BRUNA DE SÁ TAVARES

Adequação para a Engenharia de Requisitos dos requisitos extraídos a partir dos *feedbacks* de multidão de usuários de APPS

SÃO PAULO

2023

BRUNA DE SÁ TAVARES

Adequação para a Engenharia de Requisitos dos requisitos extraídos a partir dos *feedbacks* de multidão de usuários de APPS

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em ciências

Área de concentração: Engenharia de Computação

Orientador: Prof. Dr. Fábio Levy Siqueira

SÃO PAULO

2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 27 de junho de 2023

Assinatura do autor: Bruna de Sá Tavares

Assinatura do orientador: Fábio S. S

Catalogação-na-publicação

Tavares, Bruna de Sá
Adequação para a Engenharia de Requisitos dos requisitos extraídos a partir dos feedbacks de multidão de usuários de APPS / B. S. Tavares -- versão corr. -- São Paulo, 2023.
103 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Engenharia de Requisitos 2.Mineração de Dados 3.Processamento de Linguagem Natural 4.Feedback de usuário 5.Aplicativo móvel I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Agradeço aos meus pais, Monica de Sá e Fabio Sergio Tavares que sempre me apoiaram a seguir meu sonho de estudar na Poli, pela paciência do dia a dia e por me motivarem a não desistir do mestrado.

Agradeço meu noivo Caio Rodrigues Magrini por me apoiar, me ouvir e me incentivar desde o início do mestrado.

Agradeço ao meu orientador Professor Doutor Fábio Levy Siqueira pelo “casamento” nessa jornada, por ter me dado a oportunidade de desenvolver um novo tema e todos os ensinamentos ao longo deste mestrado.

Agradeço aos meus amigos por me apoiarem, me incentivarem e me motivarem a terminar o mestrado.

Agradeço aos meus colegas de trabalho, chefes e líderes pelo incentivo.

Agradeço a todos os colegas e professores da Poli com quem tive convivência e aulas muito ricas para a construção do conhecimento.

RESUMO

TAVARES, BRUNA S. **Adequação para a Engenharia de Requisitos dos requisitos extraídos a partir dos *feedbacks* de multidão de usuários de APPS. 2023.** Tese (Mestrado em Engenharia de Computação - Escola Politécnica da Universidade de São Paulo, Universidade de São Paulo, São Paulo, 2023.

O aumento na quantidade e no uso de aplicativos móveis têm gerado muitos comentários de usuários. Esses comentários, ou *feedbacks*, incluem textos que podem ser usados para melhorar o processo de desenvolvimento de software e, especialmente, as atividades de Engenharia de Requisitos. Devido ao volume muito grande de dados, novas formas de reduzir o esforço humano para analisar os *feedbacks* e aumentar o valor do uso têm sido propostas usando técnicas de processamento de linguagem natural (NLP) e mineração de dados. Apesar de existirem trabalhos usando essas abordagens no contexto da Engenharia de Software, existem poucas evidências do uso das informações extraídas em projetos reais. O objetivo deste trabalho é analisar a adequação para a Engenharia de Requisitos do requisito extraído automaticamente através de *feedbacks* explícitos de usuários de aplicativos móveis. O método usado é o de pesquisa exploratória, utilizando uma revisão sistemática da literatura. Como resultado, observou-se que a maioria dos trabalhos não altera o formato da saída; 53% dos trabalhos extraem pedidos de *features* dos *feedbacks* e 50% extraem bug *requests*; 35% dos trabalhos identifica requisito não funcional em diferentes níveis de detalhe no *feedback*; e apenas 15% dos trabalhos tem uma avaliação da saída por um especialista de requisitos. A saída do processo de extração está longe de se assemelhar a um requisito em formato e em conteúdo de uma *feature*, que é um nível mais geral de requisito e que ainda necessita de refinamento para se tornar requisito de software. Este trabalho discute a necessidade da evolução tanto do processo de extração quanto em técnicas de NLP que produzam requisitos. Também se discute a importância da geração de conteúdos mais específicos de forma automatizada para o uso da Engenharia de Requisitos e a necessidade de avaliação da saída do processo com relação ao formato e conteúdo extraído.

Palavras-chave: Mineração de dados. Extração de requisitos. *Feedback* explícito. Lojas de aplicativo. Engenharia de Requisitos.

ABSTRACT

TAVARES, BRUNA S. **Suitability for Requirements Engineering of the requirements extracted from the feedbacks of crowd of APP users. 2023.** Tese (Mestrado em Engenharia de Computação - Escola Politécnica da Universidade de São Paulo, Universidade de São Paulo, São Paulo, 2023.

