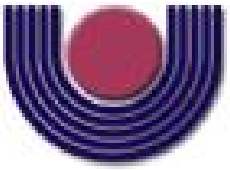

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Um Estudo de Caso Exploratório sobre a Associação entre a Taxa de Defeitos e o Desenvolvimento Ágil

Lucas Soares da Silva

CASCAVEL
2015

LUCAS SOARES DA SILVA

UM ESTUDO DE CASO EXPLORATÓRIO SOBRE A ASSOCIAÇÃO ENTRE A TAXA DE DEFEITOS E O DESENVOLVIMENTO ÁGIL

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof.Ivonei Freitas da Silva

CASCAVEL
2015

LUCAS SOARES DA SILVA

Um Estudo de Caso Exploratório sobre a Associação entre a Taxa de Defeitos e o Desenvolvimento Ágil

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Ivonei Freitas da Silva (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Victor Francisco Arraya Santander
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Ms. Sidgley Camargo de Andrade
Engenharia de Computação, UTFPR

Cascavel, 10 de março de 2016

AGRADECIMENTOS

Agradeço primeiramente a minha família, principalmente meus pais Sirlei e Gilso por todo apoio nos bons e maus momentos.

Agradeço ao meu orientador Ivonei pela dedicação e paciência.

Agradeço aos colegas e amigos que fiz na Unioeste.

Agradeço aos professores do curso de Ciência da Computação na Unioeste por terem me dados as bases do conhecimento.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
ADM	Administração de Empresas
CC	Ciência da Computação
ES	Engenheiro de Software
IEC	*International Engineering Consortium*
ISO	*International Organization for Standardization*
NBR	Norma Brasileira
PO	*Product Owner*
TADS	Tecnologia em Análise e Desenvolvimento de Sistemas
TI	Tecnologia da Informação
XP	*Extreme Programming*

Sumário

Resumo

O desenvolvimento ágil de software vem sendo adotado por diversas organizações com o objetivo de melhorar alguns fatores. A taxa de defeitos é um indicativo de qualidade de software e reduzir o número de defeitos é uma meta comum a qualquer empresa que desenvolve software. Este trabalho realiza um estudo de caso exploratório em uma empresa que desenvolve software utilizando metodologias ágeis. Objetivo do estudo de caso é diagnosticar se existe associação entre o desenvolvimento ágil e a taxa de defeitos. Ao final do estudo constatou-se por meio de evidências empíricas que existe associação, assim pode-se realizar trabalhos futuros a fim de diagnosticar os impactos do desenvolvimento ágil e a taxa de defeitos.

Palavras-chave: Desenvolvimento ágil de Software; Estudo de Caso; Defeitos.

Capítulo 1

Introdução

O ciclo de vida do desenvolvimento de um software é composto por diversas etapas, tais como análise, projeto, implementação, testes e manutenção [Sommerville 2011].

Assume-se que os ciclos de vida de software possam ser adaptados, muitos projetos de software adaptaram diferentes modelos de desenvolvimento, no entanto, de acordo com uma análise realizada pelo The Standish Group em 2009, apenas 32% desses projetos de software foram relatados como bem-sucedidos. Tais resultados foram considerados insatisfatórios, o que motivou a busca por novos modelos de desenvolvimento de software, e neste contexto os métodos ágeis surgiram como alternativa [Tarhan e Yilmaz 2014].

Este capítulo aborda as razões para a realização deste trabalho, o objetivo e a organização dos demais capítulos.

1.1 Desenvolvimento ágil de Software

Em meados da década de 1990 viu-se o surgimento de um conjunto de análise informal e abordagens de design que seriam conhecidos como métodos ágeis [Highsmith 2002].

Enquanto os modelos tradicionais valorizam o planejamento rigoroso, processos pré-definidos e documentação regular [Sommerville 2011], os métodos ágeis foram definidos valorizando indivíduos e interações sobre processos e ferramentas, software em funcionamento sobre documentação abrangente, colaboração com o cliente sobre negociação de contratos e responder mudanças sobre seguir o planejamento [Tarhan e Yilmaz 2014].

Métodos ágeis estão em conformidade com o Manifesto ágil [Beck et al. 2001], conjunto de 4 valores e 12 princípios criados por 17 desenvolvedores que são também autores de métodos

ágeis, consultores e líderes da comunidade de desenvolvimento de software.

1.2 Taxa de Defeitos

Não há nenhuma definição universalmente aceita para defeito, logo uma distinção frequentemente citada é erro, falha ou imperfeição[Radatz, Geraci e Katki 1990].

Defeitos são um fator significativo na qualidade de software. Em sistemas de software de grande escala, o número de defeitos pode ser muito alto, e, por conseguinte, tornar-se difícil de gerir. [Sandusky e Gasser 2005].

O número de defeitos tem sido geralmente considerado um indicador importante de qualidade de software [Reel 1999].

Problemas com a qualidade do software foram inicialmente descobertos na década de 1960 com o desenvolvimento dos primeiros grandes sistemas de software, e continuaram a atormentar a engenharia de software ao longo do século 20. [Sommerville 2011]

1.3 Relação do Desenvolvimento Ágil com a Taxa de Defeitos

Não há dúvida de que o processo de desenvolvimento usado tem uma influência significativa sobre a qualidade do software e que os bons processos são mais propensos a levar a bom software de qualidade. Gestão da qualidade e melhoria de processos pode levar a menos defeitos no software a ser desenvolvido. No entanto, é difícil de avaliar atributos de qualidade do software, tais como manutenibilidade, sem usar o software por um longo período. Consequentemente, é difícil dizer como características do processo podem influenciar estes atributos [Sommerville 2011].

Estudos anteriores utilizaram várias abordagens para mostrar que a qualidade melhorou com a adoção dos métodos ágeis. Estas abordagens incluem análises do número de defeitos, revisões formais de código e pesquisas que exploram a percepção de qualidade feita por pessoas na organização [Korhonen 2010].

1.4 Motivação

Do ponto de vista acadêmico, este trabalho utiliza métodos da área da engenharia de software empírica, sendo parte de um conjunto de pesquisas que procuram relacionar o desenvolvimento ágil de software com temas como motivação dos desenvolvedores e satisfação do cliente.

De um ponto de vista industrial, este trabalho se faz exploratório, pois busca compreender melhor o desenvolvimento ágil em empresas de desenvolvimento de software e através de evidências coletadas nessas empresas e com base na literatura, tentar perceber associações entre o desenvolvimento ágil de software e a qualidade do mesmo sob o aspecto da métrica selecionada, neste caso a taxa de defeitos.

1.5 Objetivos

O objetivo desta pesquisa é investigar o processo de desenvolvimento ágil de software em uma empresa e diagnosticar se existe relação entre a taxa de defeitos do software e o processo de desenvolvimento ágil.

O resultado deste trabalho deve gerar um Estudo de caso exploratório, que pode servir como base para trabalhos futuros, como estudos que avaliem o impacto das práticas ágeis na taxa de defeitos ou ainda estudos que busquem validar hipóteses geradas por este trabalho.

1.6 Organização do Trabalho

- Capítulo 2 Revisão de literatura sobre desenvolvimento ágil de software.
- Capitulo 3 Revisão de literatura sobre taxa de defeitos de software.
- Capitulo 4 Protocolo do estudo de caso.
- Capítulo 5 Resultados das Evidências Coletadas
- Capítulo 6 Análise e Discussão dos Resultados.
- Capítulo 7 Conclusões obtidas após a execução do estudo de caso e trabalhos futuros.

Capítulo 2

Desenvolvimento Ágil de Software

O termo ágil pode ser rastreado até a indústria manufatureira em 1991, quando o desenvolvimento *lean* surgiu com o objetivo de eliminar o desperdício, amplificar a aprendizagem, proporcionar a maior rapidez possível e capacitação das equipes [Poppendieck e Poppendieck 2003].

Agilidade pode ser definida como a capacidade de acomodar alterações (esperadas ou não) em um ambiente dinâmico porém simples, econômico, e com qualidade, utilizando de curtas iterações, aplicando conhecimento prévio e gerando novo conhecimento a partir da experiência.

No ano de 2001, houve uma reunião com um grupo de 17 desenvolvedores experientes, consultores e líderes da comunidade de desenvolvimento de software. Dentre os membros desse grupo estavam os criadores de métodos de desenvolvimento como *Scrum, Extreme Programming, Adaptive Software Development*, entre outros [Highsmith 2001].

Desse encontro emergiu o Manifesto para desenvolvimento ágil de software e foi assinado por todos os membros do grupo. [Highsmith 2001].

2.1 O Manifesto ágil

O Manifesto Ágil [Beck et al. 2001] enumera um conjunto de 4 valores os quais servem de base para o desenvolvimento ágil de software. Juntamente com estes valores, existem também de 12 princípios. Princípios são diretrizes de domínio específicos para a vida, que mostra como os valores podem ser aplicados em diferentes áreas. E por último, existem práticas, que são aplicações específicas dos valores e princípios [Lagerberg et al. 2013].

A tabela 2.1 específica os 4 valores ágeis propostos no Manifesto.

A tabela 2.2 enumera o conjunto dos 12 princípios ágeis do Manifesto.

1. Indivíduos e interação entre eles mais que processos e ferramentas.
2. Software em funcionamento mais que documentação abrangente.
3. Colaboração com o cliente mais que negociação de contratos.
4. Responder a mudanças mais que seguir um plano.

Tabela 2.1: Valores ágeis do Manifesto Ágil

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 adequam 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 à 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 equipe 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. Contínua atenção à 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, a equipe reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Tabela 2.2: Princípios ágeis do Manifesto Ágil

2.2 Métodos Ágeis de Desenvolvimento

Métodos ágeis são em sua essência baseados em valores e princípios definidos pelo Manifesto Ágil e composto por práticas ágeis [Jalali e Wohlin 2010]. Práticas ágeis devem ajudar a cumprir os princípios ágeis em um método e podem ser agrupados em práticas de gestão, práticas de processo de software e práticas de desenvolvimento de software [Lee e Yong 2013]. Exemplo de práticas de gestão são: cliente no local, reuniões diárias em pé. Práticas de processo de software incluem design simples e propriedade coletiva do código. A programação em pares e testes unitários são exemplos de práticas de desenvolvimento de software [Campanelli e Parreiras 2015].

Métodos ágeis de desenvolvimento de software são usados para produzir software de alta

qualidade no mais curto período de tempo. Uma pesquisa realizada mostra que 41% dos projetos de desenvolvimento adotaram alguma metodologia ágil e práticas ágeis estão sendo usados em 65% desses projetos [Ahmed et al. 2010].

Apesar do crescente debate sobre como métodos ágeis poderiam ser praticados no desenvolvimento de software, são amplamente aceitos [Kayes, Sarker e Chakareski 2013]. De acordo com o relatório feito pelo The Standish Group em 2012, projetos ágeis são bem sucedidos três vezes mais frequentemente do que projetos não-ágeis[Chan 2013]. Esta estatística é consistente com o relatório do The Standish Group de 2006 onde é afirmado que 41% dos projetos ágeis são bem sucedidos em oposição aos 16% dos projetos em cascata [Wailgum 2007]. Além disso, no relatório de 2012, 49% dos participantes dizem que a maioria de suas empresas estão usando desenvolvimento ágil. O relatório também afirma que o número de organizações que planeja implantar métodos ágeis em projetos futuros aumentou de 59% em 2011 para 83% em 2012. A pesquisa também mostra que a adoção de métodos ágeis melhora o gerenciamento geral do processo de desenvolvimento, o relacionamento com os clientes, diminui a quantidade de horas extras e aumenta a satisfação do cliente [Mann e Maurer 2005]

Alguns métodos ágeis de desenvolvimento populares são XP, Scrum e Lean.

2.2.1 Programação Extrema (XP)

XP foi criado para ajudar as pequenas equipes para desenvolver software quando os requisitos são vagos e mudam com frequência. É considerada uma metodologia leve e centra-se na redução de custos, testes de unidade antes e durante as atividades de código, integração contínua do sistemas, programação em pares, design simples e entregas frequentes de software funcionando [Beck 2000].

2.2.2 Scrum

Scrum é uma estrutura de processo para entregar produtos com o maior valor possível e lidar com problemas complexos ou situações [Schwaber e Sutherland 2011]. Usa abordagem iterativa e incremental para desenvolver produtos utilizando equipes multifuncionais (Deemer et al., 2010). Com base no empirismo, Scrum foca no conhecimento, experiência e tomada de decisão com base no que é conhecido [Schwaber e Sutherland 2011].

2.2.3 Lean

No surgimento do desenvolvimento ágil de manufatura, lean tem sido adaptado ao contexto de desenvolvimento de software [Petersen 2010]. Segundo o lean, pensar em atividades que não agregam valor para o cliente são considerados resíduos [Ikonen et al. 2010]. A ideia por trás dele, no contexto de desenvolvimento de software, é entregar as funcionalidades necessárias para os clientes [Poppendieck e Poppendieck 2003].

O Kanban [Sugimori et al. 1977], atualmente pode ser chamado de método ágil adaptativo, foi criado como uma ferramenta enxuta para gerenciar operações de fabricação [Ikonen et al. 2010]. Também tem sido aplicado no desenvolvimento de software e tem sido considerado adaptável. Uma vez que se baseia em princípios *lean, kanban* tenta remover os resíduos do processo de produção. As principais regras são: visualizar o fluxo de trabalho, limitar o trabalho em andamento e medir o tempo para terminar as tarefas [Ikonen et al. 2010]

2.3 Dificuldades encontradas ao Utilizar Métodos Ágeis

Na prática, os princípios ágeis são por vezes difíceis de serem realizados devido a diversos fatores, tais como [Sommerville 2011]:

1. Embora a ideia de envolvimento do cliente no processo de desenvolvimento é um atrativo, o seu sucesso depende de ter um cliente que está disposto e capaz para passar o tempo com a equipe de desenvolvimento e que pode representar todos os stakeholders do sistema. Frequentemente, os representantes dos clientes estão sujeitas a outras pressões e não podem participar plenamente no desenvolvimento de software.

2. Os membros da equipe individualistas não podem ter personalidades adequadas para o envolvimento intenso que é típico dos métodos ágeis, e, portanto, não interagem bem com outros membros da equipe.

3. Priorizar mudanças pode ser extremamente difícil, especialmente em sistemas para os quais há muitas partes interessadas. Normalmente, cada stakeholder dá prioridades diferentes para diferentes alterações.

4. Manter a simplicidade requer trabalho extra. Sob pressão dos prazos de entrega, os membros da equipe podem não ter tempo para levar a cabo medidas de simplificação do sistema desejáveis.

5. Muitas organizações, especialmente as grandes empresas, passaram anos a mudar sua cultura para que os processos sejam definidos e seguidos. É difícil para eles mudar para um modelo de trabalho em que os processos são informais e definidos por equipes de desenvolvimento.

