UNIVERSIDADE FEDERAL DO PAMPA

Peterson Luiz da Rosa Rodrigues

Estudos Empíricos sobre a Aplicabilidade de Especificação Formal de Requisitos em Projetos Ágeis

Alegrete
2017

Peterson Luiz da Rosa Rodrigues

Estudos Empíricos sobre a Aplicabilidade de Especificação Formal de Requisitos em Projetos Ágeis

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Mestre João Pablo Silva da Silva

Coorientador: Prof. Doutor Elder de Macedo Rodrigues

Alegrete
2017

Peterson Luiz da Rosa Rodrigues

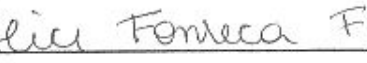
Estudos Empíricos sobre a Aplicabilidade de Especificação Formal de Requisitos em Projetos Ágeis

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 30 de Novembro de 2017
Banca examinadora:

Prof. Mestre João Pablo Silva da Silva
Orientador
Unipampa

Prof. Doutor Elder de Macedo Rodrigues
Coorientador
Unipampa



Prof. Mestre Alice Fonseca Finger
Unipampa

Bel. Alex Malmann Becker
UFSCar

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me amparado nos momentos mais difíceis que enfrentei ao longo deste trabalho, tanto pessoalmente quanto profissionalmente.

Agradeço a minha mãe Neiva que me apoiou em todos momentos, em que mesmo convivendo comigo todo meu período acadêmico, eu estava ausente mentalmente resolvendo estresses do dia a dia. Minha maior motivadora, que sempre sonhou com um filho graduado.

Também estendo meus agradecimentos à minha namorada Stefane, que me apoiou em diversos momentos durante a graduação e foi muito importante para concluir com êxito minhas atividades.

Agradeço meu Orientador João e meu Coorientador Elder que contribuíram muito para meu aperfeiçoamento acadêmico e também pessoal. Além de todos os professores que tive o prazer de trabalhar e que, também de alguma forma, auxiliam na execução deste trabalho.

Por fim, agradeço a todos meus amigos e colegas que tive o prazer de conhecer e estabelecer laços fortes. Em especial, agradeço a Analice Trevisan que me mostrou o significado da persistência, luta e força de vontade. Hoje, não está aqui conosco para defender seu próprio TCC, mas sei que onde quer que esteja, estará contente por ter continuado e não desistido daquilo que mais quero, mesmo com a perda abrupta de sua presença:

"Nunca saberemos o quão forte somos até que ser forte seja a única escolha."

RESUMO

Uma das características das abordagens ágeis é utilizar técnicas informais para a especificação de requisitos. O uso exclusivo dessas técnicas pode causar alguns problemas de especificação, como ambiguidade, omissão de informações, além de erros de interpretação. Esses problemas são enfrentados por uma determinada empresa de desenvolvimento de software, situada no Rio Grande do Sul, ao qual está impactando na produtividade do time ágil. Uma possível alternativa para mitigar esses problemas é a adoção de métodos de especificação formal que utilizam a matemática discreta para escrita de requisitos. É possível encontrar alguns trabalhos na literatura que propõem a utilização de métodos formais para especificações de requisitos, como a notação Z e *Vienna Development Method* (VDM), conjuntamente com abordagens ágeis, como *Extreme Programming* (XP) e Scrum. Mesmo identificando tais iniciativas, percebemos que há uma carência de trabalhos que avaliem empiricamente a utilização de métodos formais em projetos ágeis. Dessa forma, neste trabalho, buscamos investigar os métodos de especificação formal como alternativa para especificar requisitos em projetos ágeis, proporcionando evidências dessa aplicação para a empresa parceira deste estudo. Para tanto, foram aplicados diferentes estudos empíricos. Inicialmente, realizamos um mapeamento sistemático da literatura para identificar os métodos de especificação formal que comumente são utilizados em ambientes ágeis para desenvolvimento de sistemas. Posteriormente, realizamos diferentes estratégias para avaliar empiricamente a aplicabilidade de métodos formais como técnica de especificação. A primeira estratégia empírica consistiu na condução de uma pesquisa de opinião com profissionais da indústria de desenvolvimento ágil de software. Na segunda estratégia empírica realizamos um estudo de caso incorporado exploratório, em duas empresas de desenvolvimento de software, para avaliar a opinião de desenvolvedores ágeis após utilizar a notação Z para especificação de requisitos. Por fim, na terceira estratégia foi conduzido um experimento controlado no ambiente da empresa parceira neste estudo onde foi observado a produtividade e qualidade das especificações quando utilizada a notação Z e Caso de Uso descritivo pelo time de desenvolvimento ágil da empresa. Os resultados das duas primeiras estratégias mostraram que profissionais indicaram alguns benefícios na utilização de especificação formal em projetos ágeis, como por exemplo: especificação formal ajuda a tornar os requisitos mais claros e mais completos. Contudo, o principal fator limitador da utilização dessas técnicas é relacionado ao pré-conceito dos desenvolvedores. Os resultados do experimento indicaram que o uso da notação Z melhorou o processo de elicitação de requisito, impactando na produtividade do time, visto que o uso da notação Z resultou em uma maior quantidade de software entregue pelo time ágil, além de proporcionar uma maior cobertura das especificações.

Palavras-chave: Estudos Empíricos. Métodos Formais. Especificação Formal. Abordagem Ágil. Scrum. Notação Z. Caso de Uso.

ABSTRACT

One of the characteristics of agile approaches is to use informal techniques for specifying requirements. The exclusive use of these techniques can cause some problems, such as ambiguous specifications, omission of information and interpretation mistakes. These problems are faced by a particular software development company, in Rio Grande do Sul, resulting in the unproductiveness of an agile team. A possible alternative is to adopt Formal Methods, which are, in resume, methods that use mathematical logic to formally specify requirements. It is possible find some papers in the literature that proposes the use of formal methods for requirements specification, as the Z notation and *Vienna Development Method* (VDM) with agile approach, as *Extreme Programming* (XP) and Scrum. Even identifying such initiatives, we realize the lack of papers that empirically evaluate the use of formal methods in agile projects. In this way, in this paper, we seek to investigate formal specification methods as an alternative to specify requirements in agile projects, providing evidence of this application to the partner company of this study. For that, different empirical studies were applied. Initially, we performed a systematic mapping to identify formal specification methods that are commonly used for developing systems in agile environments. Subsequently, we performed different empirical strategies to empirically evaluate the applicability of Formal Methods as a specification technique. The first empirical strategy consisted in conducting an opinion survey with acting agile software developers. In the second empirical strategy, we carried out an exploratory embedded case study in two software development companies to evaluate the applicability of Z notation for requirements specification. Finally, in the third strategy, we carried a controlled experiment in partner company of this study, where the productivity and quality impacts of the specifications were analyzed using the formal notation Z and Descriptive Use Case. The results of the first two strategies pointed out that the professionals' perceptions tended to agree on some benefits in the use of formal specification in agile projects, for example: formal specification helps to make the requirements clearer and more complete. However, the main limiting factor in using these techniques is related to the developers' preconceptions. The experiment results indicated that the use of Z notation improved the elicitation process, impacting on team productivity, as the use of Z notation resulted in the delivery of more software functionalities and software with better specification coverage by the agile team.

Key-words: Empirical Studies. Formal Methods. Formal Specification. Agile Approach. Scrum. Z Notation. Use Caso.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

EF Especificação Formal

EI Especificação Informal

MF Método Formal

MI Método Informal

SM *Scrum Master*

TD Time de Desenvolvimento

TS Time Scrum

LISTA DE SIGLAS

ACM *Association Computing Machinery*

APF Análise de Ponto de Função

DBC *Design by Contract*

IEEE *Institute of Electrical and Electronics Engineers*

LOTOS *Language Of Temporal Ordering Specification*

PF Pontos de Função

PO *Product Owner*

SAF *Scalable Agile Formal*

SCR *Software Cost Reduction*

SMS *Systematic Mapping Study*

SOLF *Structured Object-Oriented Formal Language*

TDD *Test Driven Development*

TFD *Test First Development*

TI Tecnologia da Informação

UML *Unified Modeling Language*

VDM *Vienna Development Method*

VDM-RT *Vienna Development Method Real-Time*

VDM-SL *Vienna Development Method Specification Language*

XP *Extreme Programming*

XPF *Extreme Programming Formal*

XXM *Extreme X-Machines*

SUMÁRIO

1 INTRODUÇÃO

Na última década a forma como software é entregue evoluiu drasticamente. Grandes empresas como Amazon, Google, Nokia e Facebook buscam as melhores práticas para desenvolver sistemas de software dentro do prazo e orçamento previstos, com qualidade e que satisfaça as necessidades do cliente (WOLFF, 2012; DUVALL, 2012).

As abordagens ágeis surgiram como uma alternativa para maximizar a taxa de sucesso em projetos de software. Essas abordagens emergiram a partir do Manifesto Ágil (BECK et al., 2001) que propõe a valorização das pessoas e das interações entre elas, do software em funcionamento, da colaboração com cliente e da resposta regular às mudanças.

Uma das principais preocupações no desenvolvimento de sistemas é na etapa de engenharia de requisitos. Essa etapa fornece o mecanismo para entender o que o cliente deseja, realizando a elicitação, análise e gerência de requisitos, porém ainda podem surgir problemas nessa fase (SOMMERVILLE, 2011; KASSAB; NEILL; LAPLANTE, 2014). Nesse processo são centralizados os requisitos do usuário e do sistema em um documento de requisitos. Essa especificação deve ser clara, inequívoca, fácil de entender, completa e consistente (UNTERKALMSTEINER; GORSCHEK, 2017).

Em abordagens ágeis, comumente se utiliza Método Informal (MI) para especificação de requisitos onde é utilizada a linguagem natural para escrever as necessidades do usuário final (SOMMERVILLE, 2011). Utilizando linguagem natural a especificação é intuitiva e universal. Também é potencialmente vago, ambíguo, e seu significado depende do entendimento do leitor. Como resultado, houve muitas propostas de formas alternativas de escrever requisitos (SOMMERVILLE, 2011; KASSAB; NEILL; LAPLANTE, 2014).

A utilização exclusiva dessas técnicas pode acarretar em problemas de especificação no desenvolvimento de software, como ambiguidades nos requisitos e omissão de informação. Esses problemas causam retrabalho em requisitos durante o desenvolvimento do requisito. Para reduzir esses tipos de problemas propõe-se o uso de diferentes métodos para especificação de requisitos, como os métodos baseados em lógica matemática (KASSAB; NEILL; LAPLANTE, 2014; WOLFF, 2012).

Esse tipo de método é chamado de Método Formal (MF) que são conjuntos de práticas baseadas em técnicas matemáticas utilizadas em desenvolvimento de sistemas de computador (WOLFF, 2012). Eles servem para descrever, verificar e analisar propriedades das especificações ou do sistema e possibilitam geração de bons casos de teste (DEHARBE et al., 2000). Neste trabalho, estamos interessados nas técnicas formais de especificação de requisitos.

Os métodos de Especificação Formal (EF) utilizam a matemática discreta para especificar os requisitos que o sistema deve atingir (WOLFF, 2012). O uso de métodos de EF permitem diminuir a ambiguidade durante a análise do projeto, pois a semântica

do formalismo fornece regras precisas de interpretação. Lamsweerde (2000), apresenta outras vantagens características do formalismo, semelhante as que Larsen, Fitzgerald e Wolff (2010) descrevem. Esses autores afirmam que a inclusão do formalismo no desenvolvimento de software é motivada por possibilitar a minimização dos defeitos no sistema entregue, pois é possível identificá-los assim que eles surgem. Entretanto, na indústria ainda há receio na sua utilização, pois o emprego do formalismo gera uma curva íngreme de aprendizagem (WOLFF, 2012).

Ao utilizar métodos de EF em projetos ágeis pode-se potencialmente, trazer o melhor dos dois mundos, como potencializar a qualidade dos produtos de software através do formalismo, bem como aumentar o uso das abordagens ágeis no desenvolvimento de projetos de grande escala, como sistemas críticos de segurança (MOCHIO; ARAKI, 2012).

1.1 Motivação

A Empresa A[1], localizada no Rio Grande do Sul – Brasil, enfrenta problemas no seu processo de engenharia de requisitos. Ela é uma fábrica de software que atua no setor há mais de sete anos e tem processos de desenvolvimento baseados em princípios ágeis. A engenharia de requisitos é baseada em especificações ágeis, especificamente histórias de usuários (LUCASSEN; AL., 2016).

A utilização exclusiva de técnicas informais, como histórias de usuários, para descrição de requisitos em alto nível está levando problemas de especificação dentro da equipe de desenvolvimento dessa empresa, como omissão de informação, má interpretação, entre outros. Esses problemas afetam a implementação e causam retrabalho em requisitos incorretos, tornando o time de desenvolvimento improdutivo. Para reduzir esses tipos de problemas se propõe o uso de diferentes métodos para especificação de requisitos, como os métodos baseados em lógica matemática.

Considerando que há alguma iniciativa de EF em ambientes ágeis, como Olszewska e Waldén (2015), Wolff (2012), Shafiq e Minhas (2014), acreditamos que o uso de EF pode ajudar a mitigar os problemas enfrentados pela Empresa A. Mesmo assim, incluir ou alterar as práticas no desenvolvimento de software dessa Empresa A requer certo cuidado, pois deve trazer benefícios consideráveis para o desenvolvimento, para assim, resultar em produtos de qualidade, aumentando seu valor agregado.

Por fim, notamos que na literatura há pouca avaliação empírica dessa utilização, limitando a confiança da Empresa A em adotar essa prática. Assim, visto que existe uma demanda de interesse nesse estudo, fica evidenciado a necessidade de estudos que tragam evidências dos impactos da utilização de EF em projetos ágeis.

[1] Empresa motivadora deste trabalho, interessada diretamente nos nossos resultados. Por fins de confidencialidade omitidos seu nome.

1.2 Objetivo

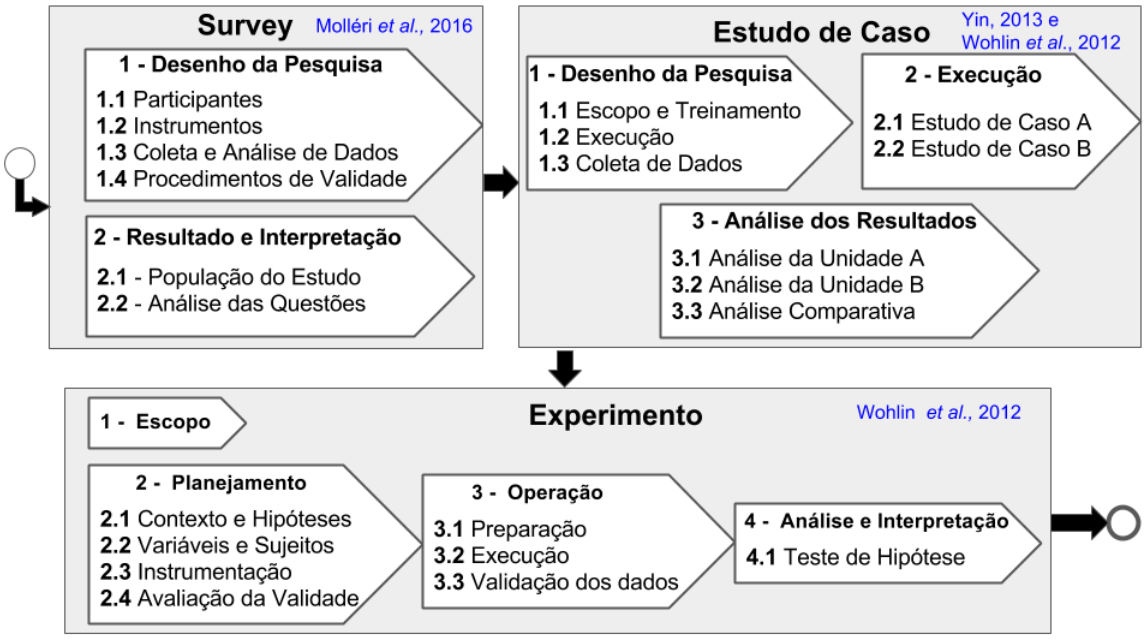
Observando o contexto e a motivação, este trabalho teve por objetivo gerar evidências através de estudos empíricos da aplicabilidade de EF em ambientes ágeis, como uma alternativa possível de resolver os problemas enfrentados pela Empresa A.

Para alcançar esse objetivo, investigamos o estado da arte de EF aplicada em ambientes ágeis, através de um Mapeamento Sistemático da Literatura (SMS), conduzimos uma pesquisa exploratória com profissionais de desenvolvimento ágil de software, executamos um estudo de caso incorporado e exploratório em duas empresas de software e, por fim, apresentamos um experimento controlado em Empresa A, principal interessada neste trabalho.

1.3 Metodologia

Nas primeiras etapas do desenvolvimento deste trabalho foi realizado um SMS buscando explorar trabalhos relacionados acerca da utilização dos MF para especificação de requisitos em projetos ágeis. Após esse estudo preliminar, dividimos nosso trabalho em três pacotes de atividades, envolvendo três estudos de avaliação empírica. A Figura 1 apresenta uma síntese das atividades envolvendo cada estratégia empírica.

Figura 1 – Síntese da Metodologia do Trabalho



Fonte: Adaptado de (MOLLÉRI; PETERSEN; MENDES, 2016), (YIN, 2013) e (WOHLIN et al., 2012).

Como ilustra a Figura 1, a primeira estratégia de avaliação empírica é a realização

de uma pesquisa exploratória, também chamada de *Survey*, utilizando o trabalho de Molléri, Petersen e Mendes (2016) como guia. Nesse *Survey*, o objetivo é explorar a opinião dos profissionais desenvolvedores ágeis de software sobre a inclusão do formalismo como prática para especificação dos requisitos.

Na sequência, planejamos e executamos um estudo de caso incorporado e exploratório, guiado pelo trabalho de Yin (2013) e Wohlin et al. (2012). Nesse estudo, analisamos as percepções de profissionais de duas empresas de desenvolvimento de software, após especificarem formalmente requisitos de projetos de cada empresa.

Por fim, após coletarmos as opiniões de profissionais conduzimos um experimento controlado, baseado no trabalho de Wohlin et al. (2012). Nesse experimento, buscamos analisar um determinado método para EF com intuito de avaliar com respeito a qualidade das especificações e a produtividade na perspectiva do pesquisador no contexto de uma empresa de desenvolvimento de software que utiliza abordagens ágeis como prática condutora de engenharia.

1.4 Principais Contribuições

Este trabalho se difere dos demais por trazer evidências empíricas através de estudos realizados sistematicamente. Nesses estudos são utilizados guias conceitualizados da literatura, que auxiliaram no planejamento, execução e análise dos resultados gerados. Por exemplo, na última estratégia de avaliação empírica (experimento controlado) buscamos analisar os impactos gerados do uso de EF e Especificação Informal (EI) na perspectiva da produtividade e qualidade das especificações em um contexto ágil de desenvolvimento de software.

Os nossos resultados gerados nos nossos estudos expressam indícios de interesse por parte dos desenvolvedores ágeis de software em utilizar métodos de EF. Os participantes do *survey* sugeriram a possibilidade em utilizar EF, acreditando que esses métodos podem contribuir para mitigar problemas causados pela EI de requisitos. Mesmo assim, algumas barreiras foram relatados na pesquisa, como o tempo necessário para aprendizado de uma notação formal.

Na segunda estratégia empírica nós observamos dois sujeitos de duas empresas de software. Percebemos nesse estudo que os sujeitos evoluíram suas percepções após utilizar um método de EF na prática. No final, eles relataram que EF colabora pra eliminar alguns problemas de especificação, como: dificuldade na compreensão de requisitos; erros de interpretação e ambiguidade nas especificações. Além disso, eles também afirmaram que com o tempo utilizar EF vai se tornando menos demorado e menos complexo de entender.

Os nossos resultados no estudo experimental contribuíram para a Empresa A entender alguns aspectos da utilização de EF. Identificamos que os sujeitos insistiram a não acreditar nas vantagens em utilizar a EF no seu ambiente. Mas ao final do estudo,

os resultados mostraram indícios da sua aplicabilidade, ao modo que o time de desenvolvimento obteve uma maior produtividade e garantiu maior cobertura nas especificações quando utilizado MFs para especificação requisitos do que uma notação informal.

A principal contribuição deste estudo esta em apresentar que exitem limitadores para propagação da prática de EF em ambiente ágil, como: complexidade em aprender a utilizar técnicas formais que utilizam lógica matemática. Mas o principal fator, presentes em todos estudos realizados, é o pré-conceito dos desenvolvedores em admitir que os métodos de EF podem ser uma alternativa de ferramenta para especificação de requisitos em projetos ágeis.

Por fim, nossas contribuições inciais já estão disponíveis na literatura. Por exemplo, no trabalho intitulado "Métodos formais como condutores para especificação de requisitos aplicados em metodologias ágeis de desenvolvimento" (RODRIGUES et al., 2016) apresentamos o estado da arte a cerca dos método de EF aplicados em abordagens ágeis através da condução de um SMS. Além disso, apresentamos em um resumo expandido (RODRIGUES et al., 2017) uma síntese dos resultados desse SMS.

1.5 Organização do trabalho

Neste capítulo foram apresentadas a contextualização no qual este trabalho está inserido, bem como sua motivação e objetivos. Na sequência, o Capítulo 2 apresenta os fundamentos necessários para entendimento deste trabalho, como as práticas ágeis e métodos para EF. Destacamos o Capítulo 3, que descreve o planejamento, execução e análise dos resultados de um SMS realizado para nos dar embasamento sobre o estado da arte do contexto do trabalho. O Capítulo 4 apresenta a elaboração e condução da pesquisa exploratória, ou *Survey*, contendo seu desenho da pesquisa, execução e análise dos resultados. Seguindo, no Capítulo 5 é descrita a estruturação, execução e interpretação dos resultados gerados através de um estudo de caso incorporado e exploratório. O Capítulo 6 apresenta a definição do experimento controlado que realizamos para evidenciar a aplicabilidade dos métodos de EF em abordagem ágil. Por fim, apresentamos no Capítulo 7 as principais considerações sobre o trabalho, bem como os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os fundamentos teóricos necessários para a compreensão do nosso trabalho. O capítulo está organizado da seguinte forma: na seção 2.1 são discutidas as definições para abordagens ágeis, suas características e princípios, bem como um exemplo de uma abordagem comumente utilizada; na seção 2.2 e seção 2.3 apresentamos, respectivamente, os conceitos sobre EF e EI de requisitos, além de exemplos de uso; na seção 2.4 apresentamos os conceitos sobre estudos empíricos; por fim, na seção 2.5 descrevemos as considerações e lições aprendidas do capítulo.

2.1 Abordagens Ágeis

As definições de desenvolvimento de software ágil evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados". Contudo, a definição oficial de Desenvolvimento Ágil de Software foi constituída através do Manifesto Ágil (BECK et al., 2001) publicado em fevereiro de 2001 por um grupo de 17 profissionais da área de Tecnologia da Informação (TI).

Essas abordagens surgiram em virtudes das falhas em metodologias convencionais, que enfrentaram dificuldades em lidar com a realidade do desenvolvimento de software em um mercado competitivo e volátil.

Em virtude disso, as práticas ágeis têm sido adotadas com o intuito de estabilizar os casos de sucesso em projetos de software. As adesões dessas abordagens se dão, principalmente, pelas suas características em descobrir maneiras melhores de desenvolver software. Dessa forma, Beck et al. (2001) propuseram:

> **Indivíduos e interações** mais que processos e ferramentas;
> **Software em funcionamento** mais que documentação abrangente;
> **Colaboração com o cliente** mais que negociação de contratos;
> **Responder a mudanças** mais que seguir um plano.

Beck et al. (2001) ressaltam que essas afirmações não significam que os itens à direita não sejam importantes, mas sim, que os itens à esquerda (em negrito) são considerados de maior relevância.

De acordo com Mochio e Araki (2012), o significado de "ágil" já é bem definido. Seus princípios auxiliam na iteração do desenvolvimento com base em poucos requisitos, prezando pela boa comunicação entre as partes interessadas e por responder regularmente a mudanças, que é algo que acontece muito no decorrer do desenvolvimento do software.

Com as iterações do ágil não é preciso definir o escopo total do projeto para iniciar seu desenvolvimento. Garantir uma boa e contínua comunicação os *stakeholder*, permite aos desenvolvedores garantir a satisfação das necessidades atuais do cliente e manter o produto com alto valor agregado. Iteração em curto prazo de desenvolvimento permite aos desenvolvedores lidar com as mudanças do mercado e do cliente de forma flexível (MOCHIO; ARAKI, 2012; WOLFF, 2012).

