FÁBIO RODRIGUES PEREIRA

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA DETECÇÃO DE ANOMALIAS DE REQUISITOS DE *SOFTWARE*

LAVRAS – MG

2023

FÁBIO RODRIGUES PEREIRA

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA DETECÇÃO DE ANOMALIAS DE REQUISITOS DE *SOFTWARE*

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Engenharia de *Software* e Banco de dados, para a obtenção do título de Mestre.

Prof. Paulo Afonso Parreira Júnior
Orientador

LAVRAS – MG
2023

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a)

Pereira, Fábio Rodrigues
Análise Comparativa de Ferramentas para Detecção de Anomalias de Requisitos de *Software* / Fábio Rodrigues Pereira. – Lavras : UFLA, 2023.
71 p. : il.

Dissertação(mestrado acadêmico)–Universidade Federal de Lavras, 2023.
Orientador: Prof. Paulo Afonso Parreira Júnior.
Bibliografia.

1. Engenharia de requisitos. 2.Anomalias de requisitos de software. 3. Qualidade de software.

FÁBIO RODRIGUES PEREIRA

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA DETECÇÃO DE ANOMALIAS DE REQUISITOS DE *SOFTWARE*
COMPARATIVE ANALYSIS OF TOOLS FOR DETECTING SOFTWARE REQUIREMENTS ANOMALIES

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Engenharia de *Software* e Banco de dados, para a obtenção do título de Mestre.

APROVADA em 21 de Julho de 2023.

Prof. Paulo Afonso Parreira Júnior	UFLA
Prof. Mauricio Ronny De Almeida Souza	UFLA
Profa. Ana Carolina Gondim Inocêncio	UFJ

Prof. Paulo Afonso Parreira Júnior
Orientador

LAVRAS – MG
2023

Dedico esta dissertação aos meus pais, Eugênio e Maria.

AGRADECIMENTOS

Primeiramente a Deus que proporcionou oportunidades, perseverança e coragem para superar todos os desafios.

Aos meus pais, Eugênio e Maria, por todo suporte, encorajamento e compreensão.

Ao meu orientador Prof. Dr. Paulo Afonso Parreira Júnior, pela confiança, paciência, amizade, mas principalmente, pelo acolhimento num momento em que me encontrava sem orientador. Muito obrigado pelos ensinamentos e a excelente orientação.

Ao meu amigo Marcos Fonseca da Silva pelo incentivo, apoio e auxílio constante.

À minha namorada Fernanda Cristina da Fonseca pela dedicação oferecida, pelos momentos de companheirismo e compreensão aos meus momentos de ausência.

Por fim, todos aqueles que de forma direta ou indireta colaboraram para a realização deste trabalho.

Você pode encarar um erro como uma besteira a ser esquecida ou como um resultado que aponta uma nova direção. (Steve Jobs)

RESUMO

Um requisito de *software* indica uma capacidade ou uma característica que um *software* deve apresentar para ter valor aos seus *stakeholders*. É fundamental garantir que a descrição dos requisitos seja clara e inequívoca, a fim de permitir seu correto entendimento e facilitar a sua evolução. Porém, como a maioria dos requisitos de *software* são descritos em linguagem natural, pode ser que eles contenham subjetividades e inconsistências em sua descrição, ao que convencionou-se chamar de "Anomalia de Requisitos de *Software*". Diversos trabalhos têm proposto ferramentas com o objetivo de contribuir para a detecção de anomalias de requisitos. Contudo, pode-se notar que poucos desses trabalhos têm avaliado a efetividade (cobertura e precisão) das ferramentas propostas, comparando-as umas com as outras e procurando identificar seus pontos fortes e fracos. Por conseguinte, este trabalho, candidatou-se a realizar um estudo experimental para cobrir esse *gap* da literatura. Ao todo, 3 (três) ferramentas para detecção de anomalias foram criteriosamente selecionadas da literatura (*RETA*, *Tactile Check* e *Tiger Pro*) e analisadas com base em documentos de requisitos de diferentes domínios, contendo mais de 85 anomalias. Como resultado, têm-se que a ferramenta com maior cobertura é a *RETA*, tendo identificado corretamente 85.19% das anomalias existentes. Porém, considerando a média entre cobertura e precisão, a ferramenta *Tactile Check* foi a que apresentou o melhor resultado. Apesar disso, todas as ferramentas comparativamente analisadas neste trabalho tiveram um nível insatisfatório de cobertura e precisão, ficando abaixo de, respectivamente, 66% e 57%. Isso se deve, em grande parte, pelas palavras que compõem o dicionário não estarem bem calibradas com as anomalias que a ferramenta se dispõe em detectar e pela técnica de detecção não considerar o contexto da descrição do requisito do qual o termo anômalo é inserido.

Palavras-chave: Engenharia de requisitos. Anomalias de requisitos de software. Qualidade de software

ABSTRACT

A software requirement indicates a capability or a characteristic that a software must have in order to be of value to its stakeholders. It is essential to ensure that the description of the requirements is clear and unambiguous, in order to allow their correct understanding and facilitate their evolution.However, as most software requirements are described in natural language, they may contain subjectivities and inconsistencies in their description, which is called “Software Requirements Anomaly”. Several works have proposed tools with the objective of contributing to the detection of requirements anomalies. However, it can be noted that few of these works have evaluated the effectiveness (coverage and accuracy) of the proposed tools, comparing them with each other and trying to identify their strengths and weaknesses. Therefore, this work, applied to perform an experimental study to cover this gap in the literature. In all, 3 (three) anomaly detection tools were carefully selected from the literature (RETA, Tactile Check and Tiger Pro) and analyzed based on requirements documents from different domains, containing more than 85 anomalies. As a result, the tool with the highest coverage is RETA, having correctly identified 85.19% of the existing anomalies. Nevertheless, considering the average between coverage and accuracy, the Tactile Check tool was the one that presented the best result. Despite this, all the tools comparatively analyzed in this work had an unsatisfactory level of coverage and accuracy, falling below 66% and 57% respectively. This is largely due to the words that make up the dictionary not being well calibrated with the anomalies that the tool is willing to detect and the detection technique not considering the context of the description of the requirement of which the anomalous term is inserted.

Keywords: Requirements engineering. Software requirements anomalies. Software quality.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Produtos de *software* estão amplamente presentes nos ambientes profissional e pessoal dos indivíduos (FABBRI; FERRARI; CAMARGO, 2014). Nos últimos anos, o *software* vêm contribuindo para a inovação em diversas áreas da sociedade, à guisa de exemplo, comunicação, educação, economia, comércio, lazer, entre outros (PRESSMAN; MAXIM, 2021). Devido à sua relevância no mundo moderno, há uma constante preocupação com a qualidade dos produtos de *software* desenvolvidos e entregues aos *stakeholders* [1] (MACHADO, 2018).

Um requisito de *software* indica uma capacidade, um atributo, uma qualidade ou uma característica que um *software* deve apresentar para ter valor aos seus *stakeholders*. Em outras palavras, requisitos de *software* podem ser entendidos como uma descrição das propriedades que o *software* deve manifestar para atender às necessidades de algo ou alguém, quando ele estiver finalizado (FERNANDES; MACHADO, 2017). A Engenharia de Requisitos (ER), uma especificidade da Engenharia de *Software*, consiste em descobrir, analisar, documentar e verificar os requisitos para desenvolvimento ou manutenção de um *software* (SOMMERVILLE, 2011). Ela engloba uma série de atividades que, uma vez realizadas, pode resultar em um documento textual com a descrição dos requisitos que o *software* deve contemplar, conhecido como "Documento de Requisitos - DR" (CHITCHYAN et al., 2006).

De acordo com Valente (2020), a ER pode colaborar diretamente para a qualidade do *software*, uma vez que ela buscar alinhar o que está sendo especificado no DR com a real expectativa dos *stakeholders*. Todavia, a elaboração de um DR de qualidade não é uma tarefa simples. No geral, requisitos de *software* são descritos em linguagem natural, o que pode fazer com que contenham subjetividades e inconsistências, comprometendo sua compreensão e interpretação (NASCIMENTO et al., 2018). Isso, por sua vez, pode levar a implementações equivocadas e/ou defeituosas, gerando retrabalho, aumento dos custos de desenvolvimento e desmotivação da equipe, dentre outros prejuízos (FEMMER et al., 2017).

O conceito de "Anomalias de Requisitos de *Software*" (*Software Requirements Smells*) surge no contexto da busca por qualidade em um DR. Anomalias de requisitos[2] são indicadores

[1] Em um projeto de *software*, normalmente, usa-se o termo *stakeholder* para designar as partes interessadas no mesmo, ou seja, os *stakeholders* são aqueles que afetam ou que são afetados pelo *software*, podendo ser pessoas físicas ou organizações (VALENTE, 2020).

[2] Por questão de simplicidade, daqui adiante, o termo "Anomalias de Requisitos de *Software*" será tratado apenas como "Anomalias de Requisitos".

de que há "algo errado" com a descrição de um requisito. Alguns tipos de anomalias, por exemplo, referem-se ao uso de termos comparativos, tais como "melhor desempenho" ou "menor tempo de resposta", na especificação de requisitos do *software* (FEMMER et al., 2017). Esses termos são considerados anomalias, pois trazem subjetividade ao texto do DR, prejudicando sua compreensão e interpretação. Em outras palavras, o significado de "menor tempo de resposta" para uma pessoa pode diferir do de outra, podendo levar a implementações que não satisfazem às reais expectativas dos *stakeholders*.

Diversos trabalhos na literatura têm proposto ferramentas para detecção de anomalias de requisitos. De acordo com um Mapeamento Sistemático da Literatura (MSL)[3] realizado por Nascimento et al. (2018), 41 estudos foram publicados sobre esse assunto no período de 2013 (quando o termo "Anomalias de Requisitos de *Software*" foi introduzido pela primeira vez) até 2018, a partir dos quais 22 ferramentas para detecção de anomalias são propostas. Contudo, são escassos os trabalhos que se propõem a avaliar a efetividade[4] das ferramentas propostas, comparando-as umas com as outras e procurando identificar seus pontos fortes e fracos (NASCIMENTO et al., 2018). Estudos com essa temática são relevantes, pois servem de apoio para: (i) construir ferramentas mais efetivas ou aprimorar a efetividade das preexistentes; e (ii) facilitar a tomada de decisão por parte de pesquisadores e profissionais da indústria quanto à escolha das ferramentas que melhor atendam às suas necessidades.

Neste contexto, esta pesquisa teve como objetivo geral cobrir esse *gap* da literatura, ao realizar uma análise comparativa da efetividade de ferramentas para detecção de anomalias de requisitos, identificando seus pontos fortes e fracos. Para alcançar o objetivo proposto, o seguinte método de pesquisa foi planejado e executado: (i) atualizar o MSL realizado por Nascimento et al. (2018), a fim de identificar as ferramentas mais recentes para detecção de anomalias de requisitos; (ii) selecionar as ferramentas a serem comparativamente analisadas, com base em critérios pré-definidos; (iii) eleger um conjunto de DR com anomalias pré-catalogadas, a serem utilizados durante a análise das ferramentas; (iv) analisar a efetividade das ferramentas selecio-

[3] Um tipo de estudo secundário, que propõe uma ampla revisão de estudos primários, identificando evidências sobre um determinado tópico; pode indicar lacunas nas quais novos estudos primários são necessários ou agrupamentos de evidências que podem levar a novas revisões sistemáticas (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

[4] Neste trabalho, o termo "efetividade" corresponde à cobertura e à precisão apresentadas por uma ferramenta de detecção de anomalias de requisitos; mais detalhes são descritos na Seção 4.2.

nadas, utilizando as métricas “cobertura” e “precisão”; e (v) descrever os pontos fortes e fracos de cada ferramenta, a partir dos resultados da análise realizada.

Ao todo, 3 (três) ferramentas foram selecionadas para a análise comparativa, sendo elas: *RETA* (ARORA et al., 2015), *Tactile Check* (WILMINK; BOCKISCH, 2017) e *Tiger Pro* (ARENDSE; LUCASSEN, 2016). Todas foram submetidas a 3 (três) DR de diferentes domínios (sistema de biblioteca integrado, sistema de casa inteligente, sistema integrado para segurança pública), contendo mais de 85 anomalias de diferentes tipos. Como resultado, a ferramenta que teve a maior cobertura foi a *Tactile Check*, tendo identificado 65,63% das anomalias existentes. A ferramenta *RETA* obteve uma cobertura de 65,43%, contudo, ela representou o menor índice de precisão, uma vez que muitos falsos positivos foram indicados por ela. Isso pode ter acontecido pelo fato de a ferramenta não considerar o contexto do qual o termo anômalo esteja inserido. O maior valor de precisão foi encontrado na ferramenta *Tiger Pro*, entretanto, ela obteve o menor índice de cobertura. Ou seja, trata-se de uma ferramenta que identifica poucas anomalias existentes, mas que, em contrapartida, gera menos falsos positivos. Provavelmente, isso se deve à qualidade do dicionário utilizado pela ferramenta para detecção de anomalias (ARENDSE; LUCASSEN, 2016). Com base na média entre cobertura e precisão, a ferramenta *Tactile Check* foi a que teve o melhor resultado e, portanto, a que se destacou nas avaliações entre as demais ferramentas. Além disso, no que tange à facilidade de preparação do ambiente, instalação, uso e apresentação dos resultados essa ferramenta se destacou entre as demais ferramentas.

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2, é apresentada uma breve fundamentação teórica sobre anomalias de requisitos e sua relação com a qualidade de *software*, bem como os trabalhos relacionados a esta pesquisa. No Capítulo 3, é apresentada a seleção das ferramentas e artefatos utilizados durante a análise comparativa. No Capítulo 4, apresenta-se a análise comparativa das ferramentas selecionadas, destacando-se seus resultados e a discussão sobre os mesmos. Por fim, no Capítulo 5, é apresentada considerações finais, limitações desta pesquisa e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo descreve a fundamentação teórica e os trabalhos relacionados aos principais conceitos com os quais se trabalhou nesta pesquisa. Desse modo, as informações estão organizadas da seguinte forma: nas Seções 2.1 e 2.2, são apresentados, respectivamente, conceitos sobre anomalias de requisitos de *software* e ferramentas para detecção de anomalias. Na Seção 2.3, são apresentadas as contribuições provenientes de trabalhos correlatos, bem como uma discussão acerca das características em que esses trabalhos divergem desta pesquisa.

2.1 Anomalias de requisitos de *software*

Para Sommerville (2011), requisitos de *software* expressam os atributos e as funcionalidades necessárias que o *software* deve possuir para a execução de tarefas que satisfaçam os objetivos dos *stakeholders*. A abordagem mais comum para redigir requisitos de *software* é por meio do uso de Linguagem Natural (LN) (ARENDSE; LUCASSEN, 2016; ZHAO et al., 2020). A utilização de LN facilita a comunicação dos requisitos com os *stakeholders*, por ser de fácil compreensão por pessoas não ligadas à área de desenvolvimento de *software* (KROTH, 2012; PALDêS et al., 2016). Porém, problemas como ambiguidade e inconsistência são comuns quando LN é utilizada, o que pode possibilitar múltiplas interpretações e omissões de características importantes que o *software* deveria possuir (PALDêS et al., 2016; KIYAVITSKAYA et al., 2008). Esses aspectos negativos, quando encontrados na especificação de requisitos, são reconhecidos pelo conceito de "Anomalias de Requisitos de *Software*" (*Software Requirements Smells*) (FEMMER, 2013).

O termo "*smell*", no domínio da Engenharia de *Software* (ES), indica uma solução inadequada para um problema recorrente. Esse termo foi cunhado, primeiramente, por Fowler (1999), se referindo a trechos de código-fonte de baixa qualidade (*code smell*), que precisavam ser refatorados. Desde então, essa metáfora vem sendo atribuída a outros artefatos de *software*, como por exemplo, em código de teste (*test smell*) (DEURSEN et al., 2001) e em descrição de requisitos de *software* (*requirements smells*) (FEMMER, 2013). As anomalias de requisitos surgem durante o processo de elaboração e manutenção do Documento de Requisitos (DR) e possuem as seguintes características em comum (NASCIMENTO et al., 2018; PALDêS et al., 2016; FEMMER et al., 2017):

a) **indicam violações de qualidade do `DR`**: aqui, entende-se a qualidade do `DR` em termos dos seus (potenciais) efeitos nas atividades subsequentes do ciclo de vida do *software*, que dependem do `DR`;

b) **não levam necessariamente a um defeito e, portanto, devem ser julgadas pelo contexto**: se a descoberta de uma anomalia é ou não um problema, isso deve ser decidido individualmente, com base no contexto;

c) **possuem uma localização concreta, por exemplo, uma palavra ou uma sentença**: anomalias de requisitos sempre fornecem um ponteiro para um determinado local onde deve-se inspecionar;

d) **possuem um mecanismo de detecção concreto**: devido à sua natureza concreta, é possível propor estratégias para sua detecção, entretanto, essas estratégias podem ser mais ou menos precisas, dependendo do contexto e do tipo de anomalia a ser detectada.

A literatura mais recente sobre o assunto descreve um catálogo com um total de 11 (onze) anomalias de requisitos (NASCIMENTO; GUIMARãES; LUCENA, 2021). O Quadro 2.1 apresenta cada anomalia com seu nome, uma breve descrição e um exemplo prático sobre ela.

2.2 Ferramentas para detecção de anomalias de requisitos

Especificações de requisitos imprecisas podem levar a disputas entre *stakeholders*, quando são mal interpretadas (HABIB; WAGNER; GRAZIOTIN, 2021). Além disso, elas custam caro ao projeto de *software*, tanto em termos de tempo quanto de orçamento (IQBAL; ELAHIDOOST; LUCIO, 2018). Por esses motivos, é fundamental que anomalias de requisitos sejam identificadas e corrigidas ainda no início do ciclo de desenvolvimento do *software*, a fim de se evitar prejuízos maiores. Em outras palavras, quanto antes os indícios de má qualidade na especificação dos requisitos forem identificados e tratados, menor será o custo de produção e manutenção do *software* e maiores serão as chances de o projeto ser bem-sucedido (PAGLIUSO et al., 2020).

Diversas ferramentas têm sido propostas para detecção de anomalias de requisitos (NASCIMENTO et al., 2018). Esta seção apresenta duas destas ferramentas, a saber: *Smella* e *RETA*.