2.4 Práticas ágeis

Prática Ágil
Avaliação De Riscos Ágeis
Cliente Presente
Código Limpo
Código Padronizado
Documentação Tardia
Entregas Frequentes
Estimativas em Pontos
Integração Contínua
Jogo do Planejamento
Preparação do *Backlog*
Priorização de Requisitos
Programação em Par
Projeto Simples
Propriedade Coletiva
Quadro *Kanban*
Rastreamento de *Bugs*
Refatoração
Reunião de Planejamento
Reunião de Retrospectiva
Reunião de de Revisão
Reunião Diária
Ritmo Sustentável
Tempo em Progresso Limitado
Testes Automatizados
Testes de Aceitação
Testes de Unidade
Timebox Fixa
Times Auto-Organizáveis

Tabela 2.3: Exemplo de Práticas ágeis

A tabela 2.3 lista alguns exemplos de práticas ágeis, estas foram escolhidas porque são adotadas pela empresa ABC, como será visto no capítulo 4. São práticas comuns dentro do desenvolvimento ágil de software e pertencem a métodos ágeis populares como *XP* e *Scrum*

2.5 Métodos ágeis e a Qualidade de software

Métodos ágeis de desenvolvimento de software têm uma série de benefícios relatados na produtividade, visibilidade do projeto, qualidade de software e outras áreas. Há também efei-

tos negativos relatados. No entanto, a base de evidência necessita de mais estudos empíricos. [Lagerberg et al. 2013]

Estudos recentes propõem que as práticas ágeis, como programação em par e integração contínua ajudam a melhorar a qualidade do código. Como resultado, o número de defeitos pode reduzir, e este fator tem encorajado organizações de software para começar a transformação ágil [Korhonen 2010].

Um ponto de vista ágil propõe que os defeitos são vistos como resíduos. Os defeitos devem ser corrigidos assim que são encontrados e, portanto, os defeitos devem ser relatados com uma ferramenta específica. De acordo com este ponto de vista, um resultado imediato de adoção de métodos ágeis seria uma redução do número de defeitos. Tais resultados foram de fato relatados em estudos anteriores [Korhonen 2010].

Práticas ágeis, por exemplo, trabalhando como uma equipe scrum, testes automatizados e desenvolvimento orientado a testes (TDD), são relatados para ajudar a melhorar a qualidade do produto e reduzir bastante o número de falhas[Korhonen 2010]. Os resultados dos estudos de caso na Microsoft e IBM [Nagappan et al. 2008] indicam que o defeito pré-lançamento densidade de quatro produtos diminuiu entre 40% e 90% em comparação com projetos similares que não usaram a prática TDD [Korhonen 2010].

Capítulo 3

Taxa de Defeitos do Software

Este capítulo faz uma revisão de literatura sobre a taxa de defeitos do software e sua relevância industrial.

3.1 Trabalhos Relacionados

Sob o aspecto do desenvolvimento ágil, defeitos de software são vistos como resíduos [Poppendieck e Poppendieck 2003]. Os defeitos devem ser corrigidos assim que são encontrados e, portanto, relatar defeitos com uma ferramenta não deve sequer ser necessária. De acordo com este ponto de vista, um resultado imediato da adoção ágil seria uma redução do número de defeitos. Tais resultados foram de fato relatado em estudos anteriores [Ilieva, Ivanov e Stefanova 2004], [Layman, Williams e Cunningham 2004], [Lindvall et al. 2004].

Práticas ágeis, em equipes que trabalham com *srum*, testes automatizados e desenvolvimento orientado a testes (TDD), foram relatadas como de ajuda para melhorar a qualidade do produto e reduzir o número de falhas. Os resultados dos estudos de caso da Microsoft e IBM [Nagappan et al. 2008] indicam que a densidade de defeitos do pré-lançamento de quatro produtos diminuiu entre 40% e 90% em relação a projetos similares que não usaram a prática TDD [Korhonen 2010].

Um conjunto de práticas ágeis foi selecionado para um piloto da Ericsson [Auvinen et al. 2005] em vez de tentar implementar um processo completamente ágil. Essas práticas incluíram a programação em pares, a propriedade coletiva de código, o jogo do planejamento e o cliente no local. Para avaliar o impacto do piloto sobre a qualidade do código, a contagem de defeitos a partir do piloto foi comparada com a contagem de defeitos de uma entrega anterior. A análise

mostrou uma redução de 5.5% em defeitos. Em [Schatz e Abdelshafi 2005], práticas ágeis básicas, incluindo multifuncionais, equipes auto-gerenciadas e iterações de tempo, foram aplicados e a organização experimentou um aumento de 30% em termos de qualidade como medido pelo número de defeitos relatados por clientes nos primeiros nove meses após o lançamento. Depois de implementar TDD, a contagem de defeitos caiu para menos de 10 por equipe, o que representa mais uma melhoria de 75% nos índices de defeitos em comparação com a versão anterior[Korhonen 2010].

No entanto, estes resultados positivos também mostram que, embora o número de falhas tenha diminuído, ainda existem falhas e os restantes devem ser manuseados de forma sistemática [Korhonen 2010].

3.2 Definição de Defeitos

Não parece haver nenhuma definição universalmente aceita para defeito. Uma distinção frequentemente citada é erro, falha ou imperfeição[Radatz, Geraci e Katki 1990].

É considerado que um software contém uma falha, se produz um resultado insatisfatório quando executado [Laitenberger 1998].

Imperfeição é considerada como uma etapa, processo ou definição de dados como incorretos, ou seja, um código incorreto é uma imperfeição mesmo que não resulte em um comportamento inesperado do sistema [Radatz, Geraci e Katki 1990]

3.3 Métrica Utilizada

As bases para seleção de métricas dependem das metas de negócios para o produto e das necessidades do avaliador. Necessidades são especificadas por critérios de medidas. O modelo desta parte da NBR ISO/IEC 9126 apoia uma variedade de requisitos de avaliação [ISO/IEC 2003].

Como o objetivo deste trabalho é diagnosticar se há associação entre o uso das práticas ágeis e a taxa de Defeitos do Software, nós selecionamos o atributo de qualidade confiabilidade, e dentre as métricas externas que apresentam indícios da confiabilidade do software, a métrica escolhida foi a taxa de defeitos por *sprint*.

A métrica foi escolhida de acordo com as regras de negócio estabelecidas pelo caso. O chamado caso é a empresa onde este estudo foi realizado.

Outras métricas surgiram ao longo do processo, porém as equipes divergiam quanto ao registro de dados, o que atrapalharia a coleta de evidências. A taxa de defeitos por *Sprint* foi a métrica comum a todos as equipes de desenvolvimento ágil na empresa onde o estudo foi realizado.

Artigos na literatura vem indicando defeitos como desvios para a qualidade do software [Gilb, Graham e Finzi 1993], [Laitenberger e DeBaud 2000], [Humphrey 1995] e [Runeson e Wohlin 1998].

3.4 Qualidade de Software

A norma ISO/IEC 9126 é um padrão definido pela IEEE, que define características relacionadas à qualidade de software [Zeiss et al. 2007]. Para isso, a norma descreve um modelo de qualidade composto em duas partes: qualidades interna e externa, e qualidade de uso. A primeira especifica seis características para qualidade interna e externa, que são subdivididas em subcaracterísticas manifestadas externamente quando o software é utilizado como parte de um sistema. Já a segunda, especifica quatro características de qualidade em uso, que é, para o usuário, o efeito combinado das seis características de qualidade do software [ver figura 3.1].

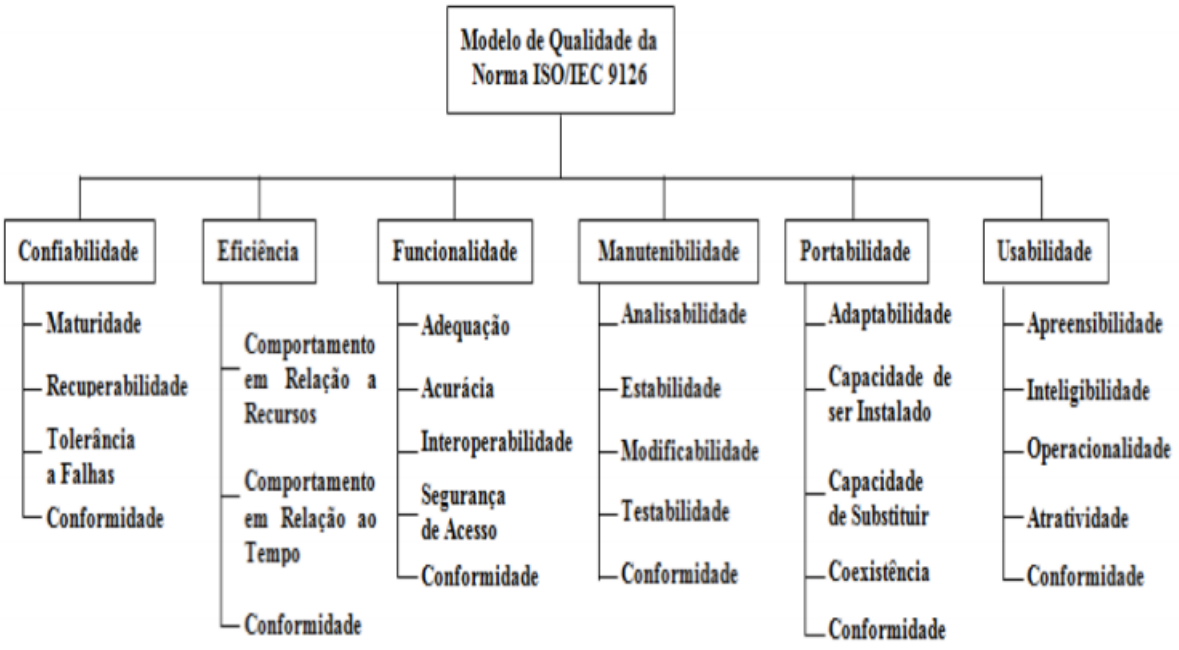

Figura 3.1: Modelo de qualidade externa e interna da NBR ISO/IEC 9126

A figura 3.1 expressa um modelo de qualidade externa interna definida pela [ISO/IEC 2003], a métrica adotada serve como indicativo para qualidade sob um ou mais aspectos.

O número de defeitos tem sido geralmente considerado um indicador importante de qualidade de software. Sabe-se que não é possível adicionar qualidade a um software, porém em algum momento percebe-se que há um problema de qualidade e provavelmente é tarde demais para corrigi-lo [Reel 1999].

Desenvolver software com qualidade é um desafio, pois se trata de questões abstratas, sendo uma tarefa difícil encontrar parâmetros para medir pontos de qualidade [Pressman 2010].

Capítulo 4

Protocolo do Estudo de Caso

Este protocolo foi baseado em [Wohlin e Aurum 2014], o estudo precisou passar por fases de tomadas de decisão.

A Figura 4.1 ilustra uma estrutura de tomada de decisões em uma pesquisa cientifica [Wohlin e Aurum 20

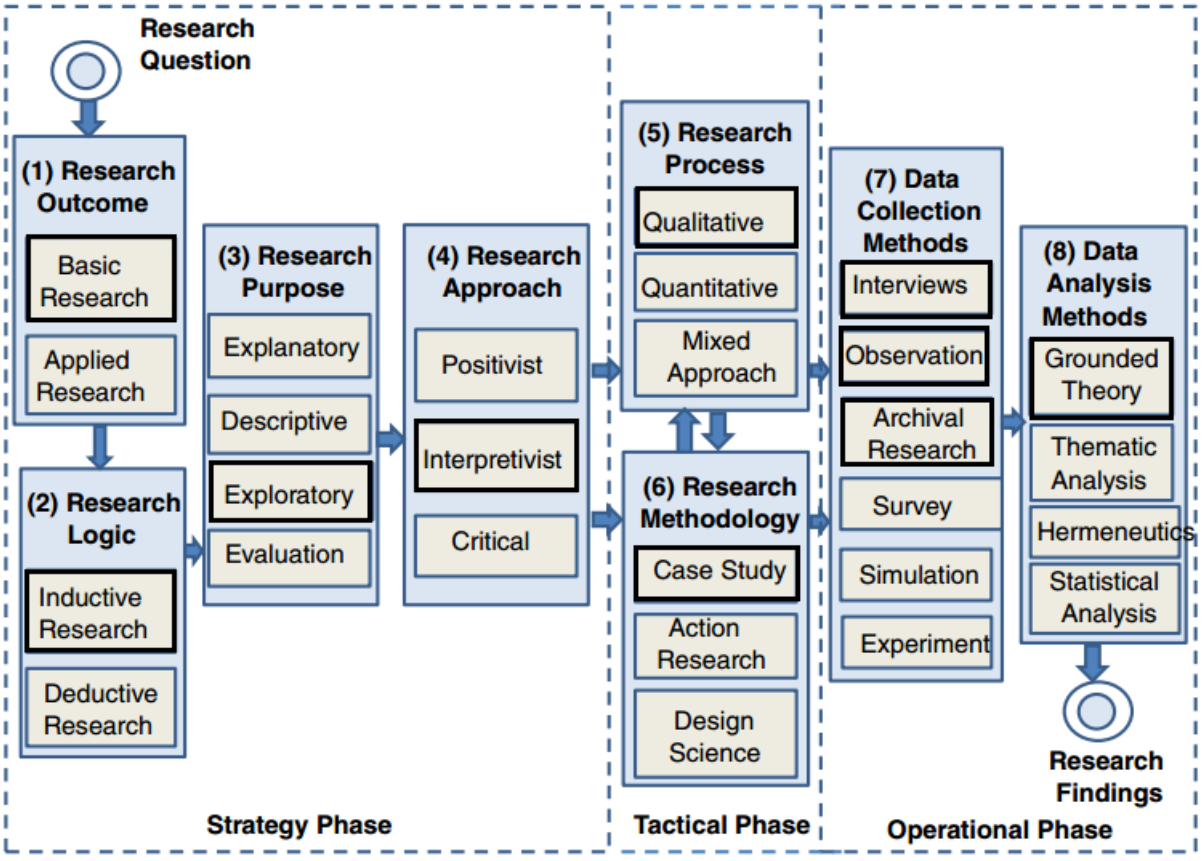

Figura 4.1: Estrutura de tomada de decisão em pesquisa cientifica [Wohlin e Aurum 2014]

4.1 Fases de Tomada de decisão

[Wohlin e Aurum 2014] sugere uma estrutura para tomadas de decisão dividida em 3 fases, são chamadas fase Estratégica, Tática e Operacional como pode ser visto na figura 4.1.