The increase in the quantity and use of mobile applications has generated a lot of user feedback. These feedbacks include texts that can be used to improve the software development process, especially Requirements Engineering activities. Due to the large volume of data, new ways to reduce human effort for analyzing feedback and increasing the value of use have been proposed using natural language processing (NLP) and data mining techniques. Despite the existence of studies using these approaches in the context of software engineering, there is little evidence of the use of the extracted information in real projects. The objective of this study is to analyze the suitability of automatically extracted requirements through explicit feedback from mobile application users for Requirements Engineering. The method used is exploratory research, using a systematic literature review. As a result, it was observed that most studies do not change the output format; 53% of studies extract feature requests from feedback and 50% extract bug requests; 35% of studies identify non-functional requirements at different levels of detail in feedback; and only 15% of studies have an evaluation of the output by a requirements specialist. The output of the extraction process is far from resembling a requirement in format and content of a feature, which is a more general level of requirement and still requires refinement to become a software requirement. This study discusses the need for evolution of both the extraction process and NLP techniques that produce requirements. It also discusses the importance of generating more specific content automatically for the use of Requirements Engineering and the need to evaluate the output of the process in terms of extracted format and content.

Keywords: Data mining. Requirements extraction. Explicit feedback. App Stores. Requirements Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CrowdRE	*Crowd-based Requirements Engineering*
DM	Data Mininig
ES	Engenharia de Software
ER	Engenharia de Requistos
KDD	*Knowledge Discovery in Database*
ML	*Machine Learning*
NLP	*Natural Language Processing*
NLP4RE	*Natural Language Processing for Requirements Engineering*
RSL	Revisão Sistemática da Literatura

SUMÁRIO

1 INTRODUÇÃO

Os softwares estão presentes em todos os aspectos de nossas vidas (BENNACEUR *et al.*, 2018), com isso as pessoas vêm adquirindo uma percepção muito maior do quão útil um software é para elas e como as funcionalidades as permitem realizar tarefas do dia a dia. Uma pesquisa feita anualmente pela consultoria Activate (2022), mostrou um aumento de 45 minutos diários no uso de tecnologia no ano de 2020, totalizando mais de 13 horas de uso. Esse aumento foi impulsionado pela COVID e, ainda assim, é esperado que se mantenha nesse patamar o mesmo total de horas de uso nos próximos anos.

Com o aumento no uso, os usuários de software desejam novas funcionalidades e querem agradecer ou reclamar sobre o software que estão usando. Isso pode ser feito através de diferentes canais que conectam milhares de pessoas (KHAN *et al.*, 2019) à empresa que desenvolve o software. Em especial, as lojas de aplicativos móveis são canais de acesso público que servem como meio oficial para o usuário ter acesso ao software e enviar *feedbacks* para o time de desenvolvimento (PAGANO; MAALEJ, 2013). Em qualquer loja, para que os produtos sejam comprados – ou baixados, no caso os aplicativos móveis (apps) – é importante que eles tenham uma boa nota na loja, pois apps bem avaliados tendem a ter mais downloads (HARMAN; JIA; ZHANG, 2012; PALOMBA *et al.*, 2015).

Uma boa nota é reflexo de que as necessidades do usuário estão sendo atendidas e, por isso, o comentário que vem junto da nota de avaliação é muito importante. Segundo Pagano e Maalej (2013) cerca de um terço dos comentários da Apple Store inclui tópicos sobre requisitos de software e experiência do usuário. Assim, essas informações poderiam ser usadas para identificar, priorizar e gerenciar requisitos de produtos de software (MAALEJ *et al.*, 2016), contribuindo para a evolução do processo de Engenharia de Requisitos (ER) e aproximando o time de desenvolvimento aos usuários.

Como os comentários são feitos por uma multidão de usuários - conhecidas como *crowd*, fazer a análise e extração de informações deles de forma manual leva tempo. Por exemplo, no trabalho de Guzman et al. (2018) os 3 coautores realizaram a atividade de rotulação manual para identificar o conteúdo de um total de 1.706

comentários de usuários, na qual cada coautor rotulou dois terços dos comentários totais extraídos para garantir que um comentário tivesse dois rótulos de duas origens diferentes para reduzir o viés da escolha do rótulo. Apesar de a rotulação envolver apenas 3 tipos de rótulos e ser apoiada por uma ferramenta, foi reportado que essa atividade durou 18 horas em média para cada pessoa, ou seja, 54 horas para ser completada.