Misra et al. (2012) apresentam outras vantagens no emprego de abordagens ágeis no desenvolvimento de software, tais como: flexibilidade em aceitar exigências de mudança por partes dos clientes; não há a necessidade de definir o escopo final do produto; capacidade adaptativa das equipes de acordo com as mudanças nos requisitos de negócios. Por outro lado, tem havido comentários sobre os problemas próprios das abordagens ágeis de desenvolvimento, como por exemplo a falta de capacidade de desenvolver sistemas críticos de segurança e falta de escalabilidade (WOLFF, 2012).

Uma ampla gama de abordagens ágeis já existem em consequência as necessidades da indústria, que vão desde métodos individuais e comportamentais, como apresenta as práticas do XP (LINDSTROM; JEFFRIES, 2004), até estruturas de processos orientados para o gerenciamento de projeto, como por exemplo o Scrum (SCHWABER; AL, 2016). Muitos desses métodos possuem uma ampla aceitação na indústria de software como uma forma de garantir que os sistemas são criados no tempo e dentro do orçamento planejado (WOLFF, 2012).

Mesmo com avanço dessas abordagens há algumas restrições que limitam sua empregabilidade em desenvolvimento de alguns tipos de sistemas, como os críticos de segurança (WOLFF, 2012). Evidências de pesquisas em projetos ágeis indicam limitações na sua adoção na indústria, algumas relacionadas a engenharia de requisitos (PAETSCH; EBERLEIN; MAURER, 2003).

2.1.1 Scrum – Um *Framework* Ágil para Gestão de Projeto

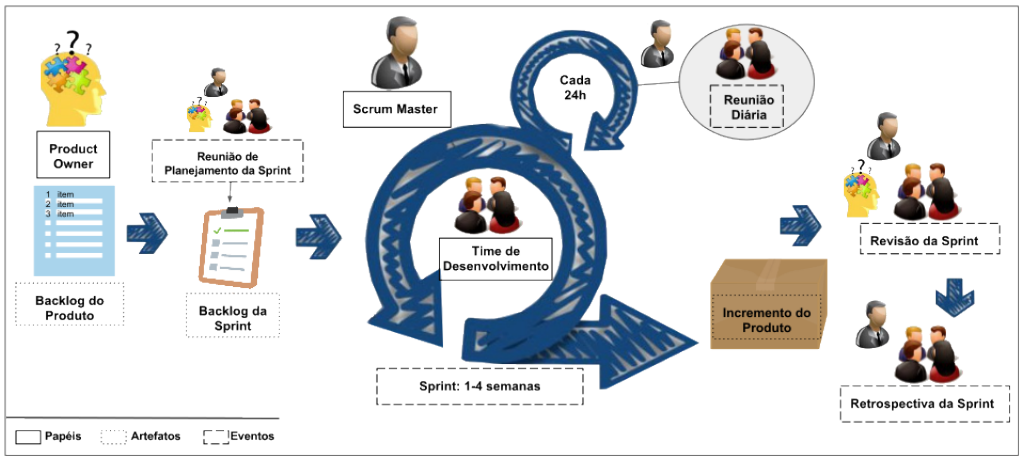
Scrum é um *framework* ágil estrutural para gerenciar o desenvolvimento de software, que vêm sendo utilizado desde 1990 (WOLFF, 2012). Ele não é uma técnica para construir software, mas sim um *framework* composto de diversas práticas que provém a integração de estratégias para eventos, artefatos e papéis, garantindo assim flexibilidade às mudanças (SCHWABER; AL, 2016).

O Scrum tem o propósito de garantir *feedback* constante dos clientes e interações curtas para o desenvolvimento do sistema. Ao envolver o usuário continuamente no processo iterativo é assegurado que o produto final realmente satisfaz as necessidades do usuário final. Porém, isso necessita de uma grande capacidade de adaptação dos processos e da equipe de desenvolvimento, a fim de apoiar as mudanças refletidas nas necessidades do usuário (WOLFF, 2012). De acordo com Cardozo et al. (2010), Scrum utiliza um processo empírico com base em flexibilidade, adaptabilidade e produtividade.

Como pode ser observado na Figura 2, o Scrum consiste nas pessoas associadas a papéis, eventos, artefatos e regras. Cada componente dentro do Scrum tem um propósito específico e é essencial para o seu uso e sucesso. O Time Scrum (TS) é composto por três papéis (SCHWABER; AL, 2016):

O ***Product Owner* (PO)** é a pessoa, ou conjunto de pessoas, que representa os interesses do cliente. É de sua responsabilidade gerenciar ou delegar ao time de desenvol-

Figura 2 – Ilustração do ciclo do Scrum.



Fonte: Adaptado de Schwaber e al (2016)

vimento, o escopo do projeto no ***Backlog* do Produto** e priorizar o que deve ser feito primeiro. O ***Scrum Master* (SM)** é descrito como o facilitador e tem como responsabilidade garantir que o Scrum está sendo entendido por todos envolvidos e executado da forma correta, além de garantir que a equipe não está distraída da tarefa em questão. Por fim, o **Time de Desenvolvimento (TD)** é representado pelo conjunto de desenvolvedores que trabalham colaborativamente para entregar o escopo selecionado para cada *Sprint*.

De acordo com Schwaber e al (2016), além dos papéis, o Scrum é dividido em diferentes eventos e artefatos. O ***Sprint*** é um intervalo de tempo de 1 a 4 semanas, onde o TD desenvolve determinados itens do *Backlog* do Produto, produzindo incrementos do produto final. ***Backlog* do Produto** é o artefato que é gerenciado pelo PO e contém itens que correspondem a versões generalistas e resumidas de partes do sistema, conjuntamente com suas prioridades. ***Backlog* da Sprint** é o artefato que contém os itens do *Backlog* do Produto que devem ser implementados no *Sprint*. **Reunião de Planejamento do *Sprint*** é a reunião onde o TD se encontra conjuntamente com PO e SM para planejar quais itens irão compor o *Backlog* da *Sprint*. Ainda nessa reunião, o TS define as atividades necessárias para construção do incremento. A **Reunião Diária** acontece todos os dias com o TD juntamente com o SM e dura de 10 a 15 minutos, onde cada um responde a três perguntas: "O que fiz no dia anterior?"; "Quais os impedimentos que tive para resolução das tarefas'?'; "O que farei hoje?". A **Reunião da *Sprint*** acontece para a equipe entregar o incremento para o PO e ele decide se aceitará ou não. Além disso, o *Backlog* do Produto é refinado. As decisões tomadas aqui provém as entradas para a reunião de planejamento da *Sprint*. Por fim, na **Retrospectiva da *Sprint***, que acontece após a Revisão da *Sprint*, e é focado nos processos, técnicas e relações que estão dando

certo e precisam continuar ou que precisam ser revisadas e melhoradas.

2.2 Especificação Formal (EF) de Requisitos

As EFs podem ser descritas como técnicas que utilizam um conjunto de notações, representadas por linguagens formais, em algum nível de abstração, com uma coleção de propriedades de alguma parte do sistema (SHAFIQ; MINHAS, 2014; LAMSWEERDE, 2000). EF é expressa em uma linguagem cujo vocabulário tem uma sintaxe e semântica bem definida. A necessidade de uma definição formal significa que as linguagens de especificação devem ser baseadas em conceitos matemáticos, onde suas propriedades são bem compreendidas. A matemática utilizada é a discreta e seus conceitos matemáticos são extraídos da lógica, álgebra e teoria dos conjuntos (DEHARBE et al., 2000; SOMMERVILLE, 2011).

Nesse cenário, o formalismo permite diminuir a ambiguidade das especificações que, ocasionalmente, ocorrem com utilização de linguagem natural para especificar requisitos (LAMSWEERDE, 2000). A utilização da semântica do formalismo fornece regras precisas de interpretação, permitindo a eliminação de muitos problemas encontrados na linguagem. Além disso, o formalismo utiliza linguagens com estruturas mais "ricas" e bem definidas. Ferramentas de apoio ao formalismo podem ser utilizadas para agilizar o processo de especificação (SOMMERVILLE, 2011; GAUDEL, 1994).

De acordo com Lamsweerde (2000), o "problema" que está sendo especificado pode ser algum tipo de modelo, como: modelo descritivo do domínio de interesse; modelo prescritivo do software e do seu ambiente; modelo prescritivo do software por si só; modelo para a interface do utilizador; modelo arquitetural do software; modelo de algum processo a ser seguido; entre outros. As propriedades consideradas podem referir-se: as metas de alto nível; requisitos funcionais; aos requisitos não funcionais; ao desempenho; à precisão; à segurança; às suposições ambientais; dos serviços prestados pelos componentes da arquitetura; e dos protocolos de interação entre esses componentes.

A precisão permite apenas uma interpretação para as informações descritas, bem como a consistência que garante que a especificação não contenha duas ou mais informações contraditórias, além disso a objetividade da especificação contém apenas informações relevantes para o desenvolvimento do software (LAMSWEERDE, 2000; SHAFIQ; MINHAS, 2014).

Vale ressaltar que em alguns casos a aplicação de métodos de EF para determinada parte do sistema pode ser feita com a finalidade de garantir um melhor entendimento e detalhamento do requisito. Da mesma forma, que ao tentar especificar esses detalhes utilizando linguagens naturais potencialmente pode ocasionar no mal-entendimento e ambiguidade desses requisitos (DEHARBE et al., 2000; LIPSKI, 2011; LARSEN; FITZGERALD; WOLFF, 2010).

A ideia principal é fazer o analista pensar mais na hora de especificar, pois as-

sim, o que era ambíguo na linguagem natural, passa a ser errado quando exemplificado formalmente em função da melhor visualização do erro. É necessário um maior esforço para especificar, mas a garantia de entendimento do problema e diminuição de custos, seja de retrabalho ou tempo, no desenvolvimento faz com que o esforço seja menor que o benefício do seu uso.

Segundo Sommerville (2011) existem duas abordagens fundamentais para a EF e que são utilizadas para escrever especificações detalhadas para sistemas de software industriais, são eles:

- **Abordagem algébrica:** onde o sistema é descrito em termos de operações e seus relacionamentos. Exemplos: *Software Cost Reduction* (SCR) ver (SAEED et al., 2014); CASL, ver (DEHARBE et al., 2000); *Language Of Temporal Ordering Specification* (LOTOS), ver (LOTOS, 1988).

- **Abordagem baseada em modelo:** onde um modelo de sistema é construído usando construções matemáticas, tais como conjuntos, sequências e operações do sistema, esses são definidos pela forma como eles modificam o estado do sistema. Exemplos: VDM, ver (FERREIRA; RACHID; SCHUARTZ, 2009) e Z ver (MADEY, 1990).

2.2.1 Notação Z para Especificação Formal de Requisitos

O MF para especificação de requisito Z foi criado na Universidade de Oxford no fim da década de 70 (MENEGAZZO; VIEIRA, 2007). A notação gerada pela linguagem Z está baseada em princípios da lógica e matemática discreta (MADEY, 1990). Em Z, os sistemas são modelados utilizando conjuntos e as relações entre os estados do sistema (SOMMERVILLE, 2011). Adicionalmente, Menegazzo e Vieira (2007) complementam que Z descreve a relação dos comportamentos e propriedades de um sistema e também algumas operações com utilização de pré e pós condições. Os símbolos de *Z* tem aumentado seus conceitos matemáticos inserindo cada vez mais conjuntos operacionais (SOMMERVILLE, 2011).

A especificação em Z é composta por um número de esquemas ou modelos, que são decomposições do requisito em partes menores. Estes esquemas podem ser utilizados e combinados em outros esquemas para caracterizar a relação das partes do sistema (MADEY, 1990). Uma das características fortes do Z são os símbolos da notação matemática, que por consequência herdou as propriedades e formalidades dessas notações (MENEGAZZO; VIEIRA, 2007).

Estruturalmente, Z pode ser definido por alguns elementos. Os **aspectos** podem ser estáticos ou dinâmicos. O primeiro, refere-se aos estados em que o sistema pode estar e os relacionamentos que são mantidos quando o sistema move de um estado para outro. O segundo, se refere as mudanças de estado que acontecem no sistema e a relação

entre as variáveis de entrada e saída, bem como as operações possíveis do sistema. As **definições estruturais** podem ser definidas por declarações, abreviações, definições axiomáticas, definições genéricas e tuplas. Além disso, os **conjuntos** podem ser utilizados para compreensão, produto cartesiano, pertinência, conjunto potência, união e interseção generalizada. As **relações** podem ser descritas pela imagem, domínio, imagem relacional, relação inversa, relação de fechos e composição. As **funções** representadas pela notação podem ser parciais, totais, funções injetoras, sobrejetoras, bijetoras e sobrecarga.

Além disso, Z também possui a estrutura de **sequências** que podem ser sequência vazia, filtros, operadores *head* e *tail*, recursão em sequências, concatenação de sequências e *bug*. Os **conceitos de lógica proporcional** são usados para representar conjunções, disjunções, implicações, negações, equivalências, tautologia e contradição. A **lógica de predicados** é relacionada com o quantificador universal, substituição e quantificador existencial. Ainda temos as **condições** que podem ser caracterizadas por dois momentos, pré-condições e pós-condições. A primeira, refere-se a definição de uma pré-condição para a operação resultar em sucesso. A segunda, apresenta o estado em que o valor está após a manipulação da operação. Ainda, as **invariantes** se referem aos domínios que um valor pode significar ou pertencer. Por fim, os **comportamentos** se referem as alterações de estados de um sistema devido a um determinado evento, ou seja, as mudanças de estado dele. No Anexo A é apresentado a estrutura completa de símbolos da notação em Z.

Expressamos o uso do Z a partir do exemplo apresentado por Madey (1990), onde é proposta a especificação de um sistema que registra aniversários de pessoas, esse sistema é capaz de, além de cadastrar, emitir um aviso quando o dia chega. O primeiro esquema elaborado com a utilização do Z é apresentado como *BirthdayBook*, onde uma Pessoa é representada por nome e data de nascimento, sendo *NAME* para nome e *DATE* para data de nascimento.

$$
\begin{array}{|l}
\hline
BirthdayBook \\
known : \mathbb{P}\ NAME \\
birthday : NAME \pfun DATE \\
\hline
known = \mathrm{dom}\ birthday \\
\hline
\end{array}
$$

Podemos observar no esquema *BirthdayBook* temos o elemento *known* que é o conjunto de nomes com aniversários registrados no *BirthdayBook*. O elemento *birhday* é uma função que quando aplicada a um *NAME* retorna uma *DATE* associada a ela. Por fim, a definição *known dom* representa que o *known* é do mesmo domínio (*dom*) da função *birthday*, ou seja, esse relacionamento é invariante no sistema, nunca muda estado. Um possível estado do sistema pode ser descrito com três pessoas para o *known* e com três aniversários para função *birthday*, como apresentado no exemplo a seguir:

known = {John, Mike, Susan} e *birthday* = {John $\mapsto$ 25-Mar, Mike $\mapsto$ 20-Dez, Susan $\mapsto$ 20-Dez}.

Nesse caso, a invariante é satisfeita, pois *birthday* registra uma data para cada nome conhecido. Na sequência, apresentamos o esquema *AddBirthday* que ilustra a operação de adicionar um novo aniversário:

$$\begin{array}{|l} \hline \Delta\ AddBirthday \\ name? : NAME \\ date? : DATE \\ \hline name? : \notin known \\ birthday' = birthday \cup \{name? \mapsto date?\} \\ \hline \end{array}$$

Nesse esquema a declaração Δ *AddBirthday* descreve a mudança de estado que introduz três variáveis, sendo elas: *known*, *birthday* e *birthday'*. O simbolo Δ (Delta) significa que essa operação mudará o estado do sistema. O ponto de interrogação (?) significa um valor informado pelo usuário. Além disso,é definida a pré-condição, sendo que o nome informado não deve ser um dos conhecido do sistema. Por fim, também temos a pós-condição *birthday'* que descreve o estado do sistema após a alteração do estado, caso não haja alteração o valor não atualizará.

2.3 Especificação Informal (EI) de Requisitos

Os métodos para EI são os mais difundidos para representação de requisitos em desenvolvimento de software. Isso está diretamente ligado a facilidade de entendimento por todos envolvidos no projeto.

Caracteriza-se por informal, os métodos que utilizam a linguagem natural, onde utiliza-se a estrutura lógica da linguagem para representar as informações do sistema. Alguns métodos são amplamente utilizados para especificação, como História de Usuário (LUCASSEN; AL., 2016) e Caso de Uso (CHALFOUN; AL., 2014).

2.3.1 Casos de Uso para Especificação Informal de Requisitos

Furlan (1998) afirma que um diagrama *Unified Modeling Language* (UML) de caso de uso descreve a visão externa de um sistema e suas interações com o mundo. Um caso de uso expressa o comportamento do sistema em algumas situações respondendo a requisições de um dos *stakeholders* (COCKBURN, 2005).

Embora casos de uso possam ser escritos usando fluxograma, fundamentalmente são utilizadas formas textuais para escrevê-los. O caso de uso é classificado por três conceitos: escopo, onde define qual o sistema que está sob discussão; ator primário, quem possui o objetivo do contexto; nível, quão alto ou baixo é o nível do objetivo. De acordo com Cockburn (2005) o **Ator** é alguém ou algo com comportamento. Os ***Stakholders*** são os interessados no comportamento do sistema sob discussão. Ainda, o **Ator Primário** é o interessado que inicia uma interação com sistema, com finalidade de alcançar um objetivo. As **pré** e **pós condições** são o que deve ser verdadeiro antes da execução e o

estado posterior a dela, respectivamente. Os **cenários** são descritos por um cenário de sucesso e quando há algo errado com cenários alternativos. Por fim, as **extensões** que podem ocorrer durante o caso de uso também são descritas.

Um exemplo de caso de uso descritivo apresentado em Cockburn (2005) é ilustrado na Tabela 1.

Tabela 1 – Exemplo de Caso de Uso

Ator Primário	Requerente
Escopo	Seguradora (“MyInsCo”)
Nível	Resumido
Stakeholders e Interesses	Requerente: Receber o máximo possível; MyInsCo: pagar a menor quantia apropriada; Departamento de Seguro: observar se todas as diretrizes foram seguidas.
Pré-condição	nenhuma
Garantia Mínima	MyInsCo registra a reivindicação e todas as atividades.
Garantia de Sucesso	Requerente e MyInsCo concordam sobre a quantia a ser paga; requerente a receber.
Acionador	Requerente submete a reivindicação
Cenário de Sucesso Principal	1. Requerente submete reivindicação com dados substâncias; 2. Seguradora verifica que requerente possui uma apólice válida; 3. Seguradora designa agente para examinar caso; 4. Seguradora verifica que todos os detalhes estão dentro das diretrizes da apólice; 5. Seguradora paga requerente e fecha arquivo
Extensões	1a. Dados submetidos estão incorretos: 1a1. Seguradora solicita informação ausente; 1a2. Requerente fornece informação ausente. 2a. Requerente não possui uma apólice válida: 2a1. Seguradora nega reivindicação, notifica requerente, registra tudo isso, termina requerimento. 3a. Nenhum agente está disponível no momento: 3a1. (O que a seguradora faz aqui?). 4a. Acidente viola diretrizes básicas da apólice: 4a1. Seguradora nega reivindicação, notifica requerente, registra tudo isso, termina requerimento. 4b. Acidente vila algumas diretrizes secundárias da apólice: 4b1. Seguradora inicia negociação com requerente sobre quantia de pagamento a ser feito.

Fonte: (COCKBURN, 2005)

2.4 Engenharia de Software Experimental

Estudos experimentais têm sido utilizados como um mecanismo para adquirir conhecimento através de uma abordagem científica baseada na medição de fenômenos em

diferentes áreas (YIN, 2013; WOHLIN et al., 2012). Estudos empíricos são importantes para apoiar a realização desses requisitos e se enquadram no contexto da engenharia de software industrial e acadêmica. Esses estudos requerem o uso científico de dados quantitativos e qualitativos para entender e melhorar algum aspecto da Engenharia de Software. O objetivo principal é evoluir o conhecimento em Engenharia de Software, a partir da aplicação de uma abordagem sistemática na construção de novos métodos e técnicas para apoio ao desenvolvimento de software (WOHLIN et al., 2012).

De acordo com Wohlin et al. (2012), dependendo do propósito da avaliação (técnicas, métodos, ferramentas ou investigação empírica), existem três tipos diferentes de estratégias empíricas para investigação, sendo eles: *survey*, estudo de caso e experimento.

Os autores Wohlin et al. (2012) e Molléri, Petersen e Mendes (2016) afirmam que um ***Survey*** é uma estratégia para coletar informações de uma amostra representativa, de interesse geral, para descrever, comparar ou explicar seu conhecimento, atitudes e comportamento. Nesse tipo de pesquisa as principais coletas de dados são através de informações qualitativas ou quantitativas, gerados a partir de entrevistas ou questionários. Um *survey* pode ser conduzido para explicar ou explorar um determinado comportamento. *Survey* de carácter explicativo permite asserções sobre uma determinada população, por exemplo entender e explicar o porque desenvolvedores preferem usar uma determinada técnica. Já um *survey* exploratório, é geralmente utilizado como uma pré-pesquisa de um estudo mais aprofundado, pois coleta indícios e uma visão geral sobre uma determinada área.

Em um survey exploratório os resultados gerados servem como guia para melhoria de futuros estudos, além de levantar indícios sobre o objetivo de estudo da pesquisa mais completa (WOHLIN et al., 2012). Em Engenharia de Software, esse tipo de *survey* é um dos métodos de pesquisa mais utilizados para a realização de estudos de investigação empírica para coleta de percepção. Como em qualquer estudo empírico, para condução de um *survey* é preciso: definir os objetivos de pesquisa; identificar as características do público alvo; preparar os artefatos; elaborar os questionários; realizar um piloto; divulgar o questionário e analisar os resultados gerados através de medidas quantitativas ou qualitativas (MOLLÉRI; PETERSEN; MENDES, 2016).

Um **Estudo de Caso** se baseia em múltiplas fontes de evidências para investigar uma instância (ou um pequeno número de instâncias) de um fenômeno estudado em engenharia de software, dentro do seu contexto real, especialmente quando a fronteira entre fenômeno e contexto não pode ser claramente especificada (WOHLIN et al., 2012; YIN, 2013). Estudos de casos são comumente utilizados para pesquisar projetos, atividades ou atribuições. Os dados são coletados para um propósito específico ao longo do estudo. Geralmente as pesquisas com estudos de caso são conduzidas através de método observacional, onde o pesquisador participa pessoalmente do andamento da experiência coletando informações (WOHLIN et al., 2012; YIN, 2013).

Semelhante ao *survey*, um estudo de caso pode ser exploratório, descritivo ou explicativo (WOHLIN et al., 2012). Em um estudo de caso existe menos exigência de haver um planejamento completo do estudo. Por exemplo, o estudo pode ter objetivos iniciais, mas pode ir evoluindo conforme a execução. Da mesma forma que os objetivos, as questões de pesquisa podem ir evoluindo conforme a execução. Estudos de caso podem ser caracterizados como holísticos, onde existe um caso e uma unidade de análise, ou estudos incorporados, onde podem existir múltiplas unidades de análise para um mesmo estudo. Por exemplo, executar dois projetos distintos em duas empresas distintas para estudar as práticas ágeis (YIN, 2013; WOHLIN et al., 2012).

Nesses estudos podem ser utilizados diferentes instrumento para coleta de dados, incluindo questionários e entrevistas. Esse último é geralmente utilizado para gerar grande quantidade de informações, para serem compiladas e analisadas posteriormente. Essas entrevistas podem seguir uma diferente estruturação, como por exemplo não estruturadas e focadas (MARCONI; LAKATOS, 2003), onde uma lista de tópicos é definida como um guia, mas podendo eliminar ou adicionar tópicos específicos de acordo com as respostas.

Os **experimentos** são estudos que manipulam um fator ou variável de configuração estudada. Wohlin et al. (2012) afirma que diferentes tratamentos são aplicados com diferentes sujeitos, mantendo outras variáveis constantes para medir os efeitos desses tratamentos em um ambiente estudado. Em geral, as experiências são executadas em um laboratório controlado, sendo esse tipo de execução caracterizado como execução **in-vitro**. Além disso, também é possível executar experimentos em ambientes reais, industriais, se caracterizando em execução *in-vivo*.

Nos experimentos diferentes variáveis são definidas. As **variáveis dependentes** descrevem os valores das medições que representam o desempenho e eficácia das abordagens testadas. As **variáveis independentes** descrevem os tratamentos, ou seja, as técnicas utilizadas, sendo as variáveis para as quais devem ser avaliados os efeitos do experimento. As variáveis de controle servem para garantir um controle do *design*. Para os tratamentos, são formulados hipóteses para o experimento, onde se objetiva refutar a hipótese nula. Para isso, os dados gerados passam por testes estatísticos de hipóteses, identificando assim se as diferenças dos dados são significativamente diferentes (WOHLIN et al., 2012).