Quadro 2.1 – Descrição das anomalias de requisitos catalogadas por Nascimento, Guimarães e Lucena (2021)

Anomalia	**Descrição**
Advérbios e adjetivos ambíguos	Refere-se a advérbios e adjetivos que são inespecíficos por natureza, tais como: "quase sempre", "muito pouco", "significativo" e "mínimo". **Exemplo**: Se a qualidade (...) for muito baixa, uma falha deve ser gravada na memória de erros.
Lacunas ou brechas	Refere-se a palavras que possibilitam aos *stakeholders* ignorarem especificações, tais como: "se possível", "conforme apropriado" e "se aplicável". **Exemplo**: Na medida do possível, as entradas devem ser verificadas.
Termos abertos e não-verificáveis	Refere-se a termos difíceis de verificar, pois oferecem várias escolhas de possibilidades, por exemplo, aos desenvolvedores. **Exemplo**: O sistema deve oferecer ao usuário todas as formas de pagamento necessárias.
Superlativos	Refere-se a advérbios e adjetivos que expressam uma relação do sistema com todos os outros sistemas, tais como: "melhor desempenho" e "menor tempo de resposta". **Exemplo**: O sistema deve fornecer o sinal ao cliente na melhor resolução.
Comparativos	Refere-se a advérbios e adjetivos que expressam uma relação do sistema com outro sistema ou situação anterior, tais como "melhor que" e "maior que". **Exemplo**: A nova interface de usuário deve conter menos problemas de usabilidade que a anterior.
Palavras negativas	Refere-se a declarações de funcionalidades que o sistema não deve fornecer, o que pode levar à falta de explicação sobre o comportamento real do sistema em tais casos. **Exemplo**: O sistema não deve desconectar usuários devido a *timeouts*.
Pronomes vagos	Refere-se a relações pouco claras de um pronome. **Exemplo**: O sistema deve implementar serviços para aplicativos, que deve se comunicar com os aplicativos do controlador implantados em outros controladores.
Referências incompletas	São referências que o leitor não consegue encontrar no texto do `DR`. **Exemplo**: [1] "Artigo desconhecido". Pedro Miller.
Linguagem subjetiva	São palavras cuja semântica não é objetiva, tais como: "amigável", "fácil de usar" e "econômico". **Exemplo**: A nova interface de usuário do sistema deve ser amigável.
Voz passiva	Refere-se a situações nas quais não está claro o ator que está desempenhando uma determinada ação no sistema. **Exemplo**: O sistema deve exibir uma mensagem de sucesso assim que o registro for salvo.
Funcionalidade duplicada	Refere-se à repetição de interações entre sistemas e atores em vários requisitos do sistema. **Exemplo**: O sistema verifica se o usuário está logado antes de prosseguir.

Fonte: adaptado de Nascimento, Guimarães e Lucena (2021)

A ferramenta *Smella* é um protótipo para dar suporte à detecção automatizada de anomalia de requisitos. Ela é implementada sobre o *kit* de ferramentas de análise de qualidade de *software ConQAT*[1], uma plataforma para análise de código-fonte, que foi estendida pelos auto-

[1] http://www.conqat.org

res com os recursos de Processamento de Linguagem Natural (PLN) necessários (FEMMER et al., 2017).

O processo de detecção da *Smella* usa artefatos de requisitos em vários formatos (*MS Word*, *MS Excel*, *PDF*, texto simples, valores separados por vírgula) e consiste em quatro etapas, conforme pode ser visto na Figura 2.1 (FEMMER et al., 2017):

a) **(*Parsing*) Análise dos artefatos de requisitos em itens únicos (por exemplo, seções ou linhas)**. A ferramenta é capaz de processar vários formatos de arquivo e, dependendo do formato, os arquivos são analisados de maneiras diferentes;

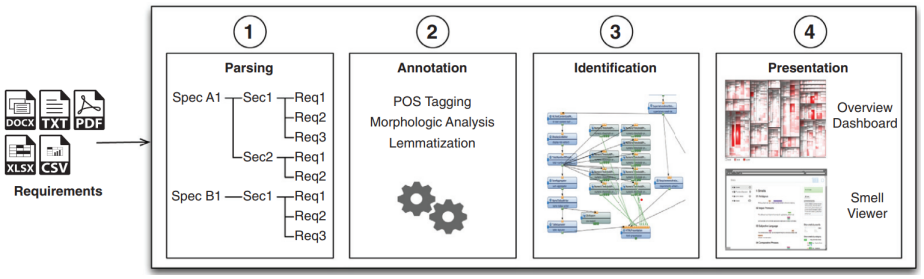
b) **(*Annotation*) Anotação dos requisitos com meta-informações**. Para as etapas de anotação e detecção de anomalias, são empregadas três técnicas de Processamento de Linguagem Natural (PLN):

   - **Dicionário e lematização**: foram utilizados dicionários de palavras, baseados na norma (ISO, 2011) e também em um trabalho anterior dos autores desta ferramenta (FEMMER et al., 2014). Além disso, foi aplicada a lematização para essas palavras, que é uma técnica de normalização que reproduz a forma original de uma palavra. Por exemplo, se um *lemmatizador* for aplicado às palavras "foi", "é" ou "são", ele retornará para todas as três palavras, a palavra "ser";
   - ***POS tagging***: a técnica *POS tagging* determina o papel e a função de cada palavra em uma frase escrita em linguagem natural. A saída é uma *tag* para cada palavra, indicando, por exemplo, se uma palavra é um adjetivo, uma partícula ou um pronome possessivo;
   - **Análise morfológica**: com base no resultado da técnica *POS tagging* (etapa anterior), é feita uma análise mais detalhada do texto para determinar a inflexão de uma palavra, isto é, determinar o tempo de um verbo ou a comparação de um adjetivo.

c) **(*Identification*) Identificação de anomalias nos requisitos, com base nas anotações**. Esta etapa realmente encontra as partes de um artefato que exibem anomalias de requisitos. Dependendo da anomalia, são usadas diferentes técnicas, conforme mostrado no Quadro 3.2. Vale ressaltar que a anomalia do tipo "Referências Incompletas" (*Incomplete References*) não fez parte da ferramenta *Smella*;

d) (***Presentation***) **Apresentação em uma visualização legível para humanos das anomalias identificadas**. Por fim, as anomalias encontradas são apresentadas por meio do protótipo chamado de *Smella* (*Smell Analysis*), como pode ser visto na Figura 2.2.

Figura 2.1 – Processo de funcionamento da ferramenta *Smella*



Fonte: Femmer et al. (2017)

Quadro 2.2 – Técnicas para detecção de anomalias utilizadas pela ferramenta *Smella*

Nome da Anomalia	**Mecanismo de detecção**
Linguagem subjetiva	Dicionário
Advérbios e Adjetivos Ambíguos	Dicionário
Lacunas ou brechas	Dicionário
Termos abertos e não verificáveis	Dicionário
Superlativos	Análise morfológica (*POS tagging*)
Comparativos	Análise morfológica
Palavras Negativas	Análise morfológica e dicionário
Pronomes Vagos	Análise morfológica: Substituindo pronomes.
Referências Incompletas	Não está no escopo deste estudo

Fonte: Adaptado de Femmer et al. (2017)

Smella é uma ferramenta baseada na *web* que permite visualizar, revisar e listar descobertas de anomalias de requisitos. A apresentação das anomalias por parte desta ferramenta se dá em um estilo de corretor ortográfico. Isso segue a ideia de que anomalias são apenas indicações, as quais devem ser avaliadas em seu contexto.

Como será observado na Seção 3.2, essa ferramenta não foi selecionada para análise comparativa nesta pesquisa, pois o projeto foi descontinuado. Entretanto, ela foi selecionada para ser detalhada neste capítulo, pelo fato de suas características corresponderem amplamente ao funcionamento geral das ferramentas para detecção de anomalias de requisitos de *software*.

Figura 2.2 – Tela com amostras de anomalias detectadas pela ferramenta *Smella*

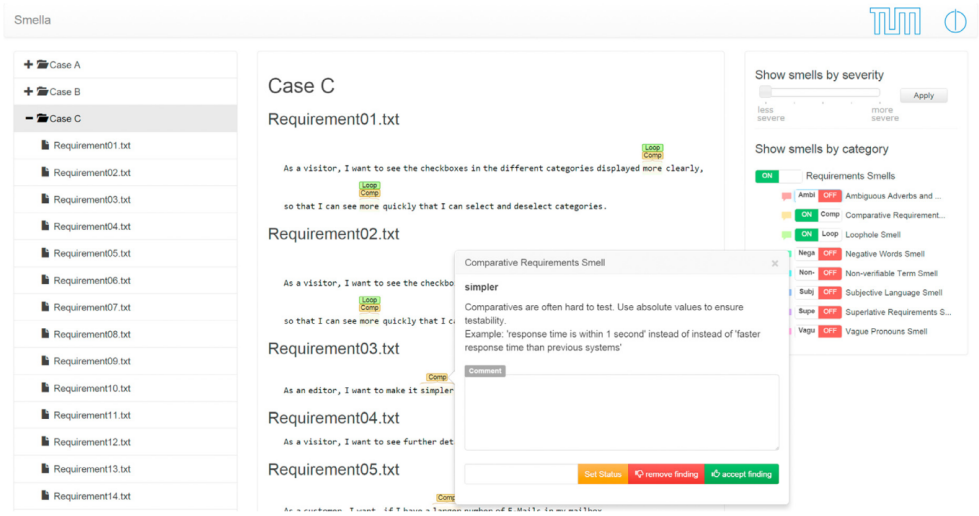


Fig. 3. A sample output from the smell detection tool (detailed artifact view) with some smells disabled and some findings blacklisted.

Fonte: Femmer et al. (2017)

Já o trabalho realizado por Arora et al. (2015) propôs uma abordagem para automatizar a verificação de conformidade de DR com relação a modelos pré-determinados. A ferramenta computacional *RETA*[2] (*REquirements Template Analyzer*) permite que os analistas verifiquem automaticamente a conformidade de um DR com vários tipos de modelos e obtenham diagnósticos sobre construções sintáticas potencialmente problemáticas em declarações de requisitos (ARORA et al., 2015).

A principal tecnologia de Processamento de Linguagem Natural (PLN) utilizada pela ferramenta *RETA* é o *text chunking*, que consiste em decompor uma frase em segmentos não sobrepostos. Por exemplo, um agrupamento da declaração do requisito R na Figura 2.3(a) produziria os segmentos mostrados na Figura 2.3(b), em que, NP (*Noun Phrase*) corresponde à uma Frase Nominal[3], VP (*Noun Phrase*) à uma Frase Verbal[4] e PP (*Prepositional Phrase*) à uma Locução Prepositiva [5]. Essa representação permite a definição de regras de correspondência para verificar a conformidade com diversos tipos de modelos.

[2] https://sites.google.com/site/retanlp/

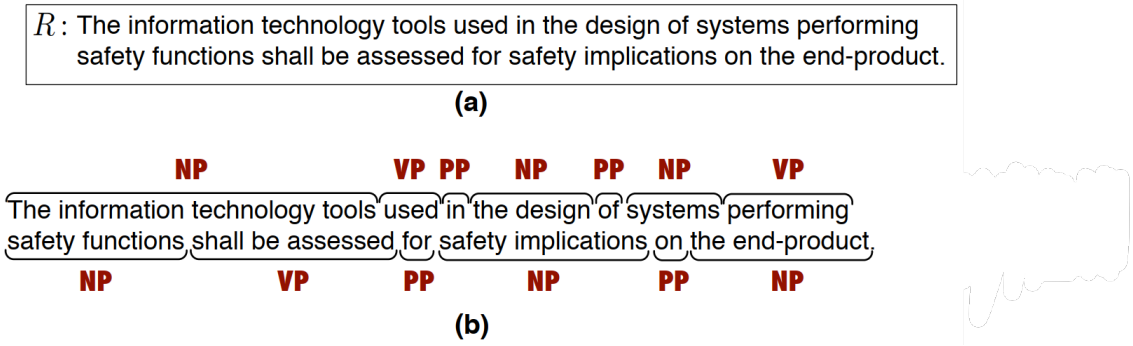
[3] Uma "Frase Nominal" é um segmento que pode ser sujeito ou objeto de um verbo (ARORA et al., 2015).

[4] Uma "Frase Verbal" é um segmento que contém um verbo com qualquer modal, auxiliar e modificador associado (ARORA et al., 2015).

[5] As locuções prepositivas são duas ou mais palavras que exercem papel de preposição, por exemplo, "apesar de", "a respeito de", "para com", "além de", entre outros (PERINI, 2017).

Para essa verificação de correspondência, é utilizada a *JAPE* (*Java Annotation Patterns Engine*), uma linguagem de correspondência de padrões baseada em expressões regulares, disponível como parte da plataforma de PLN chamada *GATE*[6]. Em outras palavras, as regras para verificar a conformidade com modelos são fornecidas como *scripts* escritos na linguagem *JAPE*, dentro da plataforma *GATE*.

Figura 2.3 – Exemplo de segmentação de uma declaração de requisito



Fonte: Arora et al. (2015)

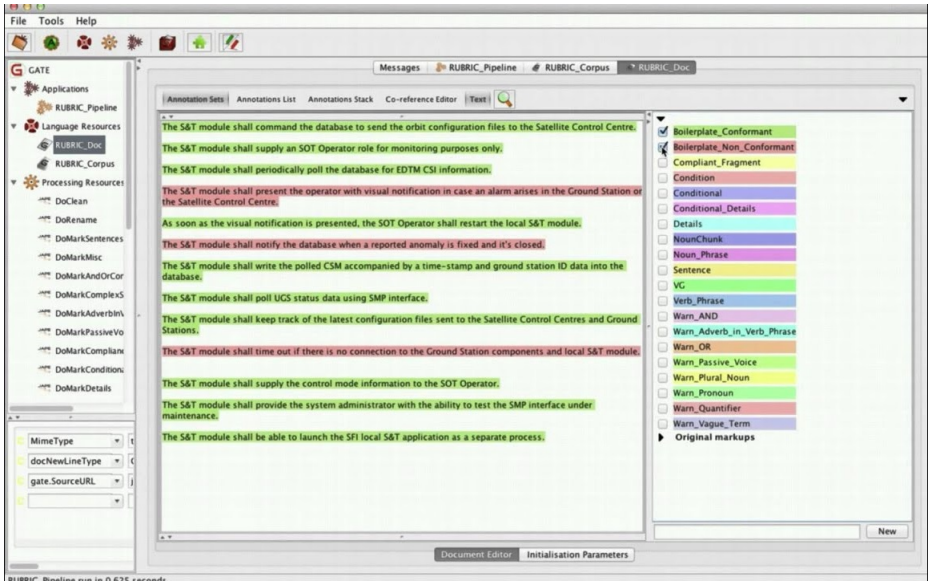
Considerando o contexto de detecção de anomalias de requisitos, após a segmentação do texto de um requisito, a ferramenta *RETA* realiza a verificação nessas segmentações, com o objetivo de detectar anomalias concernentes à ambiguidade, isto é, lacunas/brechas, termos abertos e não verificáveis, entre outros. A ferramenta *RETA* usa a interface de usuário da plataforma *GATE* para exibir diagnósticos sobre a não conformidade do modelo e desvios das melhores práticas de redação de requisitos. Uma visão geral da interface da *GATE*, após a execução da ferramenta *RETA*, é mostrada na Figura 2.4.

2.3 Trabalhos relacionados

Segundo MACHADO (2018), as necessidades e o nível de exigências dos *stakeholders* evoluem a cada dia, contribuindo para o aumento da complexidade dos produtos de *software*. Com o desenvolvimento de produtos de *software* cada vez mais complexos, a especificação de requisitos é considerada como um dos principais fatores críticos de sucesso de um projeto (VALENTE, 2020). Com isso, vários estudos têm sido realizados com o objetivo de propor novas abordagens para garantir a qualidade dos documentos de requisitos, sobretudo, a respeito de anomalias de requisitos de *software*.

[6] https://gate.ac.uk/

Figura 2.4 – Resultado da detecção de anomalias na ferramenta *RETA*



Fonte: Arora et al. (2015)

Um Mapeamento Sistemático da Literatura (MSL) foi conduzido por Nascimento et al. (2018), em busca de trabalhos que relatassem sobre abordagens e técnicas para detecção de anomalias de requisitos de *software*. Os autores do MSL consideraram as publicações entre Janeiro de 2013 até Março de 2018. Para guiar o processo de busca e mapeamento dos estudos, foram estipuladas 4 questões de pesquisa (NASCIMENTO et al., 2018):

a) **Qual definição está sendo utilizada para *Requirements Smells*?** O objetivo foi verificar o consenso dos pesquisadores quanto à definição de anomalias de requisitos. Como resultado, foi observado que existe um consenso sobre a definição do termo *Requirement Smell* entre os diversos estudos e que, de um modo geral, são eles considerados como indicadores de má qualidade na especificação de requisitos de *software* em linguagem natural;

b) **Existe algum catálogo sobre *Requirements Smells* sendo utilizado?** O objetivo foi verificar se existia um ou mais catálogos que poderiam ser aplicados para detecção de *Requirements Smells*. Como resultado, observou-se que algumas pesquisas estão investigando problemas de má qualidade na especificação de requisitos e fazendo o uso do catálogo de *Requirements Smells* proposto por Femmer et al. (2014) e apresentado em Femmer et al. (2017). De modo geral, a maioria dos estudos investiga problemas relacionados à ambiguidade, termos não compreensíveis e sentenças negativas;

c) **São propostos métodos ou ferramentas para detecção de *Requirements Smells*?** O objetivo foi investigar se existia suporte ferramental para auxiliar na detecção de *Requirements Smells*. Como resultado, verificou-se que apoios ferramentais estão sendo propostos para detecção automatizada de anomalias de requisitos. Além disso, foram encontrados trabalhos que fazem referências às ferramentas utilizadas para detecção de anomalias de requisitos propostas na literatura;

d) **Como *Requirements Smells* está sendo aplicado na prática?** O objetivo foi verificar se as pesquisas sobre *Requirements Smells* estavam sendo aplicadas na área de Engenharia de Requisitos e/ou demais áreas da Engenharia de *Software*. Como resultado, observou-se que a definição e suporte ferramental para detecção de anomalias de requisitos vem sendo aplicados na prática em análise de especificações de requisitos de projetos reais na indústria.

A pesquisa de Nascimento et al. (2018) assemelha-se com este trabalho no que diz respeito ao interesse em relatar aspectos de qualidade de *software* com foco em anomalia de requisitos. Porém, enquanto o trabalho de Nascimento et al. (2018) procura reunir e categorizar abordagens e ferramentas disponíveis na literatura para detecção de anomalia de requisitos, este trabalho busca comparar ferramentas computacionais para detecção de anomalias entre si, por meio de um experimento controlado.

O trabalho de Femmer et al. (2017) fornece uma discussão rica e uma definição precisa sobre anomalias de requisitos de *software*. Além disso, foi proposta uma ferramenta computacional para detecção automática das anomalias relacionadas (apresentada na Seção 2.2). Em suma, o trabalho se dispõe a:

a) Definir o conceito de *Requirements smells* e explicar como ele foi derivado da norma ISO 29148;

b) Apresentar a ferramenta computacional *Smella*; e

c) Relatar o estudo empírico construído para avaliar a ferramenta proposta.