4.1.1 Fase Estratégica

Esta fase permite ao pesquisador realizar sua pesquisa de forma sistemática, e posicionar a pesquisa em relação a diferentes abordagens gerais [Wohlin e Aurum 2014]. Uma estratégia de pesquisa eficaz requer a compreensão do tema de pesquisa, bem como ter conhecimento sobre cada ponto de decisão apresentada na Figura 4.1 [Wohlin e Aurum 2014]. A estratégia de investigação envolve decisões sobre o resultado da pesquisa, a lógica de pesquisa, fins de pesquisa e abordagem de pesquisa [Wohlin e Aurum 2014].

Durante a fase estratégica desta pesquisa foram escolhidas a pesquisa básica, indutiva, exploratória e interpretativista.

A pesquisa é básica porque é aplicada a um problema onde a ênfase é a compreensão do problema, em vez de fornecer uma solução. Portanto, a contribuição principal é o conhecimento gerado a partir da pesquisa. É indutiva porque o pesquisador infere conceitos teóricos e padrões de dados observados. O pesquisador começa com observações específicas, detecta padrões teóricos, e desenvolve algumas conclusões gerais ou teorias [Basili 1993]. A pesquisa é exploratória porque não há muita informação disponível na área e o pesquisador pretende reunir alguns *insights* sobre o problema. O objetivo é explorar a área do problema e fornecer informações de base que pode ser usado para a pesquisa descritiva ou explicativa. A pesquisa é interpretativista pois visa compreender as atividades humanas em uma situação específica do ponto de vista dos participantes, portanto, enfatiza o contexto [Klein e Myers 1999, Myers et al. 1997].

4.1.2 Fase Tática

A fase tática envolve decisões sobre como operacionalizar as atividades de pesquisa em termos de abordagem da questão de pesquisa especificamente[Wohlin e Aurum 2014]. Os pontos de decisão são o processo e a metodologia da pesquisa[Wohlin e Aurum 2014]. A fase tática foca na seleção do processo atual e no uso da metodologia para atingir o objetivo da pesquisa [Wohlin e Aurum 2014].

Durante a fase tática foi definido que a pesquisa seria um processo Qualitativo com a metodologia Estudo de Caso.

Esta pesquisa é chamada qualitativa, pois realizada uma investigação que tem como objetivo estudar o fenômeno social e cultural. Ele é realizado quando um pesquisador tem como objetivo compreender as perspectivas dos seus sujeitos de pesquisa [Wohlin e Aurum 2014]

Estudo de caso envolve uma abordagem interpretativa e naturalista do mundo, investigando coisas na sua forma natural, tentando interpretar fenômenos e termos dos significados trazidos pelas pessoas [Denzin e Lincoln 2005]. É uma metodologia de pesquisa que utiliza vários métodos de coleta de dados para reunir informações a partir de uma variedade de fontes, com o objetivo de investigar um fenômeno em seus ambientes naturais [Benbasat, Goldstein e Mead 1987].

Este estudo de caso foi definido seguindo as diretrizes documentadas em [Runeson e Höst 2009, Yin 2008].

4.1.3 Fase Operacional

A fase operacional envolve decisões sobre as ações que serão tomadas quando a pesquisa for executada, incluindo os métodos de coleta de dados e técnicas para análise de dados. Assim, planejando os detalhes e para coletar os dados para ser capaz de responder à questão de pesquisa [Wohlin e Aurum 2014]. A fase operacional está focada realmente em realizar a pesquisa empírica através da coleta e análise dos dados[Wohlin e Aurum 2014].

Na fase operacional, três métodos de coleta foram utilizados, entrevistas, observação em campo e Análise de documentação, a fim de permitir a triangulação dos métodos [Runeson e Höst 2009].

Para a análise foram escolhidas as técnicas Análise estatística descritiva e a Teoria Fundamentada em Dados (do inglês, *grounded theory*), um método qualitativo de pesquisa cujo objetivo é gerar uma teoria a partir de dados sistematicamente reunidos e analisados [Strauss e Corbin 2008]. No entanto, neste trabalho estamos interessados em fazer uma análise qualitativa para responder as questões de pesquisa. Tendo respondido tais questões, é possível gerar hipóteses que podem servir como base em trabalhos futuros.

4.2 Questões de Pesquisa

1. As práticas ágeis utilizadas pelas equipes de desenvolvimento estão associadas com a taxa de defeitos encontrados durante as *sprints*?

   (a) As práticas ágeis utilizadas são adaptadas?

   (b) As equipes conhecem as práticas ágeis?

   (c) As equipes possuem maturidade quanto ao uso das práticas?

   (d) Existe uma preocupação das equipes em reduzir o número defeitos do software?

   (e) Alguma prática se destaca mais que outra quanto a associação entre a quantidade de defeitos e as práticas ágeis? Se sim, quais práticas e como isso pode ser percebido?

4.3 Objetivo

Investigar a relação entre a taxa de defeitos encontrados em um software e as práticas ágeis adotadas pela equipe de desenvolvimento ágil em uma empresa tipica de desenvolvimento de software do ramo agroindustrial.

4.4 Design

O Design do estudo de caso é único e embutido[Yin 2008]. É considerado único porque o caso (Empresa ABC) é uma empresa típica no ramo de desenvolvimento de software. É chamada embutido porque o caso possui várias unidades de análise.

Na empresa ABC, existem 4 equipes de desenvolvimento ágil, todas utilizam o método *Scrum*, neste estudo cada equipe é considerada uma unidade de análise, ou seja, as coletas de evidências e análises dos resultados devem ser realizadas para cada uma delas. No decorrer do texto falaremos mais sobre as equipes.

4.4.1 Objeto de Estudo

O objeto de estudo é baseado no processo de desenvolvimento de software com o uso de práticas ágeis na Empresa ABC.

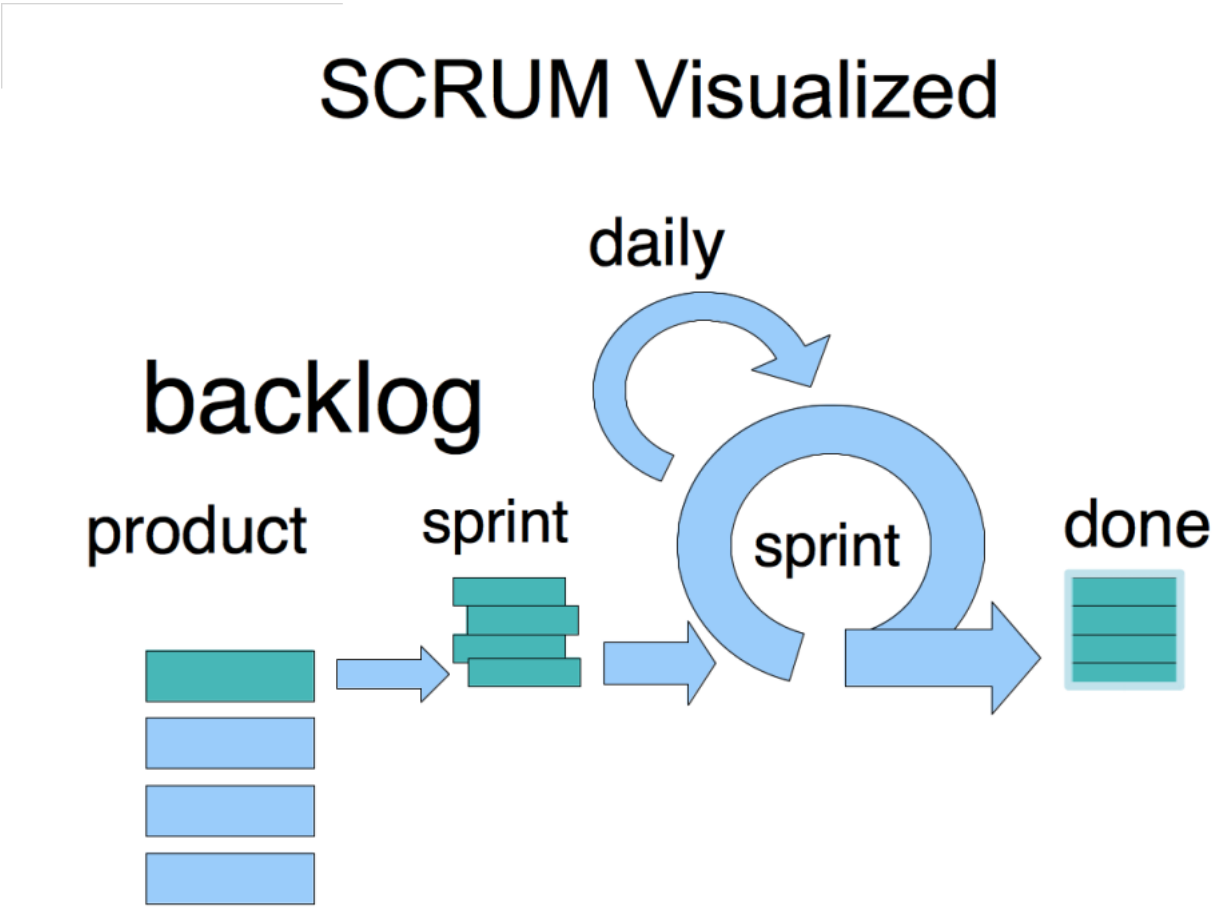

Figura 4.2: Fluxograma do Processo de Desenvolvimento Scrum seguido pela empresa ABC

O processo de desenvolvimento de software da empresa ABC segue o método *Scrum*, o que pode ser visto na figura 4.2.

As práticas ágeis relacionadas ao código são aplicadas dentro da *Sprint*.

4.5 Seleção do Caso

A empresa foi escolhida devido ao interesse da mesma em colaborar com a pesquisa, cedendo o tempo dos seus colaboradores e recebendo os pesquisadores no recinto. Para os pesquisadores tal situação é favorável porque o cenário é um ambiente natural.

A Empresa ABC trabalha desde 1991 no domínio de softwares para gestão de cooperativas e é localizada em Cascavel (Paraná, Brasil), mas atua em todo o território nacional, principalmente nos estados do Paraná, Santa Catarina, Rio Grande do Sul, Minas Gerais, Mato Grosso, Mato Grosso do Sul, São Paulo e Goiás. Projeta e desenvolve soluções integradas para dar

suporte aos processos operacionais das empresas, que também auxiliam no planejamento e na execução de ações estratégicas para o relacionamento entre empresa e produtor rural. A empresa ABC conta com uma equipe de mais de 130 colaboradores especialistas em diversas áreas.

A empresa em questão realiza diversas atividades além do desenvolvimento, tais atividades devem ser desconsideradas na pesquisa. A fábrica de software é dividida em duas partes, desenvolvimento ágil e desenvolvimento não-ágil. Nesta empresa a parte que desenvolve software sem métodos ágeis, realiza um trabalho diferente e a mais tempo, logo seria inviável fazer um estudo comparativo.

No inicio da pesquisa, foram coletadas alguns dados para auxiliar na montagem do cenário para o estudo de caso, tais dados revelaram informações como a composição dos times, o módulo de trabalho, e as práticas ágeis empregadas em cada equipe.

4.6 Papéis e Procedimentos do Estudo de Caso

Fábrica é o termo utilizado pela empresa ABC para descrever o local onde ocorre o desenvolvimento ágil de software, é formada por quatro equipes de desenvolvimento [ver tabela 4.2], todas elas utilizam o método *scrum* para a gerência de projetos e pelo menos mais um método ágil.

As equipes de desenvolvimento da empresa ABC trabalham com um princípio onde todos os membros são Engenheiros de Software, um termo adotado internamente, e possuem um foco específico dentro da equipe de acordo com suas capacidades e perfis.

A tabela 4.1 mostra a composição das equipes da empresa ABC.

As equipes não são fixas, logo podem ocorrer alterações na composição das equipes ao longo da pesquisa.

Equipe	Formação	Papel no Time
Equipe A	Graduação	*Scrum Master*
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Analista
	Graduação	Testador
	Pós-Graduação	Desenvolvedor
Equipe B	Pós-Graduação	*Scrum Master*
	Pós-Graduação	Desenvolvedor
	Pós-Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Testes
Equipe C	Graduação	*Scrum Master*
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	*Product Owner*
Equipe D	Graduação	*Scrum Master*
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor
	Graduação	Desenvolvedor

Tabela 4.1: Composição das equipes de Desenvolvimento

4.7 Coleta de Evidências

Durante a coleta de evidências foram utilizadas várias fontes de informação, a fim de mitigar viés a partir de uma única fonte de dados [Runeson e Höst 2009]. Além disso, três métodos de coleta foram utilizados, entrevistas, observação em campo e Análise de documentação, a fim de permitir a triangulação dos métodos [Runeson e Höst 2009].

A informação extraída dos participantes da empresa será confidenciada apenas aos pesquisadores e aos participantes da pesquisa.

Entrevistas

Entrevista é um método curioso para se coletar informações sobre processos, produtos, etc., envolvendo ao menos um pesquisador falando com um entrevistado. [Shull, Singer e Sjøberg 2008, Seaman 2008]. As entrevistas serão realizadas com os membros das equipes de desenvolvimento que utilizam práticas ágeis no desenvolvimento de software. As etapas das entrevistas no

estudo de caso são:

- Entrevista para obtenção de um *background* do cenário do estudo de caso
- Entrevista específica para obter a opinião dos membros da equipe quanto a relação entre práticas ágeis adotados e a taxa de defeitos do software.

As entrevistas aplicadas se encontram na seção Questionários em Forma de Entrevistas ao final do capítulo.

Observação em Campo

A observação é a interação social entre o pesquisador e os participantes no estudo de caso, os os dados são coletados de forma sistemática [Seaman 2008]. A observação proposta neste estudo é do tipo participativa, onde o pesquisador não está ativamente envolvido no trabalho que está sendo observado [Seaman 2008], e observação sombreamento, onde o pesquisador segue o participante volta e registra suas atividades [Shull, Singer e Sjøberg 2008]. A proposta é realizar a observação na empresa, acompanhando cerimônias das equipes de desenvolvimento e percebendo sua rotina de trabalho em um ambiente real.

Análise de documentação

Análise de documentação é uma técnica centrada na documentação gerada por engenheiros de software [Shull, Singer e Sjøberg 2008]. Neste estudo de caso serão analisados os documentos gerados pelas equipes de desenvolvimento que indicam os defeitos encontrados no sistema.

Nesse contexto, cada equipe tem seus critérios para registro de defeitos encontrados na etapa de desenvolvimento, porém todas as equipes utilizam a mesma ferramenta para esse registro. Os documentos analisados foram os relatórios gerados pela ferramenta em questão.

4.8 Análise

A análise deve indicar se há associação entre a taxa de defeitos e as práticas ágeis adotadas pelas equipes de desenvolvimento. Identificando a forma que tais práticas são realizadas e se seguem ou não os fundamentos dos princípios, logo será possível diagnosticar a relação dos princípios e a taxa de defeitos no software.

