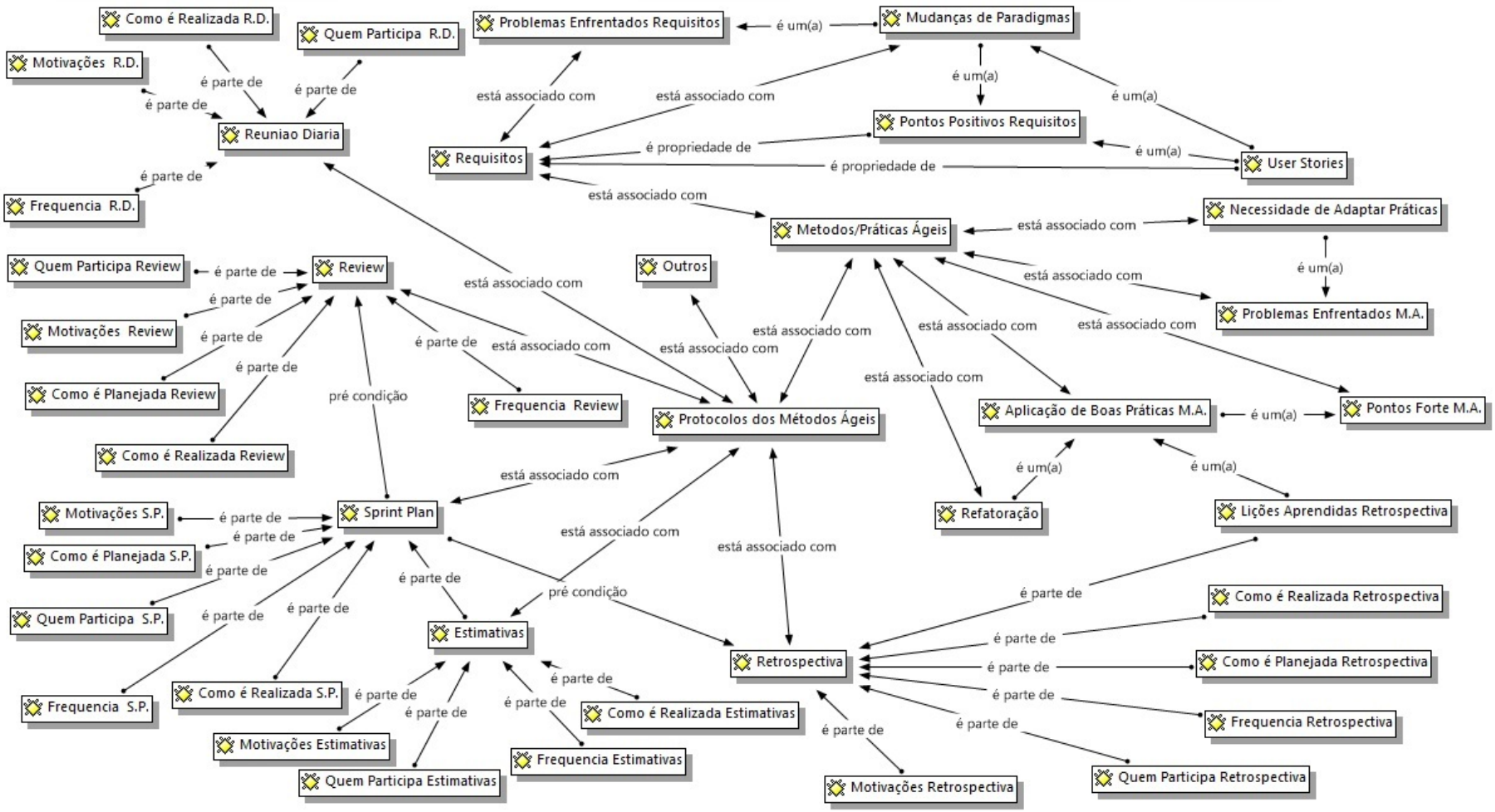

Figura 15 - 2ª versão do mapa de códigos da GT (Todos os Resultados): Visão Métodos Ágeis