Esta pesquisa recorreu ao trabalho de Femmer et al. (2017) para fundamentação teórica dos conceitos acerca de anomalias de requisitos. O resultado dessa conceituação pode ser vista no Quadro 2.1 (Seção 2.1). Contudo, esta pesquisa se difere do trabalho de Femmer et al. (2017) e dos demais que seguem a mesma linha de pesquisa, por não buscar propor uma abordagem nova, mas sim avaliar as já existentes, com o intuito de conhecer a real efetividades das mesmas.

No trabalho de Arendse e Lucassen (2016), os autores realizaram a comparação de desempenho de três ferramentas para detecção de defeitos e desvios em documentos de requisitos escritos em LN. Os autores focaram em dois tipos de anomalias específicas: ambiguidade e atomicidade [7]. O método utilizado para realização da pesquisa foi dividido em três etapas:

a) **Compilar uma lista de ferramentas elegíveis**. Os autores conduziram um estudo da literatura utilizando a técnica *snowballing*, que levou a 50 artigos descrevendo ferramentas que aplicam PLN à análise de requisitos. Em seguida, aplicaram os seguintes critérios para seleção das ferramentas a serem analisadas:

   - Fornece informações técnicas suficientes para entender o funcionamento interno da ferramenta;
   - Aceita requisitos escritos em LN como entrada;
   - É totalmente automatizada, isto é, não requer pré-processamento manual, como a criação de *links* de rastreabilidade;
   - Encontra-se disponível para *download* na Internet ou por meio de contato direto com o(s) autor(es).

   Após a aplicação dos critérios de inclusão, 3 (três) ferramentas foram selecionadas para a análise comparativa: *Qualicen* [8], *RQA* [9] e *TIGER-PRO* [10].

b) **Padrão de qualidade**. Definiram um padrão de qualidade ao qual um requisito deve estar em conformidade; este padrão consista em 16 critérios de violação relatados por (BERRY; KAMSTIES; KRIEGER, 2003);

c) **Avaliação comparativa**. Para testar a efetividade das ferramentas, foi utilizada uma coleção de requisitos de conjuntos de dados fornecidos pelos desenvolvedores de todas as três ferramentas e também quatro conjuntos de requisitos do mundo real. As duas medidas principais utilizadas pelos autores para comparação das ferramentas foram: precisão e cobertura (mais detalhes sobre essas métricas são apresentadas na Seção 4.2).

À partir do resultado da análise comparativa entre as ferramentas supra-citadas, os autores, realizaram um projeto de uma ferramenta com base nos pontos fortes encontrados nas

[7] Um requisito com ambiguidade é aquele que pode ter múltiplas interpretações; já um requisito com problema de atomicidade é aquele que possui mais de uma função (ARENDSE; LUCASSEN, 2016).
[8] https://www.qualicen.de/
[9] https://www.reusecompany.com/rqa-quality-studio
[10] https://tiger-pro.soft32.com/

ferramentas avaliadas. Ao todo, foram definidos 16 pontos de melhorias. Em teoria, uma ferramenta de "próxima geração", que implemente esses 16 pontos, deve obter pelo menos 92% de cobertura e 68% de precisão para detecção de anomalias relativas à atomicidade e 76% de cobertura e 77% de precisão para detecção de anomalias relativas à ambiguidade (ARENDSE; LUCASSEN, 2016).

O trabalho de Arendse e Lucassen (2016) foi utilizado para nortear a metodologia de análise comparativa que esta pesquisa se dispõe a realizar. Como diferencial desta pesquisa, tem-se que foi realizada a verificação de outros tipos anomalias (além da ambiguidade e atomicidade) e de outras ferramentas, bem como a utilização de conjunto de `DR` retirados de um repositório aberto e publicamente disponível, o que permite a replicabilidade desta pesquisa.

3 PROCEDIMENTOS PRÉ-ANÁLISE COMPARATIVA

Os procedimentos pré-análise comparativa constituem-se na seleção das ferramentas e artefatos utilizados durante a análise apresentada no prórixmo capítulo. A Figura 3.1 exibe, em destaque, as etapas executadas com esse intuito. A Seção 3.1 apresenta a etapa "Atualizar MSL", que visa descrever o resultado da atualização do Mapeamento Sistemática da Literatura (MSL) de Nascimento et al. (2018), realizada pelo autor dessa dissertação. A Seção 3.2 apresenta a etapa "Selecionar abordagens para análise", que descreve os procedimentos utilizados para seleção das ferramentas a serem comparativamente analisadas. A Seção 3.3 apresenta a etapa "Selecionar conjuntos de requisitos", que dispõe-se a relatar sobre a obtenção dos Documentos de Requisitos (DR) utilizados durante a análise das ferramentas. As demais etapas, isto é, "Identificar pontos fortes e fracos" e "Realizar análise comparativa" serão apresentadas no Capítulo 4.

Figura 3.1 – Etapas iniciais do método de pesquisa

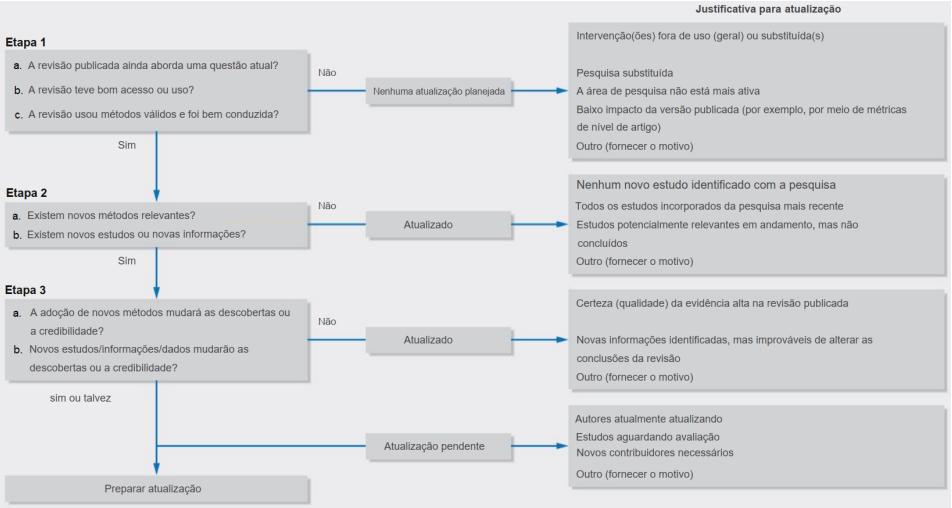
Fonte: Do Autor (2023)

3.1 Etapa 1 - Atualizar Mapeamento Sistemático da Literatura

Ao iniciar a pesquisa por ferramentas para compor a análise comparativa proposta neste trabalho, a publicação de Nascimento et al. (2018) ganhou evidência por se tratar de um MSL em que foram reunidos 41 estudos científicos sobre detecção de anomalias de requisitos, dos quais 20 propuseram, citaram ou utilizaram alguma ferramenta computacional. Entretanto, as ferramentas catalogadas no trabalho de Nascimento et al. (2018) foram publicadas entre os anos de 2013 e 2018, logo, cogitou-se a necessidade da atualização desse MSL.

A necessidade de atualizar ou não o MSL de Nascimento et al. (2018) foi norteada pelo *framework* de decisão de atualização de Revisão Sistemática da Literatura (RSL) proposto por Garner et al. (2016)[1]. A estrutura de decisão proposta pelo *framework* inclui três etapas (Figura 3.2), a serem aplicadas sequencialmente:

Figura 3.2 – *Framework* de decisão para atualização de uma RSL



Fonte: GARNER et al. (2016, tradução nossa)

a) A **Etapa 1** se concentra em avaliar o quão atual é a RSL, observando a relevância de seu tópico para pesquisa e prática. Isso inclui se a RSL publicada teve impacto na pesquisa e/ou na prática, usando métricas como número de citações, e se a RSL foi realizada adequadamente, usando uma metodologia sólida (por exemplo, critérios de inclusão/exclusão claros e avaliação de qualidade dos estudos primários). Conforme mostra a Figura 3.2, se a resposta a pelo menos uma das perguntas da Etapa 1 for "NÃO", então não há necessidade de se atualizar a RSL. Observa-se que, no caso particular em que as perguntas (a) e (b) são "SIM" e a pergunta (c) "NÃO", uma atualização da RSL original não é garantida; no entanto, a RSL original deve, de fato, ser refeita, visto que o método originalmente usado não é considerado válido. Ao ser refeita, também pode ampliar sua busca para incluir estudos primários mais atualizados. Após aplicar a Etapa 1 ao MSL de Nascimento et al. (2018), obteve-se "SIM" para todas as questões;

b) A **Etapa 2** se concentra em identificar se há novos métodos propostos e/ou novos estudos publicados após a publicação da RSL. Exemplos de novos métodos podem ser novas

[1] Embora esse *framework* tenha sido desenvolvido com o foco em revisões sistemáticas da literatura, de acordo com os próprios autores, ele também pode ser utilizado para MSL.

formas de buscar evidências e fazer a síntese das evidências, a inclusão de outros tipos de *design* de estudo (por exemplo, pesquisa qualitativa), novas técnicas estatísticas ou melhorias nas técnicas existentes usadas para meta-análise, entre outros. A pergunta (b) refere-se especificamente se novos estudos adicionais foram encontrados. Nesta etapa, desde que pelo menos uma questão tenha resposta afirmativa, pode-se prosseguir para a Etapa 3. As possíveis respostas são "SIM" e "NÃO". O MSL de Nascimento et al. (2018) retornou "NÃO" para a pergunta (a), porém, retornou "SIM" para a pergunta (b);

c) Por último, a **Etapa 3** visa avaliar se a adoção de novos métodos e/ou novos estudos pode afetar as conclusões, quando comparadas com as conclusões da RSL original. Percebe-se que em relação ao item (3.b), há uma gama de diferentes aspectos a serem considerados. Por exemplo, as conclusões podem mudar devido à adição de novos estudos primários à RSL original. Esses seriam identificados observando os resultados das pesquisas *snowballing* na base de dados, com base na leitura de títulos e resumos. Outros exemplos seriam alterações feitas nos critérios de inclusão ou *checklist* de qualidade que, quando reaplicados aos estudos na RSL original, levam a mudanças nas conclusões. Aplica-se aqui o mesmo que na Etapa 2, ou seja, desde que pelo menos uma questão tenha resposta afirmativa, recomenda-se seguir para o próximo passo. As respostas possíveis são "SIM/TALVEZ" e "NÃO". O MSL de Nascimento et al. (2018) retornou "SIM" para todas as questões da Etapa 3.

Ao final da Etapa 3, **ficou visível a necessidade de se atualizar o MSL** de Nascimento et al. (2018). Os parâmetros do protocolo, utilizados para essa atualização, tais como questões de pesquisa, bibliotecas digitais, *string* de busca, critérios de inclusão e exclusão, foram todos os mesmos estipulados por Nascimento et al. (2018). A única diferença foi concernente ao período da data de publicação das pesquisas, considerando-se de Janeiro de 2013 até Maio de 2023. Com a atualização realizada, foi possível encontrar mais 8 (oito) estudos publicados à partir de 2018, sendo 1 advindo da biblioteca digital ACM, 2 da IEEE e 5 da *ScienceDirect*. Desses, apenas 3 delas propuseram ferramentas computacionais para detecção de anomalias de requisitos, sendo os artigos de ID 21, 22 e 23 do Quadro 3.2.

Portanto, após a atualização do MSL de Nascimento et al. (2018), o rol de estudos que utilizaram, citaram ou propuseram ferramentas computacionais para detecção de anomalias de requisitos subiu para 23. O Quadro 3.2 apresenta a relação desses artigos.

Quadro 3.1 – Estudos que utilizam e/ou propõem ferramentas para detecção de anomalias de requisitos (continua)

ID	Título	Referência
1	*Automated Checking of Conformance to Requirements Templates using Natural Language Processing*	(ARORA et al., 2015)
2	*Flexible Ambiguity Resolution and Incompleteness Detection in Requirements Descriptions via an Indicator-Based Configuration of Text Analysis Pipelines*	(BäUMER; GEIERHOS, 2018)
3	*Rapid Requirements Checks with Requirements Smells: Two Case Studies*	(FEMMER et al., 2014)
4	*Rapid quality assurance with Requirements Smells*	(FEMMER et al., 2017)
5	*Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain*	(ROSADINI et al., 2017)
6	*On the Ability of Lightweight Checks to detect Ambiguity in Requirements Documentation*	(WILMINK; BOCKISCH, 2017)
7	*Toward tool mashups comparing and combining NLP RE Tools*	(ARENDSE; LUCASSEN, 2016)
8	*Initial Investigation on the influence of requirement smells on test case design*	(BEER et al., 2017)
9	*Automated Extraction and Clustering of Requirements Glossary Terms*	(ARORA et al., 2017)
10	*Using Argumentation to Explain Ambiguity in Requirements Elicitation Interviews*	(ELRAKAIBY et al., 2017)
11	*Automatic Requirements Reviews - Potentials, Limitations and Practical Tool Support*	(FEMMER, 2017)
12	*Quality Assurance of Requirements Artifacts in Practice: A Case Study and a Process Proposal*	(FEMMER et al., 2016)
13	*Detecting Domain-Specific Ambiguities: An NLP Approach Based on Wikipedia Crawling and Word Embeddings*	(FERRARI; DONATI; GNESI, 2017)
14	*Interview Review: Detecting Latent Ambiguities to Improve the Requirements Elicitation Process*	(FERRARI et al., 2017)
15	*Natural Language Requirements Processing: From Research to Practice*	(FERRARI, 2018)
16	*Detecting requirements defects with NLP patterns: an industrial experience in the railway domain*	(FERRARI et al., 2018)
17	*PURE: A Dataset of Public Requirements Documents*	(FERRARI; SPAGNOLO; GNESI, 2017)
18	*Using human error information for error prevention*	(HU et al., 2018)
19	*Improving agile requirements: the Quality User Story framework and tool*	(LUCASSEN et al., 2016)

Quadro 3.2 – Estudos que utilizam e/ou propõem ferramentas para detecção de anomalias de requisitos (conclusão)

ID	Título	Referência
20	*Detecting requirements defects with NLP patterns*	(FERRARI et al., 2018)
21	*Quality Requirements and the Requirements Quality*	(CALAZANS et al., 2019)
22	*Ambiguous requirements A semi-automated approach to identify and clarify ambiguity in large-scale projects*	(ASADABADI et al., 2020)
23	*Hidden fuzzy information Requirement specification and measurement of project provider performance using the best worst method*	(ASADABADI et al., 2019)

Fonte: Do Autor (2023)

3.2 Etapa 2 - Selecionar abordagens para análise

A partir da lista de estudos do Quadro 3.2, foi realizada a seleção das ferramentas para detecção de anomalias de requisitos a serem comparativamente analisadas nesta pesquisa. Para isso, os critérios propostos por Arendse e Lucassen (2016) foram utilizados, a saber:

a) **Automatização:** optou-se por analisar ferramentas totalmente automatizadas, buscando evitar que o conhecimento prévio do pesquisador em "manusear" tais ferramentas influenciasse nos resultados. Também foram descartados os trabalhos que dispunham apenas da abordagem para detecção de anomalias, sem um apoio de uma ferramenta computacional, pelo fato da baixa aplicabilidade no mercado e da inviabilidade de se desenvolver uma ferramenta computacional para essas abordagens, considerando o limite de prazo e de escopo desta pesquisa;

b) **Disponibilidade:** a ferramenta computacional deve estar disponível para *download* gratuitamente ou deve possuir uma licença gratuita para testes, caso seja uma ferramenta comercial;

c) **Documentação:** o artigo (ou página *Web*) que descreve a ferramenta computacional deve possuir informação clara e compreensível para sua instalação e uso. É importante que a ferramenta apresente documentação sobre suas funcionalidades, limitações e forma de operação, para que não seja penalizada durante a análise comparativa, devido ao manuseio inadequado de seus recursos; e

d) **Formato de especificação dos requisitos:** o formato principal utilizado na construção de requisitos é a Linguagem Natural LN (PALDêS et al., 2016). Por esse motivo, a ferramenta computacional deve trabalhar com requisitos especificados nesse formato.

Dos 23 artigos que propuseram, utilizaram ou citaram alguma ferramenta para detecção de anomalias, foi possível recuperar 3 (três) ferramentas, as quais foram analisadas com base nos critérios descritos acima. O resultado dessa análise é apresentado na Tabela 3.1, na qual, a primeira coluna (ID) representa o número de identificação da ferramenta, a segunda coluna (Ferramenta) representa seu nome, a terceira coluna (Referência - Art. ID) representa o número de identificação do artigo que faz menção à ferramenta (ver Tabela **??**). A quarta coluna até a sétima representam os critérios explicitados anterior, isto é, disponibilidade (C1), automatização (C2), documentação (C3) e formato de especificação dos requisitos (C4). Por fim, a última coluna representa se a ferramenta foi (ou não) aceita, com base na aplicação dos critérios C1-C4.

Como pode ser observado na Tabela 3.1, apenas 3 ferramentas foram aceitas para avaliação: *RETA*, *Tactile Check* e *Tiger-PRO*.

A maioria das ferramentas foi rejeitada pelos critérios "Disponibilidade (C1)" e de "Documentação (C3)". A guisa de exemplo, a ferramenta *Scout* é uma solução da empresa *Qualicen*, a qual não libera licença para fins acadêmicos para nenhum de seus produtos; para a ferramenta *GATE*, embora os desenvolvedores da mesma tenham enviado, por *e-mail*, as regras para serem importadas na ferramenta e uma explicação sobre elas, não foram encontradas informações sobre como configurar tais regras para execução da ferramenta e, por isso, ela não atendeu ao critério "Documentação"; a ferramenta *Teamscale* lida apenas com requisitos que estejam vinculados à um código-fonte, não atendendo ao critério "Formato de Especificação dos Requisitos". As justificativas para exclusão das demais ferramentas podem ser consultadas no Apêndice A deste trabalho.