As técnicas de análise utilizadas para identificar se há associação foram análise estatística para quantificar os dados, e a Teoria Fundamentada em Dados (do inglês, *grounded theory* foi

escolhida devido ao contexto do trabalho, pois possibilita realizar a associação entre códigos estabelecidos na etapa axial.

A análise estatística é comum em pesquisa quantitativas, porém pode ser utilizada para analisar dados qualitativos, desde que estes possam ser quantificados [Wohlin e Aurum 2014]. Na análise estatística os dados podem ser analisados de forma descritiva ou inferencial [Wohlin e Aurum 2014], neste trabalho foi selecionada a forma descritiva, que envolve a síntese dos dados através da descrição, e os dados são agregados e apresentados através de técnicas estatísticas.

Teoria fundamentada em dados é um método qualitativo de pesquisa, onde coleta e codificação/análise de dados são realizadas iterativamente. [Strauss e Corbin 2008] ressaltam que logo que o pesquisador tenha coletado um conjunto de dados - por exemplo, uma pilha de entrevistas - deve-se seguir para a etapa de codificação. No processo de codificação, o pesquisador poderá ter novas ideias, pensamentos e interpretações sobre os dados sendo codificados e isso influenciará a tomada de decisões sobre retornar aos mesmos entrevistados ou entrevistar novos participantes, ou realizar leitura em documentos específicos ou, ainda, realizar observações em lugares específicos.

Codificação Aberta

A codificação aberta é a quebra, a análise, a comparação, a conceituação, e a categorização dos dados [Conte, Cabral e Travassos 2009]. Como o nome sugere, a ênfase não é delimitada para um fenômeno particular sobre os dados, mas uma investigação aberta a todos os tipos de eventos, ocorrências e entidades de interesse. Durante a codificação atribui-se um conceito ou categoria a uma parte dos dados, normalmente um trabalho linha a linha sobre as respostas de um questionário, ou sobre documentos ou sobre as descrições das observações. Durante esse processo, também lê-se o parágrafo por inteiro para pegar o contexto mais amplo. As categorias devem ser detalhadas em subcategorias e dimensões, enriquecendo-as e contribuindo para a realização dos próximo passos.

Opcionalmente, os códigos podem ser in vivo, cuja terminologia é oriunda dos participantes. Nesse caso, o pesquisador não abstrai ou cria códigos para os dados coletados, mas sim, usa os próprios termos do contexto, dos participantes.

Codificação Axial

Na codificação axial pretende-se identificar os relacionamentos entre categorias de interesse.

O termo axial é utilizado porque a codificação ocorrem em torno do eixo de uma categoria, ligando as categorias no nível das propriedades, próximo do que ocorre na relação entre classes na orientação a objetos. Esse exercício com os códigos/conceitos/dados pode aumentar o entendimento sobre eles, dando mais substância e definindo seu significado precisamente, ampliando o conjunto de ocorrências conhecidas sobre os dados e caracterizando melhor as propriedades.

Nessa codificação inicialmente estamos interessados em definir o fenômeno central. É a categoria central da qual se desenvolve a teoria ou o modelo visual da mesma. Depois, é importante saber as causas da ocorrência desse fenômeno, identificando ou coletando/codificando novos categorias e suas relações que expliquem a origem do fenômeno central. Posteriormente, é necessário identificar quais estratégias ou ações foram realizadas em função desse fenômeno central. Normalmente, um determinado fenômeno acarreta em ações para complementá-lo, justificá-lo, expandi-lo, reforçá-lo, etc. Depois, o contexto e as condições intervenientes que influenciam as estratégias são investigados. O contexto são condições específicas nas quais as estratégias ocorrem; e as condições intervenientes são mais abrangentes, podendo ser sociais, econômicas, políticas, que influenciam as estratégias em resposta ao fenômeno central. Por fim, investigam-se as consequências resultantes dessas estratégias. Ou seja, o resultado das estratégias tomadas pelos participantes, podendo ser positivo, negativo ou neutro [Strauss e Corbin 2008].

Em resumo, a codificação axial será a remontagem dos dados que foram quebrados na codificação aberta, onde o pesquisador busca por respostas à questões do tipo por quê, quando, onde e como entre as categorias. Responder a essas questões ajuda na contextualização do fenômeno, ou seja, ajuda na localização dele dentro de uma estrutura condicional [Strauss e Corbin 2008]. Na codificação axial também se define um diagrama lógico, que é o modelo teórico desenvolvido representando as categorias, suas propriedades e os relacionamentos entre elas, semelhante a um diagrama de classes na orientação objetos, ou diagrama lógico de banco de dados, ou ainda um modelo visual de uma ontologia. Os itens do diagrama são desenhados com caixas e setas indicando o fluxo das atividades.

4.9 Validade da Pesquisa

Segundo [Yin 2003], existem quatro testes comumente utilizados para estabelecer a qualidade de qualquer pesquisa social empírica, tal qual estudo de caso.

Os testes são validade de construção, validade interna, validade externa e confiabilidade [Yin 2003].

- validade de construção - usaremos várias fontes de evidência, estabelecendo cadeias de evidência (observação em campo, questionários em forma de entrevistas e análises de documentação). Segundo [Yin 2003], este teste é importante para estabelecer as medidas operacionais corretas para os conceitos estudados, evitando que o julgamento ”subjetivo” do pesquisador.

- validade interna - é adequada apenas para estudos explanatórios, não é adequada para estudos exploratórios.

- validade externa - este estudo pode ser generalizado em empresas com cenários semelhantes (domínio, metodologia, organização, etc.). A generalização não é automática, logo seria necessário replicar este estudo em diversos cenários semelhantes, onde os mesmos resultados devem ocorrer. Uma vez que as replicações sejam feitas, os resultados podem ser aceitos e passarem a servir de suporte para a teoria.

- confiabilidade - este teste consiste em seguir os mesmos procedimentos e conduzir o mesmo estudo de caso outra vez, com o objetivo de minimizar os erros [Yin 2003]. Por questões de cronograma esse teste não foi realizado neste trabalho.

Esta pesquisa buscou tomar as precauções necessárias para garantir a confiabilidade do estudo.

4.9.1 Ameaças à validade da Pesquisa

- A baixa participação dos membros das equipes pode afetar os resultados apresentados pelo estudo de caso.

- os resultados não podem ser considerados conclusivos – são somente indícios da aplicabilidade da abordagem avaliada [Conte e Travassos 2009].

- A rotatividade de membros das equipes pode influenciar nos resultados.

4.10 Relatórios

Este estudo de caso será relatado como uma monografia de conclusão de curso de graduação em Ciência da Computação.

4.11 Questionários em Forma de Entrevistas

Foram aplicados três questionários, o primeiro de *backgroumd*, o segundo de práticas e o terceiro de *feedback*.

4.11.1 Questionário de *Background*

Os questionários de *background* foram aplicados a todos os membros, sem distinção de equipe e os resultados estão na tabela 4.2.

Questionário de *Background*

DADOS DO RESPONDENTE

Qual o seu cargo na empresa?

Qual sua formação? Especifique.

Quanto tempo de formação?

Quanto tempo de experiência em projetos ágeis? E em desenvolvimento de software em geral?

Qual sua função atual dentro da equipe?

Quais outras funções você já desempenhou em projetos ágeis?

DADOS DA EQUIPE DE DESENVOLVIMENTO

Quanto tempo na sua atual equipe de desenvolvimento? E em outras equipes, se houver.

Qual/Quais projeto(s) sua equipe trabalha atualmente?

Quais métodos ágeis sua equipe adota no processo de desenvolvimento?

( ) XP (Extreme Programming)

( ) Scrum

( ) Kanban

( ) Lean

( ) FDD (Feature-Driven Development)

( ) ASD (Adaptive Software Development)

( ) Outros (Cite):

DADOS TÉCNICOS

Quais práticas ágeis são realizadas pela sua equipe? Alguma adaptação realizada na prática?

Sobre princípios ágeis, cite quais você conhece. Na sua equipe, é possível perceber algum princípio ágil sendo empregado? Qual (is)?

4.11.2 Questionário de Práticas

Os questionários de prática foram aplicados apenas aos *Scrum Masters* de cada equipe. Este questionário serviu como base para o questionário de *feedback*.

Questionário de *Práticas*

Quais dessas Práticas Ágeis são aplicadas em sua equipe de desenvolvimento?

Prática ágil	Assinalar
Avaliação De Riscos Ágeis	
Cliente Presente	
Código Limpo	
Código Padronizado	
Documentação Tardia	
Entregas Frequentes	
Estimativas Em Pontos	
Integração Contínua	
Jogo Do Planejamento	
Preparação Do *Backlog*	
Priorização De Requisitos	
Programação Em Par	
Projeto Simples	
Propriedade Coletiva	
Quadro Kanban	
Rastreamento De Bugs	
Refatoração	
Reunião De Planejamento	
Reunião De Retrospectiva	
Reunião Diária	
Reunião de Revisão	
Ritmo Sustentável	
Tempo Em Progresso Limitado	
Teste Automatizado	
Teste De Aceitação	
Teste De Unidade	
Timebox Fixa	
Times Auto-Organizáveis	

4.11.3 Questionário de *Feedback*

Os questionários de *feedback* tiveram como base os questionários de práticas ágeis, logo foi necessária a aplicação de uma versão de questionário para cada equipe. Cada versão é ligeiramente semelhante, porém as práticas listadas são especificas para cada equipe.

4.11.3.1 Equipe A

Questionário de *Feedback*

As práticas ágeis abaixo são utilizadas pelo seu time de desenvolvimento. Assinale, baseado na sua opinião, quais delas estão associadas positivamente, negativamente ou não estão associadas com os defeitos (bugs, defeitos escapados, etc.) do software. Onde:

Associada positivamente indica que o uso da prática ágil tem contribuído para reduzir os defeitos do software. Associada negativamente indica que o uso da prática ágil tem contribuído para aumentar os defeitos do software.

Não associada indica que o uso da prática ágil não possui associação com a ocorrência de defeitos do software. Apenas Associada indica que o uso da prática ágil tem alguma associação, mas não está (ou não se sabe se está) reduzindo nem aumentando o número de defeitos do software. Não sei indica que não se sabe se existe associação entre o uso da prática ágil e a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Código Limpo					
Entregas Frequentes					
Estimativas Em Pontos					
Preparação Do *Backlog*					
Priorização De Requisitos					
Programação Em Par					
Propriedade Coletiva					
Quadro Kanban					
Refatoramento					
Reunião de Planejamento					
Reunião de Retrospectiva					
Reunião de Revisão					
Reunião Diária					
Ritmo Sustentável					
Tempo Em Progresso Limitado					
Teste Automatizado					
Teste De Unidade					
Timebox Fixa					
Times Auto-Organizáveis					

4.11.3.2 Equipe B

Questionário de *Feedback*

As práticas ágeis abaixo são utilizadas pelo seu time de desenvolvimento. Assinale, baseado na sua opinião, quais delas estão associadas positivamente, negativamente ou não estão

associadas com os defeitos (bugs, defeitos escapados, etc.) do software. Onde:

Associada positivamente indica que o uso da prática ágil tem contribuído para reduzir os defeitos do software. Associada negativamente indica que o uso da prática ágil tem contribuído para aumentar os defeitos do software.

Não associada indica que o uso da prática ágil não possui associação com a ocorrência de defeitos do software. Apenas Associada indica que o uso da prática ágil tem alguma associação, mas não está (ou não se sabe se está) reduzindo nem aumentando o número de defeitos do software. Não sei indica que não se sabe se existe associação entre o uso da prática ágil e a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Cliente Presente					
Código Limpo					
Código Padronizado					
Entregas Frequentes					
Preparação Do *Backlog*					
Priorização De Requisitos					
Propriedade Coletiva					
Quadro *Kanban*					
Refatoramento					
Reunião de Planejamento					
Reunião de Retrospectiva					
Reunião de Revisão					
Reunião Diária					
Ritmo Sustentável					
Tempo em Progresso Limitado					
Teste de Aceitação					
Times Auto-Organizáveis					

4.11.3.3 Equipe C

Questionário de *Feedback*

As práticas ágeis abaixo são utilizadas pelo seu time de desenvolvimento. Assinale, baseado na sua opinião, quais delas estão associadas positivamente, negativamente ou não estão associadas com os defeitos (bugs, defeitos escapados, etc.) do software. Onde:

Associada positivamente indica que o uso da prática ágil tem contribuído para reduzir os defeitos do software. Associada negativamente indica que o uso da prática ágil tem contribuído

para aumentar os defeitos do software.

Não associada indica que o uso da prática ágil não possui associação com a ocorrência de defeitos do software. Apenas Associada indica que o uso da prática ágil tem alguma associação, mas não está (ou não se sabe se está) reduzindo nem aumentando o número de defeitos do software. Não sei indica que não se sabe se existe associação entre o uso da prática ágil e a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas Associada	Não sei
Avaliação de Riscos Ágeis					
Código Limpo					
Código Padronizado					
Documentação Tardia					
Estimativas em Pontos					
Integração Contínua					
Jogo do Planejamento					
Preparação do *Backlog*					
Priorização de Requisitos					
Programação em Par					
Projeto Simples					
Propriedade Coletiva					
Quadro Kanban					
Rastreamento de *Bugs*					
Refatoramento					
Reunião de Planejamento					
Reunião de Retrospectiva					
Reunião de Revisão					
Reunião Diária					
Ritmo Sustentável					
Testes Automatizados					
Teste de Aceitação					
Teste de Unidade					
Timebox Fixa					
Times Auto-Organizáveis					

4.11.3.4 Equipe D

Questionário de *Feedback*

As práticas ágeis abaixo são utilizadas pelo seu time de desenvolvimento. Assinale, baseado na sua opinião, quais delas estão associadas positivamente, negativamente ou não estão associadas com os defeitos (bugs, defeitos escapados, etc.) do software. Onde:

Associada positivamente indica que o uso da prática ágil tem contribuído para reduzir os defeitos do software. Associada negativamente indica que o uso da prática ágil tem contribuído

para aumentar os defeitos do software.

Não associada indica que o uso da prática ágil não possui associação com a ocorrência de defeitos do software. Apenas Associada indica que o uso da prática ágil tem alguma associação, mas não está (ou não se sabe se está) reduzindo nem aumentando o número de defeitos do software. Não sei indica que não se sabe se existe associação entre o uso da prática ágil e a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Código Limpo					
Entregas Frequentes					
Estimativas Em Pontos					
Preparação Do *Backlog*					
Priorização De Requisitos					
Programação Em Par					
Rastreamento de *Bugs*					
Refatoramento					
Reunião de Planejamento					
Reunião de Retrospectiva					
Reunião de Revisão					
Reunião Diária					
Tempo em Progresso Limitado					
Teste Automatizado					
Teste de Aceitação					
Teste de Unidade					
Times Auto-Organizáveis					

Capítulo 5

Resultados da Coleta de Evidências

Neste Capítulo são expressos os resultados de todas as evidências coletadas, como respostas dos questionários, pareceres sobre a observação e a contabilização de defeitos por *sprint* proveniente da documentação.