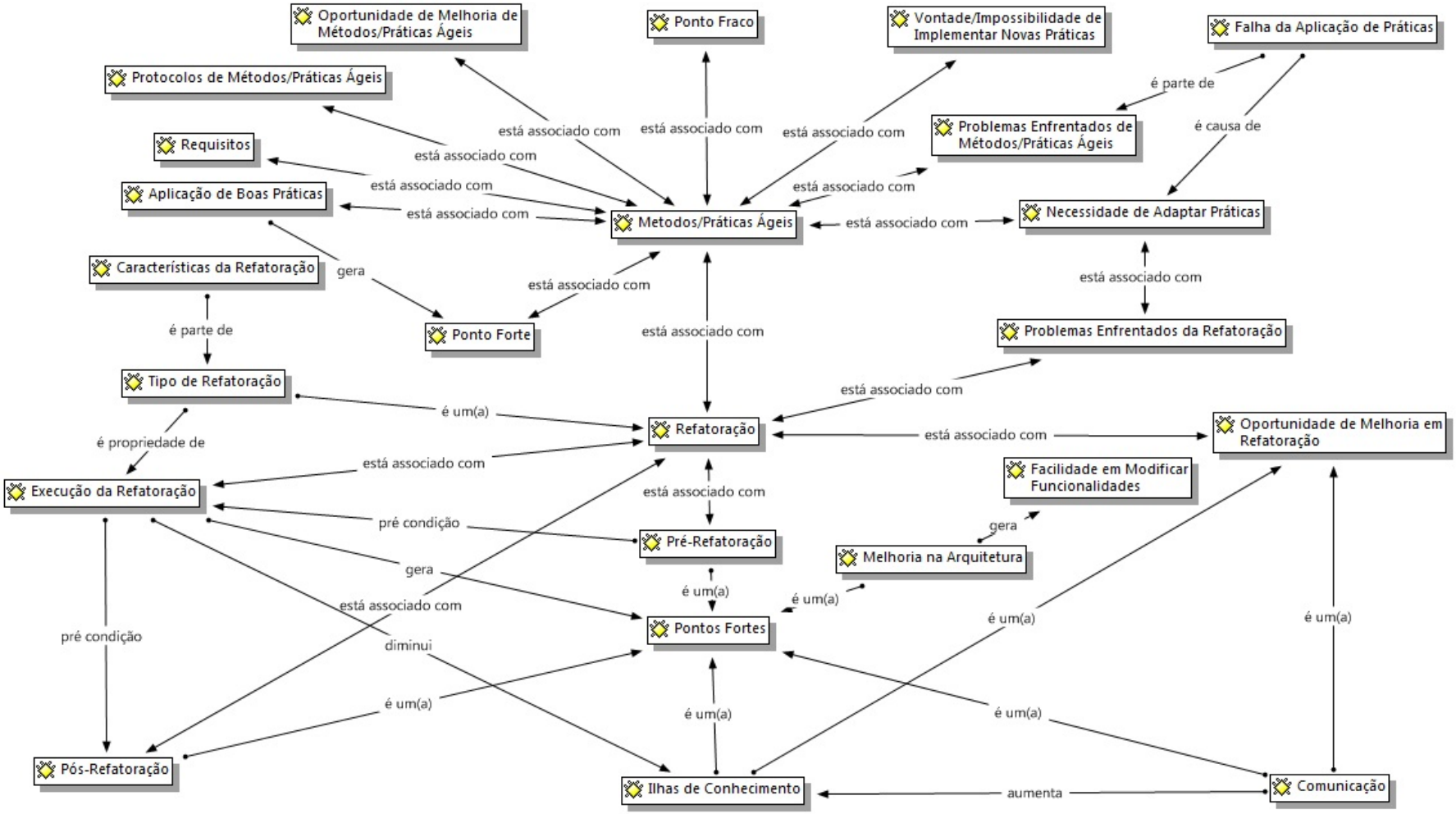

Figura 16 - 3ª versão do mapa de códigos da GT (Todos os Resultados): Visão Geral