De acordo com Wohlin et al. (2012) um processo experimental deve identificar claramente as preocupações sobre os diferentes tipos de ameaças para a validade da experiência. Para todo tipo de estudo, seja *survey*, estudo de caso ou experimentos é preciso identificar as ameaças para ser possível garantir um melhor controle das experiências, podendo prever possíveis acontecimentos que podem dificultar ou até mesmo invalidar o estudo. Wohlin et al. (2012) define a classificação de esquema dividido em quatro tipos de ameaças. As ameaças à **validade a conclusão** afetam a capacidade de realizar conclusões sobre as relações dos tratamentos (técnicas utilizadas) e os resultados gerados,

elas podem ser relacionadas por exemplo em: confiabilidade das medidas; irrelevâncias na configuração experimental; heterogeneidade dos participantes.

Além disso, as ameaças à **validade interna** garantem que o relacionamento observado entre o tratamento e os resultados são casuais, e não sendo influenciados por outro fator não controlado, o qual pode estar relacionado a: história; maturidade; seleção. As ameaças à **validade externa** tem a ver com a generalização dos resultados. Se existe uma relação causal entre a causa e o efeito, sendo que o resultado do estudo pode ser generalizado fora do alcance do nosso estudo, podendo ser: dos sujeitos; das tarefas; dos efeitos ou da experiência. Por fim, as ameaças à validade da construção consideram os relacionamentos entre a teoria e a observação, garantindo que o tratamento reflete a causa e efeito do experimento.

2.5 Lições do Capítulo

Ao longo desse capítulo foi possível compreender o significado das abordagens ágeis para o desenvolvimento de software. Também percebemos algumas limitações que as abordagens ágeis enfrentam, em especial, as que referem-se ao uso exclusivo especificação e avaliação informal.

Da mesma forma, percebemos uma boa oportunidade para inclusão do formalismo para o desenvolvimento de software, devido a possibilidade de especificar formalmente os requisitos e validá-los dentro do processo de desenvolvimento ágil. As EF proporcionam clarificar as necessidades do cliente através do método Z, por exemplo. Esse método possibilita descrever aspectos dos sistemas como propriedades e estados e que "forçam" o analista a escrever requisitos mais elaborados e mais concretos.

Também observamos no capítulo a apresentação de uma instância da UML para representação dos requisito, Caso de Uso. Nele, podemos escrever detalhadamente cenários de interação do usuário final com o sistema ainda não desenvolvido. Por fim, descrevemos os fundamentados em torno de engenharia de software experimental, ilustrando os conceitos estudos empíricos como *survey*, estudo de caso e experimento.

3 TRABALHOS RELACIONADOS

O propósito deste capitulo é apresentar o mapeamento sistemático realizado da literatura, o qual objetivou estudar o estado da arte da aplicação de EF em abordagem ágil. Esse mapeamento foi executado conforme guia proposto por Petersen et al. (2008). Na seção 3.1 apresentamos o protocolo de pesquisa. Adicionalmente na seção 3.2 são ilustrados os resultados gerados pelo mapeamento. Por fim, na seção 3.3 discutiremos sobre os aprendizados e percepções obtidas.

3.1 Metodologia do SMS

O propósito deste mapeamento sistemático é estudar o estudo da arte da aplicação de EF em abordagens ágeis. O primeiro passo é a definição do escopo do mapeamento, ao qual se dá a definição de um conjunto de pesquisa. Definimos nosso escopo em saber quais métodos de EF são comumente utilizados para especificar requisitos em abordagens ágeis. Bem como descobrir quais desafios e benefícios que essa aplicação proporciona para o desenvolvimento de sistemas. Consolidando essas dúvidas levantadas, definimos as seguintes questões:

QP1. *Quais são os métodos de especificação formal aplicáveis em abordagens ágeis?*

QP2. *Quais são as limitações na utilização de especificação formal em abordagens ágeis?*

QP3. *Quais são os benefícios no uso de especificação formal em abordagens ágeis?*

Estabelecemos o nosso processo de busca a partir da definição das bases de dados. Utilizamos o *Scopus* como base de dados por indexar uma boa quantidade de dados e por indexar outras bases de pesquisa como *Institute of Electrical and Electronics Engineers* (IEEE) e *Association Computing Machinery* (ACM) Digital Library. A seguinte *Query* foi executada:

TITLE-ABS-KEY[1] *(agile PRE/0 (methodology OR development OR approach)) AND ALL (formal PRE/0 (specification OR technique OR method)) AND PUBYEAR > 2009*

Para pesquisas complementares e encontrar alguns trabalhos interessantes, que não apareceram no *Scopus*, acrescentamos a busca no *Google Scholar*. Utilizamos a *Query* abaixo:

"formal specification" AND ("formal techniques" OR "formal method" OR "especificação formal") AND ("agile methodologies" OR "agile development" OR "metodologia ágil" OR "desenvolvimento ágil" OR "agile approach")

[1] Significa que a busca será realizada no título, resumo e palavras-chave dos trabalhos.

Na sequência, para eliminar os resultados falsos-positivos e refinar os trabalhos encontrados no estudo primário definimos os seguintes critérios de seleção:

- **Critérios para inclusão de trabalhos:**
  - Trabalhos que propõem a aplicação de práticas de especificação formal em metodologia ágeis.
- **Critérios de exclusão como sendo trabalhos que:**
  - descrevem apenas aspectos da área de especificação formal sem mencionar abordagens ágeis;
  - descrevem apenas aspectos da área de abordagens ágeis sem mencionar especificação formal;
  - quando publicados em mais de uma base de dados (revistas, periódicos e conferências), utilizaremos a versão mais completa, isto é, aquela que explicar em um nível com maior detalhes;
  - trabalhos de no mínimo 6 páginas;

Realizamos a leitura dos *Abstracts* dos trabalhos indexados pela pesquisa para podermos classifica-los e agrupa-los. Além disso, em alguns casos foi necessário realizar uma leitura mais apurada do restante do trabalho para obter um melhor entendimento.

Para cada trabalho, realizamos *Keywording* e agrupamos de acordo com as questões de pesquisa respondidas.

3.2 Resultados do SMS

Inicialmente realizamos a pesquisa preliminar dos trabalhos utilizando as *Queries* nas bases de dados, totalizando em 206 trabalhos relacionados. Desses, 31 foram indexados pelo *Scopus* e outros 175 pelo *Google Scholar*. Após aplicarmos os critérios de seleção totalizamos em doze trabalhos finais para fundamentar o mapeamento sistemático deste presente trabalho.

A Tabela 2 sumariza quantitativamente os trabalhos selecionados em cada etapa nas bases de dados. Adicionalmente na Tabela 3 é descrito os doze trabalho relacionados selecionados.

A seguir, apresentamos as questões definidas no protocolo de pesquisa, bem como as respectivas respostas dos autores dos trabalhos relacionados selecionadas no mapeamento.

Tabela 2 – Quantidade de trabalhos selecionado por etapa e base de dados

Etapa	**Google Scholar**	**Scopus**	**Total**
Pesquisas da busca automática sem aplicar critérios	175	31	**206**
Depois de aplicar o critério de inclusão no título e resumo	12	14	**26**
Depois de fazer uma análise mais apurada, aplicando critérios de exclusão	7	5	**12**

Fonte: Elaborado pelo autor

Tabela 3 – Trabalhos selecionados no mapeamento sistemático

Título	**Local de Publicação**	**Autor**
Formality, Agility, Security, and Evolution in Software Development	Journal - Computer	Bowen et al. (2014)
An approach to applying SOFL for agile process and its application in developing a test support tool	Journal - Innovations in Systems and Software Engineering	Liu (2010)
DevOps meets formal modelling in high-criticality complex systems	Proceedings of the 1st International Workshop on Quality-Aware DevOps - QUDOS 2015	Olszewska e Waldén (2015)
Agile methods for open source safety-critical software	Journal - Software: Practice and Experience	Gary et al. (2011)
UI-driven test-first development of interactive systems	Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '11	Bowen e Reeves (2011)
Requirements Modeling in Agile Methodologies with X-Machines	Journal - Biuletyn Instytutu Systemów Informatycznych	Lipski (2011)
VDM++ as a Basis of Scalable Agile Formal Software Development	Proceedings - 9th Overture Workshop on VDM	Mochio e Araki (2012)
Integrating Formal Methods in XP—A Conceptual Solution	Journal of Software Engineering and Applications	Shafiq e Minhas (2014)
Are Formal Methods Ready for Agility? A Reality Check	2nd International Workshop on Formal Methods and Agile Methods	Larsen, Fitzgerald e Wolff (2010)
Mapping Formal Methods to Extreme Programming (XP) –A Futuristic Approach	Journal - International Journal of Natural and Engineering Sciences	SAEED et al. (2014)
Scrum goes formal: agile methods for safety-critical systems	2012 1st International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches, FormSERA 2012 - Proceedings	Wolff (2012)
Supporting agile development by facilitating natural user interaction with executable formal specifications	Journal - ACM SIGSOFT Software Engineering Notes	Nummenmaa et al. (2011)

Fonte: Elaborado pelo autor

3.2.1 Quais são os métodos de especificação formal aplicáveis em abordagens ágeis?

A partir da leitura dos trabalhos relacionados, percebemos que os métodos de EF VDM e Z são os mais utilizados para exemplificar a aplicação de formalismo em abordagem ágil. VDM é apresentado em 50% dos trabalhos e o Z em 30%. A utilização de mais de um método de EF é ilustrada em 40% dos trabalhos, como indica nos trabalhos de Liu (2010), SAEED et al. (2014), Wolff (2012), Shafiq e Minhas (2014).

Na Tabela 4 demonstramos os trabalhos que descrevem a aplicação de EF em abordagens ágeis.

Tabela 4 – Métodos de especificação formal apresentados nos trabalhos selecionados

Autor	**SCR**	**VDM**	**DbC**	**SOLF**	**Event-B**	**DisCo**	**Z**	**Outros**
Liu (2010)		X		X			X	
Lipski (2011)								X
SAEED et al. (2014)	X		X					
Olszewska e Waldén (2015)					X			
Wolff (2012)		X		X	X			
Bowen e Reeves (2011)							X	
Mochio e Araki (2012)		X						
Shafiq e Minhas (2014)		X					X	X
Larsen, Fitzgerald e Wolff (2010)		X						
Nummenmaa et al. (2011)						X		

Fonte: Elaborado pelo autor

Em SAEED et al. (2014) é proposta uma nova abordagem de desenvolvimento denominada *Extreme Programming Formal* (XPF) que integra práticas formais com XP. Esta abordagem utiliza EF algébrica através do método SCR. O método SCR possibilita a construção de modelos formais que descrevem os requisitos e todos seus aspectos. Os autores também utilizam o método *Design by Contract* (DBC) na etapa de teste de aceitação para especificar as pré-condições, invariantes e limitações sobre as funções do sistema. Os autores afirmam que métodos para especificar formalmente requisitos não são tão caros e terríveis como são considerados nos últimos anos e que é viável a aplicação de MF através das EF nas abordagens ágeis.

No trabalho de Lipski (2011), é descrito o método de EF X-Machine e sua extensão *Extreme X-Machines* (XXM) para EF e modelagem de requisitos em abordagens ágeis. O autor afirma que esta integração objetiva alcançar sistemas corretos de uma forma ágil. Resultante disso, uma extensão de X-Machine denominado XXM foi proposta. Os autores destacam a facilidade de entender e aplicar essa abordagem, sendo independente de qualquer processo de desenvolvimento, encontrando a eficácia inclusive para processos ágeis.

Liu (2010) propõe uma nova abordagem de desenvolvimento denominada *Structured Object-Oriented Formal Language* (SOLF). Os autores afirmam que os métodos VDM e Z podem ser utilizados para especificar formalmente os requisitos nesta abordagem. A abordagem SOLF divide as especificações em três etapas: construção da EF; inspeção baseado nas especificações; e testes para validação e verificação. Dentre essas etapas as

EF são escritas apenas para ajudar na compreensão das declarações ambíguas na especificação semi-formal. O trabalho objetiva estabelecer o equilíbrio entre entrega rápida de produto utilizando abordagem ágil e a entrega de produto com qualidade com aplicações de formalismo. Essa abordagem foi testada em um projeto para o desenvolvimento de um suporte de ferramenta automatizada para testes baseados em especificações, porém apenas especificações semi-formais foram utilizadas.

Em Shafiq e Minhas (2014) é proposta uma solução conceitual que utiliza as práticas do XP e os métodos de EF Z e VDM, através da extensão *Vienna Development Method Specification Language* (VDM-SL). Diferentemente de Liu (2010), as EF são escritas a partir das *User Stories* geradas na análise do problema. A partir disso, são gerados modelos abstratos e extraídas as especificações concretas do sistema. Os autores ressaltam que com modelos formais elaborados a partir das especificações formalizadas é possível aplicar práticas de *Test Driven Development* (TDD) e verificação formal antes mesmo dos requisitos serem implementados. Um estudo de caso foi organizado para verificar essa proposta. Dez estudantes foram divididos em dois grupos onde um executou XP formal e o outro o XP tradicional. Para o XP formal foram utilizados os métodos para EF VDM e Z. Os autores compararam os resultados gerados pelos dois grupos e notaram que utilizando o XP formal o tempo global do desenvolvimento foi reduzido, mesmo que nas fases de planejamento e design tenha sido mais demoradas que no XP tradicional. Além de garantir uma melhor qualidade do software baseado nos testes executados.

Em outro trabalho, Wolff (2012) inclui EF na abordagem ágil de gestão de projeto SCRUM. O autor afirma que os métodos de EF SOLF, Event-B e VDM, com suas extensões VDM-SL, VDM++ e *Vienna Development Method Real-Time* (VDM-RT), podem ser utilizados no SCRUM para a formalização dos requisitos. Entretanto, Wolff (2012) ressalta que o desenvolvedor pode aplicar qualquer outro método de EF. Nesta abordagem o *Sprint Backlog* é dividido em especificar formalmente os requisitos críticos e não entendidos e em implementar as tarefas convencionais com maior entendimento. O autor apenas propôs essa abordagem e ressalta que a proposta ainda precisa de experimentos para ser considerada uma boa prática para produzir software. Por fim, ele acredita que com a integração de EF em abordagens ágeis é possível aumentar o uso de abordagens ágeis para desenvolver sistemas críticos de segurança e garantir a qualidade do software.

A abordagem FormAgi proposta por Olszewska e Waldén (2015) utiliza o método de EF Event-B integrado as práticas do SCRUM. Esse método é proposto para refinar os requisitos e representar o sistema em diferentes níveis de abstração. Essa abordagem repeita os princípios do contexto DevOps. Esse contexto é definido por conjuntos de métodos que combinam mecanismos de garantia da qualidade, com operações de TI dentro das práticas de engenharia de software para modelagem formal. Os autores relatam que ainda é uma proposta e que é preciso realizar estudos de caso para validar esta abordagem.

Bowen e Reeves (2011) propõem uma abordagem adequada para apoiar *Test First*

Development (TFD) para sistemas interativos por meio de testes abstratos. Esses testes podem ser gerados automaticamente a partir de modelos de interface gráfica. As etapas propostas pela abordagem são: análise e EF dos requisitos; elaboração de protótipo baseado nas especificações; validação com cliente(s); geração de testes automáticos utilizando o *framework* JUnit. Para especificar formalmente os requisitos gerados na análise, é utilizada a linguagem formal Z. Os resultados do trabalho apoiam a automação de testes garantindo a cobertura do sistema e a utilização de EF em práticas ágeis como TDD e TFD.

Os autores Nummenmaa et al. (2011) apresentam um método de EF flexível para facilitar a interação natural do usuário com especificações executáveis através da interface final. Nesse trabalho é afirmado que esta interação pode ser considerada como uma forma ágil e formal, pois fornece componentes de software testável através de opiniões das partes interessadas. A aplicação do método de EF DisCo baseia-se em uma modificação na execução do desenvolvimento, onde é utilizado um software que cria e anima especificações formalizadas. Após realizarem um estudo exploratório os autores afirmam que EF e abordagens ágeis não são conflitantes e se combinados corretamente podem trazer muitos benefícios.

Por fim, Larsen, Fitzgerald e Wolff (2010) realizam um comparativo onde são apresentados os doze princípios das abordagens ágeis e como os MF podem contribuir com cada um deles. Para exemplificar o uso de MF em abordagens ágeis os autores descrevem o método de EF VDM. Eles apresentam alguns usos do VDM nas indústrias. No decorrer do trabalho, os autores levantam um questionamento: “Mas há realmente uma necessidade de combinar técnicas formais e ágeis?”. Eles afirmaram que devido ao crescimento de desenvolvimento de software na indústria é preciso apoiar técnicas que trazem benefícios para o processo de desenvolvimento, e que é possível combinar formalismo e abordagens ágeis, porém é preciso criar ferramentas para apoiar as práticas formais no desenvolvimento.

Adicionalmente, Mochio e Araki (2012) também apresentam o VDM como método de especificação para utilização em abordagens ágeis. Neste trabalho é apresentado a linguagem de EF VDM++, extensão orientada a objeto da descrição de EF do VDM, aplicado no método de desenvolvimento *Scalable Agile Formal* (SAF). O SAF é utilizado para desenvolvimento de sistemas críticos e de alta escala de uma maneira ágil. Após, eles apresentam a estrutura do VDM para SAF. Logo apresenta alguns usos desta abordagem utilizando o VDM. Mochio e Araki (2012) afirmam que o VDM++ é uma linguagem de EF que se assemelha a linguagens de programação comuns na descrição de seu estilo e que por isso, torna-se fácil de utilizar e aprender, facilitando o uso em abordagens ágeis.

3.2.2 Quais são as limitações na utilização de especificação formal em abordagens ágeis?

De acordo com os trabalhos relacionado, a principal dificuldade na aplicação de EF em abordagens ágeis de desenvolvimento, descrita em 70% dos trabalhos, é a complexidade em aprender e escrever formalmente os requisitos. O tempo para especificar e aplicar corretamente o formalismo é apresentado em 30% dos trabalhos. Por fim, em 20% dos trabalhos os autores afirmam que ainda é preciso aumentar o aspecto ágil do formalismo para elevar seu uso em ambiente ágil, pois ainda é necessário criar ferramentas com boa usabilidade para automatizar o formalismo. Apontamos na Tabela 5 os problemas ilustrados nos trabalhos selecionados que comumente são encontrados na aplicação de EF em abordagens ágeis.

Tabela 5 – Limitações apresentadas pelos trabalhos relacionados

Autor	**Aprendizado**	**Tempo para especificar**	**Aplicação correta**	**Ferramentas de apoio**	**Outros**
Shafiq e Minhas (2014)			X		
Lipski (2011)					X
SAEED et al. (2014)	X	X	X		
Wolff (2012)	X		X		X
Olszewska e Waldén (2015)	X	X			
Bowen et al. (2014)	X			X	
Mochio e Araki (2012)	X				
Liu (2010)	X	X			
Nummenmaa et al. (2011)					X
Larsen, Fitzgerald e Wolff (2010)	X			X	

Fonte: Elaborado pelo autor

De acordo com Bowen et al. (2014) as dificuldades em utilizar as ferramentas de apoio às EF podem acarretar em atrasos no desenvolvimento. Larsen, Fitzgerald e Wolff (2010) fortalecem a necessidade em criar mais ferramentas para apoiar as EF, pois dessa forma, especificar formalmente os requisitos não se torna uma tarefa onerosa e demorada. Os autores ainda afirmam que só é possível obter benefícios do formalismo se a técnica ou notação formal for fácil de aprender e aplicar, tendo em vista que talvez quem vá ler e desenvolver os modelos formais não obtém domínio claro de formalismo.

Como afirmam os autores Shafiq e Minhas (2014), SAEED et al. (2014), há uma grande necessidade de além de aprender e escrever especificações formalizadas, se deve saber utiliza-las corretamente, pois utilizações equivocadas resultam em retrabalho (LIPSKI, 2011). De acordo com Lipski (2011) e Nummenmaa et al. (2011) outro aspecto que resultam em retrabalho é as mudanças de requisitos em última hora, dado a necessidade de primeiramente refatorar os requisitos formais, para assim realizar ajustes na

implementação do software produzido.

SAEED et al. (2014), Olszewska e Waldén (2015), Liu (2010) afirmam que o tempo em escrever formalmente as especificações muitas vezes é longo, dessa forma pode tornar mais demorada a fase de especificação e consequentemente atrasar o resto do desenvolvimento. Nummenmaa et al. (2011) ressalta que as formas dos clientes visualizarem os resultados de uma EF executável, muitas vezes é criticada por ser difícil de entender, portanto, sendo um grande desafio para as técnicas de EF em abordagens ágeis.

Por fim, no trabalho de Wolff (2012) é descrito outros desafios importantes e emergentes encontrados na aplicação de EF em abordagens ágeis, sendo eles: risco de aumentar o custo e não fornecer sistemas a tempo; carga de trabalho adicionada na análise do projeto; custo de aprendizado muitas vezes supera os benefícios potenciais de integração; e problemas em equilibrar o número de tarefas formais e convencionais.

3.2.3 Quais são os benefícios no uso de especificação formal em abordagens ágeis?

Em (54.54%) dos trabalhos estudados do mapeamento é indicado que os principais benefícios dessa integração são melhorar a compreensão dos requisitos e o aumento da qualidade do software. Em 27.27% dos trabalhos é afirmado que com especificações bem claras é possível garantir que especificações concretas sejam geradas em pouco tempo. O aumento da utilização de abordagem ágil em desenvolvimento de sistemas críticos de segurança também é uma das vantagens citadas por 27.27% dos autores. 36.36% dos trabalhos descrevem que melhorar a comunicação entre as partes interessadas no projeto é um benefício resultante dessa integração, seja para o melhor entendimento dos requisitos pelos desenvolvedores ou até mesmo pelos usuários finais. Outros benefícios foram indicados por 36.36% dos trabalhos, como: diminuir o risco do projeto e clarificar os requisitos para o usuário através da visualização das EF.

Na Tabela 6 sumarizamos as contribuições, apresentadas pelos trabalhos relacionados, que a integração de práticas formais em abordagens ágeis podem trazer para produção de software.

Os autores SAEED et al. (2014), Nummenmaa et al. (2011), Mochio e Araki (2012), Lipski (2011), Bowen et al. (2014) afirmam que com a aplicação de EF em abordagens ágeis não só eleva a possibilidade de alcançar alta qualidade dos softwares através da EF como também garantem a interação contínua com usuário final através das abordagens ágeis.

Bowen et al. (2014) afirmam que a inclusão das EF nas abordagens ágeis proporcionam para os desenvolvedores orientação no desenvolvimento. EF também trazem segurança para os testes devido seu formalismo. Além disso, os autores afirmam que é possível obter software seguro, pois os requisitos do software são formalizados e consequentemente garantem especificações concretas.

Tabela 6 – Benefícios apresentados pelos trabalhos relacionados

Autor	**Alta qualidade do software**	**Melhor compreensão dos requisitos**	**Especificações concretas**	**Potencializar o uso de metodologia ágil no desenvolvimento de sistemas críticos de segurança**	**Melhorar a comunicação entre os *stakeholders***	**Outros**
Mochio e Araki (2012)	X					X
SAEED et al. (2014)	X		X	X		
Wolff (2012)	X			X		X
Olszewska e Waldén (2015)	X				X	
Bowen et al. (2014)		X	X			
Shafiq e Minhas (2014)	X	X				X
Lipski (2011)		X			X	
Nummenmaa et al. (2011)	X	X			X	
Liu (2010)		X			X	X
Larsen, Fitzgerald e Wolff (2010)		X				
Gary et al. (2011)			X	X		

Fonte: Elaborado pelo autor

No trabalho de SAEED et al. (2014), são apresentados benefícios animadores. Os autores afirmam que a utilização de EF nas abordagens ágeis diminuem riscos do projeto em 99%. Os custos também são reduzidos principalmente nas fases subsequentes a de engenharia de requisitos. Adicionalmente, SAEED et al. (2014) e Wolff (2012) concordam que as EF podem auxiliar na qualificação de abordagens ágeis para o desenvolvimento de sistemas críticos de segurança, trazendo benefícios como documentação e avaliação formal dos projetos.

De acordo com Gary et al. (2011) as EF reforçam a segurança e confiabilidade dos sistemas críticos de segurança. Com a geração dos modelos formais é possível garantir especificações concretas ao ponto de serem validadas antes mesmo de serem implementadas. Mochio e Araki (2012) ressaltam que através do formalismo em abordagens ágeis é possível produzir software mais eficiente, porque cada um pode se livrar do problema do outro. Essa integração não só é possível descrever a arquitetura do sistema como também especificar, verificar e gerar automaticamente códigos-fonte consistentes.