Se esse mesmo trabalho de rotulação for feito para o aplicativo do Facebook para Android, que tem um acumulado de 10,5 milhões de *feedbacks* escritos em inglês, com uma média de 250 mil comentários diários em inglês (APP ANNIE, 2022), e usar a mesma quantidade de pessoas, estratégia para diminuir viés e tempo de rotulação, cada pessoa do time do Facebook teria que avaliar 166,6 mil comentários diários o que levaria 73 dias e 6 horas para cada pessoa ou 219 dias 18 horas no total para analisar manualmente os comentários.

Essa dificuldade da extração manual de requisitos em aplicativos que recebem muitos comentários motivou a discussão de que a forma de reduzir o esforço humano para analisar os *feedbacks* e aumentar o valor do uso de comentários da multidão (*crowd*) em Engenharia de Requisitos é através da automação do processo (DALPIAZ, 2021), tanto por conta da escassez de tempo quanto por conta da escassez de pessoas com nível de conhecimento e experiência desejado para fazer essa atividade.

1.1 Objetivo

Este trabalho tem o objetivo de analisar a adequação para a Engenharia de Requisitos do requisito extraído automaticamente através de *feedbacks* explícitos de usuários de aplicativos móveis. Para isso foi analisado o formato e o detalhamento do resultado do processo de extração que contém o processo de *knowledge discovery in database* (KDD), o qual aplica técnicas de Data Mining para a extração de requisitos.

A extração de requisitos consiste em um processo que tem como entrada o texto do *feedback* dado pelo usuário de forma explícita. No processamento desse texto, ele pode ser modificando e informações contidas nele podem ser descobertas através do processo KDD (explicado no Capítulo 3). Esse processo tem como saída informações

que vieram da descoberta de padrões no texto do comentário. No exemplo da Figura 1, o comentário que entrou no processo de extração de requisitos teve informações sobre a intenção (“Pedido”) e os assuntos (“Feature/Funcionalidade”, “Interface gráfica de usuário”, e “Melhoria”) encontradas. Se adequadas, essas informações podem fazerem parte de processos na ER.

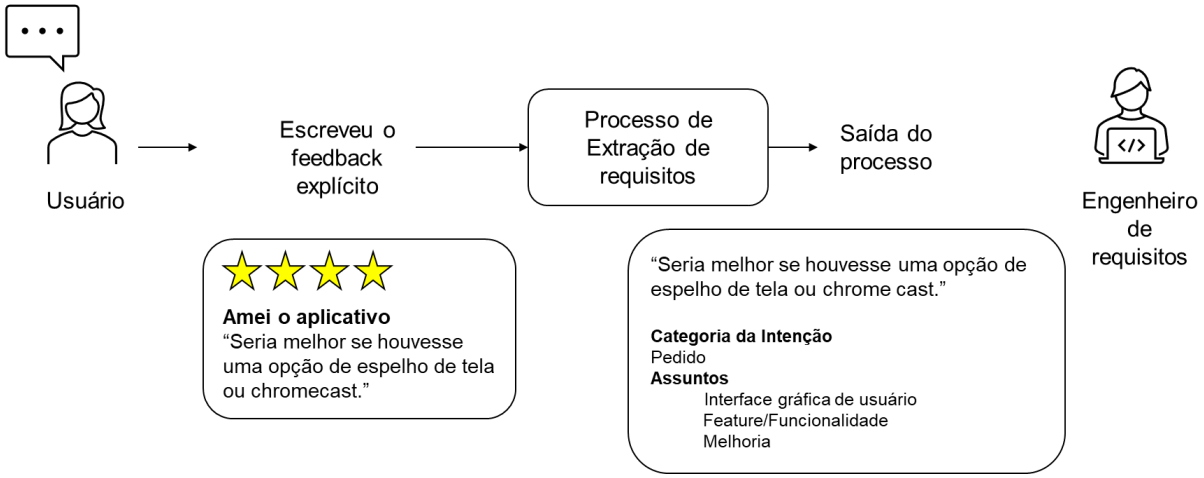

Figura 1 - Processo de extração de requisitos vindo de *feedback* de aplicativo móvel.

Fonte: (DI SORBO *et al.*, 2017, tradução nossa)

Para atingir o objetivo foram formuladas três perguntas de pesquisa, apresentadas a seguir:

- QP1: Qual o formato de saída dos *feedbacks* explícitos no processo de extração de requisitos?
- QP2: Quais as classes de saída dos *feedbacks* explícitos no processo de extração de requisitos?
- QP3: Qual a avaliação de saída dos *feedbacks* explícitos no processo de extração de requisitos?