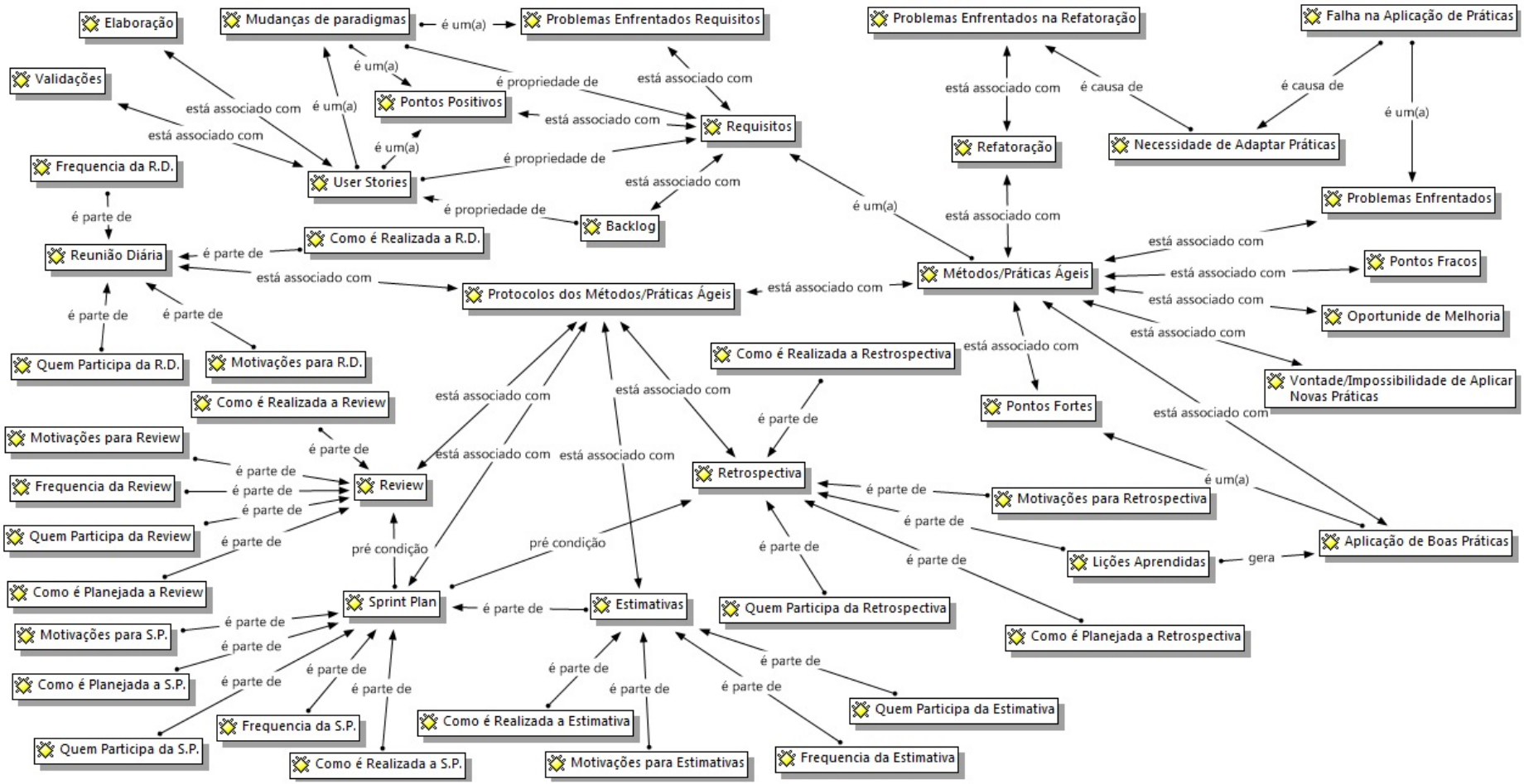

Figura 17 - 3ª versão do mapa de códigos da GT (Todos os Resultados): Visão Métodos Ágeis