Olszewska e Waldén (2015) afirmam que a mesclagem de EF e abordagem ágil acelera a entrega de artefatos, garantindo simultaneamente a sua qualidade e apoia iterações mais curtas. Essa mesclagem também facilita a comunicação intra-projeto envolvendo a equipe de desenvolvimento e os atores do sistema.

Além desses benefícios citados até o momento, os autores (SHAFIQ; MINHAS, 2014) afirmam que as EF podem elevar a qualidade dos produtos de software, reduzir taxa

de erro e melhorar a eficiência de tempo de entrega. O formalismo pode ser facilmente transformado em testes conduzido por código automatizado, portanto produzindo código livre de erro de requisitos.

Em Wolff (2012) são apresentadas outras vantagens no uso de EF em abordagens ágeis. Essa integração pode gerar uma contribuição valiosa para a fase de implementação, além de permitir reutilizar casos de testes diminuindo o retrabalho e a carga de trabalho global, como também aumentar qualidade sobre código limpo, bem como entregar software no prazo e dentro do orçamento previsto, mas ainda facilitar o uso de EF em projetos industriais.

Por fim, Lipski (2011) apresenta outros benefícios para integração do método de EF *X-Machine* em abordagens ágeis, sendo eles: suporte para um desenvolvimento modular disciplinado, permitindo aos desenvolvedores decompor o sistema em questão, assim ajudando a lidar com a modelagem de sistemas de grande escala; fornece um estilo apropriado de desenvolvimento de modelos, permitindo o reuso de modelos de componentes já existentes; facilita o entendimento do usuário dos documentos produzidos na fase de modelagem, isso melhora a comunicação entre o usuário e o time de desenvolvimento; ferramentas para a animação automática do modelo auxiliando o usuário a monitorar o modelo proposto pelo time de desenvolvimento permitindo um *feedback* mais completo e imediato. Como consequência, ao fim de cada iteração o usuário fornece um valoroso *feedback* no começo do processo de desenvolvimento; time de desenvolvimento e os usuários aprendem enquanto desenvolvem com as abordagens ágeis; todos aprendem rápido devido a intuitividade do formalismo de *X-Machine*.

3.3 Lições do Capítulo

Através do mapeamento realizado, descobrimos que já existem inciativas na utilização de EF em abordagens ágeis. Existem alguns desafios que devem ser considerados, porém os benefícios proporcionados por essa integração são animadores e consideráveis quando analisado seus resultados.

Os principais aprendizados obtidos através de nossa pesquisa foram: a) é preciso demandar tempo e esforço para o aprendizado de métodos de EF; b) EF é apenas mais uma maneira de realizar engenharia de requisitos e pode ser utilizada em qualquer processo de desenvolvimento, incluindo os que adotam abordagens ágeis; c) preferencialmente, se deve ter um profissional especializado em formalismo dentro da equipe de desenvolvimento para agilizar o processo de especificação e não atrasar o resto do desenvolvimento; d) EF em abordagens ágeis aumenta a possibilidade de uso de práticas ágeis no desenvolvimento de sistemas críticos de segurança; e) dependente da complexidade do projeto, EF não contribuem para seu desenvolvimento; f) se combinados corretamente, um pode auxiliar nos pontos fracos do outro.

Entendemos que para desenvolver software devemos definir um conjunto de téc-

nicas para suprir as necessidades nas diferentes áreas do desenvolvimento. Sendo assim, concluímos que EF e abordagem ágil são apenas técnicas que podem ser utilizadas conjuntamente para auxiliar no desenvolvimento de um software, objetivando alcançar a satisfação dos *stakeholders* e a qualidade do projeto.

4 SURVEY – UMA PESQUISA EXPLORATÓRIA

Este capítulo tem por propósito apresentar resultados gerados através da primeira estratégia empírica com a realização de uma pesquisa exploratória com profissionais desenvolveres ágeis de software. O capítulo está estruturado da seguinte forma: seção 4.1 é apresentado o planejamento da pesquisa, como o desing geral, as questões de pesquisa e seleção dos participantes; seção 4.2 apresentamos a operacionalização do **survey**; por fim, seção 4.3 e seção 4.5 descrevemos os resultados gerados e suas interpretações, além das lições aprendidas.

4.1 Planejamento

Elaboramos e conduzimos uma pesquisa de opinião descritiva (WOHLIN et al., 2012). Nesta pesquisa, buscamos identificar como EF pode contribuir com a especificação de requisitos em ambientes ágeis na perspectiva de profissionais atuantes em empresas de desenvolvimento de software. Para isso, elaboramos duas questões de pesquisa:

QP1) De que forma os métodos formais para a especificação de requisitos contribuem para projetos ágeis? Nesta questão, tratamos dois aspectos. O primeiro refere-se a determinar quando as práticas formais podem ser integradas em um estágio ou em um momento específico do processo de desenvolvimento, além de descobrir em que contexto eles devem ou podem ser usados. O segundo aspecto concentra-se nos principais benefícios que podem ser gerados pelo uso de MF.

QP2) Quais são as limitações que impedem que os métodos formais se tornem uma prática comum em projetos ágeis? Embora não tenhamos encontrado estudos que avaliem empiricamente a aplicação de EF em projetos ágeis, existem alguns estudos (OLSZEWSKA; WALDÉN, 2015; LIU, 2010; WOLFF, 2012) que apontam algumas limitações, como dificuldades de aprendizagem, complexidades de linguagem e tempo de especificação muito longo. Neste contexto, buscamos descobrir, por meio das opiniões dos profissionais, quais são os desafios que podem surgir ao utilizar EF em projetos ágeis.

Inicialmente, definimos o perfil dos participantes, sendo eles desenvolvedores ágeis atuantes em empresas de desenvolvimento de software. Para alcançar os participantes contactamos diferentes companhias de desenvolvimento de projetos ágeis.

Para identificar o perfil dos participantes, foram criadas algumas questões: a) tempo de experiência em projetos ágeis; b) experiência com métodos para EF; c) papéis assumidos pelos participantes sobre esses projetos. Uma vez que os problemas originados na especificação de requisitos em projetos ágeis são o contexto desta pesquisa, esses parâmetros (a, b, c) foram adotados para que possamos distinguir os perfis ágeis de cada participante.

Na sequência, definimos pesos para cada possibilidade de resposta. Assim, para questão sobre o tempo de experiência em projetos ágeis, atribuímos os seguintes pesos: de 1 até 6 meses, peso 1; mais de 6 até 12 meses, peso 2; mais de 1 até 3 anos, peso 4; e

quando a resposta tinha mais de 3 anos, o peso era 8. Na questão sobre os papéis assumidos pelos participantes nos projetos, atribuímos peso 1 para cada papel assumido. Nós desconsideramos participantes que não estão envolvidos diretamente com desenvolvimento de software, como por exemplo: Participantes que só assumiram papel de SM.

Por fim, os participantes são classificados em uma das seguintes categorias, de acordo com suas pontuações. As pontuações foram definidas de forma quadrática: **Iniciante**, menos de dois pontos; **Pouco experiente**, para pontuação entre 3 e 5; **Experiente**, para pontuação entre 6 e 9; **Muito Experiente**, para aqueles que obtiveram pontuações superiores a 9.

4.2 Operacionalização

O questionário foi dividido em quatro partes. Inicialmente (Parte 1) são apresentados conceitos básicos de abordagens ágeis e métodos para EF de requisitos. Adicionalmente, nós apresentamos um exemplo de EF e EI de um requisito fictício.

Na sequência (Parte 2), questões para a identificação do perfil dos participantes foram feitas, conforme mencionado no tópico "Participantes". Depois disso (Parte 3), três perguntas descritivas foram feitas para conhecer suas percepções sobre como a EF pode contribuir em um ambiente ágil.

Por fim, 12 questões objetivas (Parte 4) foram apresentadas para capturar as possíveis limitações e benefícios de EF em projetos ágeis, onde os participantes transmitiram suas percepções através de 5 escalas: Totalmente Discordo, Discordo, Indiferente, Concordo, Concordo Totalmente. A Tabela 7 apresenta as questões.

O formulário foi disponibilizado online e foi submetido aos participantes da pesquisa através de uma lista de e-mail de empresas conhecidas na região. Para analisar esses dados foi utilizado o método de análise de conteúdo (MORAES, 1999) para observar as respostas das perguntas descritivas (Parte 3). As análises passaram perfeitamente pelos estágios de 1) **preparação** de informações, 2) **unitização** de conteúdos, 3) **classificação** de unidades, 4) **descrição** das categorias formadas e 5) **interpretação** dos resultados. A escala Likert foi utilizada para análise das questões objetivas (Parte 4) observando a medida moda gerada. Quando há heterogeneidade nas respostas dos participantes, a mediana é adotada para identificar a resposta central. Uma comparação é realizada entre as questões objetivas e descritivas com o objetivo de corroborar as respostas de um mesmo participante, enfatizando a percepção do indivíduo.

O instrumento de pesquisa foi revisado por um membro da Empresa A, parceira deste trabalho, e por outros dois pesquisadores com especialidades em estudos empíricos. Assim, problemas e melhorias com relação à inferências, ambiguidades e redundâncias foram mitigadas. Por fim, uma execução piloto foi realizada com seis alunos nos semestres finais do curso de Engenharia de Software da Universidade Federal do Pampa, com o objetivo de validar o entendimento do instrumento.

Tabela 7 – Questionário de perguntas aplicados aos participantes

(1) Questões afirmativas avaliadas pela escala Likert
S1. Especificação formal permite melhor compreender as necessidades do sistema.
S2. Não é garantido eliminar erros de especificação com técnicas de especificação formal
S3. O tempo necessário para utilizar a especificação formal é sempre maior que ao utilizar especificações informais.
S4. A aplicação de métodos de especificação formal sobre abordagens ágeis potencializa a comunicação entre as partes interessadas.
S5. A especificação formal não elimina os erros de especificação, como ambiguidade, falta de completude, etc.
S6. Métodos de especificação formal são altamente recomendados para todos os contextos de produção de software, principalmente para sistemas críticos.
S7. São necessários matemáticos experientes para utilizar os métodos de especificação formal em projetos ágeis.
S8. Métodos formais para especificar requisitos aumentam o tempo total de produção do software.
S9. Todo e qualquer indivíduo pode utilizar métodos de especificação formal.
S10. Para o uso de métodos de especificação formal, não é necessário possuir conhecimentos sobre a tecnologia utilizada para o desenvolvimento do sistema.
S11. Após praticar, a especificação formal torna-se rápida e pode exigir o mesmo tempo que o uso de técnicas informais, como casos de uso, histórias de usuários, etc.
S12. É mais rápido e mais simples escrever especificações com formalismo do que com práticas informais.
(2) Questões Abertas
Q1. Quais são as vantagens e desvantagens que ocorrem ao utilizar especificação formal em projetos ágeis.
Q2. Que desafios você identifica no uso de especificações formais em projetos ágeis.
Q3. Você usaria métodos formais para especificação de requisitos em um projeto conduzido com práticas ágeis.

Fonte: Elaborado pelo autor

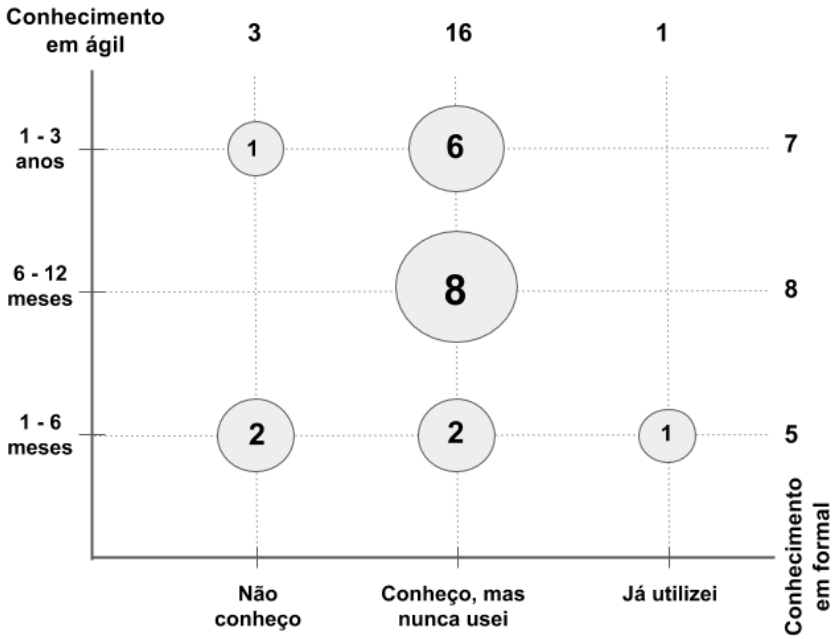
4.3 Análise e Interpretação

No *Survey* obtivemos um total de 20 respondentes de diferentes perfis, como podemos observar na Figura 3.

Podemos observar na Figura 3 que a maioria dos participantes (16) conhecem os conceitos de EF, entretanto eles são inexperientes na prática. Seis participantes tem atuado em projetos ágeis há mais de um ano. Além disso, oito participantes trabalharam em projetos ágeis por mais de seis meses. Observamos que um participante já utilizou EF e também atua em projetos ágeis. A distribuição dos participantes por empresa é apresentada na Figura 4.

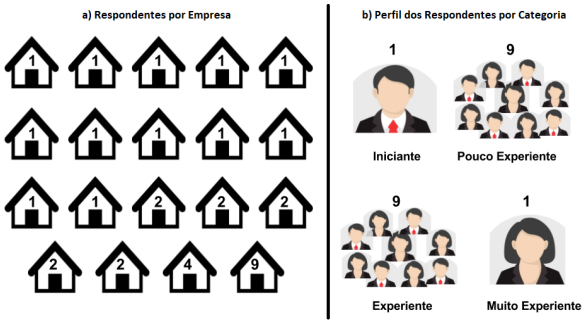
Como podemos observar na Figura 4 (a), os participantes do *Survey* atuaram em diferentes empresas de software, representando 19 empresas. Além disso, após a

Figura 3 – Perfil dos Participantes do Survey



Fonte: Elaborado pelo autor.

Figura 4 – Perfil dos Respondentes Categorizados do Survey



Fonte: Elaborado pelo autor.

categorização dos participantes (b) percebemos que a distribuição deles variam entre **Pouco Experiente** e **Experiente**.

4.3.1 Análise dos Resultados das Questões Abertas

A análise de conteúdo (MORAES, 1999) das questões descritivas iniciou na etapa de **preparação** da informação, onde as respostas do *survey* foram exportadas para a ferramenta automatizada, ATLAS.TI [1] que deu suporte a análise de conteúdo.

No estágio de **unitarização**, um total de 97 unidades foram geradas. Na questão Q1, foram geradas 34 unidades no total, onde 11 unidades representam as desvantagens

[1] <http://atlasti.com/product/what-is-atlas-ti/>

de EF e 23 descrevem as vantagens de EF em ágil. A questão Q2 gerou 18 unidades e, finalmente, na questão Q3 45 unidades foram geradas.

Na etapa de **classificação** 10 categorias foram geradas na questão Q1. Para questão Q2, 5 categorias foram mapeadas, além de 13 categorias para questão Q3. Essas categorias, na fase de **descrição**, foram nomeadas com um rótulo comum que resume suas unidades de conteúdo. As Tabelas 8, 9 e 10 sumarizam o estágio de descrição de análise de conteúdo onde sua frequência (%) dentro de cada categoria é apresentada. Categorias isoladas foram omitidas. Em seguida, é realizada a fase de **interpretação** de análise de conteúdo, discutindo algumas categorias de cada tabela.

Tabela 8 – Respostas da Questão Q1.

ID Categoria	%
Desvantagem	
1. Especificação formal demanda muito tempo	36%
2. Especificação formal é mais complexo que técnicas informais de especificação	28%
3. Especificação formal demanda treinamento	18%
4. Existe uma maior curva de aprendizado para usar a especificação formal	18%
Vantagem	
1. Especificação formal padroniza a escrita de requisitos	32%
2. Especificação formal elimina erros de ambiguidade na especificação	26%
3. Especificação formal é consistente e precisa	26%
4. Não existe erro de interpretação com especificação formal	16%

Fonte: Elaborado pelo autor

Na questão Q1 (Tabela 8), a análise de conteúdo indica que as principais desvantagens quando utilizado EF em projetos ágeis são relacionadas a complexidade e compreensão de MF. Nesse sentido, é possível que essas respostas possam representar a falta de conhecimento de MF pelos respondentes, dado o perfil dos participantes da pesquisa. Por outro lado, essa aplicação pode resultar em benefícios, como a padronização de especificação dos requisitos do software, levando a redução de erros de especificação.

Tabela 9 – Respostas da Questão Q2.

ID Categoria	%
1. Falta de conhecimento de especificação formal por parte dos *stakeholders*	38%
2. Desafios na quebra do tabu sobre o uso de métodos para especificação formal	22%
3. Tempo elevado para o aprendizado e compreensão dos métodos formais	16%
4. Para utilizar especificação formal é preciso aplicar estudos prévios	7,14%

Fonte: Elaborado pelo autor

Analisando a questão Q2 (Tabela 9) percebemos que os desafios são relacionados a falta de conhecimento sobre formalismo por parte dos desenvolvedores. Assim, o préconceito dessa pratica é um fator limitador para disseminação de MF para especificação.

Tabela 10 – Respostas da Questão Q3.

ID Categoria	%
1. Adotaria especificação formal parcialmente	16%
2. Somente aplicaria especificação formal em requisitos criticos	11%
3. Adoção de métodos para especificação formal demandaria estudos e preparação	11%
4. Adoção depende do conhecimento do time	9%
5. Adotaria exclusivamente durante a fase de especificação de requisitos	9%
6. Especificação informal inicialmente, posteriormente complementaria com especificação formal	7%
7. Adotaria especificação formal totalmente	7%
8. Não tenho conhecimento para opinar	7%
9. Adoção depende do time-box do projeto	7%

Fonte: Elaborado pelo autor

Dividimos a intenção de uso do formalismo da Tabela 10, através de análise tácita de sentimento (LIU, 2012), em três sentimentos: a) Não usaria, b) Depende, c) Usaria.

Como resultado, observamos que 23,1% refletiram o sentimento de que não utilizaria especificação formal. 29,4% dos sujeitos apresentaram o sentimento de que utilizariam formalismo. Contudo, a maioria (47,5%) indicaram o sentimento de que utilizariam o formalismo dependendo do contexto. Devemos notar que os participantes relataram que o uso da EF depende do problema à ser resolvido, o que está relacionado a percepção de que qualquer método de desenvolvimento deve ser usado de acordo com a necessidade do problema.

Adicionalmente, nós encontramos que dos 29,4% que disseram que usariam EF, 75% correspondem ao grupo de participantes da categoria **Pouco Experiente**. Enquanto, dos 23,1% que não utilizariam, 81% são participantes do grupo dos **Iniciante**.

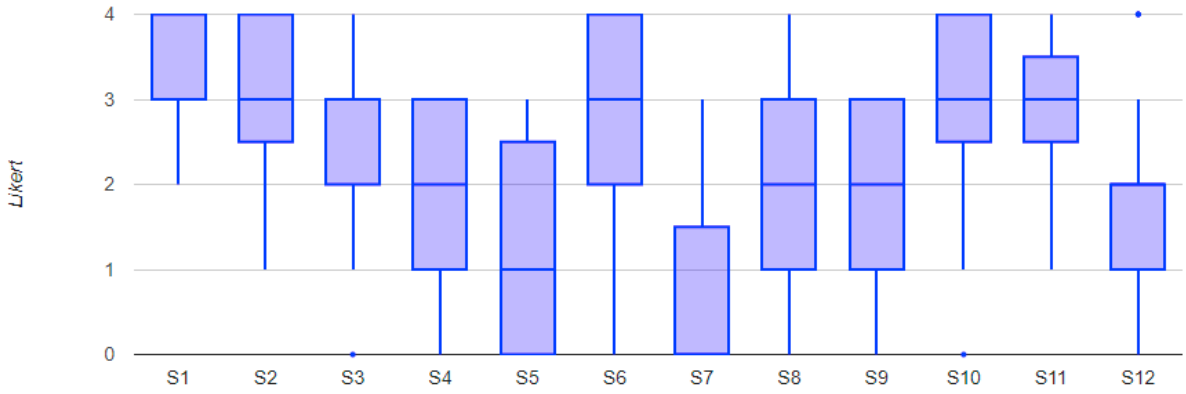
4.3.2 Análise dos Resultados das Questões Fechadas

Na Figura 5 as respostas sobre as 12 perguntas afirmativas são apresentadas e discutidas.

A análise das respostas revelou afirmações positivas de aplicação de formalismo em projetos ágeis, onde os participantes tendem a concordar com S1, S6 e S11, onde a mediana está em **Concordo** e a variação (IQR) é mais indiferente. A mediana indica tendência em **Indiferente** relacionada às afirmações S4 e S12, mostrando que os entrevistados não estavam preocupados com possíveis pontos negativos na aplicação de EF em ambientes ágeis.

As declarações negativas da aplicação de EF em projetos ágeis (S2, S5) revelaram que os participantes estão dispostos a concordar com a declaração S2. Na declaração S5, é possível observar a tendência ao **Discordo**, procurando a mediana e isso corrobora as informações na Tabela 8 nas categorias 4, 5 e 6, onde é descrito que MF elimina erros de

Figura 5 – Respostas para cada questão afirmativa do Survey.



Fonte: Elaborado pelo autor.

especificação. A declaração S10 mostrou uma dispersão de opinião limitada e a mediana está em **Concordo**, então a tendência dos participantes é concordar com a afirmação.

Nas declarações S7 e S9, que estão relacionadas a necessidade de um profissional especializado em MF, foi apontado que os participantes **Discordam Totalmente** com a afirmação S7, que pode ser vista pela mediana. No entanto, a declaração S9 revelou a mediana em **Indiferente**. Com isso, pode-se notar que os participantes concordam com o fato de que eles não precisam de profissionais experientes para usar MF, mas são indiferentes ao responder que qualquer indivíduo pode usá-lo.

Finalmente, na declaração S3 é apontado que existe uma tendência para concordar com a afirmação, com base na mediana que está em **Concordo**. As respostas da afirmação S8 não apresentam tendência e a mediana está em **Indiferente**. Essas duas tendências corroboram com a informação na categoria 1 da Tabela 8, onde é indicado que um dos desafios é o tempo necessário para a aplicação.

Observamos uma certa tendência nas respostas com base no perfil dos participantes. Os participantes da categoria **Experiente** tendiam a concordar com as declarações S1, S2, S5 e S7. Os participantes do grupo de **Pouca Experiência** tenderam a concordar com as afirmações S4 e S11 e discordar com S8. Como temos apenas um participante das categorias **Iniciante** e **Muito Experiente**, não é necessário analisar as tendências dessas categorias.

4.3.3 Análise Geral dos Resultados

Discutimos os principais resultados que observamos nesse estudo, respondendo as duas questão norteadoras do *Survey*.

QP1: De que forma os métodos formais para especificação de requisitos contribuem para projetos ágeis?

Os resultados da pesquisa sugerem que MF pode contribuir principalmente para eliminar problemas causados por práticas informais, como erros de interpretação. Isso está relacionado a uma sintaxe formal e semântica bem definida, o que facilita a compreensão e padroniza a escrita de especificações. No entanto, a complexidade da aprendizagem de EF pode ser uma restrição, e se demorar muito para adaptar a equipe, pode ser muito caro para o projeto. Algumas alternativas podem ser adotadas, tais como, adotando o formalismo apenas para requisitos cruciais ou após a validação da EI.

QP2: Quais são as limitações que impedem que os métodos formais se tornem uma prática comum em projetos ágeis?

Observamos que os métodos para EF de requisito podem contribuir claramente para projetos ágeis, mas precisam abordar um fator determinante: aceitação das pessoas envolvidas. Mudança implica em sair da sua zona de conforto e isso sempre será um desafio. Por exemplo, como convencer um analista que utiliza História de Usuário a utilizar MF. No entanto, o benefício desses métodos não excederá o custo de tal mudança? Assim, outras investigações precisam ser conduzidas para tentar responder a esta pergunta.

4.4 Ameaças à Validade

Nesta seção, descrevemos o esquema de classificação de ameaça de Wohlin et al. (2012) para a validade da experiência e como trabalhamos para mitigar os mesmos.

A principal ameaça à **validade à conclusão** do *Survey* refere-se ao número de respondentes. Nós obtemos apenas 20 respostas e isso inviabiliza a generalização dos resultados apresentados. Entretanto, podemos considerar esses indícios como resultados promissores, uma vez que o *Survey* foi realizado com profissionais que atuam no desenvolvimento de software com adoção de abordagens ágeis.

Instrumentos incorretos e medidas inadequadas podem causar diferenças de resultados, tornando assim uma ameaça à **validade interna**. Assim, os instrumentos foram elaborados e uma execução piloto foi realizada, observando quais estratégias de medidas poderiam ser usadas para uma melhor análise.