A QP1 busca entender se o formato extraído segue uma representação de requisitos específica (como história do usuário ou afirmação em linguagem natural estruturada). Na QP2, o objetivo é entender o conteúdo das classes: qual o nível de detalhe ou de refinamento (se é requisitos do *stakeholder*, de sistema, de software (ISO/IEC/IEEE, 2018)), o tipo do requisito (requisitos funcionais, requisitos não funcionais ou alguma outra forma) e se existem outras informações. Na QP3 busca-se entender se houve avaliação do resultado obtido pelo processo de extração de requisitos para prática em

ER, que tipo de avaliação foi feita, se a avaliação foi feita por um especialista em desenvolvimento de software e quais os principais assuntos abordados pela avaliação.

1.2 Método

Para conduzir essa pesquisa exploratória foi utilizado o método de revisão sistemática da literatura, seguindo o guia da Kitchenham e Charters (2007). O Capítulo 4 apresentará em detalhe esse método.

1.3 Justificativa

Os *feedbacks* de usuários de aplicativos móveis provenientes de lojas aplicativos e redes sociais vêm sendo analisados há mais de 10 anos, sabendo que eles contém tópicos sobre requisitos e experiência do usuário (CARRENO; WINBLADH, 2013; GUZMAN; IBRAHIM; GLINZ, 2017; PAGANO; MAALEJ, 2013).

Cada vez mais pesquisadores extraem os requisitos desses *feedbacks* (DĄBROWSKI *et al.*, 2022) para integrá-los ao processo de Engenharia de Software de forma automatizada, aplicando pré-processamento de texto e técnicas de *data mining,* dentro do processo de *knowledge discovery in database* (KDD).

Esse aumento de estudos incentivou a comunidade a realizar revisões sistemáticas (DĄBROWSKI *et al.*, 2022; GENC-NAYEBI; ABRAN, 2017; TAVAKOLI *et al.*, 2018) que analisaram os dados extraídos, técnicas de mineração e avaliação dessas técnicas usadas para loja de aplicativos. Os estudos tiveram diferentes focos: conceitos de *crowd* em Engenharia de Requisitos (ER) (KHAN *et al.*, 2019; WANG *et al.*, 2019), no uso específico de técnicas de aprendizado de máquina (ZAMANI; ZOWGHI; ARORA, 2021) e processamento de linguagem natural (ZHAO *et al.*, 2022) com ER. Suas principais descobertas foram:

- Engenheiros de software consideram as abordagens de mineração úteis e com desempenho promissor para gerar diferentes análises de avaliação de aplicativos. No entanto, ainda não está claro até que ponto essas abordagens

já são boas o suficiente para serem usadas na prática. (DĄBROWSKI *et al.*, 2022);

- Poucos estudos que extraem *feedback* com técnicas de processamento de linguagem natural (N*atural Language Processing*, NLP) são aplicados na indústria, o que é explicado segundo Zhao et al. (2022) pela falta de recursos específicos de ER, como coleções de textos de requisitos (vide Capitulo 3). Essa falta dificulta o desenvolvimento de abordagens eficazes e com resultados confiáveis na área de Processamento de Linguagem Natural para Engenharia de Requisitos (NLP4RE), pois usa recursos genéricos de linguagem para treinar modelos de linguagem específicos de domínio de requisitos (ZHAO *et al.*, 2022).

Este estudo foi motivado pelas descobertas de outras revisões sistemáticas da literatura, que iniciam a análise do processo de extração de requisitos sob diferentes perspectivas, mas que não tem um foco em aplicação na Engenharia de Requisitos com fontes de dados de *feedback crowd* para aplicativos móveis e um olhar mais amplo do processo de extração de dados.

Além disso, nenhum dos trabalhos relacionados avalia e discute o formato dos requisitos, considerando uma representação de requisitos especifica. Tais trabalhos também não analisam de forma quantitativa e qualitativa o nível de detalhe do requisito extraído, tipos de requisito extraído e outras informações. Por fim, eles não discutem com um olhar crítico a adequação do processo de extração de requisitos ponto de vista da Engenharia de Requisitos.