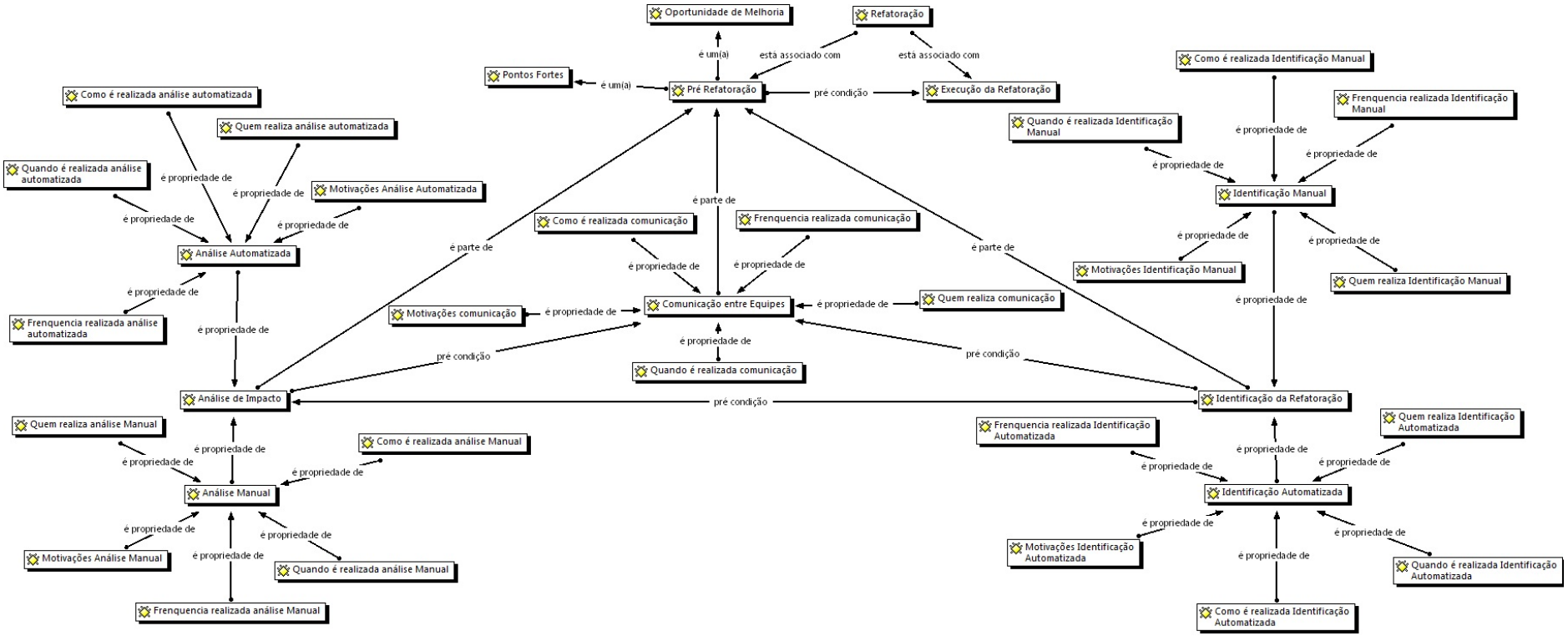

Figura 18 - 3ª versão do mapa de códigos da GT (Todos os Resultados): Visão Pré Refatoração