3.3 Etapa 3 - Selecionar conjuntos de requisitos (DR) e Identificar Anomalias

Um importante artefato para analisar a eficácia de uma ferramenta para detecção de anomalias de requisitos de *software* é o Documento de Requisitos (DR). Para que as ferramentas pudessem ser comparadas em um cenário mais próximo do mundo real, estabeleceu-se alguns critérios para seleção dos DR a serem utilizados neste trabalho, a saber:

Tabela 3.1 – Resultado do processo de seleção das ferramentas para análise comparativa

ID	Ferramenta	Artigo ID	C1	C2	C3	C4	Selecionada
1	*RETA*	1,15,16,17	x	x	x	x	Sim
2	Protótipo - sem nome (sn)	2	x			x	Não
3	*ConQAT*	3,18	x			x	Não
4	*Smella*	4,15,16,17,20				x	Não
5	*GATE*	5,15,16,20	x			x	Não
6	*Tactile Check*	6	x	x	x	x	Sim
7	*Holmes - Qualicen*	7				x	Não
8	*RAQ*	7			x	x	Não
9	*Tiger-PRO*	7	x	x	x	x	Sim
10	*Teamscale*	8		x	x		Não
11	*QuaRS Express*	10,14,19	x			x	Não
12	*SREE*	10,14,15,20			x	x	Não
13	*Scout - Qualicen*	11			x	x	Não
14	*AQUSA*	12,19		x	x	x	Não
15	*Ambiguity Detection*	14	x			x	Não
16	*Poirot*	19					Não
17	*Dowser*	19	x			x	Não
18	*RAI*	19					Não
19	*ARM tool*	20	x			x	Não
20	*Nvivo*	21		x		x	Não
21	Protótipo sn	25		x	x	x	Não
22	Protótipo sn	26		x	x	x	Não

Fonte: Do Autor (2023)

a) **Possuir mais de um domínio**: dependendo da técnica de detecção de anomalias adotada pela ferramenta, pode ser que em áreas específicas, ela possa ser mais efetiva. Com intuito de evitar esse viés, determinou-se o uso de DR de domínios variados.

b) **Possuir, no mínimo, 30 requisitos**: quanto maior o número de requisitos, maior a probabilidade da ocorrência e da diversidade de anomalias. Além disso, um número maior de requisitos representa um cenário mais próximo de um ambiente real em uma fábrica de *software*.

c) **Deve estar na língua inglesa**: a maior parte das ferramentas para detecção de anomalias de requisitos disponíveis na academia e no mercado lida com a língua inglesa (ARENDSE; LUCASSEN, 2016). Não obstante, a partir de uma análise das ferramentas catalogadas por Nascimento et al. (2018), todas elas suportam apenas requisitos escritos no idioma inglês.

d) **Possuir mais de um tipo de anomalia**: uma característica importante de uma ferramenta de detecção de anomalias é a variedade de anomalias com a qual ela consegue lidar (ARENDSE; LUCASSEN, 2016). Nesse sentido, optou-se em selecionar DR que apresentem mais de um tipo de anomalia.

Os *DR* utilizados neste trabalho foram obtidos a partir do *dataset* PURE (*PUblic REquirements dataset*) [2], com base nos critérios elencados anteriormente. O *dataset* PURE provê uma base de dados composta por 79 documentos de requisitos públicos, escritos em Linguagem Natural e que abrangem diversos domínios (FERRARI; SPAGNOLO; GNESI, 2017). O objetivo dessa base de dados é permitir que esses documentos sejam utilizados para atividades relacionadas à Engenharia de Requisitos e Processamento de Linguagem Natural, como por exemplo, detecção de ambiguidade, categorização de requisitos, entre outras. O *dataset* PURE se torna útil, visto que, atualmente, a maioria dos trabalhos realizados sobre detecção de anomalias de requisitos utilizam documentos próprios e poucos disponibilizam suas bases de requisitos, dificultando a replicação dos experimentos, bem como a generalização dos resultados (FERRARI; SPAGNOLO; GNESI, 2017).

A seleção dos DR, conforme os critérios mencionados anteriormente, bem como a detecção das anomalias presentes em seus requisitos, foram realizadas por uma aluna de IC (Iniciação Científica) e os resultados foram revisados pelo autor desta pesquisa e outro pesquisador com experiência em Engenharia de Requisitos. Foram selecionados, do *dataset* PURE, 3 (três) DR[3] para serem utilizados na análise comparativa das ferramentas identificadas na etapa anterior. As anomalias presentes nos DR foram manualmente identificadas e validadas, criando-se uma espécie de oráculo, que serviu como parâmetro de avaliação durante a análise comparativa. Destaca-se que as anomalias consideradas para a elaboração desse oráculo foram as catalogadas por Nascimento et al. (2018) e que estão relacionadas na Tabela 2.1 no Capítulo 2.

A Tabela 3.2 apresenta a lista de *DR* selecionados. Cada cada *DR* é apresentado um identificador, seu título e a quantidade de requisitos que o compõe. A quantidade e os tipos de anomalias identificadas nesses *DR* são comentadas mais adiante neste texto. Destaca-se que os DR selecionados atenderam aos critérios definidos nesta pesquisa, isto é, possuem mais de 30 requisitos, estão na língua inglesa e são de domínios diferentes.

[2] https://fmt.isti.cnr.it/nlreqdataset/
[3] https://shre.ink/aycW

Tabela 3.2 – Documentos de requisitos selecionados

ID	Título	Quantidade de requisitos
DR01	Integrated Library System Specification FRS	51
DR02	DigitalHome Software Requirements Specification	92
DR03	Crime And Criminal Tracking Network And Systems (Cctns)	122

Fonte: Do Autor (2023)

O DR01, de nome *Integrated Library System Specification* (ILS), foi especificado por *Lori Ayre* e *Lucien Kress* em janeiro de 2009. Esse documento de requisitos descreve os requisitos funcionais e não funcionais para o Módulo de Administração de um "Sistema Integrado de Biblioteca" (ILS). Os requisitos foram desenvolvidos especificamente para o Sistema de Bibliotecas do Condado de *King* (*Texas*), porém, eles são adequados para muitos sistemas de bibliotecas grandes, urbanos, de múltiplas filiais e centralizados. A organização dos requisitos foi feita por categorias, sendo elas: Geral, *Consoles e Dashboards*, Regras de Negócios, Recuperação de Dados, Segurança, Manutenção, Gestão de Clientes, Consultas e Relatórios. A Figura 3.3 apresenta exemplos de requisitos que fazem parte da categoria "Geral" do DR01. Destaca-se que os textos dos DR foram mantidos em inglês, pois foram submetidos às ferramentas dessa forma e, ademais, evita-se a perda da semântica, que pode ser gerada por intermédio da tradução.

Figura 3.3 – Exemplos de requisitos do DR01

Sigla	Requisito
RE01	System runs on a fully relational, SQL-based database system. Ability to run SQL queries against any table in the database. Ability to access database as an ODBC source. All data tables and data storage are fully accessible.
RE02	The system provides real-time processing. For example: pull lists are up to date at time of viewing or printing; system supports live shelf reading and weeding.
RE03	Ability for multiple staff members and patrons to simultaneously access and update patron and item records, including on staff check-in and check-out terminals, on self check-out stations, through SIP2/NCIP2 and similar protocols and APIs, and in OPAC. Depending on assigned privileges, staff can view all patron and item fields; patrons can access only selected fields. Record changes are applied in a reasonable way, with prompts to warn when a record has been changed since it was displayed.
RE04	For any patron record or item record, staff can identify where it is in use (location, user, date and time placed).
RE05	Support for individual and shared staff login accounts; access to modules is granted by use of "roles" or "privileges" that allow each account to access as many (or as few) modules as needed
RE06	System documentation is library-specific and follows standard formats for technical documentation. Documentation is specific to the particular version of the software in use at library. Documentation is web-based, indexed, organized by function, and easily searchable.
RE07	System upgrades and updates include written guidelines for updating servers and clients. Includes list of new, changed, and removed features.
RE08	System provides access to all configuration files

Fonte: Adaptado do repositório PURE (2023)

A Figura 3.4, por sua vez, apresenta uma anomalia de requisito do tipo “Lacunas ou brechas” (ver Tabela 2.1 no Capítulo 2), identificada no REQ03. O termo em destaque, “*reasonable*”, que em tradução livre para o português seria “razoável”, é ambíguo e de entendimento subjetivo, uma vez que o grau de concordância do que é (ou não) razoável varia de pessoa para pessoa.

Figura 3.4 – Exemplo de anomalia identificada no DR01

RE03	Ability for multiple staff members and patrons to simultaneously access and update patron and item records, including on staff check-in and check-out terminals, on self check-out stations, through SIP2/NCIP2 and similar protocols and APIs, and in OPAC. Depending on assigned privileges, staff can view all patron and item fields; patrons can access only selected fields. Record changes are applied in a **reasonable** way, with prompts to warn when a record has been changed since it was displayed.

Fonte: Adaptado do repositório PURE (2023)

O DR02, de nome *DigitalHome Software Requirements Specification*, foi especificado em outubro de 2010 por *Michel Jackson*. Este documento especifica os requisitos para o desenvolvimento de uma “*Smart House*”, chamada *DigitalHome* (DH), pela *DigitalHomeOwner Division* da *HomeOwner Inc.*. Trata-se de uma especificação de um sistema que permitirá a um utilizador doméstico gerir dispositivos que controlam o ambiente de uma casa. Os componentes da *Smart House* consistem em dispositivos domésticos (por exemplo, uma unidade de aquecimento e ar condicionado, um sistema de segurança e pequenos aparelhos e unidades de iluminação, entre outros), sensores e controladores para os dispositivos, entre outros. Em suma, o usuário se comunica por meio de uma página da *web* pessoal no servidor da *DigitalHome* ou em um servidor doméstico local.

O DR da *DigitalHome* é composto por uma lista das principais características do sistema. Diferentemente do DR01, esta versão do DR02 não é uma especificação abrangente ou completa dos requisitos da *DigitalHome*. Entretanto, ela cobre a maior parte de suas funcionalidades. Os requisitos estão divididos em “Requisitos Funcionais” e “Requisitos Não-funcionais”, e organizados em categorias como Geral, Segurança, Relatórios, Desempenho, entre outros. A Figura 3.5 apresenta exemplos de requisitos que fazem parte da categoria “Geral” do DR02.

A Figura 3.5 apresenta uma anomalia de requisito do tipo “Superlativo”. Essa anomalia foi identificada no REQ17, da categoria “Ambiente Operacional”, do DR02. O termo em destaque, “*widely*”, que em tradução livre para o português seria “amplamente”, trata-se de um superlativo (um tipo de flexão do adjetivo na língua portuguesa) que tem a característica de intensificar as características do substantivo de forma relativa ou absoluta. Quando utilizado para

Figura 3.5 – Exemplo de requisitos do DR02

Sigla	Requisito
RE01	The general user shall be able to use the DH system capabilities to monitor and control the environment in his/her home.
RE02	The general user is familiar with the layout of his/her home and the location of sensor and control devices (for temperature, for humidity, and for power to small appliances and lighting units).
RE03	Although the general user is not familiar with the technical features of the DH system, he/she is familiar with the use of a web interface and can perform simple web operations (logging in and logging out, browsing web pages, and submitting information and requests via a web interface).
RE04	A Master user will be designated, who shall be able to change the configuration of the system. For example, a Master User shall be able to add a user account or change the default parameter settings. He/she will have the same right as the DH Technician, described in section 3.2.2.4.
RE05	A DH Technician is responsible for setting up and maintaining the configuration of a DH system.
RE06	A DH Technician has experience with the type of hardware, software, and web services associated with a system like the DH system.
RE07	A DH Technician is specially trained by DigitalHomeOwner to be familiar with the functionality, architecture, and operation of the DH system product.
RE08	A DH Technician will have rights beyond the DH General User, capable of setting up and making changes in the configuration of the system (e.g.,setting system parameters and establishing user accounts), and starting and stopping operation of the DH System.
RE09	The “prototype” version of the DigitalHome System (as specified in this document) must be completed within twelve months of inception.

Fonte: Adaptado do repositório PURE (2023)

classificar um atributo de um requisito, traz subjetividade e pode deixar a interpretação dúbia (FEMMER et al., 2017).

Figura 3.6 – Exemplo de anomalia identificada no DR02

RE12	Where possible, the DigitalHome project will employ **widely** used, accepted, and available hardware and software technology and standards, both for product elements and for development tools. See section 3.4 for additional detail.

Fonte: Adaptado do repositório PURE (2023)

O DR03 é identificado pelo título *Crime And Criminal Tracking Network And Systems* (CCTNS) e foi especificado pelo *Ministry of Home Affairs Government Of India*. O CCTNS é um projeto do governo indiano para criar um sistema abrangente e integrado para um policiamento eficaz. O sistema inclui um programa nacional de rastreamento *on-line*, integrando milhares de delegacias de polícia em todo o país. O DR03 fornece a descrição detalhada das funcionalidades necessárias para a primeira versão do CCTNS. Os requisitos estão divididos em “Requisitos Funcionais” e “Requisitos Não-funcionais” e organizados em módulos do sistema como Navegação, Configuração, Cadastro, Investigação, Acusação, entre outros. A Figura 3.7 apresenta exemplos de requisitos (não funcionais) que fazem parte do módulo de suporte do DR03.

A Figura 3.8 apresenta uma anomalia de requisito do tipo “Termos abertos e não verificáveis”. Essa anomalia foi identificada no REQ34, da seção de requisitos não-funcionais, do DR03. O termo em destaque, “*several*”, que em tradução livre para o português seria “várias”, traz subjetividade e dificulta o entendimento correto do requisito (FEMMER et al., 2017). Não

Figura 3.7 – Exemplo de requisitos do DR03

Sigla	Requisito
RE01	Citizens can register their complaints with police and then based on the evidence, facts and following investigation, police shall take the complaint forward. The Registration module acts as an interface between the police and citizens and it eases the approach, interaction and information exchange between police and complainants.
RE02	After a complaint is initiated, police initiates the investigation process. The Investigation module of the CCTNS facilitates the investigation process and introduces operational efficiencies by automating most of the tasks that take place after initial entries are made during Registration.
RE03	The Search module of the CCTNS gives police personnel the ability to execute a basic or advanced search on cases. Using the search functionality, police personnel can search for a particular person, type of crime, modus operandi, property etc. It also gives the user the ability to customize the results view by criminal/accused or by cases. It makes reporting easy for police by enabling them to execute different types of queries such as monthly reporting, RTI related etc.
RE04	The Citizen Interface module of the CCTNS acts as a conduit for the information exchange between citizens and police units/personnel. Citizens can use it as a tool to get information or acknowledgements from police. The police in turn can use it to respond to citizens with very little turnaround time. It improves overall productivity by helping citizens and police to cut short the drudgery of large amounts of paperwork.
RE05	The Navigation module of the CCTNS provides role based landing pages which help in navigating through the CCTNS application. It shows information such as cases assigned,alerts, pending tasks etc hence helping police personnel to plan better and execute with greater efficiency.
RE06	The Configuration module of the CCTNS helps keep the application configured according to the states' requirements in addition to keeping data elements/rules up to date. With a proper configuration, information such as act and sections, state specific data, castes, tribes, property information etc. can be created updated and deleted. The functional requirements for each of the modules are provided as A1 to A7 in separate enclosures.

Fonte: Adaptado do repositório PURE (2023)

delimitar um valor exato de um atributo numérico de um requisito faz com o valor fique sujeito à interpretação do leitor.

Figura 3.8 – Exemplo de anomalia identificada no DR03

RE36	The System must be able to display **several** entities (cases, suspects) simultaneously.

Fonte: Adaptado do repositório PURE (2023)

No DR01, dentre os 51 requisitos que compõem a sua especificação, foram identificados 23 requisitos com anomalias. No DR02, apesar de o número de requisitos ser quase o dobro, se comparado com o DR01 (92 requisitos), apenas 13 apresentaram algum tipo de anomalia. Por fim, o DR03, dentre os seus 122 requisitos especificados, 49 deles apresentaram algum tipo anomalia. A Tabela 3.3 aponta os tipos de anomalias encontrados em cada DR. É importante ressaltar que alguns requisitos apresentaram, em si mesmos, mais de um tipo de anomalia.

Tabela 3.3 – Anomalias identificadas nos DR

ID	Requisitos com anomalias	Tipos encontrados
DR01	23	Lacunas ou brechas, Referências incompletas, Advérbios e adjetivos ambíguos
DR02	13	Lacunas ou brechas, Superlativo, Pronomes vagos, Termos abertos e não verificáveis
DR03	49	Advérbios e adjetivos ambíguos, Lacunas ou brechas, Termos abertos e não verificáveis, Palavras negativas

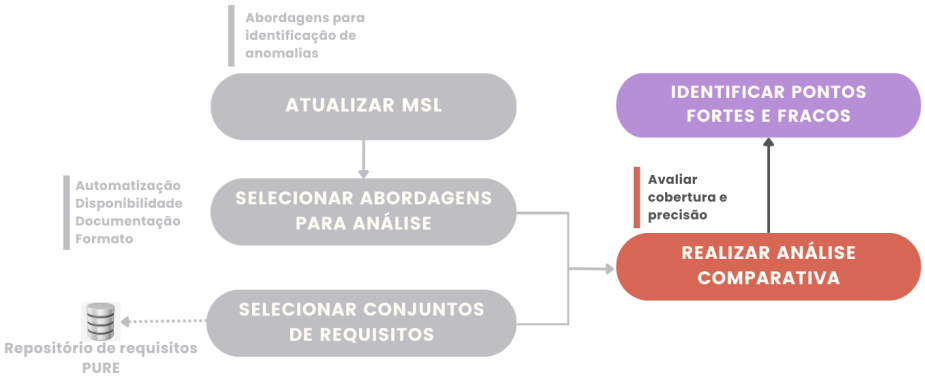
Fonte: Do Autor (2023)

4 ANÁLISE COMPARATIVA PARA DETECÇÃO DE ANOMALIAS DE REQUISITOS

Apesar de haver vários critérios para avaliação de um sistema de *software* (por exemplo, usabilidade, desempenho, segurança, entre outros), a análise comparativa proposta neste trabalho tem como objetivo avaliar a qualidade das ferramentas selecionadas, em termos de sua precisão e cobertura (mais detalhes sobre essas métricas são apresentados na Seção 4.2). Pretende-se, em trabalhos futuros, analisar outros critérios, a fim de se ter um *benchmark* mais robusto a respeito de ferramentas para detecção de anomalias de requisitos.

A Figura 4.1 apresenta as etapas executadas para a concretização deste trabalho. As etapas “Atualizar `MSL`”, “Selecionar abordagens para análise” e “Selecionar conjuntos de requisitos” foram apresentadas no Capítulo 3. Neste capítulo serão apresentadas as etapas “Realizar análise comparativa” e “Identificar pontos fortes e fracos”. As Seções 4.1 e 4.2 descrevem a etapa “Realizar análise comparativa”, na qual são apresentadas as configurações de ambiente e as especificidades de cada ferramenta, bem como os resultados obtidos a partir da análise comparativa. Por último, a Seção 4.3 descreve a etapa “Identificar pontos fortes e fracos”, na qual é apresentada uma discussão acerca dos resultados alcançados por cada ferramenta e uma análise comparativa entre eles.

Figura 4.1 – Etapas finais do método de pesquisa



Fonte: Do Autor (2022)

4.1 Etapa 4 - Realizar análise comparativa

A etapa "Realizar análise comparativa"iniciou-se com a preparação do ambiente, que consistiu em selecionar o sistema operacional, instalar *plugins* e *frameworks* necessários para

rodar as ferramentas. Além disso, nesse processo de preparação do ambiente, foi verificada a necessidade de licença de uso e os pré-requisitos mínimos de *hardware* para cada ferramenta.