Para mitigar as ameaças à **validade externa** buscamos garantir que o grupo de participantes representa a comunidade em foco. Assim, nós estudamos o perfil de participantes através da categorização do perfil, garantindo que todas as pessoas envolvidas tenham conhecimento e experiência em contexto ágil.

Os sujeitos não sabiam quais eram as questões da pesquisa, por isso não podiam adivinhar quais eram os resultados esperados. Das respostas obtidas, percebemos que alguns pontos (como alguns benefícios, por exemplo) contrariam a literatura, e isso indica que essa ameaça à **validade da construção** foi mitigada, mas não eliminada totalmente.

4.5 Lições do Capítulo

No decorrer deste capítulo apresentamos a primeira estratégia empírica para levantar indícios da aplicabilidade de EF em projetos ágeis. Nesse sentido, buscamos observar a opinião de profissionais de desenvolvimento de software em empresas que adotam abordagens ágeis em seu ambiente.

O protocolo do *Survey* e sua execução foram apresentados detalhadamente, possibilitando observar como foi tratados cada aspecto de um *Survey*, além de viabilizar a replicação do mesmo. Mesmo que tenhamos obtidos um número relativamente baixo de respondentes, acreditamos que possamos considerar como indícios da utilização de métodos de EF.

Tendo esses indícios em mente, podemos considerar justificável maiores estudos que reforcem os indícios e colaborem com resultados mais robustos e significantes para analisar a aplicabilidade de MF para especificação de requisitos em projetos ágeis.

5 ESTUDO DE CASO INCORPORADO EXPLORATÓRIO

Este capítulo tem por propósito descrever a execução de um estudo de caso incorporado (YIN, 2013) de caráter exploratório (WOHLIN et al., 2012). A estruturação do capítulo segue: seção 5.1 apresentamos as definições do protocolo do estudo; seção 5.2 discutimos como foram a preparação e execução do estudo em ambas unidades de análise (duas empresas); na seção 5.3, seção 5.4 e seção 5.5, respectivamente, realizamos a análise dos resultados de ambas unidades de análise, apresentamos as ameaças à validade e, por fim, apresentamos nossas lições aprendidas.

5.1 Planejamento

Após o primeiro estudo exploratório (ver Capítulo 4), foi possível observar indícios de que os profissionais da indústria acreditam que a EF pode contribuir para projetos ágeis, mas a inexperiência prática de formalismo torna inviável generalizar os resultados. Nesta estratégia subsequente, realizamos um estudo de caso incorporado exploratório com engenheiros de software em duas empresas distintas utilizando os trabalhos de Wohlin et al. (2012) e Yin (2013) como guias.

A execução do estudo de caso passou por quatro etapas principais etapas:

Etapa 1: Entrevista denominada pré-conhecimento. Nessa entrevista os participantes relatam suas percepções sobre os métodos de EF como técnica de especificação de requisitos em projetos ágeis.

Etapa 2: Treinamento sobre o método de EF Z (BARDEN; AL., 1995).

Etapa 3: Participantes especificaram formalmente requisitos apresentados informalmente (requisitos providos pela sua empresa).

Etapa 4: Entrevista de pós-conhecimento, com os mesmos tópicos que a primeira foi realizada para analisar a evolução das opiniões dos participantes após a experiência.

Na **etapa 2**, especificações de um projeto fictício foram usados para o treinamento, com regras de negócio que podem ser modeladas usando a notação Z. Essas especificações foram modeladas previamente pelo pesquisador e durante o treinamento as especificações modeladas foram apresentadas para os entrevistados (conjuntamente com a EI correspondente). Neste contexto, o objetivo do treinamento foi apresentar minimamente os conceitos da notação Z e demonstrar seu uso, considerando uma EF pronta.

Como parte do estudo, pedimos para cada empresa que fornecesse especificações de um conjunto de projetos reais que estavam sendo desenvolvidos. Essas especificações estavam no formato usado pela empresa. Não discutiremos estas informações em detalhes devido que, nesse momento, não é o objetivo primário dessa pesquisa.

5.2 Operacionalização

Para esse estudo duas unidades de análise, sendo a Empresa B e Empresa C. A **primeira unidade de análise (Empresa B)** disponibilizou um de seus funcionários à participar da experiência. Esse sujeito é um engenheiro de software com mais de 8 anos de experiência como desenvolvedor de projetos de software e a mais de 4 anos atua na empresa em questão. Além disso, ele participou de pelo menos 3 projetos ágeis.

A especificação disponibilizada pela Empresa B refere-se a um projeto de gerenciamento de negócios. As especificações mínimas para o Controle de Planos de Contas foram disponibilizadas para este estudo pelo gerente do projeto na empresa. Essas especificações foram selecionadas com base em desenvolvimentos futuros do projeto, portanto não sendo requisitos já desenvolvidos, em desenvolvimento ou conhecidos pelo sujeito.

O desenvolvedor participante do estudo representando a **segunda unidade de análise (Empresa C)** também é um engenheiro de software. Possui 9 anos de experiência em desenvolvimento de software, atuando em projetos ágeis nos últimos 6 anos. Nos últimos 2 anos tem atuado diretamente com práticas ágeis de desenvolvimento, como XP, além de atuar como SM em projetos de software.

O projeto da Empresa C refere-se a um sistema de gestão de Parques Científicos e Tecnológicos de Incubadoras. Esse sistema proporciona, para qualquer usuário empreendedor que deseje incubar-se no parque, elaborar seu plano de negócio, receber assistência através da pré-avaliação dos gerentes da Incubadora para melhorar e reenviar seu plano para novas pré-avaliações. Como na Empresa B, os requisitos foram fornecidos pelo gerente do projeto.

Para a execução dos estudo de caso escolhemos o método de EF Z para especificação de requisitos por facilitar a descrição do sistema em diferentes níveis de abstração e permitiu uma descrição completa da especificação (BARDEN; AL., 1995). A *Microsoft Word* foi configurada e usada com o plugin *Z Tools* para criar modelos Z pelos participantes.

Como planejado na **Etapa 1** todas questões foram abordadas. Para a execução da entrevista, nós apresentamos brevemente os conceitos de abordagens ágeis (mesmo sabendo que os participantes tenham participação ativa em projetos ágeis) e EF. Entretanto, os participantes expressaram que não tinham conhecimento prático sobre MF, fortalecendo a necessidade da próxima etapa (treinamento).

O treinamento (**Etapa 2**) foi realizado com a apresentação da especificação Z. Inicialmente, os participantes foram convidados a descrever o seu entendimento sobre a modelagem, a fim de identificar o seu nível de conhecimento. Ainda, foram discutidos o que cada símbolo da modelagem formal significava, relacionando com as regras comerciais descritas em texto informal (disponibilizado pela sua empresa).

Na sequência (**Etapa 3**), solicitamos aos participantes que modelassem com Z as especificações informais disponíveis pela empresa. Esperava-se que houvesse dificuldades

nesta atividade, no entanto, os participantes mostraram interesse e engajamento para completar o modelo de especificação.

Deve-se enfatizar que o objetivo dessa experiência foi aumentar a percepção de aplicação de MF como ferramenta para a especificação de requisitos. Portanto, a qualidade da especificação ou a produtividade desempenhada estão fora de questão. Finalmente, realizamos a entrevista de pós-conhecimento (**Etapa 4**).

Para **coleção dos dados** para o estudo, desenvolvemos duas entrevistas para coletar as percepções dos participantes, uma para o pré-conhecimento e outra para o pós-conhecimento.

As entrevistas foram não estruturadas e focadas (MARCONI; LAKATOS, 2003). Além disso, as entrevistas foram armazenadas através de gravação de áudio, posteriormente transcritas, compiladas e analisadas cuidadosamente. Todas as perguntas foram abertas para coletar dados para análise de conteúdo qualitativo (MORAES, 1999).

A entrevista de pré-conhecimento é realizada no início das atividades (**Etapa 1**) para identificar a percepção dos problemas que os participantes observam sobre as práticas atuais de sua empresa na etapa de especificação de requisitos e como as EF podem contribuir para esse contexto. A pós-conhecimento é executada na **Etapa 4**, onde as percepções são identificadas após a experiência de especificar requisitos usando uma técnica de EF.

5.3 Análise e Interpretação

Nesta seção apresentamos a análise dos dados gerados pelas duas unidades de análise, realizando uma discussão individual e, posteriormente, comparativa entre elas. Vale ressaltar que a execução nas duas empresas foi realizada separadamente, em momentos distintos.

5.3.1 Primeira Unidade de Análise (Empresa B)

A etapa de **unitarização** da análise de conteúdo (MORAES, 1999) na entrevista de pré-conhecimento da Empresa B gerou 18 unidades ao total, depois na entrevista de pós-conhecimento, 27 unidades foram geradas. Além disso, na fase de **categorização**, 13 categorias foram geradas de pré-conhecimento e 14 categorias de pós-conhecimento.

As Tabela 11 e Tabela 12 apresentam as categorias geradas de acordo com o estágio da entrevista (**Etapa 1** ou **Etapa 4**) e sua frequência (%). As unidades em categorias individuais foram omitidas.

A análise das categorias de pré-conhecimento (Tabela 11) revelam que os principais problemas da Empresa B são relacionados a falta de documentação nos seus projetos ágeis, possivelmente decorrente da interpretação da empresa sobre as abordagens ágeis. Nesse sentido, a inclusão de um MF como uma técnica alternativa de especificação pode

Tabela 11 – Entrevista de Pré-conhecimento – Categorias com sua frequência (Empresa B)

ID Categoria	%
1. Falta de documentação é um dos problemas em projetos ágeis	14%
2. Depois de algumas experiências com especificação formal, o tempo necessário para especificar vai reduzindo	14%
3. É necessário menos refatoração no software quando utilizado formalismo	7%
4. A especificação formal é interessante e pode ajudar no ambiente de desenvolvimento ágil	7%
5. É apenas uma questão de troca de técnica de especificação, sai informal entra formal	7%
6. Não há ambiguidade ao utilizar especificações formais	7%
7. O comportamento da equipe mudaria ao mudar o método	7%

Fonte: Elaborado pelo autor

contribuir para remediar esse problema.

Tabela 12 – Entrevista de Pós-conhecimento – Categorias com sua frequência (Empresa B)

ID Categoria	%
1. Utilizar especificação formal exige menos tempo que eu pensei, então, com a prática, o tempo demandado vai se reduzindo	15%
2. Eu pensei que seria mais complexo, mas não é complexo e difícil de entender	11%
3. Métodos de especificação formal ajudam a entender a especificação	11%
4. Utilizar especificação formal é mais simples, exige menos trabalho e mitiga a subjetividade do escritor e leitor	11%
5. É mais rápido escrever requisitos com formalismo do que com técnicas informais	7%
6. Gostei, é uma boa ferramenta de especificação	7%
7. A especificação formal é melhor do que o caso de uso	7%
8. A especificação informal é ambígua, torna a compreensão e a comunicação difíceis	7%

Fonte: Elaborado pelo autor

As respostas geradas na categoria de pós-conhecimento (Tabela 12) indicam uma evolução de percepção do sujeito. Essa evolução sugere que a falta de conhecimento sobre a especificação formal resulta em um pré-conceito sobre o uso desses métodos.

Acredita-se que esses métodos, após o custo inicial de aprendizagem, podem trazer benefícios significativos para projetos ágeis a longo prazo. Assim, essas informações indicam que esses métodos podem ser uma boa ferramenta para ajudar no desenvolvimento de sistemas de software.

5.3.2 Segunda Unidade de Análise (Empresa C)

A etapa de **unitarização** das respostas da entrevista de pré-conhecimento da Empresa C geraram 33 unidades de conteúdo e no pós-conhecimento foram geradas 38

unidades. No estágio de **classificação** as unidades foram agrupadas em similaridades, assim gerando 12 categorias de pré-conhecimento e 17 de pós-conhecimento. As Tabelas 13 e 14 apresentam as categorias e sua frequência (%) geradas na entrevista do sujeito da Empresa C. As categorias isoladas foram omitidas.

Tabela 13 – Entrevista de Pré-conhecimento – Categorias com sua frequência (Empresa C)

ID Categoria	%
1. Especificação formal demanda alto tempo para especificação	21%
2. O processo e o time precisam se adaptar para utilizar especificação formal	18%
3. Os problemas de especificação informal são pela falta de entendimento	12%
4. Identificar quais requisitos precisam ser especificados formalmente	12%
5. Métodos de especificação formal ajudam a entender melhor a especificação	9%
6. Métodos de especificação formal diminuem os erros de especificação de requisitos	9%

Fonte: Elaborado pelo autor

Como podemos observar na Tabela 13, as dificuldades mencionadas pela Empresa C são similares as mencionadas pela Empresa B, como problemas de subjetividade no entendimento do requisito, por exemplo. Esse tipo de problema pode ser mitigado através das EF de requisitos, entretanto, pode haver o medo da exigência de um alto custo de tempo para adaptar a equipe. Uma sugestão para adotar esses métodos é utilizá-los somente quando for necessário entender melhor o requisito, assim, por exemplo, identifica-se um determinado requisito crucial para o negócio e o modela formalmente.

Tabela 14 – Entrevista de Pós-conhecimento – Categorias com sua frequência (Empresa C)

ID Categoria	%
1. Depois de algumas experiências com especificação formal, o tempo necessário para especificar vai reduzindo, se tornando rápida	16%
2. O formalismo pode contribuir para a etapa de teste do software	11%
3. O formalismo não é difícil de entender	11%
4. A especificação formal pode ser uma ferramenta complementar para desenvolvimento ágil de software	8%
5. A leitura do requisito especificado formalmente é mais rápida do que o requisito especificado informalmente	5%
6. A falta de utilização de especificação formal pode ser devido à falta de conhecimento dos desenvolvedores	5%
7. Ágil é sobre o comportamento, então formal não é contra ágil	5%
8. Algum conhecimento de código é necessário para utilizar a especificação formal	5%
9. Identificar quais requisitos são necessários obter um melhor entendimento e especificá-los formalmente	5%

Fonte: Elaborado pelo autor

Analisando as categorias geradas na entrevista de pós-conhecimento (Tabela 14) nós podemos observar que a experiência prática de EF fez com que evoluíssem as opiniões do sujeito. Nós percebemos que houve uma mudança na opinião sobre a complexidade de entendimento dos MF, onde diferente da categoria 1 da Tabela 13, o tempo necessário para especificação é menor do que imaginado.

Finalmente, de acordo com a interpretação ágil para a Empresa C as características das abordagens ágeis promovem o uso de práticas para alcançar o sucesso no projeto. Assim, MF podem ser usados para obter especificações mais precisas e coesivas.

5.3.3 Análise Geral dos Resultados

Alguns pontos identificados nessa segunda estratégia de avaliação empírica revelaram fortes percepções, principalmente nas intersecções dos resultados. Ambas as empresas acreditam que o tempo necessário para especificar formalmente seus requisitos diminui quando os trabalhadores têm mais conhecimento sobre a técnica. Assim, sugere-se que ao utilizar continuamente EF o tempo necessário para especificação pode se igualar ao tempo demandado para especificar informalmente requisitos.

Além disso, após o processo de especificação, as percepções dos sujeitos indicaram que o formalismo não é tão difícil de entender e, especialmente, simplifica o processo de escrita de especificações.

Em relação a inclusão de métodos de EF em projetos ágeis, ambas as empresas também acreditam que deve ser identificado para quais requisitos é preciso ser garantido uma maior coesão e precisão para, assim, especificá-los formalmente. Essa prática está diretamente relacionada as abordagens ágeis, uma vez que um de seus princípios é apenas fazer o que contribuirá para alcançar o objetivo final: software em funcionamento.

Finalmente, ambas as empresas acreditam que a prática de EF pode contribuir para o processo de especificação de requisitos, sendo a EF uma boa ferramenta complementar para os projetos ágeis. No entanto, a barreira do pré-conceito sobre essa prática precisa ser combatida, para assim poder ser disseminado a prática de EF de requisitos de software em ambientes ágeis.

5.4 Ameaças à Validade

Uma possível ameaça para **validade à conclusão** é a utilização única da estratégia de análise de conteúdo. Com isso, nossa visão sobre os resultados é limitada. Entretanto, por termos um tempo curto de execução do estudo acreditamos que a estratégia adotada foi a correta, por proporcionar um processo sistemático de análise de informações geradas através de entrevistas.

Por fim, poderíamos ter ameaças relacionadas a efeito da expectativa do experimentador – o pesquisador interage intensamente com os participantes, as crenças dele

causam um efeito no sujeito (WOHLIN et al., 2012) – durante as entrevistas, visto que as entrevistas foram não estruturadas, seguindo uma linha de raciocino por tópicos. Assim, nós optamos por remover dos resultados os momentos em que o pesquisador discutia suas percepções com os sujeitos apresentando sua opinião sobre a prática.

Por termos atingidos poucos sujeitos no estudo é inviável generalizar os resultados obtidos. Entretanto, podemos corroborar os indícios achados na primeira estratégia empírica aplicada nessa pesquisa (ver Capítulo 4). Assim, entendemos que ao garantir que os sujeitos do estudo são uma amostra real da população estudada (profissionais experientes em projetos ágeis) podemos obter uma maior confiança nos resultados.

Para mitigar problemas relacionados ao tempo de execução – impor ou suprimir restrições de tempo – nós garantimos que os participantes conseguiriam modelar os requisitos solicitados, definindo um escopo pequeno para cada experimento. Nesse sentido, o intuito do estudo foi proporcionar um conhecimento mínimo sobre EF, mais especificamente a notação Z, e coletar suas percepções, independentemente da especificação gerada.

Para esse estudo, uma possível ameaça à **validade à construção** seria os participantes acharem que estavam sendo avaliados e que os resultados poderiam impactar no seu trabalho na empresa. Para mitigar essa ameaça, explicamos aos participantes que estávamos observando apenas a percepção dele quanto as práticas formais, não questões comportamentais ou e capacidades e habilidades.

5.5 Lições do Capítulo

Ao longo deste capítulo apresentamos como segunda estratégia empírica adotada na pesquisa foi conduzida e os resultados gerados por ela. Nota-se que esse segundo estudo obteve uma perspectiva diferente da primeira (Capítulo 4), pois mesmo sendo uma análise de percepção dos participantes, trouxe as opiniões após uma experiência prática do MF Z para especificação de requisitos.

Reforçando os indícios do *Survey*, percebemos que há fortes indicativos de que os métodos de EF podem contribuir para ambientes ágeis, pelo menos na perspectiva dos desenvolvedores ágeis de software. Por exemplo, antes da experiência prática os sujeitos tenderam à não acreditar que os métodos de EF podiam contribuir para esses ambientes, acreditando que o fato da equipe precisar mudar seu comportamento seria um limitador. Contudo, após utilizar a notação Z, os sujeitos evoluíram suas percepções. Por exemplo, os sujeitos indicaram que a adoção de MF não é tão complexo e demorado como pensavam, sendo apenas uma questão de modificação da técnica de especificação.

Essa evolução evidencia o pré-conceito que envolvem a possibilidade em utilizar EF em projetos ágeis. Além disso, outros benefícios foram relatados, como a rapidez em escrever e ler as especificações, bem como a melhora na compreensão, além da contribuição na etapa de teste do desenvolvimento.

Finalmente, entendemos que ainda há limitações dos resultados desse segundo estudo, que inviabiliza generalizá-los. Assim, nota-se a importância da aplicação de uma experiência controlada para analisar a aplicabilidade desses métodos em um contexto ágil de desenvolvimento de software. Subsequentemente, apresentamos tal experiência através de um experimento controlado (Capítulo 6).

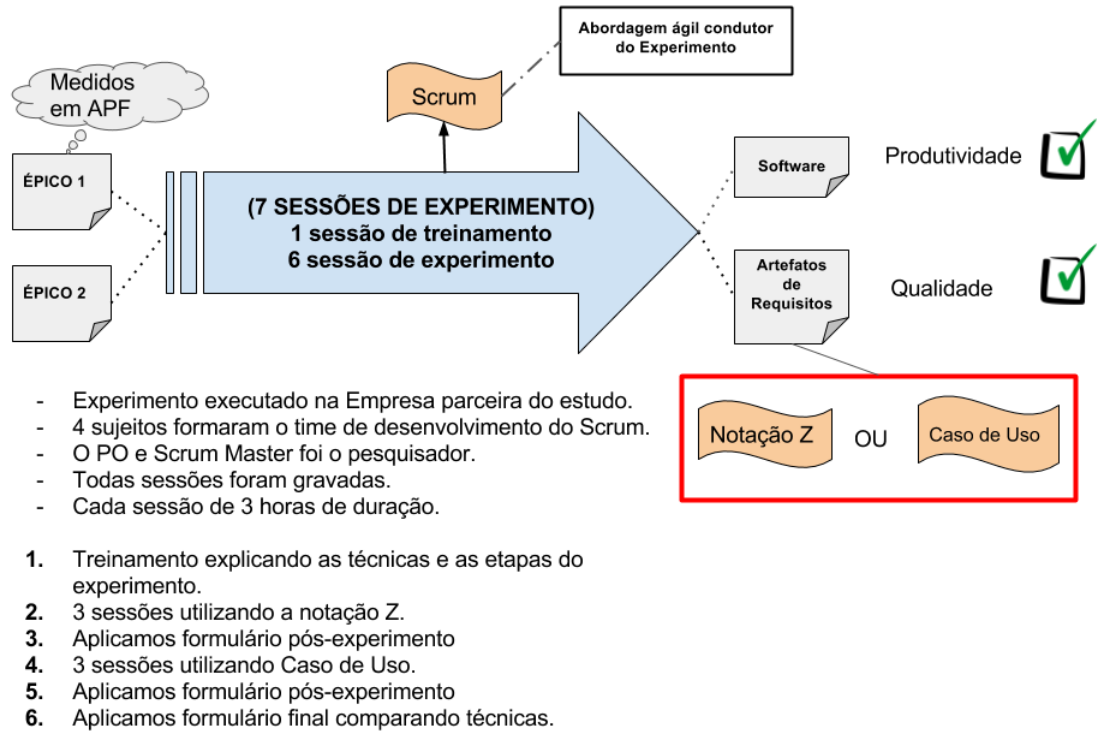
6 ESTUDO EXPERIMENTAL

O propósito deste capítulo é descrever o planejamento do experimento, bem como suas definições, objetivos e motivações. Para elaboração do experimento utilizamos o guia do Wohlin et al. (2012). O capítulo está organizado da seguinte forma: seção 6.1 é apresentado o planejamento detalhado do experimento; a seção 6.2 descreve a execução do experimento; seção 6.3 são analisados e interpretados os resultados do experimento; por fim, na seção 6.5 apresentamos as lições finais do capítulo.

6.1 Planejamento

Nesse experimento buscamos analisar as técnicas de especificação de requisitos Z e Casos de Uso descritivo, com objetivo de avaliar, com relação a qualidade das especificações e produtividade de um time ágil, na perspectiva do pesquisador, no contexto de profissionais engenheiros de software da Empresa A especificando requisitos no *framework* Scrum.

Figura 6 – Visão Geral do Experimento



Fonte: Elaborado pelo autor.

Podemos observar na Figura 6 uma síntese da estrutura definida para nosso experimento. Adotamos o *framework* Scrum como portadora do desenvolvimento do experimento, onde o ambiente respeitou todos seus eventos, papéis e restrições.

Como entrada para execução das sessões os participantes recebem especificação em Épicos [1], descritos textualmente. Como saída, os sujeitos entregam artefatos de software e os requisitos escritos com cada uma das técnicas.

As duas **variáveis independentes** do experimento foram os métodos Z e Caso de Uso descritivo. A única interferência no modo de desenvolver software no experimento é na etapa de especificação de requisitos, onde ora os participantes utilizaram a técnica de EF Z, ora utilizaram Caso de Uso para EI. Duas **variáveis dependentes** foram a produtividade do TD e a qualidade das especificações geradas (em relação ao atributo completude).

Os participantes formaram apenas um TS e utilizaram os métodos de forma pareada, utilizando inicialmente o método Z, na sequência Caso de Uso. O planejamento experimental abordou os seguintes princípios gerais:

- **Randomização:** todos participantes executaram os métodos Z e Caso de Uso, a sequência de execução foi definida de forma aleatória.
- **Bloqueio:** os participantes formaram apenas um grupo, formando um TD Scrum. Para completar, o pesquisador assumiu os papéis de PO e SM. Dentro do TD haviam participantes de diferentes níveis de experiência. Essas experiências foram levantadas a partir de um formulário antecedendo o experimento. Todos participantes possuem, pelo menos, mais de dois anos com experiência em ágil.
- **Equilíbrio:** cada técnica teve o mesmo número de participantes utilizando-a. Ainda, cada uma foi utilizada em três sessões de três horas cada.
- **Tipo de *Design*:** Baseado nas categorias apresentadas por Wohlin et al. (2012), Um Fator com Dois Tratamentos representam nosso tipo de design. O fator de nosso experimento é a especificação de requisitos e os dois tratamentos são Z e Caso de Uso.