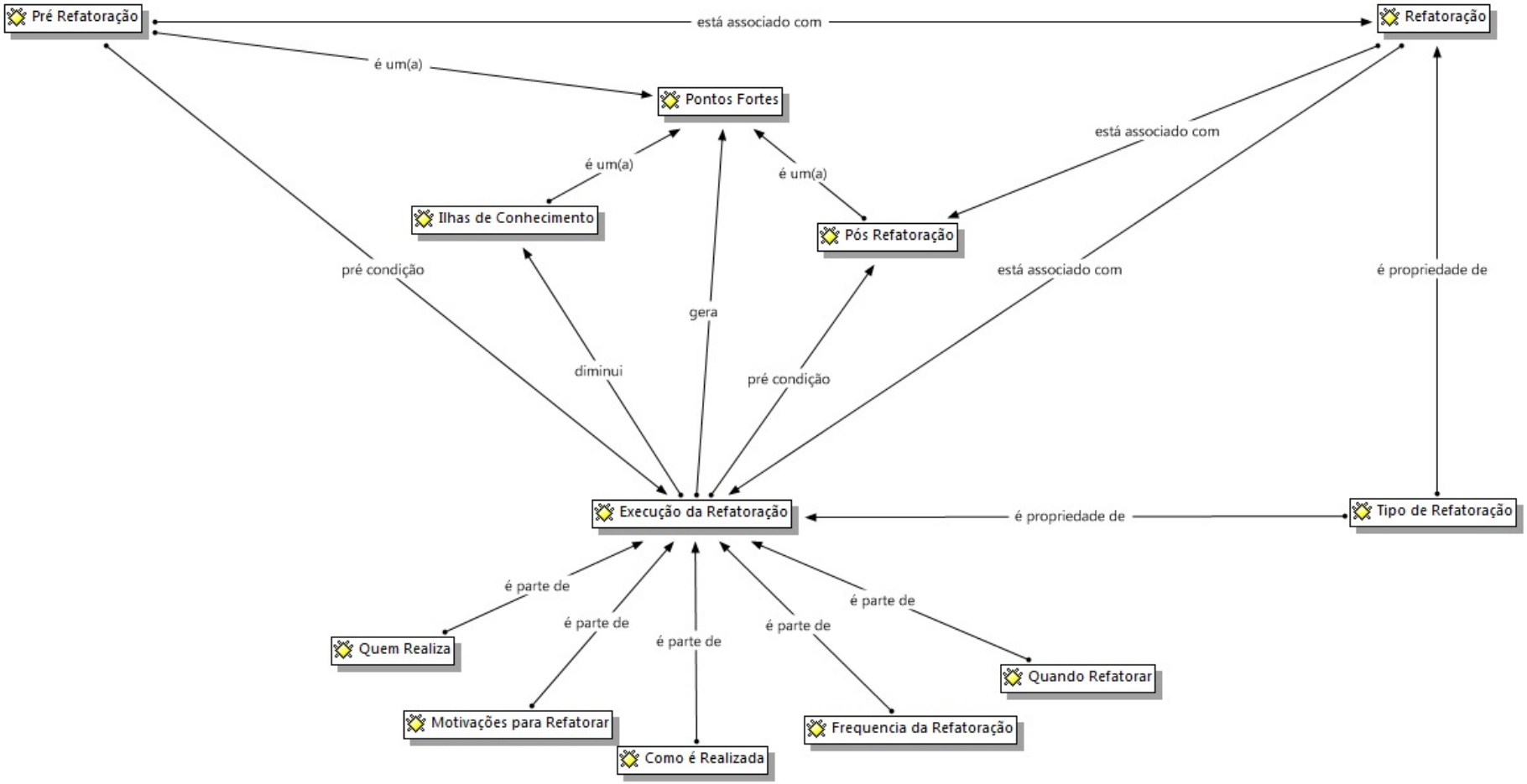

Figura 19 - 3ª versão do mapa de códigos da GT (Todos os Resultados): Visão Execução da Refatoração