Ao todo, foram 3 (três) ferramentas que compuseram a análise comparativa proposta neste trabalho, a saber: *RETA*, *Tactile Check* e *Tiger-PRO*. Somente a ferramenta *Tiger-PRO* possui licença comercial, que permite acesso a mais parâmetros de configuração, personalização da ferramenta e outros recursos voltados à especificação de requisitos. Entretanto, na *Tiger-PRO*, a identificação de anomalias (foco desta pesquisa) é um recurso nativo e habilitado para uso em sua versão gratuita. Nas demais ferramentas, isto é, na *RETA* e na *Tactile Check*, não há necessidade de licença de uso proprietária e todos os recursos estão disponíveis para utilização gratuitamente.

Cada ferramenta lida com uma quantidade e tipos específicos de anomalias. Em alguns casos, uma mesma anomalia pode ser tratada por nomes diferentes entre as ferramentas. Além desses casos, pode acontecer ainda de uma anomalia corresponder a mais de um tipo em outra ferramenta. Por esses motivos, foi realizada uma correspondência do tipo "De-Para" entre as anomalias utilizadas neste trabalho[1], e as anomalias propostas pelas ferramentas. Para realizar essa correlação, cada anomalia catalogada foi identificada com um `ID` único, como é possível conferir no Quadro 4.1. As anomalias propostas pelas ferramenta que não se encaixaram em nenhuma categoria das anomalias catalogadas, foram desconsideradas. Isso foi feito com intuito de categorizar as anomalias de igual forma, tomando como base as anomalias catalogadas, para não haver prejuízos para nenhuma ferramenta na análise comparativa.

A ferramenta ***RETA* (*REquirements Template Analyzer*)** foi desenvolvida em 2015 como um aplicativo para o ambiente de trabalho *GATE Developer*, ou apenas *GATE*. Portanto, é necessário instalar o *GATE* previamente e, somente após, realizar a importação do *RETA*. O *GATE* é um *software* de código aberto, está disponível sob a licença *GNU Lesser General Public Licence 3.0* [2] e pode ser obtida através do *website* dos autores[3]. O *software GATE* é multiplataforma, isto é, possui versão para *Linux*, *Windows* e *MAC*. Independentemente do sistema operacional, é necessário que ele possua *JAVA* em sua versão 8 ou superior *JDK*. Uma vez que o *GATE* esteja instalado corretamente, basta descompactar a ferramenta *RETA* e importar o arquivo "*application.xgapp*" no *GATE*.

[1] Daqui adiante, essas anomalias serão descritas como "anomalias catalogadas".
[2] https://www.gnu.org/licenses/lgpl-3.0.html
[3] https://gate.ac.uk/download/

Quadro 4.1 – Anomalias de requisitos catalogadas por catalogadas por Nascimento, Guimarães e Lucena (2021)

ID	Anomalia
AN1	Advérbios e adjetivos ambíguos
AN2	Termos abertos e não-verificáveis
AN3	Lacunas ou brechas
AN4	Palavras negativas
AN5	Referências incompletas
AN6	Linguagem subjetiva
AN7	Pronomes vagos
AN8	Comparativos
AN9	Superlativos
AN10	Voz passiva
AN11	Funcionalidade duplicada

Fonte: adaptado de Nascimento, Guimarães e Lucena (2021)

A ferramenta *RETA* lida com 10 (dez) tipos de anomalias ((ARORA et al., 2015)), porém, apenas 5 (cinco) delas foram possíveis de serem mapeadas para as anomalias catalogadas, a saber:

a) ***Warn Quantifier***: termos usados para quantificação como **todos** e **quaisquer** podem levar a ambiguidade, se não forem usados adequadamente. Exemplo: "Todas as luzes da sala estão conectadas a um interruptor." **Existe um único interruptor ou vários interruptores?**

b) ***Warn Pronoun***: pronomes podem levar à ambiguidade referencial. Exemplo: "As caminhonetes devem passar pelas estradas antes que elas congelem." **"Eles" se refere às caminhonetes ou às estradas?**

c) ***Warn Vague Terms***: existem vários termos vagos que são comumente usados em documentos de requisitos. Os exemplos incluem **amigável**, **suporte**, **aceitável**, **até** e **periodicamente**. Esses termos devem ser evitados em requisitos. Exemplo: "O módulo S&T deve **suportar até** cinco parâmetros de *status* configuráveis." **"Suporte" é um termo vago. Não está claro se "até" significa "até e incluindo" ou "até e excluindo".**

d) ***Warn Passive Voice***: a voz passiva desfoca o ator do requisito e deve ser evitada nos requisitos. Exemplo: "caso o módulo C&T necessite de um arquivo de configuração local, este deve ser criado a partir dos dados de configuração do sistema de banco de dados." **Não está claro se o ator é módulo C&T, banco de dados, ou outro agente.**

e) ***Warn Adverb in Verb Phrase***: frases verbais adverbiais são desencorajadas devido à imprecisão e às chances de detalhes importantes permanecerem tácitos no advérbio. Exemplo: "O módulo S&T deve pesquisar **periodicamente** o banco de dados para informações EDTM CSI." **A frequência da atividade periódica não é especificada.**

O Quadro 4.2 apresenta a relação do tipo "De-Para" entre as anomalias da ferramenta *RETA* e as anomalias catalogadas (ver Tabela 2.1 no Capítulo 2). É possível observar que a anomalia *Warn Adverb in Verb Phrase* foi mapeada para os quatro tipos de anomalias catalogadas (`AN1`, `AN2` e `AN4`). Nesses casos, a decisão sobre quando se tratava de uma ou de outra foi realizada pelo autor deste trabalho e revisada por outro pesquisador.

Quadro 4.2 – Relação de anomalias catalogadas X Anomalias da ferramenta *RETA*

Sigla	**Anomalias catalogadas**	**Anomalias da ferramenta *RETA***
AN1	Advérbios e adjetivos ambíguos	Warn Adverb in Verb Phrase
AN2	Termos abertos e não verificáveis	Warn Adverb in Verb Phrase
AN3	Lacunas e brechas	Warn Vague Terms
AN4	Palavras negativas	Warn Adverb in Verb Phrase
AN10	Voz passiva	Warn Passive Voice
AN6	Linguagem subjetiva	Warn Quantifier
AN7	Pronomes Vagos	Warn Pronoun

Fonte: Do Autor (2023)

A ferramenta *RETA* foi executada em ambiente *Microsoft Windows* (versão 11), sob o *GATE* na versão 8.1 (como sugerido pelos autores). Todos os `DR` foram transcritos para texto puro sem formatação, uma vez que a *RETA* suporta apenas requisitos em *.txt*. A detecção de anomalias, para cada `DR`, foi realizada em 3 passos:

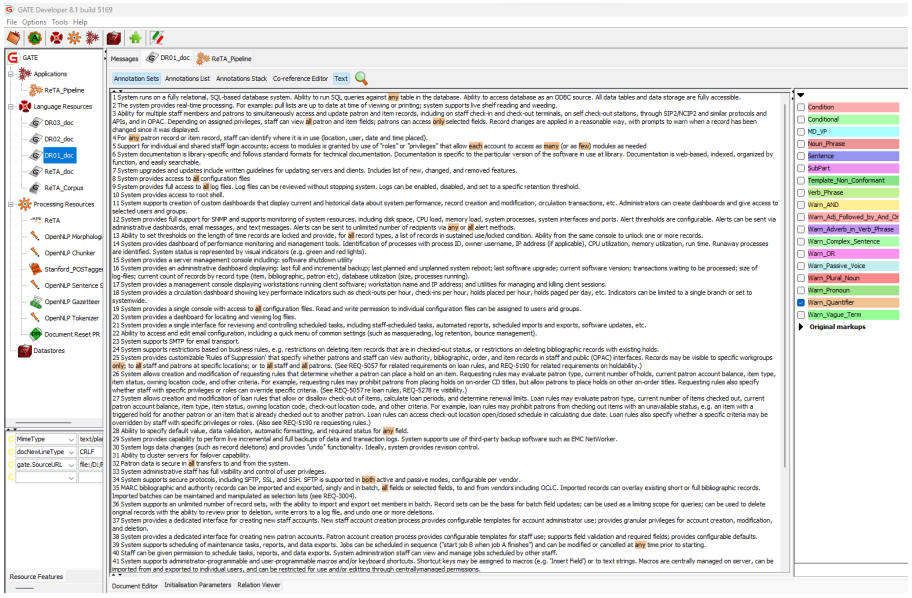
a) **Passo 1:** importar `DR`. Nessa etapa, foi necessário informar o caminho do arquivo concernente ao `DR` e um nome para referenciá-lo;

b) **Passo 2:** selecionar a anomalia. A detecção foi realizada apenas com uma anomalia por vez para facilitar a identificação das mesmas no texto do `DR`. Vale ressalta que, a *RETA* utiliza-se de cores para identificar cada anomalia no texto do requisito. Caso um termo seja marcado por mais de uma anomalia, há sobreposição de cores e dificulta a leitura do resultado. Por esse motivo, foi realizada a detecção separadamente das anomalias;

c) **Passo 3:** registrar resultados. Após a detecção, registrou-se em uma planilha eletrônica o número de ocorrências de anomalias identificadas pela ferramenta.

Um exemplo de detecção de anomalias de requisitos de *software* realizada pela ferramenta *RETA* pode ser vista na Figura 4.2.

Figura 4.2 – Exemplo detecção de anomalia - RETA



Fonte: Do autor (2023)

A ferramenta ***Tactile Check*** (TC) foi desenvolvida em 2016 como uma macro [4] *Visual Basic for Applications* (VBA) do *Microsof Word* na versão 2013. Ela adiciona anotações às frases do documento, conforme vai identificando anomalias nele. Por ser disponibilizada para *Word*, pode ser executada no mesmo ambiente usado para escrever o documento de requisitos. Contudo, apesar da ferramenta TC ser distribuída gratuitamente, vale ressaltar que ela é executa sob um produto licenciado comercialmente pela *Microsoft* (*Word*).

A TC lida com 5 (cinco) tipos de anomalias, sendo elas (WILMINK; BOCKISCH, 2017; NASCIMENTO et al., 2018):

a) ***Incompleteness***: são termos que pedem complemento e não são fornecidos no requisitos. Exemplos: **não definido**, **não determinado**, **mas não limitado a**, **no mínimo**, entre outros.

b) ***Subjective Language***: são palavras cujo significado não é objetivo. Exemplos: **amigável**, **fácil de usar**, **econômico**;

[4] Segundo a *Microsoft*, uma macro é uma "série de comandos e instruções que você agrupa como um único comando para realizar uma tarefa automaticamente".

c) ***Negative Statements***: são termos usados para indicar o que o sistema não deve fornecer. Exemplo: o sistema **não deve** aceitar registros de clientes duplicados. Essa forma de descrever um requisito pode deixar dúbio o entendimento do comportamento do sistema levando a uma espécie de "subespecificação";

d) ***Non-verifiable Terms***: são termos difíceis de verificar, por oferecer várias possibilidades de execução do sistema. Exemplo: o sistema só pode ser ativado se todos os sensores necessários (...) trabalharem com precisão de medição **suficiente**;

e) ***Loopholes***: são termos que possibilitam aos *stakeholders* ignorarem as especificações. Exemplos: **se possível**, **conforme apropriado**, **conforme aplicável**.

O Quadro (4.3) apresenta a relação do tipo "De-Para" entre as anomalias adotadas pela *Tactile Check* e as anomalias catalogadas. É possível observar que a anomalia *Subjective Language* foi mapeada para dois tipos de anomalias catalogadas (AN1 e AN6).

Quadro 4.3 – Relação de anomalias catalogadas X Anomalias da ferramenta *Tactile Check*

Sigla	**Anomalias catalogadas**	**Anomalias *Tactile Check***
AN1	Advérbios e adjetivos ambíguos	Subjective Language
AN2	Termos abertos e não verificáveis	Non-verifiable Terms
AN3	Lacunas e brechas	Loopholes
AN4	Palavras negativas	Negative Statements
AN5	Referências incompletas	Incompleteness
AN6	Linguagem subjetiva	Subjective Language

Fonte: Do Autor (2023)

A ferramenta *Tactile Check* foi executada no *Word* (versão 17), em ambiente *Windows* (versão 11), ambos produtos licenciados pela *Microsoft*. Como requerido da ferramenta, foi necessário inserir os requisitos em formato de tabela, em que, cada requisito deveria ocupar uma única linha e possuir até duas colunas, sendo uma para identificação (ID) e a outra para a descrição do requisito. A detecção de anomalias, para cada `DR`, foi realizada em 3 passos:

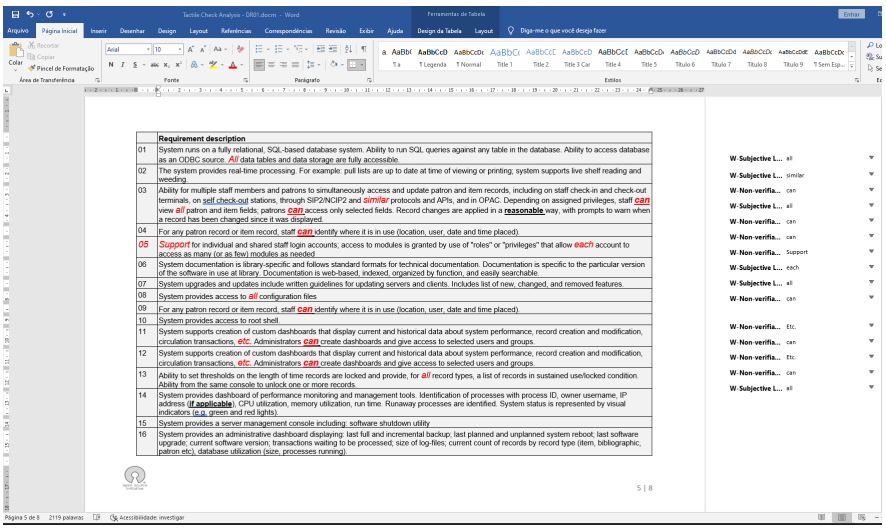
a) **Passo 1:** importar o `DR`. Nessa etapa, foi utilizado um *template* em branco fornecido pela *Tactile Check* para transcrever os requisitos em formato de tabela;

b) **Passo 2:** executar detecção automática. Diferentemente de como ocorreu na ferramenta *RETA*, não houve necessidade de executar a detecção para cada anomalia separadamente. A forma como a *Tactile Check* apresenta seus resultados permitiu realizar a detecção do documento como um todo e de uma única vez;

c) **Passo 3:** registrar resultados. Após a detecção, registrou-se em uma planilha eletrônica, o número de ocorrências de anomalias identificadas pela ferramenta.

Um exemplo de detecção de anomalias de requisitos de *software* realizada pela ferramenta TC pode ser vista na Figura 4.3.

Figura 4.3 – Exemplo detecção de anomalia - *Tactile Check*



	Requirement description
01	System runs on a fully relational, SQL-based database system. Ability to run SQL queries against any table in the database. Ability to access database as an ODBC source. *All* data tables and data storage are fully accessible.
02	The system provides real-time processing. For example: pull lists are up to date at time of viewing or printing; system supports live shelf reading and weeding.
03	Ability for multiple staff members and patrons to simultaneously access and update patron and item records, including on staff check-in and check-out terminals, on self check-out stations, through SIP2/NCIP2 and *similar* protocols and APIs, and in OPAC. Depending on assigned privileges, staff ***can*** view ***all*** patron and item fields; patrons ***can*** access only selected fields. Record changes are applied in a **reasonable** way, with prompts to warn when a record has been changed since it was displayed.
04	For any patron record or item record, staff ***can*** identify where it is in use (location, user, date and time placed).
05	*Support* for individual and shared staff login accounts; access to modules is granted by use of "roles" or "privileges" that allow *each* account to access as many (or as few) modules as needed.
06	System documentation is library-specific and follows standard formats for technical documentation. Documentation is specific to the particular version of the software in use at library. Documentation is web-based, indexed, organized by function, and easily searchable.
07	System upgrades and updates include written guidelines for updating servers and clients. Includes list of new, changed, and removed features.
08	System provides access to *all* configuration files
09	For any patron record or item record, staff ***can*** identify where it is in use (location, user, date and time placed).
10	System provides access to root shell.
11	System supports creation of custom dashboards that display current and historical data about system performance, record creation and modification, circulation transactions, *etc*. Administrators ***can*** create dashboards and give access to selected users and groups.
12	System supports creation of custom dashboards that display current and historical data about system performance, record creation and modification, circulation transactions, *etc*. Administrators ***can*** create dashboards and give access to selected users and groups.
13	Ability to set thresholds on the length of time records are locked and provide, for *all* record types, a list of records in sustained use/locked condition. Ability from the same console to unlock one or more records.
14	System provides dashboard of performance monitoring and management tools. Identification of processes with process ID, owner username, IP address (**if applicable**); CPU utilization, memory utilization, run time. Runaway processes are identified. System status is represented by visual indicators (e.g. green and red lights).
15	System provides a server management console including: software shutdown utility
16	System provides an administrative dashboard displaying: last full and incremental backup; last planned and unplanned system reboot; last software upgrade; current software version; transactions waiting to be processed; size of log-files; current count of records by record type (item, bibliographic, patron etc), database utilization (size, processes running).

Fonte: Do autor (2023)

A ferramenta ***Tiger-PRO*** (**TP**) foi desenvolvida em 2006, utilizando a linguagem de programação *Delphi* da *Borland* (*Visual Pascal*). É uma ferramenta de *front-end* para auxiliar o Engenheiro de Requisitos a escrever bons requisitos (ARENDSE; LUCASSEN, 2016). Ela roda exclusivamente sob a plataforma *Microsoft Windows*. O TP é *shareware*[5] e deve ser licenciado se usado por mais de 14 (quatorze) dias, exceto para fins acadêmicos que, para esse caso, o proprietário fornece licença gratuita. Atualmente, a TP possui duas versões: 2.0 e 2.1 (beta). A versão *beta* possui recursos adicionais, incluindo destaque de cores para cada tipo de anomalia e a capacidade de reparar alguns tipos de defeitos. Entretanto, como o foco da pesquisa é de detecção de anomalias, esta pesquisa considerou a versão 2.0, além do fato de ser a versão estável e consolidada da ferramenta.

A TP lida com 7 tipos de anomalias (KASSER et al., 2003), porém, apenas uma delas foi possível ser mapeada para as anomalias catalogadas: ***Unverifiable***. Essa anomalia diz respeito

[5] A característica do *shareware* é que ele é distribuído em uma base "experimente antes de comprar" e uma doação é esperada se o *software* for útil para o usuário. (CIFUENTES; FITZGERALD, 1997)

a termos subjetivos que não expressam um valor exato para uma funcionalidade, por exemplo, *fácil*, *rápido*, *amigável*, entre outros .