Para seleção do contexto do experimento decidimos as quatro dimensões apresentadas a seguir:

- **Processo**: usamos uma abordagem *in-vivo*, visto que o experimento foi conduzido em ambiente empresarial, controlado, sendo categorizado como *on-line*;
- **Participantes**: profissionais Engenheiros de Software funcionários da Empresa A de desenvolvimento de software;
- **Realidade**: utilizamos um problema real na experiência, onde foi utilizado o método de EF Z e método para EI Caso de Uso descritivo para especificação de requisitos;

[1] Épico é uma descrição geral do que se deseja do software.

- **Generalidade**: a análise foi realizada em objetos específicos, sendo a utilização de formalismo através do método Z no *framework* Scrum.

A partir disso foi possível formular nossas hipóteses informais. Na primeira percepção, entendemos que ao utilizar o método Z para EF em um ambiente ágil é alcançada uma maior completude nos requisitos de software do que com Caso de Uso. Assim, é possível obter uma melhor qualidade nas especificações. Com base nessa hipótese informal definimos a nossa primeira questão de pesquisa **QP1** com suas hipóteses formais, utilizando as seguintes notações:

Φ_z: representa a qualidade em utilizar Z;

Φ_{uc}: representa a qualidade em utilizar Caso de Uso;

QP1. Qual método de especificação, Z ou Caso de Uso, possibilita gerar especificações com melhor qualidade, através do atributo completude?

- H_0: $\Phi_z = \Phi_{uc}$: A qualidade das especificações é a mesma quando se utiliza Z ou Caso de Uso.

- H_1: $\Phi_z > \Phi_{uc}$: A qualidade das especificações é maior quando se utiliza Z do que Caso de Uso.

- H_2: $\Phi_z < \Phi_{uc}$: A qualidade das especificações é menor quando se utiliza Z do que Caso de Uso.

Dividimos a forma de coletar e medir os dados do experimento em duas perspectiva: Subjetiva e Objetiva. Para análise subjetiva, três questionários foram utilizados, contendo questões relacionadas a qualidade das especificações. Nesses instrumentos, questões placebo foram adicionadas. Para questão de pesquisa **QP1**, derivamos três perguntas nos dois primeiros questionários. A percepção de cada sujeito era representada pela escala Likert (LIKERT, 1932).

As perguntas dos dois primeiros questionários ("Formulário para Pós-Experimento do uso de EF" e "Formulário para Pós-Experimento do uso de EI") coletaram as opiniões dos sujeitos quanto as práticas desempenhadas, sendo o primeiro formulário sendo tendencioso para a EF como sendo melhor que EI e o segundo afirmando que as práticas de EI são melhores que EF. As questões definidas em cada questionário foram construídas propositalmente de forma tendenciosa.

Derivamos as questões $Q1_{qf}$ [2], $Q2_{qf}$ e $Q3_{qf}$, representando a variável dependente **qualidade**:

- $Q1_{qf}$ Métodos de especificação formal garantem que a especificação esta livre de erros de especificação, diferente das práticas informais.

[2] Q_{qf} refere-se a questão ressaltando as EF no aspecto qualidade.

- $Q2_{qf}$ Se comparados uma técnica de especificação formal com informal, em quesito qualidade da especificação, a especificação formal garante resultados melhores do que informal.

- $Q3_{qf}$ Com especificação formal eu consigo desenvolver artefatos de requisitos com maior qualidade, impactando diretamente no software produzido, diferente das técnicas informais.

No segundo formulário foram utilizadas as questões antônimos $Q1_{qi}$ [3], $Q2_{qi}$ e $Q3_{qi}$ para variável dependente **qualidade**:

- $Q1_{qi}$ Técnicas informais de especificação garantem que a especificação esta livre de erros de especificação, diferente das práticas formais.

- $Q2_{qi}$ Se comparado uma técnica de especificação formal e informal, em quesito qualidade da especificação, a especificação informal garante resultados melhores do que formal.

- $Q3_{qi}$ Com especificação informal eu consigo desenvolver artefatos de requisitos de maior qualidade, impactando diretamente no software produzido, diferente das técnicas formais.

Após a ordenação dos valores das percepções dos sujeitos através da escala Likert, identificamos a Mediana de percepção do TD para cada questão.

Após identificar a Mediana de cada questão, comparamos seus antônimos. Por exemplo, comparamos a Mediana do TD para $Q1_qf$ que afirma que EF é melhor pra qualidade e observamos qual foi o resultado para $Q1_qi$ que afirma que contrário. Assim, analisamos a coerência de respostas do TD para cada questão.

Com essa análise da coerência, identificamos quais dados corroboram para identificar se Z ou Caso de Uso foi melhor na percepção do time. Além disso, também identificamos os dados **inconclusivos**, onde ambas perceptivas apontaram para o mesmo lado (por exemplo: quando a Mediana de uma questão e seu antônimo foram iguais).

Para a análise **objetiva** dos artefatos gerados utilizamos a avaliação de qualidade proposta por Rocha (2006). O propósito dessa medida é *avaliar* o aspecto *qualidade* das especificações em relação ao atributo *completude* no ponto de vista do *pesquisador*.

Para aplicar a medida, buscamos identificar se todos os elementos presentes nos requisitos dos Épicos estão presentes nos modelos correspondentes. Para isso, aplicamos a seguinte métrica:

Métrica M1 = $\frac{F_1+R_1}{F+R}$, onde: F = total de funções encontradas no requisito; R = total de requisitos regras de negócio (R) presentes no requisito; F_1 = total de fun-

[3] Q_{qi} refere-se a questão ressaltando as EI no aspecto qualidade.

ções encontradas na especificação.; R_1 = total de R encontradas na especificação que correspondem aos do requisito.

Interpretação:
M1 = 1 → tradução completa.
M1 <1 → nem todos os elementos do requisito estão mapeados na especificação.

Além de avaliar a qualidade das especificações geradas, também buscamos analisar a produtividade do TD. Assim, entendemos que é possível alcançar uma maior produtividade no time ágil utilizando o método de EF Z do que com método de EI Caso de Uso.

Com base nessa hipótese informal definimos a nossa segunda questão de pesquisa **QP2** com suas hipóteses formais, utilizando as notações:

μ_z: representa a produtividade em utilizar Z;

μ_{uc}: representa a produtividade em utilizar Caso de Uso;

QP2. Qual método de especificação, Z ou Caso de Uso, possibilita uma maior produtividade para o time ágil?

- H_0: $\mu_z = \mu_{uc}$: A produtividade é a mesma quando se utiliza Z ou de Caso de Uso.
- H_1: $\mu_z > \mu_{uc}$: A produtividade é maior quando se utiliza Z do que Caso de Uso.
- H_2: $\mu_z < \mu_{uc}$: A produtividade é a menor quando se utiliza Z do que Caso de Uso.

Para análise **subjetiva** derivamos as seguintes perguntas aplicadas em ambos formulários. No primeiro formulário, temos as questões $Q1_{pf}$ [4] e $Q2_{pf}$, representando a variável **produtividade**:

- $Q1_{pf}$ Métodos de especificação formal para especificar requisitos diminuem mais o tempo de produção total do software do que com técnicas informais.
- $Q2_{pf}$ É mais rápido e simples escrever especificação com formalismo do que com práticas informais.

Além disso, no segundo formulário derivamos as questões antônimos $Q1_{pf}$ [5] e $Q2_{pf}$, representando a variável **produtividade**:

- $Q1_{pf}$ Métodos de especificação formal para especificar requisitos diminuem mais o tempo de produção total do software do que com técnicas informais.
- $Q2_{pf}$ É mais rápido e simples escrever especificação com formalismo do que com práticas informais.

[4] Q_{pf} refere-se a questão ressaltando as EF no aspecto produtividade.
[5] Q_{pf} refere-se a questão ressaltando as EF no aspecto produtividade.

Da mesma forma que executamos para as questões sobre qualidade, realizamos também para as de produtividade, sendo que para cada questão aplicamos a fórmula que identifica a Mediana de percepção do TD.

Para análise **objetiva** utilizamos a estratégia de Análise de Ponto de Função (APF) para saber quantos Pontos de Função (PF) foram entregues em cada Épico. Para isso, passamos sequencialmente por 5 etapas: 1) contagem dos PF contidos em cada épico; 2) identificação da quantidade de horas efetivas trabalhadas; 3) desenvolvimento de um projeto *client* [6] para consumir os serviços desenvolvidos; 4) contagem de quantos PF foram entregues; 5) aplicação da fórmula para descoberta de quantos PF foram entregues por hora trabalhada:

- $PF_{hora} = \frac{PFtotal\_entregue}{HRtotal\_trabalhada}$
- $HRtotal\_trabalhada = HRtotal\_horas - HRtotal\_disperdicado$

Para encontrar $HRtotal\_trabalhada$ analisamos as gravações realizadas no experimento e removemos da hora total disponível ($HRtotal\_horas$) o tempo ocioso do TD ($HRtotal\_disperdicado$).

Utilizamos alguns **objetos** e configurações para viabilizar a execução do experimento, como tutorias de uso do Z e Caso de Uso, *template* de modelo de Caso de Uso, dois projetos pré-configurados para utilização pelos sujeitos, repositórios remotos para compartilhamento de código e documentos contendo as especificações em Épicos.

Elaboramos um tutorial, utilizando conteúdos da literatura com exemplos fictícios de especificação, para suporte na utilização do método de especificação Z. Adotamos a premissa de que os participantes conheciam os conceitos de Caso de Uso, visto que nos componentes curriculares do curso de Engenharia de Software da Unipampa – Campus Alegrete é abordado esse conteúdo. Mesmo assim, especificamos um modelo padrão para escrita de Caso de Uso (apresentado na Apêndice B).

Configuramos um projeto com componentes de software necessários pra desenvolvimento do experimento. Nesse sentido, os participantes não precisaram se preocupar com conexão com banco de dados, visto que um desses componentes já acoplava essa responsabilidade.

Além dos componentes de software, realizamos a configuração de quatro repositórios remotos para versionamento do software desenvolvido pelos sujeitos, sendo dois para cada Épico. Neles, foi disponibilizado todo material necessário na experiência.

Além dos dois primeiros questionários já apresentados, desenvolvemos um terceiro formulário, aplicado ao final de todo experimento ("Formulário final, comparando ambas técnicas"). Nele, enfatizamos a necessidade dos sujeitos se posicionarem, pois eles precisam informar qual das técnicas proporciona melhores benefícios.

[6] No contexto Cliente/Servidor, um Cliente é um programa que pede um determinado serviço a outro, consumindo seus serviços.

Semelhante a medição dos dois instrumentos iniciais, analisamos as percepções dos sujeitos de forma que, ao final, pudéssemos obter uma porcentagem final, indicando qual técnica fora melhor na perspectiva dos sujeitos em relação a **produtividade** alcançada, bem como a **qualidade** das especificações.

Além das questões placebo incluímos as questões **Q1 e Q2** para **qualidade**:

- **Q1.** Qual das duas práticas você acredita que potencializou uma maior qualidade dos artefatos de requisito, contribuindo para a qualidade do software gerado?
- **Q2.** Quais fatores abaixo você acredita que potencializou a qualidade do software entregue (Dentre as opções há o método Z e o Caso de Uso).

Além disso, também definimos as questões **Q3 e Q4** para **produtividade**:

- **Q3:** Qual das duas práticas você acredita que potencializou a produtividade da equipe?
- **Q4:** Quais fatores abaixo você acredita que potencializou sua produtividade em ambos os escopos do experimento (Dentre as opções há o método Z e o Caso de Uso).

Os participantes do experimento foram quatro profissionais da Empresa A. Antecedendo a execução efetiva do experimento, um treinamento realizado. Durante esse treinamento, são apresentadas as etapas do experimento, além do detalhamento da estrutura de Z e Caso de Uso.

Os dois Épicos do experimento foram definidos juntamente com um *stakeholder* da Empresa A. Ambos Épicos se referem ao desenvolvimento de serviços através de *Web Services*[7].

Épico 1: *O serviço deve permitir a inclusão, edição, remoção e consulta de contas a receber de clientes. Para inclusão de uma nova conta, deve ser informado a data de emissão, número da duplicata, data de vencimento da conta, histórico das compras, valor da duplicata, além da informação se duplicata esta paga e o cliente vinculado a conta. Além disso, o valor da multa, bem como dos juros e valor total devem ser calculados automaticamente e mantidos pelo sistema. A consulta é realizada através da identificação da conta através do número da duplicata. Quando consultado, o sistema deve calcular o valor total da conta. Esse cálculo é realizado levando em consideração os juros (quando o vencimento passou dos dias que vai tolerar a cobrança de juros) e multas (quando o vencimento passou dos dias que vai tolerar a cobrança de multas).* A Tabela 15 apresenta as funções e regras do Épico 1, além da quantidade de PF.

[7] Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes.

Tabela 15 – Descrição do Épico 1

ID	Descrição
Funções do Épico 1	
RF1	Incluir Conta
RF2	Editar Conta
RF3	Consultar Conta
RF4	Editar Conta
RF5	Cálculo dos Juros
RF6	Cálculo da Multa
RF7	Cálculo do Valor Total
Regras do Épico 1	
R1	Ao buscar contas a receber, trazer calculado o valor total, calculando os juros e multas
R2	Todos os campos devem ser obrigatório para inserção de uma nova conta
R3	Se nas configurações do financeiro estiver marcado para calcular juros, calcula-se o juros
R4	Se nas configurações do financeiro estiver marcado para calcular multa, calcula-se a multa
R5	Se for calcular juros, deve-se verificar se o vencimento passou dos dias tolerado para cobrança de juros. Então efetiva-se o cálculo dos juros
R6	Consultar uma conta pelo número da duplicata.
R7	Se for calcular multa, deve-se verificar se o vencimento passou dos dias tolerados para cobrança de multas. Então efetiva-se o calculo da multa
R8	A edição e exclusão de uma conta deve ser através da consulta do número da duplicata
R9	Valor Pendente deve receber o valor da duplicata
R10	Apenas não poderá ser editado e nem excluído uma conta caso a mesma possua duplicata paga
Total de PF: 42	

Fonte: Elaborado pelo autor

Épico 2: *O serviço deve permitir a importação de plano de contas. Essa importação leva em consideração as informações de cada plano, como o identificador único do plano, tipo de conta de classificação, tipo de regime contábil, nota fiscal (com os produtos relacionados a nota). Esse serviço deve garantir que as restrições de propriedade de um plano de conta sejam respeitadas. Além disso, também deve ser possível consultar informações de todas notas fiscais de um determinado Plano de Contas. Por fim, também deve ser possível consultar, através de um serviço, a soma de notas fiscais de todos planos de contas de um determinado cliente.* A Tabela 16 apresenta as funções e regras do Épico 2 e a quantidade de PF.

Tabela 16 – Descrição do Épico 2

ID	**Descrição**
Funções do Épico 2	
RF1	Importar Plano de Contas
RF2	Cálculo do valor total do produto
RF3	Cálculo custo do frete por produto
RF4	Consultar todas notas fiscais de um plano
Regras do Épico 2	
R1	Tipo de Conta (1 carácter A ou S = Analítica ou Sintética)
R2	Classificação (números de 1 até 6)
R3	Tipo Regime Contábil (valores numéricos 0,1 ou 2)
R4	Data de emissão deve estar no formato (dd/mm/yyyy)
R5	Cliente da nota fiscal é o mesmo cliente do Plano de Notas
R6	A soma do total de frete de produto de todos os produtos tem que fechar o mesmo valor do total frete da nota fiscal
R7	A importação do plano de contas deve ser feito a partir de um arquivo *.txt* no formato *csv* (separação com ; e {produtos})
R8	Todas contas inválidas em um arquivo devem ser informadas para quem está importando ter a possibilidade de corrigir
R9	Todos cálculos (com exceção do total de frete da nota) são realizados quando importado o plano de contas. Somente é sobrescrito se o valor do arquivo for diferente do resultado do cálculo
R10	Consulta da soma da nota retorna um valor em *xml* com total
R11	Todos os campos são necessários, qualquer conta sem uma dessas informações fica não é cadastrada
R12	O identificador do plano de contas é único, não podendo ter outros planos com mesmo identificador
Total de PF: 54	

Fonte: Elaborado pelo autor

6.2 Operacionalização

O primeiro contato com os sujeitos foi feito através de uma sessão introdutória e foi dividida em duas partes: 1) apresentação das etapas do experimento; 2) preenchimento de formulário contendo os termos de concordância. Apresentamos o objetivo do experimento, reforçando que estávamos analisando as técnicas e não os sujeitos, além da explicação do termo de concordância.

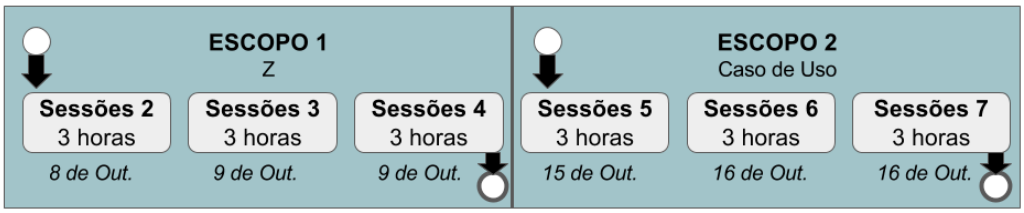
Antecipadamente, configuramos um projeto que continha componentes de software que seriam utilizados pelos sujeitos. Como componentes de conexão com banco de dados. Assim, os sujeitos apenas precisavam se preocupar em desenvolver os serviços de software

solicitado para a *Web Service*. Esse projeto foi configurado em quatro repositórios e só foi liberado para os sujeitos no início da sessão. Removemos o acesso dos sujeitos nos repositórios ao final de cada entrega de Épico, bem como excluído o projeto na máquina de desenvolvimento. Essa ação foi realizada principalmente para inviabilizar que os sujeitos reutilizassem códigos do primeiro Épico para o segundo.

Para obter melhor controle da experiência, preparamos uma máquina para gravação de toda experiência, com consentimento de todos. Por fim, elaboramos uma *checklist* utilizada para validação de cada sessão, contendo itens como: 1) a seção foi executado no tempo previsto, não menos que 3 horas; 2) não houve interferência direta do SM ePO na construção dos artefatos solicitados; 3) não houve interrupções externas durante a execução da seção; 4) foi respeitado a utilização por todos desenvolvedores da ferramenta Netbeans; 5) foi respeitado a utilização por todos desenvolvedores da ferramenta Microsoft Word; 6) foi respeitado a utilização por todos desenvolvedores da ferramenta *Z tools*; 7) todos participantes modelaram os requisitos utilizando a técnica proposta; 8) foi respeitado a utilização por todos desenvolvedores da ferramenta Git; 9) foi respeitado a utilização por todos desenvolvedores do repositório; 10) foi claramente apresentado aos participantes o objetivo da seção e o que deveria ser entregue; 11) todos do TD compareceram da seção; 12) houve problemas de infraestrutura que impediram ou atrasaram a execução da seção;

Como podemos observar na Figura 7, a execução da experiência ocorreu em outubro 2017 e foi composta por duas fases: treinamento e execução da experiência. Ao todo, foram executadas sete sessões, sendo a primeira de 1h e 30m, utilizada para a apresentação inicial e treinamento, e as demais de 3 horas cada.

Figura 7 – Organização das sessões na etapa de execução do experimento.



Fonte: Elaborado pelo autor

Na sessão de treinamento executamos o tutorial elaborado exclusivamente para o experimento. Nele, apresentamos os conceitos e exemplos práticos do uso do Z e do Caso de Uso, respectivamente. Exemplos fictícios foram usados para o treinamento. Utilizamos constantemente o quadro branco, disponível no ambiente, como forma de observar o engajamento dos sujeitos.

Posteriormente a sessão inicial, foram conduzidas as sessões da fase de execução.

Nas sessões 2, 3 e 4 os sujeitos tiveram que utilizar a notação Z e nas sessões 5, 6 e 7 utilizaram Caso de Uso descritivo. Nas sessões 2 e 5 o PO (desempenhado pelo pesquisador) apresentou o Épico que deveria ser desenvolvido. Além disso, o SM (também exercida pelo pesquisador) atuou como organizador do Scrum, fazendo com que os sujeitos realizassem os eventos do *framework*, como as reuniões diárias.

Ao final das sessões 4 e 7 os participantes tiveram 30 minutos para preencher aos formulários de pós-experimento. Além disso, ao final da sessão 7 também solicitamos que preenchessem o formulário final de comparação das técnicas. A seguir, apresentamos uma síntese dos acontecimentos das sessões:

Sessões 2, 3 e 4: Todas as sessões envolveram especificar formalmente os requisitos utilizando a notação Z. Após os sujeitos receberem o Épico 1 na sessão 1 foi necessária a intervenção do SM, justamente para dar início aos eventos do Scrum, como a reunião de planejamento. Percebemos que a postura dos sujeitos após isso foi constantemente evoluindo, ao modo que eles foram tomando propriedade do software produzido.

Os sujeitos utilizaram o quadro branco para especificar colaborativamente os requisitos. Após um tempo, eles trascreveram as especificações para a ferramenta oferecida para tal atividade, *Microsoft Word + Z tools*.

Os sujeitos passaram por todas etapas de desenvolvimento de software, iniciando pela especificação, onde consequentemente percebiam a arquitetura necessária, bem como codificação e teste. Por decisão da equipe, eles elaboraram casos de testes para testar seus serviços. Não houve o evento "Reunião de Revisão da Sprint", devido a não entrega do software por parte dos participantes.

Sessões 5, 6 e 7: Na sessão 5 os participantes receberam o Épico 2 pelo PO. Na sequência, os sujeitos especificaram os requisitos com Caso de Uso descritivo. No decorrer da experiência notamos que os participantes estavam propensos a utilizar outras técnicas de especificação, como modelo conceitual, entre outros. Isso pode estar relacionado com as experiências que eles possuem com técnicas informais. Então, foi preciso interferir nas atividades enfatizando que a única forma de especificação deveria ser Caso de Uso.

Da mesma forma que as sessões anteriores, os participantes passaram por todas etapas do ciclo de desenvolvimento. Para especificação com Caso de Uso, os sujeitos compartilharam um documento remoto para desenvolvimento compartilhado da especificação, não fazendo uso do quadro branco em nenhum momento. Ao final da sessão 7 os participantes esboçaram realizar a entrega do Épico , contudo foi percebido que faltavam algumas regras à serem desenvolvidas.

6.3 Análise e Interpretação

Nesta seção, realizamos a análise e interpretação dos resultados gerados em nossa experiência. Apresentamos um perfil dos sujeitos, tempo efetivo desempenhado nas atividades, quantidade de PF entregue em cada escopo, quantidade de requisitos identificados

e mapeados nas especificações, além dos resultados das percepções dos sujeitos na experiência.

Após levantarmos o perfil dos sujeitos do experimento notamos que 75% dos deles possuem entre 2 a 3 anos de experiência em atuação em projetos ágeis. Apenas um sujeito possui mais de dois anos de experiência. Além disso, todos sujeitos nunca utilizaram EF, mas conhecem o conceito. Por fim, também notamos que todos possuem experiências práticas com *framework* Scrum, além de especificação com Casos de Uso.

As questões de produtividade e qualidade das especificações foram analisadas em duas perspectivas: 1) Análise Subjetiva das Medidas; 2) Análise Objetiva das Medidas.

6.3.1 Análise Subjetiva

Os dados gerados através dos formulários de pós-experimento são ilustrados a seguir, onde é apontado da percepcão dos sujeitos através do valor na escala Likert (LIKERT, 1932), além da Mediana de respostas das três questões que foram derivadas para cada variável dependente. A forma de interpretação das **Medianas** das escalas foram: Mediana < 2 indica uma concordância; Mediana $= 2$ indica inconclusividade; Mediana $>$ 2 de indica discordância.

Na Tabela 17 apresentamos os dados gerados nos instrumentos que avaliam a percepção dos sujeitos sobre a **qualidade das especificações**.

Tabela 17 – Dados dos questionários de pós-experimentos (informal e formal). Atributo qualidade.