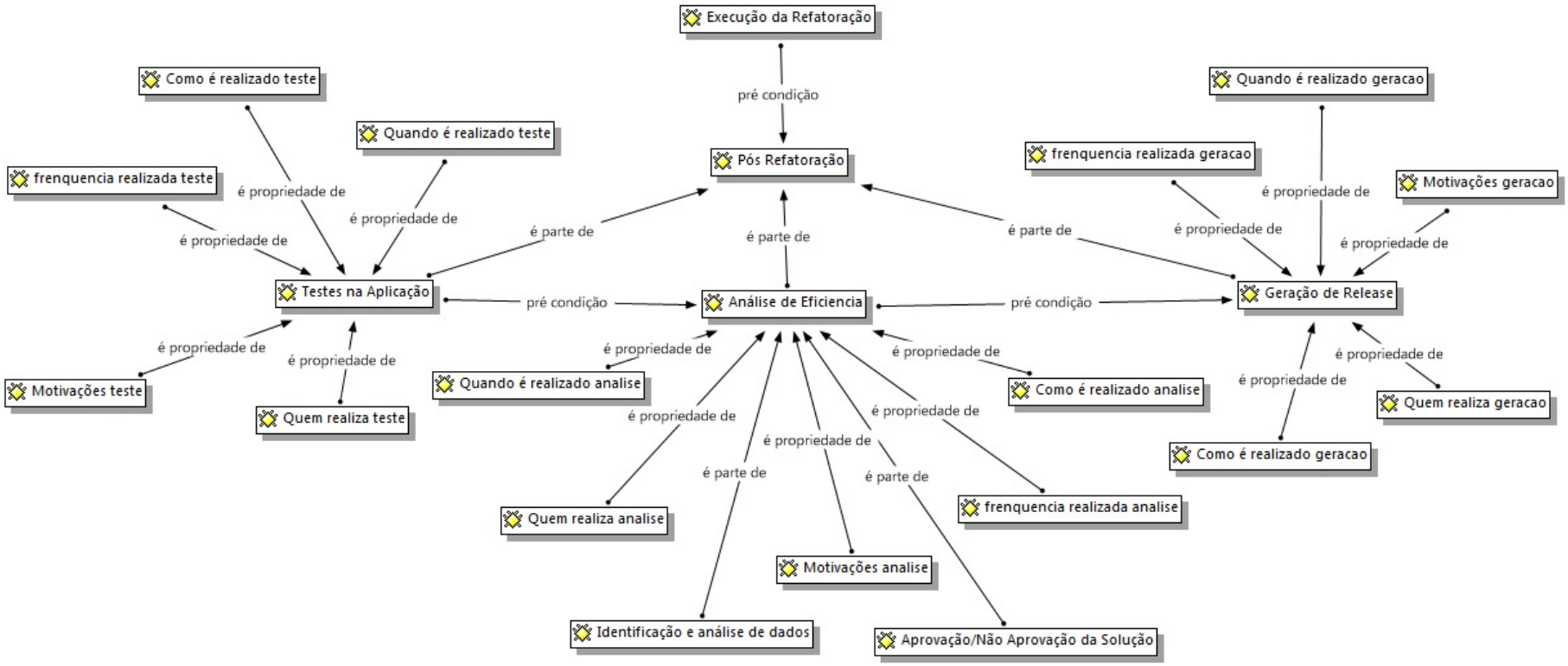

Figura 20 - 3ª versão do mapa de códigos da GT (Todos os Resultados): Visão Pós Refatoração

APÊNDICE C – DESCRIÇÃO DOS TESTES PILOTOS

Os testes piloto deste estudo foram realizados entre os dias 11/08/2009 e 24/08/2009 por cinco alunos de mestrado da universidade federal do Pará. Foram definidos os seguintes critérios de avaliação do questionário: clareza da pergunta, completude do estudo, organização das perguntas, forma com que as perguntas eram realizadas, dentre outros aspectos. Após a execução dos testes era exposto aos alunos qual a motivação das perguntas, o escopo de pesquisa e quais papéis seriam entrevistados.

Após as entrevistas foram propostas melhorias no guia no intuito de deixar o mesmo com perguntas mais claras e diretas, algumas das perguntas deveriam ser mais abertas, possibilitando ao entrevistado expor sua opinião. E algumas perguntas foram adicionadas. Após essa avaliação foi gerada o primeiro guia de pesquisa conforme exposto no APÊNDICE A – guia de entrevista.