O Quadro 4.4 apresenta a relação do tipo "De-Para" entre as anomalias catalogadas e aquelas propostas pela ferramenta *Tiger Pro*. É possível observar que a anomalia *Unverifiable* foi mapeada para os quatro tipos de anomalias catalogadas (`AN1`, `AN2`, `AN3`, e `AN4`). Nesses casos, a decisão sobre quando se tratava de uma ou de outra foi realizada pelo autor deste trabalho e revisada por outro pesquisador.

Quadro 4.4 – Relação de anomalias catalogadas X Anomalias da *Tiger Pro*

Sigla	**Anomalias catalogadas**	**Anomalias *Tiger Pro***
AN1	Advérbios e adjetivos ambíguos	Unverifiable
AN2	Termos abertos e não verificáveis	Unverifiable
AN3	Lacunas e brechas	Unverifiable
AN6	Linguagem subjetiva	Unverifiable

Fonte: Do Autor (2023)

A ferramenta *Tiger Pro* foi executada em um ambiente *Microsoft Windows* (versão 11). A *Tiger Pro* permite realizar a importação de arquivo em formato *.txt*. Entretanto, foi necessário retirar as quebras de linhas da descrição dos requisitos antes de realizar a importação do `DR`, pelo fato de a ferramenta tratar cada quebra de linha como um novo requisito. A detecção de anomalias, para cada `DR`, foi realizada em 3 passos:

a) **Passo 1:** importar `DR`. Nessa etapa, foi informado o caminho do `DR` e, em seguida, a ferramenta gerou um banco de dados para armazená-lo;

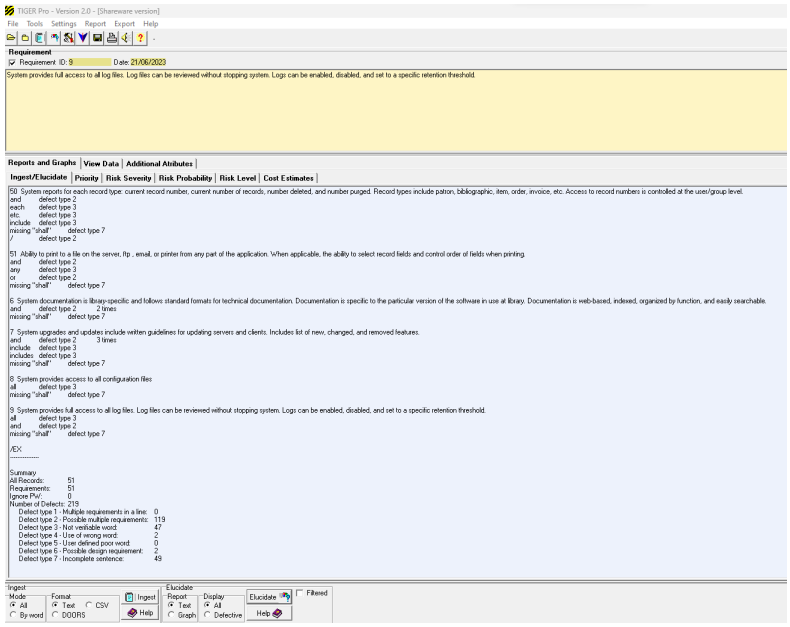
b) **Passo 2:** executar detecção automática. Semelhantemente como ocorreu na ferramenta *Tactile Check*, a forma com que os resultados desta ferramenta são apresentados permitiu que a detecção fosse feita considerando o arquivo completo;

c) **Passo 3:** registrar resultados. Após a detecção, registrou-se em uma planilha eletrônica, o número de ocorrências de anomalias identificadas pela ferramenta.

Um exemplo de detecção de anomalias de requisitos de *software* realizada pela `TC` pode ser visto na Figura 4.4.

Após a realização da correspondência entre as anomalias catalogadas e as anomalias detectadas de cada uma das ferramentas selecionadas para a análise comparativa, foi gerado o Quadro 4.5 para demonstrar com quais tipos de anomalias (catalogadas) as ferramentas lidam.

Figura 4.4 – Exemplo detecção de anomalia - `Tiger-PRO`



Fonte: Do autor (2023)

As anomalias AN8, AN9 e AN11 que, respectivamente, representam as anomalias “Comparativos”, “Superlativos” e “Funcionalidade duplicada”, não estão contidas no escopo de detecção de nenhuma das ferramentas.

Quadro 4.5 – Tipos de anomalias catalogadas que as ferramentas lidam

ID	**Anomalia**	***RETA***	***Tactile Check***	***Tiger Pro***
AN1	Advérbios e adjetivos ambíguos	X	X	X
AN2	Termos abertos e não-verificáveis	X	X	X
AN3	Lacunas ou brechas	X	X	X
AN4	Palavras negativas	X	X	
AN5	Referências incompletas		X	
AN6	Linguagem subjetiva	X	X	X
AN7	Pronomes vagos	X		
AN8	Comparativos			
AN9	Superlativos			
AN10	Voz passiva	X		
AN11	Funcionalidade duplicada			

Fonte: Do Autor (2023)

4.2 Resultados do processo de detecção de anomalias

O desempenho de algoritmos de classificação, clusterização e recuperação de informações geralmente é calculado a partir de métricas, tais como "Cobertura" (*Recall*), "Precisão" (*Precision*) e *F-measure*. (ARORA et al., 2015; NAKACHE; METAIS; TIMSIT, 2005; HERRERA et al., 2012). Sendo assim, elas podem ser aplicadas ao contexto de detecção de anomalias de requisitos, por se tratar de problema de classificação.

A métrica "Cobertura", no contexto deste trabalho, pode ser definida como o número de anomalias detectadas por uma ferramenta em um `DR`, em relação ao total de anomalias existentes neste `DR`. O resultado é um percentual obtido através da razão entre "Anomalias Identificadas" (`AI`) e "Anomalias Existentes" (`AE`), logo:

$$C = \frac{AI}{AE} * 100$$

Por exemplo: em um `DR` com 20 anomalias pré-identificadas por especialistas, se uma ferramenta identificou apenas 10 dessas anomalias, tem-se que a Cobertura = 10/20 * 100 = 50%.

Analogamente, a métrica "Precisão" pode ser definida pela quantidade de anomalias detectadas pela ferramenta que são, de fato, anomalias presentes em um `DR`. O resultado é um percentual obtido por meio da razão entre "Anomalias Reais" (`AR`) e "Anomalias Identificadas" (`AI`), logo:

$$P = \frac{AR}{AI} * 100$$

Por exemplo: se em um conjunto de 20 anomalias identificadas pela ferramenta, apenas 5 são, de fato, anomalias presente no `DR` (de acordo com seu oráculo), então, obtêm-se a Precisão: 5/20 * 100 = 25%.

Por fim, outra métrica bastante utilizada no contexto de problemas de classificação é a *F-measure*. Essa é uma métrica importante, pois combina precisão e cobertura em um único resultado. A *F-measure* é definida como uma média harmônica entre "Precisão" (**P**) e "Cobertura" (**C**), conforme apresenta a fórmula abaixo (SASAKI et al., 2007):

$$F = \frac{2PC}{P+C}$$

Para registro dos dados do processo de detecção de anomalias, utilizou-se uma planilha eletrônica. Foi gerada uma planilha para cada ferramenta, cujos *links* para acesso são apresentados a seguir: *RETA* [6], *Tactile Check* [7] e *Tiger Pro* [8]. Cada planilha possui 4 abas, uma para cada DR e uma para resultados gerais. Em cada aba, tem-se a descrição dos requisitos, a quantidade e o tipo de anomalia detectados pelas ferramentas, além dos dados do oráculo e do somatório das ocorrências identificadas.

4.2.1 Resultados da ferramenta *RETA*

A partir da planilha apresentada anteriormente, foi possível agrupar os dados relativos à quantidade de anomalias identificadas pelas ferramentas, separadas por tipo de anomalia e por documento de requisitos. Também foi possível obter o número de falsos positivos, de falsos negativos e de detecções corretas. À partir desses dados, foi possível calcular as métricas de precisão e cobertura de cada ferramenta.

Os resultados são apresentados nesta dissertação por meio de 4 tabelas, uma para cada um dos 3 (três) DR e uma com o resultado geral. Foram utilizadas siglas para facilitar a visualização dos dados na tabela. As siglas AN1, AN2, ... ANN, referem-se às anomalias catalogadas e as siglas DR01, DR02 e DR03 referem-se aos DR analisados.

A ferramenta *RETA* foi avaliada com base em 7 (sete) tipos de anomalias de requisitos, citadas na Seção 4.1. As Tabelas 4.1, 4.2 e 4.3, apresentam, respectivamente, os resultados do processo de detecção de anomalias para os DR01, DR02 e DR03.

De maneira geral, essas tabelas foram organizadas em 4 (quatro) partes: (a) as duas primeiras linhas apresentam o tipo e a quantidade de ocorrências totais para cada uma das anomalias detectadas. Essa quantidade de ocorrências é exposta tanto para o oráculo (detecção manual), quanto para a ferramenta em análise (detecção automática); (b) são apresentados os resultados da análise da detecção, isto é, dadas as anomalias detectadas pela ferramenta, em contraste com a detecção manual do oráculo, quais as anomalias foram detectadas como Falso Negativo (FN), Falso Positivo (FP) e quais anomalias foram detectadas corretamente (OK); e (c) as duas últimas linhas da tabela apresentam o percentual das métricas de cobertura e precisão para cada anomalia de requisitos; e (d) por fim, para cada linha, é exibido na última coluna, o

[6] <https://shre.ink/HMbG>
[7] <https://shre.ink/HMbS>
[8] <https://shre.ink/HMbM>

valor total referente ao somatório dos valores apresentados nas colunas de anomalias de requisitos. Nas células em que não foi possível calcular valores para as métricas, foi utilizado um traço (“-”). A guisa de exemplo, a métrica cobertura é a quantidade de anomalias identificadas corretamente por uma ferramenta. Se uma determinada anomalia não aparece no `DR`, não quer dizer que a cobertura é 0%, na verdade, não foi possível calculá-la, pois não há anomalia desse tipo no `DR`.

Tabela 4.1 – Resultado - detecção de anomalias no `DR01` pela *RETA*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**AN7**	**Total**
ORÁCULO	3	1	22	1	0	0	0	27
RETA	2	1	29	1	19	18	2	72
FN	1	0	3	0	0	0	0	4
FP	0	0	10	0	19	18	2	49
OK	2	1	19	1	0	0	0	23
Cobertura	66,67%	100%	86,36%	100%	-	-	-	85,19%
Precisão	100%	100%	65,52%	100%	0%	0%	0%	31,94%

Fonte: Do Autor (2023)

Tabela 4.2 – Resultado - detecção de anomalias no `DR02` pela *RETA*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**AN7**	**Total**
ORÁCULO	2	1	6	0	0	0	1	10
RETA	0	0	31	0	32	15	6	84
FN	2	1	0	0	0	0	1	4
FP	0	0	25	0	32	15	6	78
OK	0	0	6	0	0	0	0	6
Cobertura	0%	0%	100%	-	-	-	0%	60%
Precisão	-	-	19,35%	-	0%	0%	0%	7,14%

Fonte: Do Autor (2023)

Tabela 4.3 – Resultado - detecção de anomalias no `DR03` pela *RETA*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**AN7**	**Total**
ORÁCULO	12	13	18	1	0	0	0	36
RETA	6	10	57	14	68	31	29	162
FN	9	9	2	0	0	0	0	20
FP	7	2	41	13	68	31	29	191
OK	3	4	16	1	0	0	0	24
Cobertura	25%	30,77%	88,89%	100%	-	-	-	54,55%
Precisão	30%	66,67%	28,07%	7,14%	0%	0%	0%	11,16%

Fonte: Do Autor (2023)

A Tabela 4.4 apresenta o resultado geral dos testes, considerando os 3 (três) DR utilizados nos testes. Os cálculos utilizados para obter-se as métricas de cobertura e precisão geral seguiram o mesmo raciocínio do cálculo por DR, porém, considerando o somatório geral das variáveis FN, FP e OK.

Tabela 4.4 – Resultado geral - detecção de anomalias pela *RETA*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**AN7**	**Total**
FN	12	10	5	0	0	0	1	28
FP	7	2	76	13	119	64	37	318
OK	5	5	41	2	0	0	0	53
Cobertura	29,41%	33,33%	89,13%	100%	-	-	0%	65,43%
Precisão	41,67%	71,43%	35,04%	13,33%	0%	0%	0%	14,29%

Fonte: Do Autor (2023)

4.2.2 Resultados da ferramenta *Tactile Check*

A ferramenta *Tactile Check* foi avaliada com base em 6 (seis) tipos de anomalias de requisitos, citadas na Seção 4.1. As Tabelas 4.5, 4.6 e 4.7, apresentam, respectivamente, os resultados do processo de detecção de anomalias para os DR01, DR02 e DR03.

Tabela 4.5 – Resultado - detecção de anomalias no DR01 pela *Tactile Check*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**Total**
ORÁCULO	3	1	22	1	0	0	27
Tactile C.	1	9	19	0	0	3	32
FN	2	1	4	1	0	0	8
FP	0	9	2	0	0	3	14
OK	1	0	18	0	0	0	19
Cobertura	33,33%	0%	81,82%	0%	-	-	70,37%
Precisão	100%	0%	90%	-	-	0%	57,58%

Fonte: Do Autor (2023)

A Tabela 4.8 apresenta o resultado geral dos testes considerando os 3 (três) DR utilizados nos testes (DR01, DR01 e DR01).

4.2.3 Resultados da ferramenta *Tiger Pro*

A ferramenta *Tiger Pro* foi avaliada com base em 4 (quatro) tipos de anomalias de requisitos, citadas na Seção 4.1. As Tabelas 4.9, 4.10 e 4.11, apresentam, respectivamente, os resultados do processo de detecção de anomalias para os DR01, DR02 e DR03.

Tabela 4.6 – Resultado - detecção de anomalias no `DR02` pela *Tactile Check*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**Total**
ORÁCULO	2	2	8	0	0	0	12
Tactile C.	0	5	8	0	0	10	23
FN	2	0	0	0	0	0	2
FP	0	3	0	0	0	10	13
OK	0	2	8	0	0	0	10
Cobertura	0%	100%	100%	-	-	-	83,33%
Precisão	0%	40%	100%	-	-	0%	43,48%

Fonte: Do Autor (2023)

Tabela 4.7 – Resultado - detecção de anomalias no `DR03` pela *Tactile Check*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**Total**
ORÁCULO	11	20	20	3	0	2	48
Tactile C.	0	22	16	2	0	15	42
FN	12	6	4	1	0	0	23
FP	0	8	0	0	0	14	22
OK	0	14	16	2	0	0	34
Cobertura	25%	70%	80%	66,67%	-	100%	59,65%
Precisão	-	63,64%	100%	100%	-	12,50%	60,71%

Fonte: Do Autor (2023)

Tabela 4.8 – Resultado geral - detecção de anomalias pela *Tactile Check*

	AN1	**AN2**	**AN3**	**AN4**	**AN5**	**AN6**	**Total**
FN	16	7	8	2	0	0	33
FP	0	20	2	0	0	27	49
OK	1	16	42	2	0	2	63
Cobertura	5,88%	69,57%	84%	50%	-	100%	65,63%
Precisão	100%	44,44%	95,45%	100%	-	6,90%	56,25%

Fonte: Do Autor (2023)

Tabela 4.9 – Resultado - detecção de anomalias no `DR01` pela *Tiger Pro*

	AN1	**AN2**	**AN3**	**AN4**	**Total**
ORÁCULO	3	1	22	0	26
Tiger Pro	1	3	18	0	22
FN	2	1	10	0	13
FP	0	3	6	0	9
OK	1	1	12	0	13
Cobertura	33,33%	0%	54,55%	-	50,00%
Precisão	100%	0%	66,67%	-	59,09%

Fonte: Do Autor (2023)

Tabela 4.10 – Resultado - detecção de anomalias no DR02 pela *Tiger Pro*

	AN1	**AN2**	**AN3**	**AN4**	**Total**
ORÁCULO	2	3	8	0	13
Tiger Pro	0	8	20	2	30
FN	2	0	5	0	7
FP	0	5	17	2	24
OK	0	3	3	1	6
Cobertura	0%	100%	37,50%	-	46,15%
Precisão	-	37,50%	15,00%	0%	20%

Fonte: Do Autor (2023)

Tabela 4.11 – Resultado - detecção de anomalias no DR03 pela *Tiger Pro*

	AN1	**AN2**	**AN3**	**AN4**	**Total**
ORÁCULO	11	21	22	2	48
Tiger Pro	4	17	26	5	43
FN	9	7	13	2	31
FP	1	3	17	5	26
OK	3	14	9	0	26
Cobertura	25%	66,67%	40,91%	0%	45,61%
Precisão	75%	82,35%	34,62%	0%	50%

Fonte: Do Autor (2023)

A Tabela 4.12 apresenta o resultado geral dos testes considerando os 3 (três) DR utilizados nos testes (DR01, DR01 e DR01).

Tabela 4.12 – Resultado geral - detecção de anomalias pela *Tiger Pro*

	AN1	**AN2**	**AN3**	**AN4**	**Total**
FN	13	8	28	2	51
FP	1	11	40	7	59
OK	4	17	24	0	45
Cobertura	23,53%	68%	46,15%	0%	46,88%
Precisão	80%	60,71%	37,50%	0%	43,27%

Fonte: Do Autor (2023)

A métrica *F-measure* foi considerada neste trabalho para obter uma pontuação média da ferramenta concernente à sua precisão e cobertura. Com base nos valores dessas métricas, foi calculada a métrica *F-measure*, como pode ser observado na Tabela 4.13.

Tabela 4.13 – Resultado geral - detecção de anomalias pelas 3 ferramentas

Ferramenta	Cobertura	Precisão	*F-measure*
RETA	65,43%	14,29%	23,45%
Tactile Check	65,63%	56,25%	60,58%
Tiger Pro	46,88%	43,27%	45,01%

Fonte: Do Autor (2023)

4.3 Etapa 5 - Identificar pontos fortes e fracos

A etapa de identificar pontos fortes e fracos teve como objetivo realizar uma discussão dos resultados do processo de detecção de anomalias, bem como destacar as características positivas e negativas percebidas durante a utilização das ferramentas nas avaliações.

A ferramenta ***RETA*** apresentou uma cobertura geral de 65,43% e a precisão de 14,29%. Nas Tabelas 4.1, 4.2 e 4.3 que, respectivamente, apresentam os resultados das avaliações dos `DR01`, `DR02` e `DR03`, é possível perceber que, no geral, embora a cobertura tenha um valor acima dos 50%, a precisão está bem abaixo da metade desse valor, em todos os casos. Destaca-se que, no `DR02`, a diferença entre a precisão e a cobertura é bem discrepante, em que a precisão é de apenas 7.14%, diante da cobertura de 60%. Isso se deve ao grande número de falsos positivos referente à anomalia `AN3` (lacunas e brechas), causado pela forma de descrição dos requisitos em que são utilizados muitas vezes os termos "*support, include* e *use*", que a ferramenta RETA entende como anomalias.