5.1 Questionários

Ao todo foram aplicados três questionários nas equipes de desenvolvimento, o primeiro deles é chamado de Questionário de *Background*, que serve obter dados dos membros das equipes, como participação, formação e conhecimento sobre *Agile*. O segundo questionário foi aplicado apenas aos *Scrum Masters* das 4 equipes de desenvolvimento, e foi perguntado quais as práticas ágeis os equipes estavam utilizando. O terceiro questionário aplicado as equipes de desenvolvimento é chamado Questionário de *Feeback* e serviu para descobrir a percepção das equipes sobre a associação entre os métodos ágeis e a quantidade de defeitos encontrados no software.

As respostas do Questionário de *Background* auxiliaram na construção do cenário do Estudo de Caso. Mais informações sobre o cenário construído podem ser vistas no capítulo 4, na seção de Papéis de Procedimentos do Estudo de Caso.

Prática Ágil
Avaliação De Riscos Ágeis
Cliente Presente
Código Limpo
Código Padronizado
Documentação Tardia
Entregas Frequentes
Estimativas Em Pontos
Integração Contínua
Jogo Do Planejamento
Preparação Do *Backlog*
Priorização De Requisitos
Programação Em Par
Projeto Simples
Propriedade Coletiva
Quadro *Kanban*
Rastreamento De Bugs
Refatoramento
Reunião De Planejamento
Reunião De Retrospectiva
Reunião Diária
Reunião Revisão
Ritmo Sustentável
Tempo Em Progresso Limitado
Teste Automatizado
Teste De Aceitação Teste De Unidade
Timebox Fixa
Times Auto-Organizáveis

Tabela 5.1: Práticas ágeis utilizadas pelos Times A, B, C e D segundo os *Scrum Masters*

A tabela 5.1 mostra quais práticas ágeis são utilizadas pelos 4 Times de desenvolvimento, as equipes podem optar por não utilizar determinadas práticas de acordo com suas necessidades.

O Questionário de *Feeback* foi aplicado em 4 versões, uma para cada equipe, e cada questionário contém as práticas ágeis utilizadas pelo respectiva equipe de desenvolvimento ágil, conforme as respostas do questionário anterior.

A primeira coluna contém práticas ágeis utilizadas pela equipe de desenvolvimento ágil na empresa ABC e as colunas seguintes contém uma afirmação quanto às práticas e suas associações com os defeitos encontrados no software.

As associações consideradas foram:

- Associada positivamente: O uso da prática tem contribuído para reduzir a quantidade de defeitos do software.
- Associada negativamente: O uso da prática tem contribuído para aumentar a quantidade de defeitos do software
- Não associada: O uso da prática não está associado com a quantidade de defeitos.
- Apenas associada: O uso da prática está associado com quantidade de defeitos, porém não se sabe como exatamente.
- Não sei: Não se sabe se existe associação entre o uso da prática ágil e a ocorrência de defeitos.

As associações tem como objetivo captar as percepções das equipes de desenvolvimento ágil quanto às práticas utilizadas pelo mesmo e nesse contexto, suas associações com os defeitos encontrados no software durante a etapa de desenvolvimento.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Código Limpo	5	0	0	1	0
Entregas Frequentes	5	0	0	0	1
Estimativas em Pontos	3	0	2	1	0
Preparação Do *Backlog*	4	0	1	1	0
Priorização De Requisitos	2	0	1	3	0
Programação Em Par	6	0	0	0	0
Propriedade Coletiva	3	0	0	2	1
Quadro *Kanban*	3	0	0	3	0
Refatoração	4	0	0	2	0
Reunião De Planejamento	4	0	1	1	0
Reunião De Retrospectiva	5	0	1	0	0
Reunião Diária	3	0	1	2	0
Reunião de Revisão	3	0	1	2	0
Ritmo Sustentável	3	0	1	2	0
Tempo em Progresso Limitado	5	0	1	0	0
Testes Automatizados	6	0	0	0	0
Testes de Unidade	6	0	0	0	0
Timebox Fixa	2	1	1	2	0
Times Auto-Organizáveis	4	0	0	2	0

Tabela 5.2: Respostas dos Questionários de *Feedback* da equipe A

A tabela 5.2 contabiliza as respostas da equipe de desenvolvimento A.

Como pode ser observado, os membros dessa equipe acreditam que práticas ágeis como Programação em Par, Testes Automatizados e Testes de Unidade tem contribuído para a redução do número de defeitos no software.

Em contrapartida, a prática Estimativas em Pontos foi que a dois membros da equipe consideraram não estar associada com a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Cliente Presente	5	1	0	0	0
Código Limpo	3	0	0	1	2
Código Padronizado	4	0	0	1	1
Preparação Do *Backlog*	2	0	2	1	1
Priorização De Requisitos	3	0	1	2	0
Propriedade Coletiva	1	0	1	1	1
Quadro *Kanban*	2	0	1	1	2
Refatoração	2	2	0	1	1
Reunião De Planejamento	5	0	1	0	0
Reunião De Retrospectiva	5	0	1	0	0
Reunião Diária	4	0	0	1	0
Ritmo Sustentável	3	0	0	1	2
Tempo Em Progresso Limitado	3	0	1	1	1
Teste De Aceitação	4	0	0	1	1
Times Auto-Organizáveis	5	0	0	0	1

Tabela 5.3: Respostas dos Questionários de *Feedback* da equipe B

A tabela 5.3 Contabiliza as respostas dos membros da equipe B, não houve unanimidade em nenhuma prática ágil.

A equipe considerou as práticas ágeis Cliente Presente, Reunião de Planejamento, Reunião de Retrospectiva e Times Auto-Organizáveis as práticas que tem melhor contribuído para reduzir a quantidade de defeitos.

A prática da Refatoração foi a que obteve maior associação negativa, e a prática de Preparação do *Backlog* foi a que mais membros consideraram não possuir associação com a ocorrência de defeitos. As razões dessas respostas serão explicadas na seção de Observação em Campo.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Avaliação De Riscos Ágeis	2	0	0	2	1
Código Limpo	4	0	1	0	0
Código Padronizado	3	0	1	0	1
Documentação Tardia	1	0	2	1	1
Estimativas Em Pontos	4	0	0	0	1
Integração Contínua	5	0	0	0	
Jogo Do Planejamento	4	0	0	1	0
Preparação Do *Backlog*	5	0	0	0	0
Priorização De Requisitos	5	0	0	0	0
Programação Em Par	5	0	0	0	0
Projeto/Design Simples	3	0	1	0	1
Propriedade Coletiva	5	0	0	0	0
Quadro *Kanban*	2	0	0	3	0
Rastreamento De *Bugs*	1	0	0	0	4
Refatoração	2	0	0	1	2
Reunião De Planejamento	5	0	0	0	0
Reunião De Retrospectiva	5	0	0	0	0
Reunião Diária	5	0	0	0	0
Reunião Revisão	5	0	0	0	0
Ritmo Sustentável	3	0	1	0	1
Testes Automatizados	4	0	0	1	0
Testes de Aceitação	2	0	0	3	0
Teste de Unidade	5	0	0	0	0
Timebox Fixa	3	0	0	2	0
Times Auto-Organizáveis	4	0	0	0	1

Tabela 5.4: Respostas dos Questionários de *Feedback* da equipe C

A tabela 5.4 contabiliza as respostas da equipe C, essa equipe utiliza o maior número de práticas ágeis.

A equipe considerou que a práticas como Integração Contínua, Programação em Par, Cerimônias do método *Scrum* entre outras, estão contribuindo para reduzir o número de defeitos do software e nenhuma prática tem contribuído para aumentar esse número.

A prática de Documentação Tardia foi que a que mais membros consideraram não estar associada com a ocorrência de defeitos.

Práticas Ágeis	Associada positivamente	Associada negativamente	Não associada	Apenas associada	Não sei
Código Limpo	4	0	0	0	0
Entregas Frequentes	4	0	0	0	0
Estimativas Em Pontos	1	0	1	1	1
Preparação Do *Backlog*	2	0	1	0	1
Priorização De Requisitos	2	0	1	1	0
Programação Em Par	1	0	2	0	1
Rastreamento De *Bugs*	4	0	0	0	0
Refatoração	2	0	1	1	0
Reunião De Planejamento	3	0	0	1	0
Reunião De Retrospectiva	2	0	0	1	1
Reunião Diária	2	0	0	2	0
Reunião Revisão	3	0	0	1	0
Tempo Em Progresso Limitado	1	0	1	2	0
Testes Automatizados	4	0	0	0	0
Teste De Aceitação	4	0	0	0	0
Teste De Unidade	4	0	0	0	0
Times Auto-Organizáveis	3	0	1	0	0

Tabela 5.5: Respostas dos Questionários de *Feedback* da equipe D

A tabela 5.5 contabiliza as respostas da equipe D. Como pode ser visto na tabela, a equipe considera práticas ágeis como Código Limpo [Martin 2009], Entregas Frequentes, Rastreamento de *Bugs*/Defeitos e práticas relacionadas à testes como as que tem maior contribuição para reduzir a quantidade de defeitos.

Nessa equipe, a prática ágil Programação em Par foi considerada por mais membros como não associada aos defeitos de software, isso ocorre em contrapartida a dois outras equipes que utilizam a mesma prática, a explicação será melhor detalhada na seção de Observação em Campo.

5.2 Análise de Documentação

A Documentação analisada foi o controle de defeitos por *sprint* feito por cada equipe. Todos os 4 equipes utilizam um software específico para gerenciamento de tarefas durante uma *sprint* e este software fornece tabelas com os defeitos que foram encontrados.

Nenhuma equipe registrou quais práticas utilizou em cada *sprint*, logo consideramos que as práticas foram utilizadas no período todo.

Os períodos das *sprints* variam de acordo com as equipes.

Sprint	21	22	23	25	30	36	37	38	40	41	42	43	44	45	48	49	50	51	54	55	56	57	59
Defeitos	5	4	2	3	3	4	6	1	3	1	1	4	5	7	3	3	1	1	2	1	1	1	2

Tabela 5.6: Defeitos por *Sprint* Time A

A tabela 5.6 enumera a quantidade de defeitos por *Sprint*, o Time de desenvolvimento A só contabiliza os Defeitos escapados, ou seja, os defeitos que foram encontrados no lado do cliente. Foram coletados defeitos da *Sprint* 21 até a *Sprint* 59 e dentre essas, foram encontrados 64 defeitos escapados em 23 *Sprints*. Uma taxa de 2.78 Defeitos escapados por *Sprint*.

Sprint	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Defeitos	15	11	1	1	3	5	3	2	4	5	5	9	10	2	27	17	9	5	2	2	5	7	8	5	5	3	11	3	7	5

Tabela 5.7: Defeitos por *Sprint* Time B

A equipe de desenvolvimento B é voltada para manutenção de software, logo a maioria das tarefas está relacionada à defeitos. Foram coletadas informações sobre 31 *Sprints* e ao todo foram encontrados 196 defeitos na maioria destas, com exceção da *Sprint* 9. Uma taxa de 6.53 Defeitos por *Sprint*.

Sprint	1	2	3	4	5	6	7	8	9	10	11	12
Defeitos	9	1	5	5	10	17	6	6	5	3	8	23

Tabela 5.8: Defeitos por *Sprint* Time C

A equipe de desenvolvimento C trabalhou em um módulo específico do sistema, com um total de 12 *Sprints*, foram encontrados defeitos em todas elas, contabilizando um total de 98 defeitos. Uma taxa de 8.17 Defeitos por *Sprint*.

Sprint	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Defeitos	14	13	6	18	37	9	22	10	16	8	13	12	13	32	5	7	13	12	14	24	30	18	15	14	17	17	9	10	1

Tabela 5.9: Defeitos por *Sprint* Time D

Foram coletadas dados relacionados à 29 *Sprints* do Time de desenvolvimento D, ao todo foram encontrados 429 defeitos. Uma taxa de 14.8 Defeitos por *Sprint*.

5.3 Observação em Campo

No período de março/2015 à dezembro/2015 foram realizadas visitas à empresa ABC onde foram realizadas observações dos tipos participativa e sombra. O foco das observações foi descobrir quais práticas ágeis são utilizadas e como são, além de buscar captar se há associação de tais práticas com os defeitos encontrados durante o desenvolvimento do software.

Ao observar o trabalho das quatro equipes de desenvolvimento ágil, é possível perceber o uso das práticas ágeis do método *Scrum*, na empresa ABC, as reuniões são chamadas cerimônias, essas buscam a maior fidelidade possível com o método original.

A empresa ABC também tem como meta estratégica melhorar a qualidade dos seus produtos, essa meta se estende a fábrica.

Um caso que ocorreu no período de observação, foi uma exigência de refatoração de códigos do sistema para todos os módulos do produto, um trabalho chamado por vezes de ”mal necessário”, porém é importante para evitar problemas no futuro.

5.3.1 Equipe A

A equipe A usa as práticas ágeis do método *Scrum* da forma mais fiel possível ao método, havendo um controle rígido do uso correto das práticas. Segundo observações e opiniões de membros de outras equipes, esta equipe possui certa maturidade no uso de práticas como Testes automatizados e Programação em Par.

5.3.2 Equipe B

Por se tratar de uma equipe focada em manutenção de software, algumas prática ágeis necessitam ser adaptadas, a prática ágil Preparação do *Backlog*, citada na seção de Questionários, é feita de forma adaptada porque as requisições do cliente podem chegar com urgência e o planejamento deve ser refeito, logo alguns membros da equipe não conseguem perceber o uso da prática.

Segundo uma parte da equipe, a prática ágil Refatoração aumenta o número de defeitos, a explicação vem do fato de que quando o código é refatorado, defeitos podem ser encontrados por diversas razões, como por exemplo, dependência de outras aplicações.

Nessa equipe a prática ágil Código Limpo ainda está em uma fase inicial, logo as respostas foram diferentes das equipes que possuem maturidade quanto ao uso dessa prática.

5.3.3 Equipe C

A equipe C utiliza as práticas de forma mais fiel possível ao método. De acordo com os questionários e observações, a equipe parece concordar com o uso das práticas empregadas, o que pode ser observado na tabela 5.4, onde as respostas são bastante condizentes entre a maioria dos membros dessa equipe.

5.3.4 Equipe D

A equipe D usa práticas ágeis adaptadas, muito por conta do perfil da equipe e de situações onde a equipe é reorganizada.

A prática ágil Programação em Par, como foi comentado anteriormente na seção de Questionários, foi considerada por mais membros da equipe como Não associada a ocorrência de defeitos. Segundo membros da equipe, a prática vem sendo utilizada de forma adaptada e de forma pouco frequente.