Q1 - Qual método de especificação (formal ou informal) possibilita gerar especificações com melhor qualidade que impactam o software entregue?						
#	**Prol Formal**			**Prol Informal**		
Participantes	$Q1_{qf}$	$Q2_{qf}$	$Q3_{qf}$	$Q1_{qi}$	$Q2_{qi}$	$Q3_{qi}$
Sujeito 1	0	4	3	1	1	2
Sujeito 2	4	2	2	0	1	2
Sujeito 3	0	3	2	0	1	3
Sujeito 4	0	2	2	0	0	2
Mediana	0	2,5	2	0	1	2

Fonte: Elaborado pelo autor

Analisando das questões $Q1_{qf}$ e $Q1_{qi}$ observamos que a Mediana de cada questão da Tabela 17, percebemos que o TD discorda fortemente tanto da $Q1_{qf}$, quanto do contrário ($Q1_{qi}$). Entendemos que a afirmação de que um dos métodos garantem que a especificação esta livre de erros trouxe desconformo aos sujeitos, visto que o TD não concordou com nenhuma das questões. Apenas percebemos uma certa peculiaridade no Sujeito 2, que

concordou fortemente com a afirmação de que EF possibilita uma garantir de qualidade ao livrar os requisitos de erros de especificação.

Analisando as questões $Q2_{qf}$ e $Q2_{qi}$ notamos que, diferentemente da $Q1_{qf}$ e $Q1_{qi}$, os dados apresentados na $Q2_{qf}$ e $Q2_{qi}$ são mais expressivos. A opinião do TD quando analisado a Mediana gerada na $Q2_{qf}$ indica uma leve concordância na afirmação, sendo a Mediana de 2,5. Corroborando essa percepção, a $Q2_{qi}$ apresentou uma Mediana 1, sendo as opiniões dos Sujeitos oscilando em 0 e 1.

Essa questão foi mais específica, pois comparou claramente ambos métodos de especificação. Nesse sentido, entedemos como forte indício de confiabilidade, pois percebemos que o TD concordou em indicar o formalismo como melhor prática nesse contexto.

Observamos as questões $Q3_{qf}$ e $Q3_{qi}$ notamos que elas apresentaram uma Mediana igual, sendo 2 em cada. Quando confrontamos ambas as técnicas, uma sendo melhor que a outra, os dados dos sujeitos indicaram um conflito de informação. Dessa forma não podemos inferir precisamente a opinião do TD nessa questão.

Em uma análise geral, percebemos que os desempenhos das afirmações que ressaltam a EF como melhor técnica para especificação do que EI indicam uma maior concordância, assim percebemos que existem indícios de que as EF contribuem para qualidade das especificações, na percepção dos sujeitos. Mesmo a $Q2_qf$ indicando uma tendência, nas demais questões nós encontramos dados inconclusivos, como foi o caso das questões $Q1_q$ e $Q2_q3$.

Na Tabela 18 apresentamos os dados gerados dos questionários de pós-experimentos com relação a variável dependente **produtividade**.

Tabela 18 – Dados dos questionários de pós-experimentos (informal e formal). Atributo produtividade.

Q2 - Qual método de especificação (formal ou informal) possibilita uma maior produtividade em projeto ágil?				
#	**Prol Formal**		**Prol Informal**	
Participante	$Q1_{pf}$	$Q2_{pf}$	$Q1_{pi}$	$Q2_{pi}$
Sujeito 1	3	0	3	4
Sujeito 2	1	1	1	3
Sujeito 3	2	2	1	0
Sujeito 4	1	1	3	4
Mediana	1,5	1	2	3,5

Fonte: Elaborado pelo autor

Quando questionados sobre seu grau de concordância sobre a produtividade proporcionada por cada técnica houve uma percepção compartilhada entre os sujeitos. Observando as questões $Q1_{pf}$ e $Q1_{pi}$, notamos que o TD, em relação a produtividade, discorda

com a $Q1_{pf}$, onde é afirmado que EF diminui o tempo de produção total do software. Na respectiva questão contrária ($Q1_{pi}$) não houve uma tendência negativa ou positiva central, resultando na Mediana neutra. Notamos que as percepções dos sujeitos foram opostas, oscilando de discordo e concordo.

Nas questões $Q2_{pf}$ e $Q2_{pi}$, quando afirmado que uma das técnicas é mais rápida e simples que a outra o TD concordou em indicar o Caso de Uso como sendo a técnica com melhores resultados para esses aspectos. Essa percepção pode estar relacionada com a tendência em pensar em matemática como algo complexo e demorado de se fazer. Assim, pode ser que um desenvolvedor ágil, por ter mais experiências com práticas informais, tenha tendência em acreditar que esse tipo de método possui mais desvantagens pro ambiente ágil do que vantagens.

Igualmente aos dados sobre **qualidade das especificações**, também obtivemos dados inconclusivos. Notamos que 50% dos sujeitos indicaram que Caso de Uso foi melhor do que Z no aspecto qualidade. Além disso, outros 50% foram dados inconclusivos.

Por fim, na Tabela 19 ilustramos as respostas dos sujeitos no questionário final do experimento, onde os mesmos se posicionaram sobre qual técnica foi a melhor nas experiências. Observamos que na questão Q2, os sujeitos não indicaram que algum dos métodos tenha interferido na produtividade do TD. Essa possibilidade era prevista, pois objetivamos identificar se o uso de um dos métodos foi um dos fatores que impactaram positivamente no desenvolvimento do software. Contudo, notamos que na questão Q4, os participantes se posicionaram.

Tabela 19 – Questionário final de pós-experimento (comparativo entre as técnicas)

Questionário pós-experimento (comparativo entre os 2)				
#	**Relação a Produtividade**		**Relação a Qualidade**	
Participantes	Q1	Q2	Q3	Q4
Sujeito 1	Caso de Uso	-	Método Z	Método Z
Sujeito 2	Caso de Uso	-	Caso de Uso	Método Z
Sujeito 3	Caso de Uso	-	Método Z	Método Z
Sujeito 4	Caso de Uso	-	Caso de Uso	-

Fonte: Elaborado pelo autor

Fazendo uma análise geral da Tabela 19 percebemos que 100% dos sujeitos concordaram de que Caso de Uso foi melhor que Z promovendo uma melhor **produtividade** no ambiente ágil. Contudo, para as questões relacionadas a **qualidade** houve distribuição de opiniões. 62,50% apontaram que a notação Z proporcionou uma melhor qualidade nas especificações, além disso, 37,50% incidiram o contrário.

6.3.2 Análise Objetiva

Os resultados gerados sobre a qualidade das especificações, em relação ao atributo completude, apontaram diferenças entre os requisitos mapeados. As especificações criadas pelo TD com utilização de Z e Caso de Uso são descritas nos Anexos A e B, respectivamente.

A Figura 8 sintetiza os requisitos especificados pelo TD. Observamos que do Épico 1, 100% das funções dos requisitos (RF) foram identificadas e mapeadas. Além disso, 90% das regras (R) dos requisitos também foram mapeadas. Analisando as especificações em Caso de Uso para o Épico 2, observamos que 60% das funções foram mapeadas, além disso, 41,66% das regras do escopo também foram identificadas e mapeadas na especificação.

Figura 8 – Requisitos mapeados na especificação dos dois escopos.

Fonte: Elaborado pelo autor

– **Aplicando a métrica M1 para qualidade das especificações:** O Épico 1 possui 7 funções e 10 regras. Analisando os elementos que foram identificados e mapeados na Figura 8 das especificações com utilização do Z temos: 9 funções e 7 regras mapeadas. Então:

Funções → F = 7
Regras → R = 10
Funções → $F_1 = 7$
Regras → $R_1 = 9$

Para avaliar a completude da especificação aplica-se a seguinte métrica:

- Análise da completude:
  - **M1** = $\frac{F_1+R_1}{F+R}$
  - **M1 = (7 + 9) / (7 + 10) = 0,94**

Interpretando esse resultado, percebemos que esse valor informa que existem problemas na especificação do requisitos, assim, existem elementos nos requisitos que não foram mapeados para especificação em Z. Esse resultado confirma que em relação à completude, a especificação dos requisitos está incompleta, pois existem requisitos que não estão mapeados na especificação em Z.

O Épico 2 possui 5 funções e 12 regras. Observando os elementos que foram identificados e mapeados na especificação com utilização Caso de Uso temos: 3 funções e 5 regras mapeadas. Então:

Funções → F = 5
Regras → R = 12
Funções → $F_1 = 3$
Regras → $R_1 = 5$

Para avaliar a completude da especificação aplica-se a seguinte métrica:

- Análise da completude:
  - **M1** = $\frac{F_1+R_1}{F+R}$
  - **M1 = (3 + 5) / (5 + 12) = 0,47**

Interpretando esse resultado, também percebemos que existe problemas na especificação do requisitos, existindo elementos não foram mapeados para especificação em Caso de Uso. Esse resultado confirma que em relação à completude, a especificação está incompleta, pois existem requisitos que não estão mapeados na especificação em Caso de Uso.

Comparando os resultados dos dois escopos para cada métrica temos, então:

$M1_z = 0{,}94$ ✓ $M1_{uc} = 0{,}47$

Percebemos claramente que, mesmo não havendo uma completude de 100% dos requisitos, com nenhuma das técnicas, os resultados obtidos com Z ($M1_z$) foram melhores do que com Caso de Uso. Por exemplo, realizando um calculo geral de qualidade, percebemos que com utilização da notação Z houve um desempenho levemente melhor que os dados do Caso de Uso, no atributo completude para qualidade da especificação.

Para análise da **produtividade** do TD, inicialmente isolamos o tempo efetivo de trabalho realizado por cada sujeito. A Figura 9 apresenta o tempo desempenhado por cada sujeito para cada escopo (sessões 2, 3 e 4 para Épico 1 e sessões 5, 6 e 7 para Épico 2), removendo o tempo desperdiçado com questões fora de escopo do experimento. Dado que cada sujeito tinha 9 horas de trabalho para cada escopo, somados totalizou 36 horas disponíveis e 72 horas para todo experimento (com exceção da sessão 1). Observamos que para ambos os Épicos cada sujeito teve, pelo menos, 34 minutos reduzidos de trabalho. Para o Épico 1, 10h24m foram desperdiçados e removidos do tempo total disponível para as sessões. Assim, o TD trabalhou efetivamente 25h36m no Épico 1. Enquanto para o Épico 2 o TD obtiveram 27h35m efetivamente de esforço.

Além disso, após utilizar a aplicação *client* desenvolvida para testar os artefatos de software gerados pelos TD descobrimos que o TD entregou 34 PF dos 42 solicitados para o Épico 1. Enquanto, do Épico 2 foram estregues 35 PF dos 54 solicitados.

– **Aplicando a métrica** PF_{hora}=$\frac{PFtotal\_entregue}{HRtotal\_trabalhada}$ **para produtividade do TD:**

- Épico 1, utilizando notação Z:

Figura 9 – Esforço desempenhado em cada escopo.

Fonte: Elaborado pelo autor

  - $PFtotal\_entregue$ = **34 PF**
  - $HRtotal\_trabalhada$ = **25h36m**
  - PF_{hora}=$\frac{PFtotal\_entregue}{HRtotal\_trabalhada}$ = **1,35 por hora**

- Épico 2, utilizando Caso de Uso:

  - $PFtotal\_entregue$ = **35 PF**
  - $HRtotal\_trabalhada$ = **27h35m**
  - PF_{hora}=$\frac{PFtotal\_entregue}{HRtotal\_trabalhada}$ = **1,28 por hora**

Percebemos que o desempenho da **produtividade** do TD com utilização da notação Z, foi 4,55% melhor do que com Caso de Uso. Essa influência indica a relação com a notação formal, pois devido a necessidade de entendimento para geração dos modelos o time precisou descobrir mais informações dos requisitos, fazendo com que problemas durante o desenvolvimento tenham sido mitigados. Além disso, podemos pensar que 4,55% pode ser uma diferença relativamente baixa, contudo, se pensarmos em projetos de grande escala sua significância pode ser mais elevada.

Comparando os resultados de ambas perspectivas (subjetiva e objetiva) entendemos que, para qualidade das especificações, tanto a percepção quanto o desempenho com a utilização do método Z foram melhores que o Caso de Uso. Observando a variável produtividade, notamos que a percepção do TD indicou que escrever requisitos com Caso de Uso gerou maior produtividade para o time. Entretanto, notamos que o time entregou mais software quando utilizado a notação Z do que quando utilizado Caso de Uso. Essa diferença pode refletir na tendência em haver pré-conceito com as notações formais, onde na percepção dos sujeitos esse tipo de técnica não é bem aceita.

Fazendo um relação entre produtividade e qualidade o TD entregou mais e com maior qualidade (em relação a completude) quando utilizado a notação Z. Não podemos

generalizar esse resultado, mas podemos tomar como um indício de que se aplicado no mesmo contexto e em grande escala há, potencialmente, chance de se repetir tal resultado.

6.4 Ameaças à Validade

Uma ameaça à **validade da conclusão** é o número pequeno de sujeitos. Essa ameaça pode impactar na precisão dos resultados do experimento, mas acreditamos que com o desenvolvimento do protocolo cuidadosamente e os resultados obtidos contribuímos para clarificar as teses na utilização de formalismo em projetos ágeis.

A atuação do PO como fonte de requisito diminuiu a necessidade dos sujeitos recorrerem as suas especificações para tirar dúvidas. Entendemos essa situação como um ameaça, mas acreditamos que, havendo atuação do PO, a experiência trouxe um aspecto mais real, do que acontece na indústria.

Por envolver o julgamento humano temos ameaças nas medidas subjetivas. Para não depender unicamente dessas medidas utilizamos métricas de dados objetivos, onde é possível analisar ambas as perspectivas.

Mesmo termos identificado cuidadosamente o esforço real desempenhado nas atividades, removendo o tempo desperdiçado, como introdução involuntária de novas variáveis, podemos ter uma ameaça por não termos mapeado o impacto que a inclusão de uma nova tecnologia de desenvolvimento teve nos resultados, tanto positiva quanto negativamente. Contudo, notamos que isso ocorreu no desenvolvimento dos dois escopos, distribuindo os impactos gerados.

Como não limitamos algumas questões durante o experimento, como utilização de celular e conversas paralelas, realizamos uma análise das gravações para remover dos resultados o tempo desperdiçado com essas questões, mitigando essa ameaça.

Outra possível ameaça é em relação a diferença de escopos do experimento em PF. Essa diferença poderia ter causado uma demanda extra de esforço para especificação de requisitos de alguma técnica, por exemplo, para o Caso de Uso onde o escopo respectivo possui 54 PF. Para ambos escopos nós garantimos um escopo maior do que a capacidade de entrega do TD.

Nossa amostra não foi estratificada, contudo entendemos que o nosso propósito era investigar as práticas no contexto do time ágil da Empresa A. Além disso, nosso objetivo é levantar perguntas, através do experimento. Mesmo assim, foi realizada uma pesquisa para avaliar o conhecimento e a experiência dos sujeitos. Mas como não realizamos o bloqueio, visto que todos sujeitos são funcionários da empresa, se tornou uma ameaça ao estudo. Entendemos que essa ameaça é mitigada ao modo que todos sujeitos são profissionais da indústria, com conhecimento em práticas ágeis, mas sem conhecimento avançado em EF.

Para **validade externa**, dado que os sujeitos foram profissionais da indústria com experiência em práticas ágeis foi possível mitigar a ameaça de selecionar um grupo

de participantes que não represente a comunidade em foco.

Outra ameaça é que os sujeitos e pesquisador se conhecem e isso pode interferir nos resultados gerados. Contudo, como elaboramos a estratégia de medir objetiva e subjetivamente entendemos que essa ameaça foi mitigada, ao modo que comparamos os resultados para observar comportamento tendencioso, por exemplo, os sujeitos indicarem positivamente que formal é melhor que informal apenas por conhecer a proposta da pesquisa.

Uma possível ameaça à **validade da construção** seria os participantes acharem que estavam sendo avaliados e que os resultados poderiam impactar no seu trabalho na Empresa A. Para mitigar essa ameaça, explicamos aos participantes que estávamos observando apenas a utilização de diferentes técnicas de especificação em um projeto rodado com Scrum.

6.5 Lições do Capítulo

Esse capítulo teve por objetivo descrever o planejamento, execução e análise dos resultados geradas através de um experimento controlado. Percebemos que o planejamento é uma das atividades mais importantes de qualquer estudo exploratório, pois possibilita visualizar e clarificar os objetivos do estudo, bem como possíveis ameaças e impedimentos que possam ocorrer ao longo desse.

A principal motivação desse experimento partiu da necessidade da indústria em descobrir formas diferentes de produzir software com foco nos resultados gerados na especificação de requisitos. A indústria em questão foi uma empresa parceira que cedeu seus funcionários para participarem dessa pesquisa. Quatro sujeitos foram utilizado, onde foram observados em um experimento in-vivo, com especificações reais da empresa. Nessa experiência analisamos duas principais variáveis que impactam o desenvolvimento de software na empresa, qualidade das especificações e produtividade.

Para ambas as variáveis, foram utilizadas duas perspectiva de análise. A primeiro, objetivou utilizar instrumentos que possibilitassem observar as percepções dos sujeitos quanto a utilização da notação Z e do Caso Uso para especificação de requisitos em um projeto rodando o *framework* Scrum.

Uma análise detalhada dos dados gerados indicou que, tanto as percepções dos sujeitos quanto os artefatos gerados, há uma diferencia em utilizar Z e Caso de uso no contexto experimentado. O método Z obteve melhores resultados nos artefatos quanto a qualidade das especificações geradas. Corroborando esse resultado, os sujeitos também concordaram que Z proporcionou uma melhor qualidade nas especificações, impactando diretamente no software gerado.

Por outro lado, os sujeitos não acreditam que Z tenha potencializado a produtividade do TD. Contudo, a análise objetiva, levando em consideração a quantidade de software entregue, indicou que a produtividade foi ligeiramente melhor ao utilizar Z do

que Caso de Uso.

Através de um detalhamento do design do experimento foi possível identificarmos as principais fragilidades do estudo. Por exemplo, entendemos que os dados gerados aqui são limitados, devido as variáveis que não conseguimos controlar totalmente. Além disso, também notamos ameaças que impactaram na forma de análise dos dados. Por exemplo, a inviabilidade de testar estatisticamente nossos dados tornam os resultados com baixa generalidade. Contudo, como nosso principal objetivo foi explorar a aplicação de EF em ambientes ágeis entendemos que os dados apresentados aqui possibilitam realizar alguns questionamentos com embasamento sobre a óptica de uma experiência prática.

7 CONSIDERAÇÕES FINAIS

Este trabalho foi motivado por tentarmos contribuir para soluções de problemas de especificação enfrentados por uma empresa de desenvolvimento de software da região sul do país. Assim, objetamos realizar um amplo estudo exploratório para coletar evidências através diferentes estudos empíricos da aplicabilidade de EF em ambientes ágeis.

Para tornar essa pesquisa sistemática, inicialmente realizamos um SMS onde buscamos descobrir o estado da arte sobre a utilização de métodos para EF de requisitos em ambientes que adotam práticas ágeis de desenvolvimento de software. A partir disso, descobrimos que já existem iniciativas de estudos que buscam a utilização desses métodos. Descobrimos poucos estudos que apresentam informações robustas sobre tal aplicação. Sendo assim, elaboramos três estudos, nomeados por Estratégia Empírica, para coletar diferentes percepções de profissionais de desenvolvimento de software atuantes em projetos ágeis da indústria.

A primeira estratégia foi a condução de um *Survey* que obteve um total de 20 respondentes da indústria de software, onde coletamos suas percepções sobre a utilização de EF com prática de especificação de requisitos de software. Na sequência, a segunda estratégia foi constituída pela elaboração de um estudo de caso com participação de duas empresas parceiras na pesquisa, onde interferimos na etapa de especificação de requisitos e solicitamos aos participantes utilizarem o MF Z para especificação de requisitos. Por fim, a terceira estratégia foi a execução de um experimento controlado na empresa interessada nessa pesquisa, ao qual motivou esse trabalho. Nesse experimento, buscamos analisar a aplicação do método Z em dois atributos fundamentais para o desenvolvimento de software: qualidade das especificações e produtividade do time ágil.

Na Tabela 20 descrevemos os principais achados que sustentam a importância desta pesquisa, ilustrando os estudos da literatura que apoiam nossas descobertas, bem como os novos resultados.

No nosso estudo, notamos que as percepções dos diferentes participantes do *Survey* e estudo de caso indicam que utilizar as EF podem contribuir para uma melhor compreensão dos requisitos, trazendo benefícios para ambientes ágeis. Isso está diretamente relacionado a sintaxe e semântica definida para os MF. Nesse sentido, acreditamos que esses métodos contribuem para os ambientes ágeis, potencializando a compreensão bem como auxiliando a entregar software que satisfaça necessidade do usuário final.

De acordo com os resultados apontados nos nossos estudos entendemos que um forte limitador para utilização de MF é a complexidade de aprendizado implicada nas técnicas formais, devido ao rigor matemático. Entendemos que essas percepções tiveram uma evolução por partes dos participantes dos estudos. Por exemplo, no estudo de caso houve uma mudança de opinião, onde os sujeitos perceberam que a dificuldade é relativa até conhecer a EF na prática. Da mesma forma, no experimento não foi relatado esse tipo de percepção e também não percebemos em conversas com os sujeitos durante e execução

Tabela 20 – Mapeamento dos nossos resultados entre todas estratégias e trabalhos relacionados

	Survey	Estudo de Caso	Experimentos	Trabalhos Relacionados
EF facilita a compreensão dos requisitos	✓	✓		✓
É complexo aprender a especificar com EF	✓	✓		✓
É demorado especificar com EF, além de precisar adaptar equipe	✓			✓
É melhor EF somente em requisitos cruciais para o sistema	✓	✓		
As pessoas não aceitam EF por pré-conceito	✓	✓	✓	
Com o tempo, utilizar EF vai se tornando menos demorado	✓	✓		
É melhor ler EF do que Caso de Uso		✓	✓	
Tempo para entender EF é um fator limitador para sua utilização	✓	✓		

Fonte: Elaborado pelo autor

da experiencia.

Ao contrário dos trabalhos relacionados, nossa pesquisa inclui novas descobertas- Por exemplo, um possível cenário adequado para utilização de EF em projetos ágeis é quando há a necessidade de entender ou garantir formalmente um determinado requisito, sendo ele crucial para o sistema. Faz sentido utilizar dessa forma, pois de acordo com os princípios ágeis, devemos utilizar aquelas ferramentas que convém no qual potencializará a entrega dos incrementos de software com maior eficácia.

Durante esta pesquisa, notamos que quando utilizado EF foi percebido seus benefícios. Por exemplo, a produtividade e qualidade dos artefatos gerados quando utilizado EF foram melhores que quando utilizado Caso de Uso, por exemplo. Isso mostra claramente uma certa discordância do que se entende por utilização de EF, principalmente no entendimento dos desenvolvedores de software. Entendemos assim, que o principal limitador da disseminação de EF em abordagens ágeis é, e possivelmente continuará sendo, o pré-conceito por partes dos desenvolvedores de software.

Nós entendemos que toda e qualquer técnica/método para alguma etapa do desenvolvimento possui suas vantagens e limitações. Nesse sentido, nós temos a consciência de que EF não é uma solução para todos os problemas de especificação, devido o envolvimento de seres humanos, que por natureza então propensos a falhar. Deve ser analisado para cada necessidade se a EF vai contribuir para alcançar o objetivo, levando em considerações seus aspectos. Dessa forma que este trabalho contribui para literatura e demais empresas que queiram utilizar EF, pois proporciona uma experiência prática do seu emprego.

Enfatizamos algumas limitações do nosso trabalho como a quantidade de sujeitos

alcançados em todas experiências. Contudo, ressaltamos que todos os participantes das experiências foram profissionais da indústria com experiência em projetos ágeis, tornando nossos resultados uma reflexão das necessidades de empresas de software. Ainda, ressaltamos que tentamos executar o experimento em um ambiente laboratorial controlado, contudo não consegui alcançar uma quantidade mínima de participantes, realocando os esforços para execução in-vivo, na empresa principal interessada no estudo.

Enfrentamos a dificuldade em aplicar teste de hipótese no experimento controlado, o que limita a generalização dos dados. Mesmo assim, como o objetivo final do estudo é explorar os aspectos em torno de EF em projetos ágeis acreditamos que as evidências geradas clarificaram esses aspectos, possibilitando a empresa interessada nesse estudo tomar decisões quanto o seu emprego.

Como trabalho podemos utilizar as limitações do estudo para aplicar novas experiências, listamos alguns possíveis trabalhos:

1. Aplicar uma Revisão Sistemática da Literatura observando como os métodos para EF estão sendo utilizados em projetos ágeis, como em qual contexto, para qual projeto, perfil do time, entre outros.

2. Aplicar novas pesquisas exploratórias, através de *Survey*, com intuito de descobrir a opinião dos participantes sobre a aplicação de EF em projetos ágeis em diferentes aspectos, como: produtividade, qualidade do software, benefícios para os testes, impacto para o negócio da empresa, entre outros.

3. Desenvolver novas experiências, como experimentos e estudos de casos, onde seja possível executar com diferentes times ágeis de desenvolvimento para pode gerar dados quantitativos para uma análise robusta.