Percebe-se que, apesar de a *RETA* detectar grande parte de anomalias de requisitos catalogados, o resultado dessa detecção traz consigo muitos falsos positivos, o que pode comprometer negativamente o esforço do Engenheiro de Requisitos na análise do `DR`. Possivelmente, isso se deve ao fato dela não analisar o contexto ao qual o termo está inserido, capturando, por exemplo, tudo que é conector do tipo "*OR*" e demarcando-o como uma anomalia.

O `DR` em que a *RETA* se saiu melhor foi o `DR01`, em que a maioria das anomalias encontradas foi do tipo `AN3` (lacunas e brechas). Quanto ao `DR03`, a menor precisão foi concernente à anomalia `AN4` (palavras negativas). A correlação desta anomalia catalogada foi feita com a anomalia da *RETA* "*Warm Adverb in Verb Phrase*". Os advérbios que são compatíveis entre essa correlação são os advérbios de negação. Porém, a *RETA* considera outros advérbios (além dos de negação) em sua detecção. Para esta pesquisa, eles foram considerados como falsos positivos, pois não há relação desses demais advérbios com as anomalias catalogadas.

Apesar de não estar disponível o acesso ao dicionário da ferramenta *RETA*, se comparado às demais ferramentas, ela foi a que teve maior abrangência de correlação com as anomalias catalogadas. Porém, como já dito, a precisão da ferramenta é muito baixa se comparada à sua cobertura. Essa característica poderia ser melhorada se a ferramenta utilizasse uma análise contextual para identificar se um termo é (ou não) uma anomalia.

A ferramenta ***Tactile Check*** apresentou uma cobertura geral de 65,63% e a precisão de 56,25%. Nas Tabelas 4.5, 4.6 e 4.7 que, respectivamente, apresentam os resultados das avaliações dos `DR01`, `DR02` e `DR03`, é possível perceber que o percentual da precisão da detecção das anomalias atingiram, no mínimo 50%, da cobertura em todos os casos. Destaque para a análise do `DR03`, em que o valor da precisão se equiparou com o valor da cobertura. Esse é o melhor cenário em relação à eficácia no processo de detecção de anomalias, isto é, quando o percentual da precisão se equipara com o percentual da cobertura.

Nota-se que o maior número de falsos positivos ocorreu na avaliação da `DR02`, referente à anomalia `AN2` (termos abertos e não verificáveis). Isso se deve pelo fato de os termos "*may*" e "*can*" serem usados amplamente na descrição dos requisitos e serem considerados pela *Tactile check* como anomalias.

O tipo de anomalia detectada amplamente e corretamente em todos os `DR` pela *Tactile Check*, foi a `AN3` (lacunas e brechas). Isso pode ter ocorrido por causa da seleção das palavras que compõem o dicionário da ferramenta estar mais calibrada com os termos que configuram essa anomalia. A anomalia `AN6` (Linguagem subjetiva) teve o maior número de falsos positivos nas análise de todos os `DR`. Isso se deve, provavelmente, pelo fato da *Tactile Check* ignorar o contexto e a gramática no processo de detecção das anomalias de requisitos (WILMINK; BOCKISCH, 2017). Comparando a cobertura da *Tactile Check* com a das demais ferramentas, percebe-se que ela fica logo abaixo da ferramenta *RETA*, isto é, possui a segunda maior abrangência de correlação com as anomalias catalogadas. Embora a *Tactile Check* tenha apresentado uma precisão considerável nas anomalias do tipo `AN2` e `AN3`, acredita-se que, em detrimento das demais anomalias que tiveram muitos falsos positivos, uma análise contextual no processo de identificação das anomalias de requisitos pode resultar no aumento da precisão da ferramenta.

A ferramenta ***Tiger Pro*** apresentou uma cobertura geral de 46,88% e a precisão de 43,27%. Nas Tabelas 4.9, 4.10 e 4.11 que, respectivamente, apresentam os resultados das ava-

liações dos DR01, DR02 e DR03, é possível perceber, de maneira geral, que o percentual da cobertura não ultrapassou os 50% em nenhum dos casos.

Observa-se que, apesar da *Tiger Pro* detectar uma pequena parte de anomalias de requisitos catalogados, o resultado dessa detecção é de alta precisão, isto é, traz consigo poucos falsos positivos. Possivelmente, isso se deve ao fato da seleção das palavras que compõem o dicionário da ferramenta estar mais calibrada com os termos que configuram as anomalias que a ferramenta lida.

O DR em que a *Tiger Pro* se saiu melhor foi o DR01, em que a maior parte das anomalias encontradas foram do tipo AN3 (lacunas e brechas). Isso pode ter ocorrido pela maneira em que foram descritos os requisitos. Da mesma forma que ocorre nas demais ferramentas, a *Tiger Pro* não analisa o contexto para marcar um determinado termo como anomalia, logo, vai depender do contexto onde foi inserido o termo para saber se trata (ou não) de uma anomalia. Nesses casos, se o dicionário não é bem elaborado, a demarcação de um termo sem a contextualização pode gerar muitos falsos positivos impactando negativamente na análise do engenheiro de requisitos.

Os DR01 e DR03 tiveram um percentual de cobertura e de precisão bem equiparados. As anomalias que, em ambos os casos, foram mais bem identificadas foram AN2 (Termos abertos e não verificáveis) e AN3 (lacunas e brechas). Entretanto, na análise do DR02, o total da precisão ficou menos da metade do total do percentual da precisão. Essa discrepância se deve pelo alto número de falsos positivos encontrados na detecção da anomalia AN3. Nota-se que essa quantidade de falsos positivos foi atingida devido ao termo "*include*", que foi amplamente empregado no DR02 e a *Tiger Pro* a entende como uma anomalia de requisitos.

A cobertura da *Tiger Pro*, se comparada com as demais ferramentas, é a menor delas, isto é, possui a menor abrangência de correlação com as anomalias catalogadas. Entretanto, de maneira geral, a precisão é alta. De igual forma com as demais ferramentas, acredita-se que uma análise contextual no processo de identificação das anomalias de requisitos, pode resultar no aumento da precisão da ferramenta.

Os valores da métrica *F-measure* para as ferramentas *RETA*, *Tactile Check* e *Tiger Pro*, são, respectivamente, 23,45%, 60,68% e 45%. Logo, a ferramenta que melhor se saiu nas análises realizadas foi a *Tactile Check*. Apesar de a ferramenta *Tactile Check* ter se destacado nas avaliações, tem-se que nem ela, e nem as demais ferramentas tiveram um nível satisfatório

de cobertura e precisão. Entende-se que, para o uso profissional dessas ferramentas, ainda haveria um considerável esforço dos engenheiros de requisitos para revisão das detecções das anomalias. Logo, uma possível solução seria a realização de melhorias quanto às técnicas empregadas na detecção das anomalias para o aumento da confiabilidade das ferramentas.

Todas as ferramentas foram executadas com os parâmetros padrão (pós-instalação), ou seja, não houve qualquer tipo de personalização. Além das métricas de precisão e cobertura, foram considerados 3 (três) processos principais na análise comparativa das ferramentas: (a) preparação do ambiente e instalação; (b) operação, uso e manuseio; e (c) apresentação dos resultados. Destaca-se que, alguns critérios observados durante a instalação, configuração e operação das 3 (três) ferramentas são de caráter subjetivo. Logo, alguns termos como "complexo", " intuitivo", "mais fácil", "apenas", "simples", dentre outros, foram utilizados para categorizar processos que requisitaram menor (ou maior) esforço de pesquisa e/ou passos necessários para a sua execução.

Todas as ferramentas foram disponibilizadas há alguns anos para uso. A guisa de exemplo, a ferramenta *Tiger Pro* foi desenvolvida no ano de 2004. Logo, a execução delas seguem um conceito mais antigo de desenvolvimento, não possuindo tecnologias mais recentes como, por exemplo, computação em nuvens. Assim sendo, todas as ferramentas rodam localmente, sem ser possível a realização do compartilhamento dos trabalhos em equipe pela *internet* de forma nativa.

No processo de **preparação do ambiente e instalação**, foram percebidos os seguintes pontos positivos e negativos para cada ferramenta:

a) ***RETA***:

- **Pontos positivos**:
    a) A ferramenta é multiplataforma e foi testada nos sistemas operacionais *Windows* e *Linux*;
    b) No que tange ao ambiente (sistema operacional), tem-se como pré-requisito, apenas a versão 9 ou superior do `JAVA` instalada;
    c) Possui documentação para instalação e configuração.
- **Pontos negativos**: nenhum relevante.

b) ***Tactile Check***:

- **Pontos positivos**:

a) A ferramenta roda no *Microsoft Word*, que é um *software* amplamente utilizado para especificação de requisitos;
b) Não é necessário realizar instalação;
c) Possui documentação para operação e parametrização.

- **Pontos negativos**:

a) A ferramenta requer a execução de *macros*, que exigem que todos os parâmetros de segurança, concernentes à execução de *macros*, sejam desativados.

c) ***Tiger Pro***:

- **Pontos positivos**:

a) A ferramenta possui apenas um executável que já instala todas as dependências necessárias para rodá-la;
b) Possui documentação para instalação e configuração.

- **Pontos negativos**:

a) Roda apenas na plataforma *Windows*.

Quanto ao processo de **operação, uso e manuseio** das ferramentas, a experiência divergiu bastante de acordo com a tecnologia empregada para o desenvolvimento da ferramenta. Destaca-se que todas as ferramentas aceitaram os `DR` no formato de texto plano com o *layout* "ID - Descrição". Quanto ao uso e operação das ferramentas, foram percebidos os seguintes pontos positivos e negativos:

• ***RETA***:

- **Pontos positivos**:

a) Permite vários formatos de entrada para `DR`: *.txt*, *.xml*, *.html*, dentre outros.

- **Pontos negativos**:

a) Não possui documentação de operação. Vale ressaltar que a ferramenta *GATE* possui uma extensa documentação, entretanto, a ferramenta *RETA* (que roda sob o *GATE*) não possui um manual de utilização;
b) Não permite editar parâmetros de detecção de anomalias;
c) A interface do *GATE* não é intuitiva e não fica claro como realizar a sequência de comandos necessária para a detecção de anomalias.
d) O processo para detecção é moroso, há muitas configurações necessárias.

- ***Tactile Check***:
  - **Pontos positivos**:
    a) Interface gráfica intuitiva de fácil compreensão e aprendizado;
    b) Permite parametrizar as configurações de detecção de anomalias de requisitos de maneira bem simples;
    c) Nos casos em que a especificação esteja sendo feita no *Microsoft Word*, a detecção será feita sem a necessidade de acessar um ambiente externo.
  - **Pontos negativos**:
    a) Não é possível importar um `DR` para a detecção das anomalias. É necessário que os requisitos sejam transcritos do `DR` para o documento (*template*) do *Microsoft Word*, que possui as `macros`;
    b) Necessidade da compra de licença para utilização do *Microsoft Word*.
- ***Tiger Pro***:
  - **Pontos positivos**:
    a) Permite `DR` nos formatos *.txt* e *.CSV*;
    b) Possui documentação de utilização;
    c) Interface gráfica de fácil compreensão e aprendizado.
  - **Pontos negativos**:
    a) Roda apenas na plataforma *Windows*;
    b) A licença para estudante não permite a parametrização das configurações de detecção das anomalias de requisitos.

De maneira geral, a detecção de uma anomalia de requisitos de *software* consiste em destacar no texto do `DR` os termos que apresentam um problema em potencial, categorizando-a quanto ao seu tipo. Esse processo pode ser feito de várias formas e, dependendo da técnica empregada, pode contribuir ou dificultar a análise, por parte Engenheiro de Requisitos. Logo, no quesito **apresentação dos resultados**, foram percebidos os seguintes pontos positivos e negativos de cada ferramenta:

- ***RETA***:
  - **Pontos positivos**:
    a) Destaca cada anomalia com uma cor diferente;

b) Permite selecionar uma ou várias anomalias de uma única vez;
c) Possui legenda colorida informando o tipo da anomalia.

- **Pontos negativos**:

a) Não gera um relatório com o resultado da detecção;
b) Não permite exportar o resultado para um arquivo externo;
c) Quando selecionadas todas as anomalias, a visualização fica bem confusa e, em alguns casos, incompreensível por sobrepor as cores quando um termo atende mais de uma anomalia.

- ***Tactile Check***:

- **Pontos positivos**:

a) Destaca as anomalias, de acordo com seu tipo;
b) Cada anomalia possui uma explicação logo ao lado do documento, com uma "*tag*".

- **Pontos negativos**:

a) nenhum relevante.

- ***Tiger Pro***:

- **Pontos positivos**:

a) Sumariza o resultado pelo tipo de anomalia, apresentando logo abaixo de cada requisito o termo, a quantidade e o tipo de anomalia detectada;
b) Permite exportar o resultado em *.CSV*;
c) Apresenta o resultado através de um relatório em tela.

- **Pontos negativos**:

a) nenhum relevante.

Considerando a métrica *F-measure* e os pontos fortes e fracos quanto à preparação do ambiente para instalação, à utilização e à forma de apresentação dos resultados das ferramentas, a *Tactile check* se destacou entre as demais avaliadas. Entretanto, a questão de ter que desabilitar todos os mecanismos de segurança concernente à execução de macros para seu funcionamento, pode invalidar sua escolha para uso em um projeto real.

5 CONSIDERAÇÕES FINAIS

Garantir a qualidade dos requisitos de um *software* é essencial para que as demais atividades do ciclo de desenvolvimento ocorram com sucesso (VALENTE, 2020). Apesar de existirem diversas ferramentas para detecção de anomalias de requisitos, há escassez de estudos cujo intuito seja prover uma análise comparativa entre ferramentas já existentes. Com o intuito de mitigar essa deficiência, esta dissertação apresentou uma análise comparativa entre ferramentas que realizam detecção automática de anomalias de requisitos em Linguagem Natural e, após a avaliação dessas ferramentas, elencar os pontos fortes e fracos das mesmas.

Para realização deste estudo, foi realizado um experimento controlado para investigar as métricas de precisão e de cobertura de 3 (três) ferramentas selecionadas, a saber, *RETA*, *Tactile Check* e *Tiger Pro*. Embora haja várias ferramentas disponíveis no mercado que se propõem a realizar a detecção automática de anomalias de requisitos, apenas essas 3 (três) atenderam aos critérios de aceitação citados na Seção 3.2. As ferramentas selecionadas foram submetidas a 3 (três) Documentos de Requisitos (`DR`), extraídos de um repositório público, denominado `PURE`, a fim de testar a sua eficácia em diferentes cenários.

A ferramenta que teve a maior cobertura foi a *Tactile Check*, tendo identificado 65,63% das anomalias existentes. Embora a ferramenta *RETA* tenha obtido uma cobertura de 65,43%, teve o menor índice de precisão, uma vez que muitos falsos positivos foram indicados por ela. Isso pode ter acontecido pelo fato da ferramenta não considerar o contexto do qual o termo anômalo esteja inserido ou o dicionário não esteja bem calibrado com as anomalias que a ferramenta se dispõe a detectar. O maior valor de precisão foi encontrado na ferramenta *Tiger Pro*, entretanto, ela obteve o menor índice de cobertura. Ou seja, trata-se de uma ferramenta que identifica poucas anomalias existentes, mas que, em contrapartida, gera menos falsos positivos. Provavelmente, isso se deve ao fato da qualidade do seu dicionário, uma vez que é seu método primário de detecção de ambiguidade (ARENDSE; LUCASSEN, 2016). Com base na média entre cobertura e precisão, a ferramenta *Tactile Check* foi a que teve o melhor resultado e, portanto, a que se destacou nas avaliações entre as demais ferramentas. Apesar da ferramenta *Tactile Check* ter se destacado nas avaliações, considera-se que nem ela, e nem as demais ferramentas tiveram um nível satisfatório de cobertura e precisão. Atribui-se isso ao fato de nenhuma ferramenta realizar uma análise contextual para marcar um termo como anomalia de requisito. Quanto à preparação do ambiente para instalação, operação e apresentação dos resultados das

ferramentas, a *Tactile check* também se destacou entre as demais avaliadas. Entretanto, a questão de ter que desabilitar todos os mecanismos de segurança concernente à execução de macros para seu funcionamento, pode invalidar sua escolha para uso em um projeto real.

Dada a importância de se ter um DR de qualidade e o entendimento que uma anomalia de requisito pode prejudicar as etapas seguintes do ciclo de desenvolvimento do *software*, esse trabalho trouxe as seguintes contribuições científicas/tecnológicas: (i) atualização de um MSL sobre anomalias de requisitos; (ii) cobertura de um *gap* de pesquisa sobre análise de ferramentas para identificação de anomalias de requisitos; (iii) base de dados composta por 3 DR[1] [2] [3] com anomalias identificadas e categorizadas; e (iv) elaboração de um conjunto de pontos positivos e negativos para as ferramentas analisadas.

Como principais limitações desta pesquisa, pode-se elencar:

a) Quantidade baixa de ferramentas utilizadas no experimento. Mesmo havendo na literatura um número alto de abordagens e ferramentas para detecção de anomalias, na prática, em contato com os autores, a maioria já não está mais disponível ou foi descontinuada. Algumas foram rejeitadas da análise por serem ferramentas comerciais e não possuírem licença gratuita para o uso em pesquisas científicas.

b) Quantidade limitada de DR. Foram utilizados poucos DR nas avaliações, apenas 3 (três). Entretanto, buscou-se DR de diferentes domínios, mais robustos e mais alinhados com a indústria.

c) Elaboração do oráculo. A definição e a elaboração do oráculo foram realizadas manualmente. Com o intuito de mitigar qualquer erro na detecção das anomalias nos DR, todo o processo foi realizado e revisado por mais de um pesquisador.

d) Mapeamento "De-Para" entre as anomalias catalogadas e as detectadas pelas ferramentas. A relação entre as anomalias utilizadas pelas ferramentas e as anomalias catalogadas foi feita manualmente. Além disso, nos casos em que uma mesma anomalia utilizada pela ferramenta foi mapeada para mais de uma das anomalias catalogadas, a análise de qual das anomalias catalogadas estava realmente presente em um determinado requisito

[1] <https://shre.ink/HMbG>
[2] <https://shre.ink/HMbS>
[3] <https://shre.ink/HMbM>

também foi realizada de forma manual. Contudo, todos esses processos foram avaliados e revisados por mais de um pesquisador com experiência em Engenharia de Requisitos.

e) Avaliação apenas pelo autor da pesquisa. Não houve avaliação das ferramentas com usuários finais, logo, os critérios subjetivos avaliados nesta pesquisa foram realizados apenas pela perspectiva do autor da pesquisa.