Capítulo 6

Análise

Neste capítulo serão realizadas as análises dos resultados obtidos na pesquisa, os resultados estão expressos no capítulo anterior.

Para ser realizada a análise foram selecionadas duas técnicas, Análise Estatística e Teoria Fundamentada em Dados.

6.1 Análise Estatística

Segundo [Gomes 1990], pode-se definir a Estatística como a Matemática aplicada aos dados de observação. Mas tais dados são, em vários casos, obtidos por meio de trabalhos feitos propositalmente e em condições previamente determinadas, ou seja, especificadas. Nesse caso, temos então os dados experimentais, obtidos por meio de experimentos. O que dificulta o trabalho do experimentador e exige a análise estatística é a presença, em todos os dados obtidos, de efeitos de fatores não controlados (que podem ser controláveis ou não). Esses efeitos, sempre presentes, não podem ser conhecidos individualmente e alteram, pouco ou muito, os resultados, logo é tarefa do experimentador verificar se as diferenças observadas num experimento têm ou não têm valor, isto é, se são ou não significativas.

Nesta seção, são apresentadas estatísticas sobre a taxa de defeitos por *Sprint* e também sobre a associação com as práticas ágeis.

Equipe	Quantidade de Defeitos	Número de *Sprints*	Defeitos por *Sprint*
A	64	39	1.6
B	196	31	6.32
C	98	12	8.17
D	429	29	14.79

Tabela 6.1: Tabela de Defeitos por *Sprint* para todas as equipes

A tabela 6.1 apresenta dados das quatro equipes e suas taxas de defeitos por *Sprint*, a equipe A teve a menor taxa de defeitos enquanto a equipe D teve a maior. Como foi explicado no capítulo anterior, a equipe A só registra Defeitos escapados, por isso tem a taxa bem abaixo das demais equipes, se fossem registrados todos os defeitos encontrados pela equipe, possivelmente os números seriam diferentes.

As tabelas seguintes, 6.2, 6.3, 6.4 e 6.5, contém uma nova contabilização das respostas dos questionários de *Feedback*, considerando apenas se as práticas ágeis estão associadas ou não a taxa de defeitos, onde:

- Associada: significa que o uso da respectiva prática ágil tem associação (positiva, negativa ou não determinada) com a ocorrência de defeitos.
- Não Associada: significa que o uso da respectiva prática ágil não tem associação, ou esta é imperceptível, com a ocorrência de defeitos.

As tabelas mostram as percepções das equipes de desenvolvimento ágil sobre a associação entre o uso de práticas ágeis e a taxa de defeitos por *Sprint*.

Prática Ágil	Associada	Não Associada
Código Limpo	6	0
Entregas Frequentes	5	1
Estimativas em Pontos	4	2
Preparação Do *Backlog*	5	1
Priorização De Requisitos	5	1
Programação Em Par	6	0
Propriedade Coletiva	5	1
Quadro *Kanban*	6	0
Refatoração	6	0
Reunião De Planejamento	5	1
Reunião De Retrospectiva	5	1
Reunião Diária	5	1
Reunião de Revisão	5	1
Ritmo Sustentável	5	1
Tempo em Progresso Limitado	5	1
Testes Automatizados	6	0
Testes de Unidade	6	0
Timebox Fixa	5	1
Times Auto-Organizáveis	6	0

Tabela 6.2: Tabela Geral de Associação entre práticas e Defeitos da equipe A

Segundo 100% dos membros da equipe A, 36.8% das práticas ágeis associadas estão com a taxa de defeitos e 86.33% dos membros da equipe acreditam que 57.8% das demais práticas também estão associadas.

Por outro lado, 0.33% da equipe A acredita que 5.2% das prática ágeis, o que equivale a uma prática, não está associada com a taxa de defeitos.

Logo, a maioria dos membros dessa equipe de desenvolvimento acredita que existe associação entre práticas ágeis e a taxa de defeitos.

Prática Ágil	Associada	Não Associada
Cliente Presente	5	1
Código Limpo	4	2
Código Padronizado	5	1
Preparação Do *Backlog*	3	3
Priorização De Requisitos	5	1
Propriedade Coletiva	2	2
Quadro *Kanban*	3	3
Refatoração	3	3
Reunião De Planejamento	5	1
Reunião De Retrospectiva	5	1
Reunião Diária	5	0
Ritmo Sustentável	4	2
Tempo Em Progresso Limitado	4	2
Teste De Aceitação	5	1
Times Auto-Organizáveis	5	1

Tabela 6.3: Tabela Geral de Associação entre práticas e Defeitos da equipe B

A equipe B não respondeu de forma unanime, porém de acordo com 83.33% dos membros dessa equipe, mais de 53% das práticas ágeis utilizadas tem associação com a taxa de defeitos.

Em contrapartida, 50% dos membros dessa equipe acreditam que 20% das práticas não possuem associação.

Prática Ágil	Associada	Não Associada
Avaliação De Riscos Ágeis	4	1
Código Limpo	4	1
Código Padronizado	3	2
Documentação Tardia	2	3
Estimativas Em Pontos	4	1
Integração Contínua	5	0
Jogo Do Planejamento	5	0
Preparação Do *Backlog*	5	0
Priorização De Requisitos	5	0
Programação Em Par	5	0
Projeto/Design Simples	3	2
Propriedade Coletiva	5	0
Quadro *Kanban*	5	0
Rastreamento De *Bugs*	1	4
Refatoração	3	2
Reunião De Planejamento	5	0
Reunião De Retrospectiva	5	0
Reunião Diária	5	0
Reunião Revisão	5	0
Ritmo Sustentável	3	2
Testes Automatizados	5	0
Testes de Aceitação	5	0
Teste de Unidade	5	0
Timebox Fixa	4	1
Times Auto-Organizáveis	4	1

Tabela 6.4: Tabela Geral de Associação entre práticas e Defeitos da Equipe C

A equipe C, como um todo, percebeu associação entre as 14 de 25 práticas ágeis adotadas com a taxa de defeitos, ou seja, 100% dessa equipe acredita que 56% das práticas adotadas estão associadas com a taxa de defeitos.

Por outro lado, 80% da equipe C, acredita que a prática ágil Rastreamento de *Bugs*, não está associada com a taxa de defeitos.

Prática Ágil	Associada	Não Associada
Código Limpo	4	0
Entregas Frequentes	4	0
Estimativas Em Pontos	2	2
Preparação Do *Backlog*	2	2
Priorização De Requisitos	3	1
Programação Em Par	1	3
Rastreamento De *Bugs*	4	0
Refatoração	3	1
Reunião De Planejamento	4	0
Reunião De Retrospectiva	3	1
Reunião Diária	4	0
Reunião Revisão	4	0
Tempo Em Progresso Limitado	3	1
Testes Automatizados	4	0
Teste De Aceitação	4	0
Teste De Unidade	4	0
Times Auto-Organizáveis	3	1

Tabela 6.5: Tabela Geral de Associação entre práticas e Defeitos da equipe D

Na equipe D, houve unanimidade em 9 das 17 práticas ágeis adotados, isto é, a equipe toda considerou haver associação de mais de 50% das práticas ágeis adotadas com a taxa de defeitos.

A prática ágil que foi menos considerada como associada foi a programação em par, que como foi explicado anteriormente, é feita de maneira adaptada da forma descrita na literatura.

6.2 Teoria Fundamentada em Dados

Nesse estudo foram utilizadas as duas primeiras fases, pois após a codificação axial foi possível responder as questões de pesquisa e não se fez necessário o uso da codificação seletiva.

As codificações aberta e axial foram realizadas para as quatro equipes de desenvolvimento ágil.

A categoria Práticas Ágeis é dividida em subcategorias, sendo essa as práticas utilizadas por cada equipe.

A categoria Defeito não precisou ser subdividida, pois Defeito foi utilizado na pesquisa como termo geral, que engloba *Bugs*, falhas, erros entre outros. Esta categoria foi a principal e se relacionar com as subcategorias de Práticas ágeis.

Na codificação axial, os códigos que possuem relação direta com o código Defeitos são os que foram escolhidos por mais membros das respectivas equipes como associados, os demais códigos se relacionam entre si.

6.2.1 Equipe A

6.2.1.1 Codificação Aberta

Na etapa de codificação aberta são criados códigos para 2 categorias encontradas e suas respectivas subcategorias, como pode ser observado na tabela 6.6.

Categoria	Subcategorias
Práticas Ágeis	Código Limpo Entregas Frequentes Estimativas em Pontos Preparação do *Backlog* Priorização de Requisitos Programação em Par Propriedade Coletiva Quadro *Kanban* Refatoração Reunião de Planejamento Reunião de Retrospectiva Reunião de Revisão Reunião Diária Ritmo Sustentável Tempo em Progresso Limitado Testes Automatizados Testes de Unidade *Timebox* Fixa Times Auto-Organizáveis
Defeitos	

Tabela 6.6: Categorias e subcategorias para a codificação aberta da equipe A

As práticas ágeis adotadas pela equipe A foram selecionadas como códigos da categoria Práticas Ágeis, nessa equipe não há registros de práticas ágeis adaptadas.

6.2.1.2 Codificação Axial

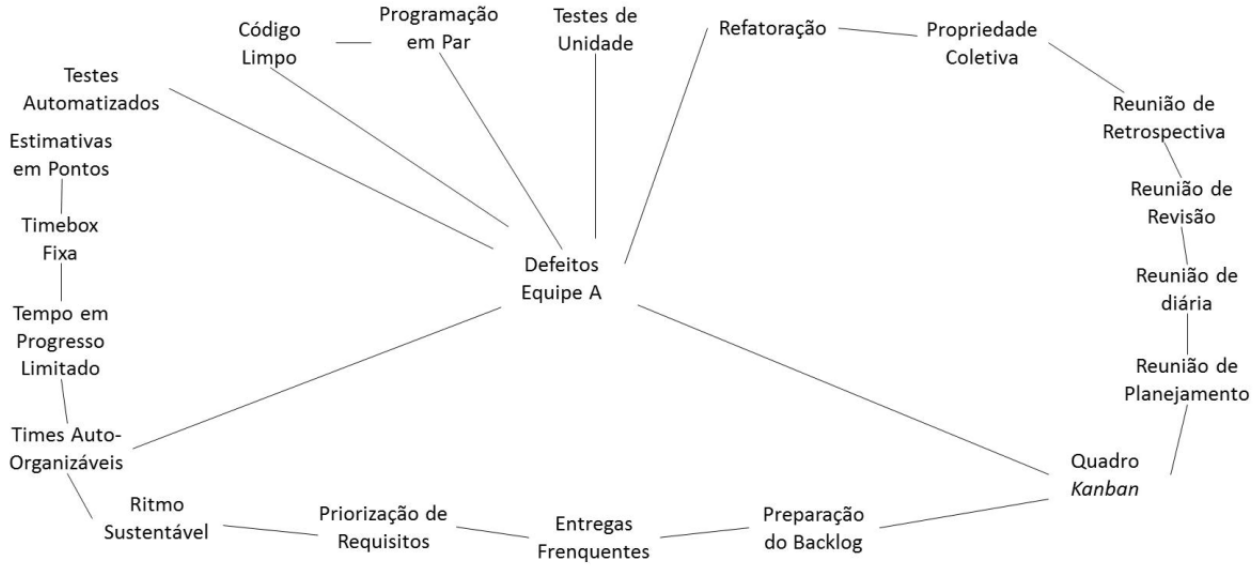

Figura 6.1: Codificação Axial da equipe A

A figura 6.1 é um grafo que expressa a codificação axial para a equipe A, as práticas ágeis possuem associação entre si, e algumas práticas estão diretamente associadas aos Defeitos.

A equipe A não relata utilizar práticas de forma adaptada, nessa equipe, 7 das 19 práticas foram consideradas associadas a taxa de defeitos.

Práticas como programação em par, testes automatizados e código limpo, evidentemente, possuem associação com os defeitos de software.

Essa equipe tem uma taxa de 1.6 defeitos por *sprint*, porém, são contabilizados apenas os defeitos escapados. Outro detalhe é a duração de cada *sprint*, que foi de aproximadamente um mês.

As práticas Times Auto-Organizáveis e Quandro *Kanban* foram escolhidas pelas equipes como diretamente associadas, porém são necessárias mais evidências empíricas para comprovar tais associações.

6.2.1.3 Equipe B

6.2.1.4 Codificação Aberta

Na etapa de codificação aberta são criados códigos para a equipe B nas 2 categorias encontradas e suas respectivas subcategorias, como pode ser observado na tabela 6.7.

Categoria	Subcategorias
Práticas Ágeis	Cliente Presente Código Limpo (Adaptada) Código Padronizado Preparação do *Backlog* (Adaptada) Priorização de Requisitos Propriedade Coletiva Quadro *Kanban* Refatoração Reunião de Planejamento Reunião de Retrospectiva (Adaptada) Reunião Diária Ritmo Sustentável Tempo em Progresso Limitado Testes de Aceitação Times Auto-Organizáveis
Defeitos	

Tabela 6.7: Categorias e subcategorias para a codificação aberta da equipe B

As práticas ágeis adotadas pela equipe B foram selecionadas como códigos da categoria Práticas Ágeis, algumas práticas adotadas por essa equipe foram ditas como adaptadas, são práticas ágeis executadas diferentemente da forma descrita pela literatura.

As práticas adaptadas nessa equipe são a Preparação do *Backlog*, o Código Limpo e a Reunião de Retrospectiva.

Os motivos que levam ao uso de tais práticas de forma adaptada são as próprias regras de negócio da equipe, que por ter foco em manutenção de software, por vezes acaba trabalhando sob demanda, logo o *backlog* do produto não se faz necessário por não haver um produto final e as reuniões de retrospectiva são realizadas eventualmente,devido ao fator tempo.

A prática de código limpo é realizada de forma adaptada pois a equipe ainda está em uma etapa de estudos e treinamento nessa prática, logo falta maturidade a equipess quanto ao uso da prática.