4. Realizar experiências empíricas para observar diferentes aspectos do desenvolvimento em projetos ágeis, como perfil de cada time, ferramentas de desenvolvimento utilizada, afinidade entre os sujeitos, entre outras questões que auxiliem a identificar em que contexto EF melhor se aplica.

5. Conduzir novas experiências para avaliar outros atributos da qualidade da especificação, como ambiguidade, consistência, classificação por importância/estabilidade, verificabilidade, modificabilidade, rastreabilidade, entre outros.

6. Desenvolver ferramentas de apoio aos métodos de EF que possibilitem, por exemplo, especificar, verificar e gerar códigos-fonte a partir do requisito modelado.

REFERÊNCIAS

BARDEN, R.; AL. et. **Z in Practice**. [S.l.]: Prentice-Hall, Inc., 1995. Citado 2 vezes nas páginas 61 e 62.

BECK, K. et al. **Manifesto for agile software development**. 2001. DisponÃvel em http://agilemanifesto.org/, acesso em 21/04/2016. Citado 2 vezes nas páginas 21 e 27.

BOWEN, J.; REEVES, S. UI-driven test-first development of interactive systems. In: ACM. **Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '11**. New York, New York, USA: ACM Press, 2011. p. 165–174. ISBN 9781450306706. Citado 3 vezes nas páginas 41, 42 e 43.

BOWEN, J. P. et al. Formality, agility, security, and evolution in software development. **Computer**, IEEE Computer Society, v. 47, n. 10, p. 86–89, oct 2014. ISSN 00189162. Citado 4 vezes nas páginas 41, 45, 46 e 47.

CARDOZO, E. et al. Scrum and productivity in software projects: a systematic literature review. In: **14th International Conference on Evaluation and Assessment in Software Engineering (EASE)**. [S.l.: s.n.], 2010. Citado na página 28.

CHALFOUN, I.; AL. et. Specification of a reconfigurable and agile manufacturing system (rams). **Int. J. Mech. Eng. Autom**, v. 1, 2014. Citado na página 33.

COCKBURN, A. **Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software**. [S.l.]: Bookman Editora, 2005. Citado 2 vezes nas páginas 33 e 34.

DEHARBE, D. et al. Introdução a métodos formais: Especificação, semântica e verificação de sistemas concorrentes. **RITA**, v. 7, n. 1, p. 7–48, 2000. Citado 3 vezes nas páginas 21, 30 e 31.

DUVALL, P. **Breaking down barriers and reducing cycle times with DevOps and continuous delivery**. 2012. Online. Acessado em Julho–2017. Disponível em: <http://try.newrelic.com/rs/newrelic/images/GigaOm-Pro-Report-Breaking-down-barriers-and-reducing-cycle-times-with-devops-and-continuous pdf>. Citado na página 21.

FERREIRA, C. L.; RACHID, E.; SCHUARTZ, F. C. VDM: uma linguagem de especificação. 2009. Citado na página 31.

FURLAN, J. D. **Modelagem de objetos através da UML-the unified modeling language**. [S.l.]: Makron books, 1998. Citado na página 33.

GARY, K. et al. Agile methods for open source safety -critical software. **Software: Practice and Experience**, Wiley Online Library, v. 41, n. 9, p. 945–962, 2011. Citado 2 vezes nas páginas 41 e 47.

GAUDEL, M.-C. Formal specification techniques. In: IEEE COMPUTER SOCIETY PRESS. **Proceedings of the 16th international conference on Software engineering**. [S.l.], 1994. p. 223–227. Citado na página 30.

KASSAB, M.; NEILL, C.; LAPLANTE, P. State of practice in requirements engineering: contemporary data. **Innovations in Systems and Software Engineering**, v. 10, n. 4, p. 235–241, 2014. Cited By 16. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84920250433&doi=10.1007%2fs11334-014-0232-4&partnerID=40&md5=9401aa3bd37b76ca75403d3021171872>. Citado na página 21.

LAMSWEERDE, A. v. Formal specification: a roadmap. p. 147–159, 2000. Citado 2 vezes nas páginas 22 e 30.

LARSEN, P. G.; FITZGERALD, J. S.; WOLFF, S. Are formal methods ready for agility? a reality check. In: **FM+ AM**. [S.l.: s.n.], 2010. p. 13–25. Citado 7 vezes nas páginas 22, 30, 41, 42, 44, 45 e 47.

LIKERT, R. A technique for the measurement of attitudes. In: **Archives of Psychology**. New York: [s.n.], 1932. Citado 2 vezes nas páginas 71 e 80.

LINDSTROM, L.; JEFFRIES, R. Extreme programming and agile software development methodologies. **Information systems management**, Taylor & Francis, v. 21, n. 3, p. 41–52, 2004. Citado na página 28.

LIPSKI, A. Requirements modeling in agile methodologies with x-machines. **Biuletyn Instytutu Systemów Informatycznych**, p. 19–24, 2011. Citado 7 vezes nas páginas 30, 41, 42, 45, 46, 47 e 48.

LIU, B. Sentiment analysis and opinion mining. **Synthesis lectures on human language technologies**, Morgan & Claypool Publishers, v. 5, n. 1, p. 1–167, 2012. Citado na página 56.

LIU, S. An approach to applying sofl for agile process and its application in developing a test support tool. **Innovations in Systems and Software Engineering**, Springer, v. 6, n. 1-2, p. 137–143, 2010. Citado 7 vezes nas páginas 41, 42, 43, 45, 46, 47 e 51.

LOTOS, I. A formal description technique based on the temporal ordering of observational behaviour. **International Organisation for Standardization-Information Processing Systems-Open Systems Interconnection, Geneva**, 1988. Citado na página 31.

LUCASSEN, G.; AL. et. Improving agile requirements: the quality user story framework and tool. **Requirements Engineering**, Springer, v. 21, 2016. Citado 2 vezes nas páginas 22 e 33.

MADEY, J. The Z notation: A reference manual. **Science of Computer Programming**, v. 15, n. 2-3, p. 253–255, 1990. ISSN 01676423. Citado 3 vezes nas páginas 31, 32 e 109.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. [S.l.]: 5. ed.-São Paulo: Atlas, 2003. Citado 2 vezes nas páginas 36 e 63.

MENEGAZZO, C. T.; VIEIRA, K. V. Um Modelo Formal para Composição de Serviços Web usando a Linguagem SAWSDL. **2o. Seminário de Pesquisa em Planejamento e Gestão Territorial**, p. 1–4, 2007. Citado na página 31.

MISRA, S. et al. Agile software development practices: evolution, principles, and criticisms. **International Journal of Quality & Reliability Management**, Emerald Group Publishing Limited, v. 29, n. 9, p. 972–980, 2012. Citado na página 28.

MOCHIO, H.; ARAKI, K. Vdm++ as a basis of scalable agile formal software development. In: CITESEER. **PROCEEDINGS OF THE 9TH OVERTURE WORKSHOP**. [S.l.], 2012. p. 50. Citado 8 vezes nas páginas 22, 27, 41, 42, 44, 45, 46 e 47.

MOLLÉRI, J. S.; PETERSEN, K.; MENDES, E. Survey guidelines in software engineering: An annotated review. In: ACM. **Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement**. [S.l.], 2016. p. 58. Citado 3 vezes nas páginas 23, 24 e 35.

MORAES, R. Análise de conteúdo. **Revista Educação**, v. 22, 1999. Citado 3 vezes nas páginas 52, 54 e 63.

NUMMENMAA, T. et al. Supporting agile development by facilitating natural user interaction with executable formal specifications. **ACM SIGSOFT Software Engineering Notes**, ACM, v. 36, n. 4, p. 1–10, 2011. Citado 6 vezes nas páginas 41, 42, 44, 45, 46 e 47.

OLSZEWSKA, M.; WALDÉN, M. Devops meets formal modelling in high-criticality complex systems. In: ACM. **Proceedings of the 1st International Workshop on Quality-Aware DevOps**. [S.l.], 2015. p. 7–12. Citado 8 vezes nas páginas 22, 41, 42, 43, 45, 46, 47 e 51.

PAETSCH, F.; EBERLEIN, A.; MAURER, F. Requirements engineering and agile software development. In: IEEE. **Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on**. [S.l.], 2003. p. 308–313. Citado na página 28.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: SN. **12th international conference on evaluation and assessment in software engineering**. [S.l.], 2008. v. 17, n. 1, p. 1–10. Citado na página 39.

ROCHA, F. Z. F. **Modelo para avaliação da qualidade da tradução entre requisitos e casos de uso**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2006. Citado na página 72.

RODRIGUES, P. et al. Métodos para especificação formal de requisitos em metodologias ágeis de desenvolvimento: Um mapeamento sistemático. **Anais do Salão Internacional de Ensino, Pesquisa e Extensão**, v. 8, n. 2, 2017. Citado na página 25.

RODRIGUES, P. L. et al. Métodos formais como condutores para especificação de requisitos aplicados em metodologias ágeis de desenvolvimento. **Anais da Escola Regional de Informática da Sociedade Brasileira de Computação (SBC)–Regional de Mato Grosso**, v. 1, n. 7, 2016. Citado na página 25.

SAEED, T. et al. Mapping Formal Methods to Extreme Programming (XP) â€“ A Futuristic Approach. **International Journal of Natural and Engineering Sciences**, Nobel International Journals, v. 8, n. 3, p. 35–42, 2014. ISSN 1307-1149. Citado 6 vezes nas páginas 31, 41, 42, 45, 46 e 47.

SCHWABER, K.; AL et. The definitive guide to scrum: the rules of the game. **Acesso em http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf**, 2016. Citado 2 vezes nas páginas 28 e 29.

SHAFIQ, S.; MINHAS, N. M. Integrating formal methods in xpâ€"a conceptual solution. **Journal of Software Engineering and Applications**, Scientific Research Publishing, v. 2014, 2014. Citado 7 vezes nas páginas 22, 30, 41, 42, 43, 45 e 47.

SOMMERVILLE, I. 27 Formal Specification. **Software Engineering**, p. 1–23, 2011. Citado 3 vezes nas páginas 21, 30 e 31.

UNTERKALMSTEINER, M.; GORSCHEK, T. Requirements quality assurance in industry: Why, what and how? In: SPRINGER. **International Working Conference on Requirements Engineering: Foundation for Software Quality**. [S.l.], 2017. p. 77–84. Citado na página 21.

WOHLIN, C. et al. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012. Citado 10 vezes nas páginas 23, 24, 35, 36, 51, 58, 61, 67, 69 e 70.

WOLFF, S. Scrum goes formal: Agile methods for safety-critical systems. In: IEEE PRESS. **Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches**. [S.l.], 2012. p. 23–29. Citado 12 vezes nas páginas 21, 22, 27, 28, 41, 42, 43, 45, 46, 47, 48 e 51.

YIN, R. K. **Case study research: Design and methods**. [S.l.]: Sage publications, 2013. Citado 5 vezes nas páginas 23, 24, 35, 36 e 61.

Apêndices

APÊNDICE A – APÊNDICE A - ESPECIFICAÇÃO COM Z

As especificação gerada do Escopo 1, onde foi utilizado o método formal Z para especificação:

$\widehat{=}$*Conta*

DataEmissao: seq_1 *Data*
DataVencimento: seq_1 *Data*
NumeroDuplicata: $\text{seq}_1 \mathbb{N}$
HistoricoCompra: $\mathbb{P}$*Compra*
ValorDuplicata: $\text{seq}_1 \mathbb{R}$
DuplicataPaga: *Boolean*
Cliente: $\mathbb{P}$*Cliente*
ValorMulta: $\text{seq}_1 \mathbb{R}$
ValorJuros: $\text{seq}_1 \mathbb{R}$
ValorTotal: $\text{seq}_1 \mathbb{R}$
ValorPendente: $\text{seq}_1 \mathbb{R}$

DataEmissao $\neq \varnothing$
DataVencimento $\neq \varnothing$
(*Cliente* $\in \mathbb{P}$*Cliente* $\wedge$ *Cliente* $\neq \varnothing$)
ValorDuplicata $\neq \varnothing$
ValorMulta $\neq \varnothing$
ValorTotal $\neq \varnothing$
ValorJuros $\neq \varnothing$
ValorPendente $\neq \varnothing$
ValorDuplicata $\neq \varnothing$
DataEmissao $\leqslant$ *DataVencimento*
ValorPendente = *ValorDuplicata*

$\widehat{=}$*Config*

Id: $\text{seq}_1 \mathbb{N}$
CalcularJuros: *Boolean*
TolerânciaAtrasoSemJuros: $\text{seq}_1 \mathbb{N}$
JurosAtraso: $\text{seq}_1 \mathbb{R}$
CalcularMulta: *Boolean*
TolerânciaAtrasoSemMulta: $\text{seq}_1 \mathbb{N}$
MultaAtraso: $\text{seq}_1 \mathbb{R}$

$\widehat{=}$*Cliente*

Id: $\text{seq}_1 \mathbb{N}$
Contas: $\mathbb{P}$ *Conta*

Id $\neq \varnothing$

$\widehat{=}$*Compra*

Id: $\text{seq}_1 \mathbb{N}$
Conta: $\mathbb{P}$*Conta*

Id $\neq \varnothing$
Conta $\neq \varnothing$

Ξ*CalcularTotal*

ValorPendente?: $\text{seq}_1 \mathbb{R}$
ValorJuros?: $\text{seq}_1 \mathbb{R}$
ValorMulta?: $\text{seq}_1 \mathbb{R}$
Total!: $\text{seq}_1 \mathbb{R}$

Total! = *ValorPendente*? + *ValorJuros*? + *ValorMulta*?

ΞCalcularJuros

ValorPendente?: $\text{seq}_1\ \mathbb{R}$
PercentualJuros?: $\text{seq}_1\ \mathbb{R}$
DiasAtraso?: $\text{seq}_1\ \mathbb{N}$
Juros!: $\text{seq}_1\ \mathbb{R}$

Juros! = (*ValorPendente*?*((*PercentualJuros*?/100)/30))**DiasAtraso*?

ΞValorMulta

ValorPendente?: $\text{seq}_1\ \mathbb{R}$
PercentualMulta?: $\text{seq}_1\ \mathbb{R}$
Multa!: $\text{seq}_1\ \mathbb{R}$

Multa! = *ValorPendente*? * (*PercentualMulta*? /100)

ΞCalcularDiasAtraso

diaAtual?: *Data*
diaVencimento?: *Data*
diasAtraso!: $\text{seq}_1\ \mathbb{N}$

---

diasAtraso!= *diaAtual*? – *diaVencimento*?

ΞCalcularValores

Conta?: *Conta*
Config?: *Config*
DiaAtual?: *Data*
ContaAtualizada!: *Conta*

---

if (*config*?.*CalcularMulta*) **then**
if(*CalcularDiasAtraso*(*DiaAtual*, *Conta*?.*DataVencimento*) > *Config*?.*TolerânciaAtrasoSemMulta*) **then**
Conta?.*ValorMulta′* = *ValorMulta*(*Conta*?.*ValorPendente*, *Config*?.*MultaAtraso*)

If(*Config*?.*CalcularJuros*)**then**
If(*CalcularDiasAtraso*(*DiaAtual*, *Conta*?.*DataVencimento*)> (*Config*?.*TolerânciaAtrasoSemJuros*))**then**
Conta?.*ValorJuros′* = *CalcularJuros*(*Conta*?.*ValorPendente*, *Config*?.*JurosAtraso*, *CalcularDiasAtraso*(*DiaAtual*?, *Conta*?.*DataVencimento*))

Conta?.*ValorTotal′* = *CalcularTotal*(*Conta*?.*ValorPendente*, *Conta*?.*ValorJuros*, *Conta*?.*ValorMulta*)

Conta? ∉ *Conta*?.*Cliente.Contas*

ContaAtualizada! = *Conta*?

ΔInserirConta

Conta?: *Conta*
Config?: *Config*
DiaAtual?: *Data*

Conta? = *CalcularValores*(*Conta*?, *Config*?, *DiaAtual*?)
Contas′ = *Contas* ∪ *Conta*?

Δ*ConsultaConta*

Cliente?: *Cliente*
NúmeroDuplicata?: $\text{seq}_1\ \mathbb{N}$
Cconta?: *Conta*
Config?: *Config*
DiaAtual?: *Data*
Conta!: *Conta*

Conta! = (*n*: *Cliente*?.*Contas* | *n*.*NúmeroDuplicata* = *NúmeroDuplicata*?)
Conta? = *CalcularValores*(*Conta*?, *Config*?, *DiaAtual*?)

Δ*ExcluirConta*

Cliente?:*Cliente*
NúmeroDuplicata?: $\text{seq}_1\ \mathbb{N}$
Conta: *Conta*

Conta′ = *ConsultarConta*(*Cliente*?, *NúmeroDuplicata*?)
If (*Conta*.*DuplicataPaga* = *false*)**then**
Cliente.*Contas′* = *Cliente*.*Contas*? - *Conta*

Δ*EditarConta*

ContaEditada?: *Conta*
Cliente?:*Cliente*
NúmeroDuplicata?: $\text{seq}_1\ \mathbb{N}$
Conta: *Conta*
Config?: *Config*
DiaAtual: *Data*

Conta′ = *ConsultarConta*(*Cliente*?, *NúmeroDuplicata*?)
If (*Conta*.*DuplicataPaga* = *false*)**then**
Conta′ = *Conta* ∪ *ContaEditada*?
Conta′ = *CalcularValores*(*Conta*, *Config*?, *DiaAtual*?)

APÊNDICE B – APÊNDICE B - ESPECIFICAÇÃO COM CASO DE USO

As Tabelas 21, 22 e 23 apresentam as especificações geradas do Escopo 2, onde foi utilizado Caso de Uso para especificação:

Tabela 21 – Caso de Uso - Consultar Notas Fiscais

Identificador	UC01
Nome	Consultar Notas Fiscais
Ator Primário	Sistema Consumidor
Pré-condição	O plano de contas deve estar preenchido em um arquivo txt.
Pré-condição	Plano de contas importado
Cenário de Sucesso Principal	1. O sistema consumidor faz a requisição POST, enviando o arquivo texto contendo todas as informações do plano de contas preenchidas (id, tipo de conta, classificação, cliente, tipo de regime contábil, lista de notas número da nota, data emissão, cliente, soma total dos valores de todos os produtos, soma total de frete, total da nota, lista de produtos, codigo de barras, cor e dimensões do grupo, descricao do produto, quantidade, valor unitario, valor de desconto, valor total, total de frete por produto). 2. O sistema deve fazer o "parser" do arquivo para as entidades relacionadas. 3. O sistema verifica as informações obrigatórias. 4. O sistema recalcula os totais e sobrescreve caso seja diferente. 5. O sistema persiste as informações no banco de dados. 6. O sistema envia uma mensagem de sucesso.
Fluxo alternativo A:	3.1 O sistema não permite continuar com as operações e retorna o status da requisição POST com um texto contendo os campos com problemas.
Fluxo alternativo B:	3.1 O conteúdo do arquivo importado não está de acordo com o padrão. 3.2 O sistema não faz a importação e retorna o status da requisição POST com um texto exibindo uma mensagem de problema.
Fluxo alternativo C:	3.1 O formato do arquivo importado não está de acordo com o padrão. 3.2 O sistema não faz a importação e retorna o status da requisição POST com um texto exibindo uma mensagem de problema.
Fluxo alternativo D:	3.1 O Cliente da Nota Fiscal e o Cliente do Plano de contas são diferentes. 3.2 O sistema não faz a importação e retorna o status da requisição POST com um texto exibindo uma mensagem de problema.
Extensões	

Tabela 22 – Caso de Uso - Importar Plano de Contas

Identificador	UC02
Nome	Importar Plano de Contas
Ator Primário	Sistema Consumidor
Pré-condição	
Pré-condição	Informações de Notas Fiscais recuperadas
Cenário de Sucesso Principal	1. O Sistema Consumidor envia uma requisição GET para o sistema contendo informações do plano de conta (id). 2. O Sistema busca no banco por todas as Notas Fiscais do plano de conta indicado. 3. O Sistema recupera as informações das Notas Fiscais e as armazena em um arquivo XML. 4. O Sistema retorna o arquivo XML para o Sistema Consumidor. 5. O Sistema Consumidor recebe o arquivo XML.
Fluxo alternativo A:	2.1 O Sistema não encontra nenhum Plano de Conta com o id especificado. 2.2 O sistema retorna uma mensagem em um arquivo XML dizendo que o Plano de Conta não foi encontrado.
Fluxo alternativo B:	2.1 O Sistema não encontra nenhuma Nota Fiscal para o plano de conta especificado. 2.2 O sistema retorna um arquivo XML dizendo que não há nenhuma Nota Fiscal no Plano de Conta.
Extensões	

Tabela 23 – Caso de Uso - Consultar Soma de Notas Fiscais

Identificador	UC03
Nome	Consultar Soma de Notas Fiscais
Ator Primário	Sistema Consumidor
Pré-condição	
Pré-condição	Informações de Notas Fiscais recuperadas
Cenário de Sucesso Principal	1.O Sistema Consumidor envia uma requisição GET para o sistema contendo as informações do Cliente (id). 2. O sistema consulta no banco de dados por todos os planos de conta daquele Cliente. 3. O sistema recupera as informações das notas fiscais de cada plano de contas do Ciente. 4. O sistema faz a soma dos valores de todas as notas fiscais de todos os planos de conta daquele Cliente e retorna-o em um arquivo XML. 5. O Sistema Consumidor recebe o valor da soma das notas fiscais.
Fluxo alternativo A:	1.1. O Sistema Consumidor faz uma requisição passando as informações de um Cliente (id) que não existe. 1.2. O sistema retorna uma mensagem em um arquivo XML dizendo que aquele cliente não foi encontrado.
Fluxo alternativo B:	2.1. O Sistema não encontra nenhum plano de contas para aquele cliente. 2.2. O sistema retorna uma mensagem em um arquivo XML dizendo que não existem plano de contas para aquele cliente.
Extensões	

Anexos

ANEXO A – ANEXO A - QUADRO DA VISÃO GERAL DE SÍMBOLOS DO Z

Tabela 24 – Quadro da visão geral de símbolos do Z

$\mathbb{P}$	power set	$\neq$	inequaly	Ξ	(xi)
$\times$	(cross)	$\notin$	Non-membership	$\mathbb{N}$	Natural numbers
$\langle\!\langle \ldots \rangle\!\rangle$	(date) constructor	$\varnothing$	(empty set)	$\mathbb{Z}$	Integers
$\triangleq$	(schema naming)	$\subseteq$	subset relation	$+$	Arithmetic operations
$==$	abbreviation definition	$\subset$	proper subset	-	Arithmetic operations
$(\ldots)$	tuples	$\mathbb{P}_1$	power set	$*$	Arithmetic operations
$\{\ldots\}$	display	$\cup$	union	$<$	Numerical comparison
λ	lambda notation (lambda)	$\cap$	intersection	$\leq$	Numerical comparison
μ	μ-notation (mu)	$\setminus$	Set difference	$\geq$	Numerical comparison
.	selection operator	$\bigcup$	Generalized union	$>$	Numerical comparison
θ	θ notation (theta)	$\bigcap$	Generalized intersection	$\mathbb{N}_1$	Strictly positive integers
$\langle \ldots \rangle$	sequence	$\leftrightarrow$	binary relations	$\upto$	Number range
$[\![\ldots]\!]$	bag	$\mapsto$	maplet	R^k	Iteration
$=$	equality	$\circ$	Backward realtional composition	$\mathbb{F}$	Finite sets
$\in$	set merbership	$\lhd$	domain restriction	$\mathbb{F}_1$	Nom-empty finite sets
$\neg$	negation (not)	$\rhd$	range restriction	$\#$	size or cardinality
$\land$	conjunction (and)	$::=$	free type	$\uplus$	(bag union)
$\lor$	disjunction (or)	$\rightarrowtail$	total, injection functions	$\twoheadrightarrow$	total, surjective functions
$\Rightarrow$	implication (implies)	R~	Relation inversion	$\frown$	Concatenação
$\Leftrightarrow$	equivalência (se e somente se)	$\_(\!\vert\_\vert\!)$	relational image	$\upharpoonright$	Extração
$\forall$	universal quantifier (for all)	$\oplus$	Overriding	$\frown/$	(Concatenação distribuída)
$\exists$	(existe)	R^+	Transitive closure	$\sharp$	(sharp)
$\exists_1$	(único existe)	R^*	Reflexive-transitive closure	$\otimes$	Bag scaling
$\setminus$	hiding	$\gg$	Piping	Δ	(delta)
$\upharpoonright$	filtro	$\rightarrow$	(funções totais)	$\sqsubseteq$	Sub-bag relation
$\fatsemi$	relational composition				

Fonte: (MADEY, 1990). Legenda: representa as sequências; comportamento e condições; elementos da lógica proposicional