Com base nas contribuições e limitações apresentadas neste capítulo, algumas propostas para trabalhos futuros podem ser citadas:

a) Extensão da pesquisa envolvendo mais ferramentas e mais `DR`, a fim de ampliar a segurança dos resultados obtidos e prover mais estudos sobre análise comparativa de ferramentas de detecção de anomalias de requisitos disponíveis no mercado.

b) Envolver os profissionais da indústria na avaliação das ferramentas, a fim de tornar o experimento mais alinhado com a atividade profissional.

c) Desenvolver uma ferramenta *open-source* que abarque os pontos positivos das 3 (três) ferramentas avaliadas.

REFERÊNCIAS

ARENDSE, B.; LUCASSEN, G. Toward tool mashups: Comparing and combining nlp re tools. In: **2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)**. [S.l.: s.n.], 2016. p. 26–31.

ARORA, C. et al. Automated checking of conformance to requirements templates using natural language processing. **IEEE Transactions on Software Engineering**, v. 41, p. 1–1, 10 2015.

ARORA, C. et al. Automated extraction and clustering of requirements glossary terms. **IEEE Transactions on Software Engineering**, v. 43, n. 10, p. 918–945, 2017.

ASADABADI, M. et al. Hidden fuzzy information: Requirement specification and measurement of project provider performance. **Fuzzy Sets and Systems**, 06 2019.

ASADABADI, M. R. et al. Ambiguous requirements: A semi-automated approach to identify and clarify ambiguity in large-scale projects. **Computers & Industrial Engineering**, v. 149, p. 106828, 2020. ISSN 0360-8352. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360835220305313>.

BEER, A. et al. Initial investigations on the influence of requirement smells on test-case design. In: **2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)**. [S.l.: s.n.], 2017. p. 323–326.

BERRY, D. M.; KAMSTIES, E.; KRIEGER, M. M. From contract drafting to software specification: Linguistic sources of ambiguity. **Univ. of Waterloo, Tech. Rep**, 2003.

BäUMER, F.; GEIERHOS, M. Flexible ambiguity resolution and incompleteness detection in requirements descriptions via an indicator-based configuration of text analysis pipelines. In: . [S.l.: s.n.], 2018.

CALAZANS, A. T. S. et al. Quality requirements and the requirements quality: The indications from requirements smells in a financial institution systems. In: **Proceedings of the XXXIII Brazilian Symposium on Software Engineering**. [S.l.: s.n.], 2019. p. 472–480.

CHITCHYAN, R. et al. In: **A Tool Suite for Aspect-Oriented Requirements Engineering**. New York, NY, USA: Association for Computing Machinery, 2006. (EA '06), p. 19–26. ISBN 1595934057.

CIFUENTES, C.; FITZGERALD, A. Copyright in shareware software distributed on the internet—the trumpet winsock case. In: **Proceedings of the 19th international conference on Software engineering**. [S.l.: s.n.], 1997. p. 456–464.

DEURSEN, A. et al. Refactoring test code. In: **Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)**. [S.l.: s.n.], 2001. p. 92–95.

ELRAKAIBY, Y. et al. Using argumentation to explain ambiguity in requirements elicitation interviews. In: **2017 IEEE 25th International Requirements Engineering Conference (RE)**. [S.l.: s.n.], 2017. p. 51–60.

FABBRI, S. C. P. F.; FERRARI, F. C.; CAMARGO, K. G. A atividade de teste sob a perspectiva de qualidade de software. **Revista TIS**, v. 2, n. 3, p. 164–166, 2014.

FEMMER, H. Reviewing natural language requirements with requirements smells – a research proposal –. In: . [S.l.: s.n.], 2013.

FEMMER, H. Automatic requirements reviews - potentials, limitations and practical tool support. In: . [S.l.: s.n.], 2017. p. 617–620. ISBN 978-3-319-69925-7.

FEMMER, H. et al. Rapid requirements checks with requirements smells: Two case studies. In: . New York, NY, USA: Association for Computing Machinery, 2014. (RCoSE 2014), p. 10–19. ISBN 9781450328562. Disponível em: <https://doi.org/10.1145/2593812.2593817>.

FEMMER, H. et al. Quality assurance of requirements artifacts in practice: A case study and a process proposal. In: . [S.l.: s.n.], 2016. p. 506–516. ISBN 978-3-319-49093-9.

FEMMER, H. et al. Rapid quality assurance with requirements smells. **Journal of Systems and Software**, v. 123, p. 190–213, 2017. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121216000789>.

FERNANDES, J.; MACHADO, R. **Requisitos em projetos de software e de sistemas de informação**. Novatec Editora, 2017. ISBN 9788575225660. Disponível em: <https://books.google.com.br/books?id=X0TCDgAAQBAJ>.

FERRARI, A. Natural language requirements processing: from research to practice. In: . [S.l.: s.n.], 2018. p. 536–537.

FERRARI, A.; DONATI, B.; GNESI, S. Detecting domain-specific ambiguities: An nlp approach based on wikipedia crawling and word embeddings. In: . [S.l.: s.n.], 2017. p. 393–399.

FERRARI, A. et al. Detecting requirements defects with nlp patterns: an industrial experience in the railway domain. **Empirical Software Engineering**, Springer, v. 23, n. 6, p. 3684–3733, 2018.

FERRARI, A.; SPAGNOLO, G.; GNESI, S. Pure: A dataset of public requirements documents. In: . [S.l.: s.n.], 2017. p. 502–505.

FERRARI, A. et al. Interview review: Detecting latent ambiguities to improve the requirements elicitation process. In: . [S.l.: s.n.], 2017. p. 400–405.

FOWLER, M. **Refactoring: improving the design of existing code**. [S.l.]: Addison-Wesley Professional, 1999.

GARNER, P. et al. When and how to update systematic reviews: Consensus and checklist. **BMJ**, v. 354, p. i3507, 07 2016.

HABIB, M. K.; WAGNER, S.; GRAZIOTIN, D. Detecting requirements smells with deep learning: Experiences, challenges and future work. 08 2021.

HERRERA, J. et al. Revealing crosscutting concerns in textual requirements documents: an exploratory study with industry systems. In: IEEE. **2012 26th Brazilian Symposium on Software Engineering**. [S.l.], 2012. p. 111–120.

HU, W. et al. Using human error information for error prevention. **Empirical Software Engineering**, v. 23, 12 2018.

IQBAL, T.; ELAHIDOOST, P.; LUCIO, L. A bird's eye view on requirements engineering and machine learning. In: . [S.l.: s.n.], 2018.

ISO, I. O. for S. **ISO/IEC/IEEE 29148: 2011–Systems and software engineering—Life cycle processes—Requirements engineering**. [S.l.]: ISO ISO/IEC/IEEE Switzerland, 2011.

KASSER, J. et al. **Prototype educational tools for systems and software (pets) engineering**. Tese (Doutorado) — Australasian Association for Engineering Education, 2003.

KIYAVITSKAYA, N. et al. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. **Requirements engineering**, Springer, v. 13, n. 3, p. 207–239, 2008.

KROTH, E. Emprego de técnicas de representação do conhecimento como forma de apoio à engenharia de requisitos. In: **XVIII Congreso Argentino de Ciencias de la Computación**. [S.l.: s.n.], 2012.

LUCASSEN, G. et al. Improving agile requirements: the quality user story framework and tool. **Requirements Engineering**, v. 21, 09 2016.

MACHADO, F. **Análise e Gestão de Requisitos de Software – Onde nascem os sistemas**. Saraiva Educação S.A., 2018. ISBN 9788536509693. Disponível em: <https://books.google.com.br/books?id=MYdiDwAAQBAJ>.

NAKACHE, D.; METAIS, E.; TIMSIT, J. F. Evaluation and nlp. In: SPRINGER. **International Conference on Database and Expert Systems Applications**. [S.l.], 2005. p. 626–632.

NASCIMENTO, R. et al. Requirements smells como indicadores de má qualidade na especificação de requisitos: Um mapeamento sistemático da literatura. In: **WER**. [S.l.: s.n.], 2018.

NASCIMENTO, R.; GUIMARãES, E.; LUCENA, M. Requirements smells como indicador de qualidade para histórias de usuários: Estudo exploratório. In: **WER**. [S.l.: s.n.], 2021.

PAGLIUSO, P. et al. Gvr - guia de validação de requisitos baseados nas técnicas pbr e ad hoc resultante de um estudo de. In: . [S.l.: s.n.], 2020. p. 183–197.

PALDêS, R. et al. A utilização da linguagem natural na especificação de requisitos: um estudo por meio das equações estruturais. In: . [S.l.: s.n.], 2016.

PERINI, M. A. **Gramática descritiva do português brasileiro**. [S.l.]: Editora Vozes Limitada, 2017.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1–18, 2015. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584915000646>.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**. 9th. ed. [S.l.]: AMGH, McGraw-Hill, 2021.

ROSADINI, B. et al. Using nlp to detect requirements defects: An industrial experience in the railway domain. In: . [S.l.: s.n.], 2017. v. 10153, p. 344–360. ISBN 978-3-319-54044-3.

SASAKI, Y. et al. The truth of the f-measure. **Teach tutor mater**, v. 1, n. 5, p. 1–5, 2007.

SOMMERVILLE, I. **Engenharia de software**. 9th. ed. [S.l.]: Pearson Education, 2011.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [S.l.]: Independente, 2020.

WILMINK, M.; BOCKISCH, C. On the ability of lightweight checks to detect ambiguity in requirements documentation. In: . [S.l.: s.n.], 2017. v. 10153, p. 327–343. ISBN 978-3-319-54044-3.

ZHAO, L. et al. Natural language processing (nlp) for requirements engineering: A systematic mapping study. In: . [S.l.: s.n.], 2020.

APÊNDICE A – Resultado seleção das ferramentas

Neste apêndice, são apresentados os resultados do processo de seleção das ferramentas para detecção de anomalias, após a aplicação dos critérios descritos na Seção 3.2.

- **1** (`ID` da Tabela **??**): apresentam a ferramenta `RETA` (*REquirements Template Analyzer*). **Situação**: ferramenta aceita.
- **2**: abordam o problema da descrição imprecisa dos requisitos (ambiguidade e incompletude). A ferramenta apresentada auxilia na detecção e na correção automática de anomalias de requisitos. **Situação**: ferramenta rejeitada. Não houve retorno dos autores quanto à disponibilidade e a documentação com a instrução de uso da ferramenta.
- **3**: apresentam uma ferramenta de detecção automática de anomalias de requisitos, a partir de critérios extraídos da norma ISO/IEC/IEEE 29148. **Situação**: ferramenta rejeitada. Em contato com os autores da ferramenta, informaram que os dados para testes são confidenciais e que não há mais informação sobre o que foi implementado. Com isso, não será possível utilizar a ferramenta proposta, pois ela não contempla o critério "Disponibilidade", citado na Seção 3.2.
- **4**: apresentam uma ferramenta chamada *Smella*, apresentada da Seção 2.2 deste trabalho. **Situação**: ferramenta rejeitada. Em contato com os pesquisadores, informaram que a ferramenta tratou-se de um protótipo e que não possuem mais o seu código-fonte.
- **5**: apresentam uma ferramenta para detecção de anomalias de requisitos, utilizando como base outra ferramenta, denominada *GATE* [4]. **Situação:** ferramenta rejeitada. Embora os os pesquisadores tenham enviado as regras *JAPE* (*Java Annotation Patterns Engine*) para serem importadas na ferramenta *GATE* e um *link* para *download* com a explicação dessas regras, não houve as demais informações de configuração necessárias para rodar a aplicação.
- **6**: apresentam uma ferramenta de análise de anomalias de requisitos chamada *Tactile Check* [5]. **Situação:** ferramenta aceita.
- **7**: realizam a comparação de 3 ferramentas existentes no mercado para detecção automática de atomicidade e ambiguidade em 4 conjuntos de requisitos de linguagem natural do

[4] https://gate.ac.uk/download/
[5] https://github.com/mwmk67/TactileCheck

mundo real (conforme apresentado na Seção 2.3 deste trabalho). As ferramentas avaliadas foram: *Holmes* da *Qualicen* [6], *RQA* [7] e *Tiger-PRO* [8]. Todas as ferramentas utilizadas nesse artigo são comerciais. Os proprietários da ferramenta *Holmes - Qualicen* não liberam licença para fins acadêmicos. **Situação: *Holmes - Qualicen***: ferramenta rejeitada. Os proprietários da ferramenta *RQA* não forneceram uma licença para utilização da sua ferramenta em nossa pesquisa. **Situação: *RQA***: ferramenta rejeitada. Uma mensagem de *e-mail* foi enviada para a empresa proprietária da ferramenta *Tiger-PRO* solicitando uma licença para estudantes e pesquisadores, a mesma foi atendida. **Situação: *Tiger-PRO***: ferramenta aceita. Em resumo, *Holmes - Qualicen* (rejeitada), *RQA* (rejeitada) e *Tiger-PRO* (aceita).

- **8**: utilizam a ferramenta comercial *Teamscale* [9] para realizar a detecção automática de anomalias de requisitos. A ferramenta lida apenas com requisitos que estejam vinculados à um código-fonte. Por esse motivo, isto é, por não atender ao critério "formato"da seleção das ferramentas, pois depende de um código-fonte inerente aos requisitos, foi rejeitada. **Situação:** ferramenta rejeitada.
- **9**: trata-se de uma ferramenta computacional para extrair termos relacionados a um glossário, a partir de documentos de requisitos escritos em linguagem natural. A ferramenta não contemplou os critérios de seleção e nem a funcionalidade de interesse desta pesquisa (detecção de anomalias de requisitos). **Situação:** ferramenta rejeitada.
- **10**: os autores citam duas ferramentas: *QuARS Express* e *SREE*. A *QuARS Express* é um analisador automático de requisitos em Linguagem Natural. A ferramenta *SREE* identifica ambiguidades baseada em padrões linguísticos. **Situação:** Quanto à ferramenta *QuARS Express*, devido a licenças de terceiros e problemas de compatibilidade de sistemas operacionais, tanto as ferramentas *QuARS* e *QuARS Express* não são mais mantidas e não estão disponíveis. **Situação: *QuARS Express***: ferramenta rejeitada. Sobre a ferramenta 10 , os autores informaram que não existe mais a ferramenta, logo, não está disponível para teste. **Situação: *SREE***: ferramenta rejeitada.

[6] https://www.qualicen.de/
[7] https://www.reusecompany.com/personalized-requirements-analysis
[8] https://therightrequirement.com/tiger-pro-tool-to-ingest-and-elucidate-requirements/
[9] https://www.cqse.eu/en/teamscale/download/

- **11**: o artigo cita a ferramenta comercial *Scout - Qualicen*. A empresa não fornece licença de uso para estudante e/ou grupo de estudo. **Situação:** ferramenta rejeitada.
- **12**: citam a ferramenta *AQUSA* [10]. A ferramenta trata-se de um protótipo com alto grau de complexidade para sua configuração e utilização. Além disso, a indisponibilidade de um manual de operação, levou a ferramenta a ser descartada dos testes. **Situação:** ferramenta rejeitada.
- **13**: esse trabalho não cita ferramentas para detecção de anomalias de requisitos, apenas ferramentas que lidam com técnicas Processamento de Linguagem Natural, logo, as ferramentas apresentadas não contemplaram os critérios de seleção e nem a funcionalidade de interesse desta pesquisa (detecção de anomalias de requisitos). **Situação:** ferramentas rejeitadas.
- **14**: esse trabalho cita 4 ferramentas, a saber, *QuARS Express*, *SREE*, *Smella* e *Ambiguity Detection*. As duas primeiras ferramentas (*QuARS Express* e *SREE*) já foram citadas no artigo de ID 10; a ferramenta *Smella* foi citada no artigo de ID 4. A ferramenta inédita trazida nesse trabalho, *Ambiguity Detection*, foi descontinuada e os autores informaram ainda que não possuem mais o código-fonte. **Situação:** ferramenta rejeitada.
- **15**: os autores citam 4 ferramentas, a saber, *GATE*, *Smella*, *RETA* e *SREE*. **Situação:** ferramentas já consideradas em outros trabalhos.
- **16**: os autores citam 3 ferramentas, a saber, *GATE*, *Smella* e *RETA*. **Situação:** ferramentas já consideradas em outros trabalhos.
- **17**: duas ferramentas são citadas nesse trabalho, *Smella* e *SREE*, ambas já citadas no artigo de ID 4. **Situação:** ferramentas já consideradas em outros trabalhos.
- **18**: relata uma abordagem automatizada para detecção de anomalias de requisitos, já citada no artigo de ID 3. Essa ferramenta foi implementada sob o *kit* de ferramentas de análise de qualidade *ConQAT*. **Situação:** ferramenta já considerada em outros trabalhos.
- **19**: cita 5 ferramentas, duas delas já foram mencionadas em outras pesquisas, sendo *AQUSA* (artigo de ID 12) e *QuARS Express* (artigo de ID 10). A ferramenta *Dowser* consiste em um protótipo para lidar com ambiguidades no documento de requisitos. A ferramenta *RAI* foi projetada para identificar termos, que é uma abstração do domínio de

[10] https://github.com/gglucass/aqusa

interesse. **Situação:** uma mensagem de *e-mail* foi enviada para os autores da *Dowser*, que responderam que o projeto não está mais ativo e nenhum suporte pode ser fornecido. **Situação:** ***Dowser***: ferramenta rejeitada. Quanto à ferramenta *RAI*, esta foi rejeitada, pois ela não contempla os critérios de seleção e nem a funcionalidade de interesse desta pesquisa (detecção de anomalias de requisitos). **Situação**: ferramenta rejeitada.

- **20**: os autores citaram 6 ferramentas, duas delas não possuem nome e não disponibilizam *link* para *download*. As demais são: *GATE* (citada no artigo de ID 5) e *Smella* e *SREE* (ambas citadas no artigo de ID 4). Por último, a ferramenta *ARM tool*, não contempla a funcionalidade de interesse desta pesquisa. **Situação:** ferramentas já consideradas em outros trabalhos ou rejeitadas por não atenderem aos critérios de inclusão.
- **21**: a publicação apresenta uma ferramenta de terceiro comercial *Nvivo*. A empresa fornece licença gratuita para pesquisas e projetos acadêmicos, entretanto, a ferramenta não realiza a detecção de anomalias de requisitos (funcionalidade de interesse desta pesquisa).**Situação:** ferramenta rejeitada.
- **22**: Abordagem semiautomatizada para resolver o problema de ambiguidade de requisitos do produto. Processamento de Linguagem Natural (PLN) e abordagem de lógica *fuzzy* para correção de ambiguidades.**Situação:** ferramenta rejeitada. Não houve retorno do autor sobre como obter a ferramenta. **Situação:** trabalho rejeitado.
- **23**: não houve retorno dos autores sobre como obter a ferramenta e informações de como operá-la. **Situação:** ferramenta rejeitada.