6.2.1.5 Codificação Axial

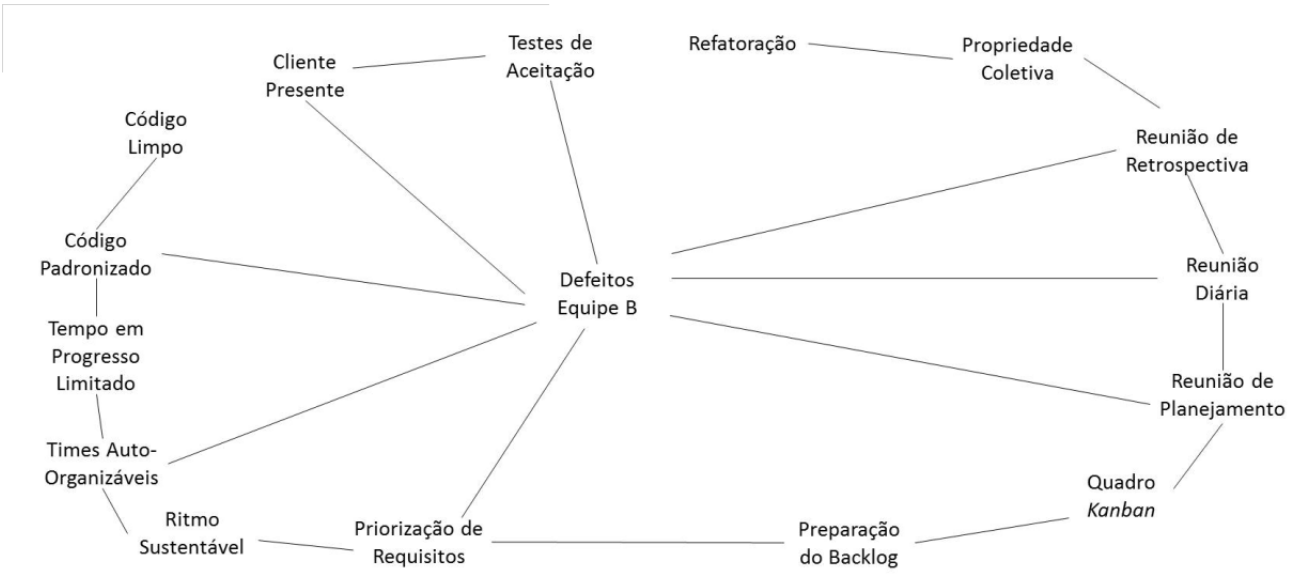

Figura 6.2: Codificação Axial da equipe B

A figura 6.2 é um grafo que expressa a codificação axial para a equipe B, as práticas ágeis possuem associação entre si, e algumas práticas estão diretamente associadas aos Defeitos.

Nessa equipe não houve unanimidade quanto a escolha de práticas associadas a taxa de defeitos, práticas como Testes de aceitação e Código padronizado estão evidentemente associadas, porém algumas práticas que poderiam estar associadas diretamente não estão e práticas que não possuem associação tão evidente foram consideradas como diretamente associadas.

A prática Código limpo poderia ser considerada como diretamente associada, porém, essa equipe está em uma etapa de estudos e treinamento nessa prática, logo a associação não foi considerada evidente por todos os membros da equipe.

Práticas de cerimônias nem sempre são consideradas como diretamente associadas, mas nessa equipe foram consideradas. Para comprovar tais associações são necessárias mais evidências empíricas ou estudos mais específicos.

6.2.1.6 Equipe C

6.2.1.7 Codificação Aberta

Na etapa de codificação aberta são criados códigos para a equipe C nas 2 categorias encontradas e suas respectivas subcategorias, como pode ser observado na tabela 6.8.

Categoria	Subcategorias
Práticas Ágeis	Avaliação de Riscos Ágeis Código Limpo Código Padronizado Documentação Tardia Estimativas em Pontos Integração Contínua Jogo do Planejamento Preparação do *Backlog* Priorização de Requisitos Programação em Par Projeto Simples Propriedade Coletiva Quadro *Kanban* Rastreamento de *Bugs* Refatoração Reunião de Planejamento Reunião de Retrospectiva Reunião de Revisão Reunião Diária Ritmo Sustentável Testes Automatizados Testes de Aceitação Testes de Unidade *Timebox* Fixa Times Auto-Organizáveis
Defeitos	

Tabela 6.8: Categorias e subcategorias para a codificação aberta da equipe C

As práticas ágeis adotadas pela equipe C foram selecionadas como códigos da categoria Práticas Ágeis, nessa equipe não há registros de práticas ágeis adaptadas.

6.2.1.8 Codificação Axial

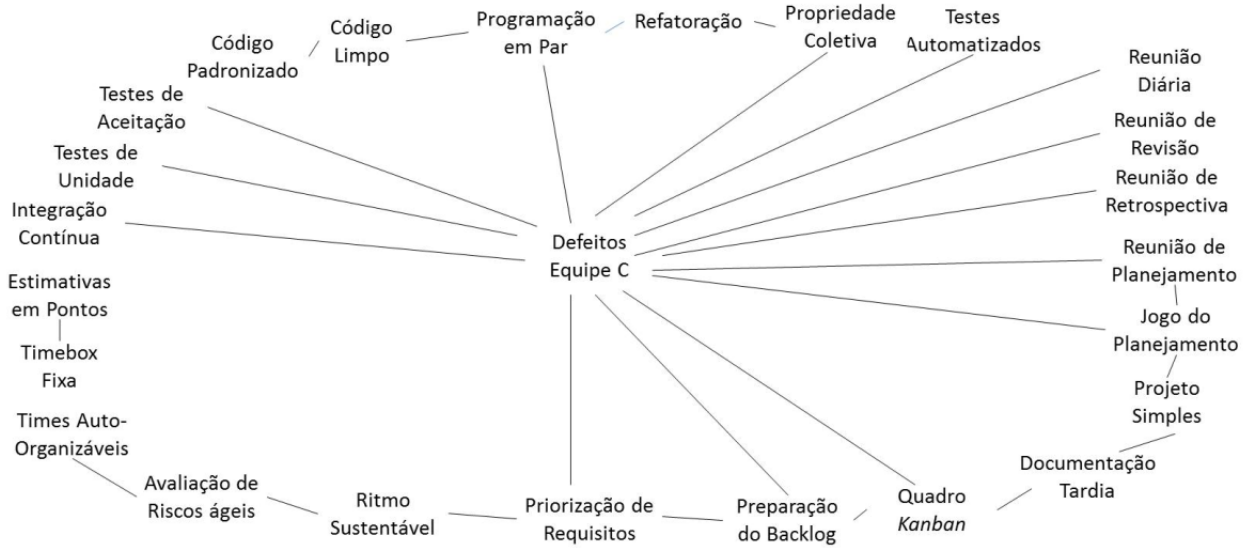

Figura 6.3: Codificação Axial da equipe C

A figura 6.3 é um grafo que expressa a codificação axial para a equipe C, as práticas ágeis possuem associação entre si, e algumas práticas estão diretamente associadas aos Defeitos.

Essa equipe obteve mais respostas unanimes quanto a associações, logo mais associações diretas.

Práticas ágeis como programação em par, testes de unidade e integração contínua estão evidentemente associadas com os defeitos.

Outras práticas não estão tão evidentemente associadas, porém, foram consideradas associadas por outras equipes, por exemplo quadro *kanban* e priorização de requisitos.

As práticas código limpo e código padronizado chamam a atenção por não estarem diretamente associadas, porém o motivo não foi constatado. Nos questionários, respostas alegaram não saber da associação e também que as práticas não estavam associadas.

Existem possibilidades para tais resultados, como por exemplo, o uso dessas práticas de forma adaptada ou a não percepção do uso de tais práticas.

Para obter o motivo real dessa não associação é necessário coletar mais evidências empíricas.

6.2.1.9 Equipe D

6.2.1.10 Codificação Aberta

Na etapa de codificação aberta são criados códigos para a equipe D nas 2 categorias encontradas e suas respectivas subcategorias, como pode ser observado na tabela 6.9.

Categoria	Subcategorias
Práticas Ágeis	Código Limpo Entregas Frequentes (Adaptada) Estimativas em Pontos Preparação do *Backlog* Priorização de Requisitos Programação em Par (Adaptada) Rastreamento de *Bugs* Refatoração Reunião de Planejamento Reunião de Retrospectiva Reunião de Revisão Reunião Diária Tempo em Progresso Limitado Testes Automatizados (Adaptada) Testes de Aceitação Testes de Unidade Times Auto-Organizáveis
Defeitos	

Tabela 6.9: Categorias e subcategorias para a codificação aberta da equipe D

As práticas ágeis adotadas pela equipe D foram selecionadas como códigos da categoria Práticas Ágeis, algumas práticas adotadas por essa equipe foram ditas como adaptadas, são práticas ágeis executadas diferentemente da forma descrita pela literatura.

As práticas adaptadas nessa equipe são as Entregas Frequentes, Programação em Par e Testes Automatizados

Os motivos que levam ao uso de tais práticas de forma adaptada são causadas pela própria gestão, organização e maturidade da equipe. Testes automatizados e programação em par estão relacionados a maturidade da equipe em executar tais práticas.

As entregas frequentes não são tão frequentes devido a regras de negócio.

6.2.1.11 Codificação Axial

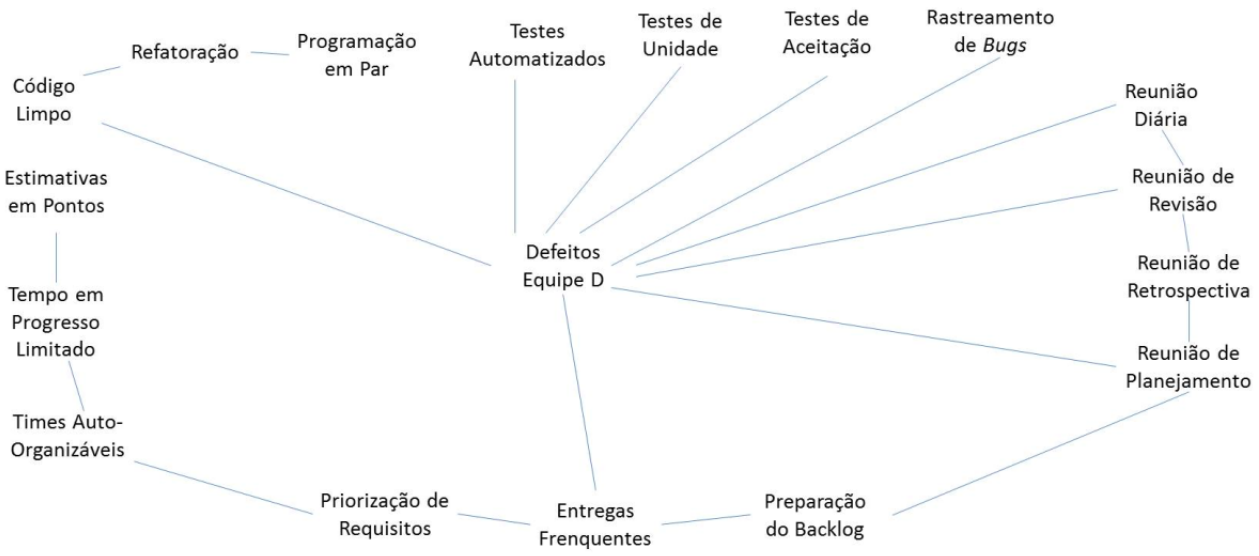

Figura 6.4: Codificação Axial da equipe D

As práticas ágeis relacionadas à testes, rastreamento de *bugs* e código limpo estão, evidentemente, associadas diretamente aos defeitos.

Práticas de cerimônias também foram consideradas como diretamente associadas, assim como ocorre em outras equipes.

A prática ágil entregas frequentes, nesse caso nem tão frequentes, foi considerada como diretamente associada a ocorrência de Defeitos, não há uma evidência que comprove o motivo dessa associação ser percebida de forma tão clara, porém o *feedback* gerado por essa prática pode ser o fator determinante para tal associação.

6.3 Discussão

Neste trabalho foram realizadas duas técnicas de análise para cada uma das quatro equipes de desenvolvimento ágil da empresa ABC.

Práticas ágeis como código limpo, preparação do *backlog*, priorização de requisitos, refatoração, reunião de planejamento, reunião de retrospectiva, reunião diária e equipes auto-organizáveis são comuns às quatro equipes, algumas equipes adotam tais práticas de forma adaptada.

Este trabalho buscou responder a questão de pesquisa: "As práticas ágeis utilizadas pelas equipes de desenvolvimento estão associadas com a taxa de defeitos encontrados durante as *sprints*?"a questão foi dividida em subquestões para abranger melhor o tema.

Para responder a questão e suas subquestões foram coletadas evidências, tais evidências foram organizadas e analisadas. O resultado final não é conclusivo, são apenas indícios que podem levar a uma teoria.

6.3.1 Hipóteses

Diversas hipóteses foram levantadas ao longo da pesquisa e outras ainda podem surgir.

- Determinadas práticas ágeis aumentam ou diminuem a taxa de defeitos?
- Se a equipe possui experiência em métodos ágeis a taxa de defeitos irá diminuir?
- O uso de práticas ágeis de forma adaptada reduz a efetividade da prática?
- O uso de práticas relacionadas ao código reduz a taxa de defeitos?
- Reduzir a taxa de defeitos aumenta a qualidade do produto?
- As cerimônias do método *Scrum* tem algum impacto sobre a taxa de defeitos?

Para validar tais hipóteses devem ser realizados trabalhos futuros, como sugerido na seção correspondente.

6.3.2 Associação de práticas ágeis com a Taxa de defeitos

As duas técnicas de análise empregadas neste trabalho indicam que existe essa associação entre o uso de práticas ágeis com a taxa de defeitos por *Sprint*, os indícios são baseados nas percepções de desenvolvedores ágeis de software.

Os resultados estão de acordo com os trabalhos relacionados citados anteriormente no capítulo 3, [Ilieva, Ivanov e Stefanova 2004], [Layman, Williams e Cunningham 2004], [Lindvall et al. 2004], o que reforça a validade deste estudo.

6.3.2.1 As práticas ágeis utilizadas são adaptadas

Segundo observações, questionários e entrevistas com membros das equipes, constatou-se que as equipes A e C procuram empregar as práticas ágeis da forma mais fiel possível ao método, apesar de que por vezes alguma prática precise de adaptação.

Nas equipes B e D as práticas ágeis são adaptadas de forma mais evidente e coincidentemente possuem as maiores taxas de defeitos por *Sprints*. Outros fatores tem influência para que tais situações ocorram. Como comentado anteriormente, a equipe B presta serviços de manutenção de software, logo, práticas como o planejamento são difíceis de se manter. No caso da equipe D pode ser percebido que a equipe ainda busca um ritmo de trabalho, é possível que ao passar do tempo o resultado seja diferente deste apresentado.

6.3.2.2 Conhecimento em práticas ágeis por parte das equipes

Dentre as evidências coletadas, a observação em campo foi a que mais contribuiu para responder essa subquestão.

De um modo geral, as equipes conhecem as práticas ágeis que utilizam, porém em certos casos o uso da prática fica implícito, logo, as equipes empregam a prática na rotina de trabalho sem perceber que está empregando.

6.3.2.3 Maturidade das equipes quanto ao uso de práticas ágeis

A maturidade da equipe em desenvolvimento ágil passa por etapas de aprendizado, nas primeiras etapas a equipe utiliza as práticas ágeis sob supervisão e orientação de alguém com mais experiência, ao longo do tempo o membro da equipe começa a captar o processo da prática e passa a compreender a necessidade daquela prática estar sendo utilizada.

Por vezes, alguns membros das equipes não souberam responder se utilizavam determinadas práticas apenas pelo nome da mesma, porém quando a prática foi descrita de forma clara e objetiva, os membros recordavam de atividades desenvolvidas.

6.3.2.4 Preocupações das equipes em reduzir o número de defeitos do software

A empresa ABC tem como meta estratégica melhorar a qualidade dos seus produtos, essa meta foi uma das razões pela qual este estudo de caso pôde ser realizado, logo, a meta se estende aos níveis mais baixos da empresa, como a fábrica por exemplo.

Nas equipes de desenvolvimento existe o objetivo de melhorar o produto. Foi comentado na seção de Observação em Campo do Capítulo 5, que a empresa requisitou que fosse feita a refatoração de todos os módulos dos produtos da fábrica, como defeitos são indícios de má-qualidade do produto, pode-se considerar que existe a preocupação em reduzir o número de defeitos do software.

6.3.2.5 Associação mais destacada entre determinadas práticas e a taxa de defeitos

Para responder sobre quais práticas tem associação mais destacada são utilizadas as evidências dos questionários aplicados, segundo a maioria dos membros das equipes, as práticas relacionadas à desenvolvimento, como Código Limpo e Programação em Par, e da categoria Testes, como Testes automatizados e Testes de Unidade, são as práticas que tem mais associação com a taxa de defeitos por *sprint*.

Em contrapartida, práticas da categoria organização, como estimativas em pontos, são práticas consideradas por desenvolvedores como práticas menos evidentemente associadas.

Capítulo 7

Conclusão

Neste capítulo é apresentado um aparato geral sobre o trabalho e como foi conduzido o estudo. São apresentadas conclusões sobre as questões de pesquisa e ao final são apresentados trabalho futuros.

Este trabalho investigou a associação entre o desenvolvimento ágil de software e a taxa de defeitos.

Para realizar tal objetivo, este trabalho consiste em uma pesquisa qualitativa com a metodologia estudo de caso embutido [Yin 2008], pois contém quatro unidades de análise.

As quatro unidades de análise mostram uma perspectiva mais abrangente, pois abordam equipes diferentes, porém, por se tratar de quatro unidades de análise, o esforço para realizar o estudo de caso se torna maior.

A análise, como mencionado anteriormente, mostrou por meio das evidências coletadas e referências da literatura, que existe associação entre o desenvolvimento ágil de software e a taxa de defeitos. O fato de que existe tal associação não necessariamente garante que o número de defeitos vai aumentar ou diminuir, pois existem outros fatores que interferem nesse número. Para constatar, com mais precisão, se o uso de determinadas práticas aumenta ou diminui a taxa de defeitos é necessário um estudo que avalie o impacto das práticas ágeis na taxa de defeitos, o que pode ser um trabalho futuro.

Ao realizar a pesquisa, constatou-se que as equipes de desenvolvimento da empresa ABC realizam a maior parte das práticas ágeis da forma o mais fiel possível, porém existem algumas adaptações.

De acordo com a ideia de desenvolvimento ágil, o sistema em funcionamento é a melhor maneira de julgar o quanto a equipe tem feito [Singh, Singh e Sharma 2014]. Por isso a impor-

tância dos conhecimento. Constatou-se que as equipes conhecem o procedimento das práticas ágeis, porém, por vezes não conhecem o termo ou o conceito aplicado. Juntamente com o conhecimento sobre desenvolvimento ágil, existe o conceito da maturidade, que faz com que membros com mais experiência em *agile* tenham mais conhecimento.

Na empresa ABC, o método de implantação de *agile* consiste em fazer com o que as equipes utilizem as práticas de forma repetitiva até adquirir certa maturidade e com ela assimilar conhecimento acerca do tema.

A empresa ABC tem uma meta para o futuro melhorar a qualidade do seu produto, conforme foi mencionado anteriormente, o que faz com que exista a preocupação em diminuir os defeitos do software. Este trabalho é uma etapa para o cumprimento da meta, servindo como base para se conhecer melhor os procedimentos da empresa.

As práticas relacionadas diretamente ao software, como testes e código limpo, por exemplo, foram sempre citadas como fundamentais para se obter uma menor quantidade de defeitos e por consequência, melhoria na qualidade sob alguns aspectos, como a confiabilidade. Um fato que ocorre é que a empresa ABC vem adotando uma ideologia onde é antiético entregar software não testado para o cliente.

7.1 Trabalhos Futuros

Para trabalhos futuros a sugestão é a aplicação de um estudo de caso explanatório [Yin 2008, Wohlin e Aurum 2014] que busca validar alguma das hipóteses levantadas por este trabalho, como comentado na seção 6.3.1.

Uma possibilidade de trabalho futuro é a avaliação do impacto de se usar uma determinada prática ágil na taxa de defeitos do software.

Glossário

Cerimônia	Reunião referente ao framework Scrum.
Stakeholder	Parte interessada (em um projeto por exemplo).
Defeito	Falha ou deformidade do sistema.

Referências Bibliográficas

[Ahmed et al. 2010]AHMED, A. et al. Agile software development: Impact on productivity and quality. In: IEEE. *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference on*. [S.l.], 2010. p. 287–291.

[Auvinen et al. 2005]AUVINEN, J. et al. *Improving the Engineering Process Area at Ericsson with Agile Practices: A Case Study*. [S.l.]: Turku Centre for Computer Science, 2005.

[Basili 1993]BASILI, V. R. The experimental paradigm in software engineering. In: *Experimental Software Engineering Issues: Critical Assessment and Future Directions*. [S.l.]: Springer, 1993. p. 1–12.

[Beck 2000]BECK, K. *Extreme programming explained: embrace change*. [S.l.]: Addison-Wesley Professional, 2000.

[Beck et al. 2001]BECK, K. et al. Manifesto for agile software development. 2001.

[Benbasat, Goldstein e Mead 1987]BENBASAT, I.; GOLDSTEIN, D. K.; MEAD, M. The case research strategy in studies of information systems. *MIS quarterly*, JSTOR, p. 369–386, 1987.

[Campanelli e Parreiras 2015]CAMPANELLI, A. S.; PARREIRAS, F. S. Agile methods tailoring–a systematic literature review. *Journal of Systems and Software*, Elsevier, v. 110, p. 85–100, 2015.

[Chan 2013]CHAN, K. *Agile Adoption Statistics 2012*. 2013. Disponível em: <http://www.onedesk.com/2013/05/agile-adoption-statistics-2012/>.

[Conte, Cabral e Travassos 2009]CONTE, T.; CABRAL, R.; TRAVASSOS, G. H. Aplicando grounded theory na análise qualitativa de um estudo de observação em engenharia de

software–um relato de experiência. In: *V Workshop"Um Olhar Sociotécnico sobre a Engenharia de Software"(WOSES 2009)*. [S.l.: s.n.], 2009. p. 26–37.

[Conte e Travassos 2009]CONTE, T.; TRAVASSOS, G. H. Técnica de inspeção de usabilidade baseada em perspectivas de projeto web. *Rio de Janeiro: UFRJ/COPPE*, 2009.

[Denzin e Lincoln 2005]DENZIN, N. K.; LINCOLN, Y. S. Qualitative research. *Denzin, NK y Lincoln YS*, 2005.

[Gilb, Graham e Finzi 1993]GILB, T.; GRAHAM, D.; FINZI, S. *Software inspection*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1993.

[Gomes 1990]GOMES, F. P. *Curso de estatística experimental*. [S.l.]: ESALQ/USP São Paulo, 1990.

[Highsmith 2001]HIGHSMITH, J. History: The agile manifesto. *Sitio: http://www. agilemanifesto. org/history. html*, 2001.

[Highsmith 2002]HIGHSMITH, J. A. *Agile software development ecosystems*. [S.l.]: Addison-Wesley, 2002.

[Humphrey 1995]HUMPHREY, W. S. *A discipline for software engineering*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1995.

[Ikonen et al. 2010]IKONEN, M. et al. Exploring the sources of waste in kanban software development projects. In: IEEE. *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*. [S.l.], 2010. p. 376–381.

[Ilieva, Ivanov e Stefanova 2004]ILIEVA, S.; IVANOV, P.; STEFANOVA, E. Analyses of an agile methodology implementation. In: IEEE. *Euromicro Conference, 2004. Proceedings. 30th*. [S.l.], 2004. p. 326–333.

[ISO/IEC 2003]ISO/IEC. *Engenharia de Software - Qualidade do Produto*. Rio de Janeiro, 2003.

[Jalali e Wohlin 2010]JALALI, S.; WOHLIN, C. Agile practices in global software engineering-a systematic map. In: IEEE. *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*. [S.l.], 2010. p. 45–54.

[Kayes, Sarker e Chakareski 2013]KAYES, I.; SARKER, M.; CHAKARESKI, J. On measuring test quality in scrum: An empirical study. *arXiv preprint arXiv:1310.2545*, 2013.

[Klein e Myers 1999]KLEIN, H. K.; MYERS, M. D. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, JSTOR, p. 67–93, 1999.

[Korhonen 2010]KORHONEN, K. Evaluating the effect of agile methods on software defect data and defect reporting practices-a case study. In: IEEE. *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*. [S.l.], 2010. p. 35–43.

[Lagerberg et al. 2013]LAGERBERG, L. et al. The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at ericsson. In: IEEE. *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. [S.l.], 2013. p. 348–356.

[Laitenberger 1998]LAITENBERGER, O. Studying the effects of code inspection and structural testing on software quality. In: IEEE. *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*. [S.l.], 1998. p. 237–246.

[Laitenberger e DeBaud 2000]LAITENBERGER, O.; DEBAUD, J.-M. An encompassing life cycle centric survey of software inspection. *Journal of Systems and Software*, Elsevier, v. 50, n. 1, p. 5–31, 2000.

[Layman, Williams e Cunningham 2004]LAYMAN, L.; WILLIAMS, L.; CUNNINGHAM, L. Exploring extreme programming in context: an industrial case study. In: IEEE. *Agile Development Conference, 2004*. [S.l.], 2004. p. 32–41.

[Lee e Yong 2013]LEE, S.; YONG, H.-S. Agile software development framework in a small project environment. *Journal of Information Processing Systems*, Korea Information Processing Society, v. 9, n. 1, p. 69–88, 2013.

[Lindvall et al. 2004]LINDVALL, M. et al. Agile software development in large organizations. *Computer*, IEEE, v. 37, n. 12, p. 26–34, 2004.

[Mann e Maurer 2005]MANN, C.; MAURER, F. A case study on the impact of scrum on overtime and customer satisfaction. In: IEEE. *null*. [S.l.], 2005. p. 70–79.

[Martin 2009]MARTIN, R. C. *Clean code: a handbook of agile software craftsmanship*. [S.l.]: Pearson Education, 2009.

[Myers et al. 1997]MYERS, M. D. et al. Qualitative research in information systems. *Management Information Systems Quarterly*, MIS RESEARCH CENTER-SCHOOL OF MANAGEMENT, v. 21, p. 241–242, 1997.

[Nagappan et al. 2008]NAGAPPAN, N. et al. Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Software Engineering*, Springer, v. 13, n. 3, p. 289–302, 2008.

[Petersen 2010]PETERSEN, K. Is lean agile and agile lean? *Modern Software Engineering Concepts and Practices: Advanced Approaches*, p. 19, 2010.

[Poppendieck e Poppendieck 2003]POPPENDIECK, M.; POPPENDIECK, T. *Lean software development: an agile toolkit*. [S.l.]: Addison-Wesley Professional, 2003.

[Pressman 2010]PRESSMAN, R. *Software engineering: a practitioner's approach.* [S.l.]: McGraw-Hill Higher Education, 2010.

[Radatz, Geraci e Katki 1990]RADATZ, J.; GERACI, A.; KATKI, F. Ieee standard glossary of software engineering terminology. *IEEE Std*, v. 610121990, n. 121990, p. 3, 1990.

[Reel 1999]REEL, J. S. Critical success factors in software projects. *Software, IEEE*, IEEE, v. 16, n. 3, p. 18–23, 1999.

[Runeson e Höst 2009]RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, Springer US, v. 14, n. 2, p. 131–164, 2009.

[Runeson e Wohlin 1998]RUNESON, P.; WOHLIN, C. An experimental evaluation of an experience-based capture-recapture method in software code inspections. *Empirical Software Engineering*, Springer, v. 3, n. 4, p. 381–406, 1998.

[Sandusky e Gasser 2005]SANDUSKY, R. J.; GASSER, L. Negotiation and the coordination of information and activity in distributed software problem management. In: ACM. *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*. [S.l.], 2005. p. 187–196.

[Schatz e Abdelshafi 2005]SCHATZ, B.; ABDELSHAFI, I. Primavera gets agile: a successful transition to agile development. *IEEE software*, IEEE, n. 3, p. 36–42, 2005.

[Schwaber e Sutherland 2011]SCHWABER, K.; SUTHERLAND, J. The scrum guide. *Scrum Alliance*, 2011.

[Seaman 2008]SEAMAN, C. B. Qualitative methods. In: *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008. p. 35–62.

[Shull, Singer e Sjøberg 2008]SHULL, F.; SINGER, J.; SJØBERG, D. I. *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008.

[Singh, Singh e Sharma 2014]SINGH, A.; SINGH, K.; SHARMA, N. Agile knowledge management: a survey of indian perceptions. *Innovations in Systems and Software Engineering*, Springer, v. 10, n. 4, p. 297–315, 2014.

[Sommerville 2011]SOMMERVILLE, I. *Software Engineering*. 9. ed. Boston: Addison Wesley, 2011.

[Strauss e Corbin 2008]STRAUSS, A. L.; CORBIN, J. *Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada*. [S.l.]: Artmed, 2008.

[Sugimori et al. 1977]SUGIMORI, Y. et al. Toyota production system and kanban system materialization of just-in-time and respect-for-human system. *The International Journal of Production Research*, Taylor & Francis, v. 15, n. 6, p. 553–564, 1977.

[Tarhan e Yilmaz 2014]TARHAN, A.; YILMAZ, S. G. Systematic analyses and comparison of development performance and product quality of incremental process and agile process. *Information and Software Technology*, Elsevier, v. 56, n. 5, p. 477–494, 2014.

[Wailgum 2007]WAILGUM, T. *From Here to Agility*. 2007.

[Wohlin e Aurum 2014]WOHLIN, C.; AURUM, A. Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, Springer, p. 1–29, 2014.

[Yin 2003]YIN, R. K. Case study research design and methods third edition. *Applied social research methods series*, Sage Publications Inc, v. 5, 2003.

[Yin 2008]YIN, R. K. *Case Study Research: Design and Methods: Design and Methods*. [S.l.]: Sage Publications, 2008.

[Zeiss et al. 2007]ZEISS, B. et al. Applying the iso 9126 quality model to test specifications. *Software Engineering*, v. 15, n. 6, p. 231–242, 2007.