1.4 Organização do Trabalho

Além desta introdução, este trabalho possui mais 7 capítulos. Os Capítulos 2, 3 e 4 apresentam a fundamentação teórica desta pesquisa, tratando dos conceitos da Engenharia de Requisitos, mineração de dados e revisão sistemática da literatura, respectivamente. O Capítulo 5 apresenta o protocolo da revisão sistemática da literatura feita e o Capítulo 6 apresenta os resultados obtidos. O Capítulo 7 apresenta os trabalhos relacionados a este trabalho. Por fim, o Capítulo 8 é apresentada a conclusão e os trabalhos futuros desta pesquisa.

2 ENGENHARIA DE REQUISITOS

Alguns comentários de usuários em lojas de aplicativos podem conter sugestões de novos requisitos ou de melhorias dos requisitos já implementados. Assim, o processo de obtenção dos requisitos a partir dos comentários pode ser visto como uma forma de elicitação de requisitos, que é uma das atividades do processo de desenvolvimento de requisitos da área de Engenheira de Requisitos.

Segundo a ISO/IEC 29148 (2018, p.5), “a Engenharia de Requisitos é uma função interdisciplinar que faz a mediação entre os domínios do comprador e do fornecedor para estabelecer e manter os requisitos a serem atendidos pelo sistema, software ou serviço de interesse”.

2.1 Requisito

Um requisito é uma afirmação relacionada a um sistema, software ou serviço ou outro item de interesse que “traduz ou expressa uma ou mais necessidades junto de suas restrições e condições associadas” (ISO/IEC/IEEE, 2018, p. 4).

As necessidades expressas pelos requisitos vêm dos objetivos e/ou metas dos *stakeholders*. Os *stakeholders* são pessoas ou organizações que têm interesse em se beneficiar com o software, tais como usuários, a empresa que desenvolve o software e instituições regulamentadoras (ISO/IEC/IEEE, 2018). Suas necessidades são definições de mais alto nível que orientam o time de desenvolvimento a entender por que um sistema de software é necessário (NUSEIBEH; EASTERBROOK, 2000; VAN LAMSWEERDE, 2001).

O requisito de *stakeholder* vem do refinamento e evolução das necessidades dos *stakeholders* (ISO/IEC/IEEE, 2018). Para isso, o engenheiro de requisitos precisa compreender as preocupações dos *stakeholders* no nível organizacional e sistêmico e, a partir de um nível inicial das necessidades, gerar um conjunto de declarações objetivamente adequadas, estruturadas e mais formais (ISO/IEC/IEEE, 2018, p. 10). Para essas declarações serem consideradas requisitos de *stakeholder* elas precisam estar bem formadas e devem atender às características de construção, características

de requisitos individuais e características de um conjunto de requisitos (ISO/IEC/IEEE, 2018).

Um usuário é um dos tipos de *stakeholders*, ele se diferencia dos outros *stakeholders* pois é aquele que “interage com um sistema ou se beneficia de um sistema durante sua utilização o sistema a pessoa que usa” (ISO/IEC/IEEE, 2017). Dessa forma um requisito de usuário é um dos tipos de requisito de stakeholder (ISO/IEC/IEEE, 2018, p. 53) e “são derivados das necessidades e capacidades do usuário, a fim de fazer uso do sistema de maneira eficaz, eficiente, segura e satisfatória” (ISO/IEC/IEEE, 2017, p. 497).

Quando o requisito de *stakeholder* recebe mais informações e detalhes sobre o sistema que será construído, ele se torna um requisito do sistema (ISO/IEC/IEEE, 2018). O requisito de sistema ao ser alocado a um software, recebe mais detalhes e se torna o requisito de software (ISO/IEC/IEEE, 2018).

Um aplicativo móvel é uma aplicação de software, portanto as necessidades relacionadas a ele são requisitos de software. Os requisitos de software podem ser divididos em requisitos funcionais, requisitos não funcionais e restrições.

O requisito funcional é aquele relacionado ao resultado de algum comportamento a ser fornecido por uma função do sistema (POHL; RUPP, 2015). Ele diz como o software se comportará ao receber uma interação do ambiente, a qual pode vir de uma pessoa ou de outro sistema. A cada interação, o software deve realizar uma ação que processa informações e pode gerar uma saída, por exemplo, mostrando algum texto na tela. Além disso, os requisitos funcionais descrevem como o software deve se comportar em situações de exceção (BOURQUE; FAIRLEY; EDS., 2014)(ISO/IEC/IEEE, 2018).

Alguns trabalhos que extraem informações de comentários de aplicativos (DI SORBO *et al.*, 2017; GUZMAN; MAALEJ, 2014) utilizam o termo *feature* para se referir a funcionalidades dos aplicativos e atributos de qualidade do aplicativo, mas sem seguir uma referência. Segundo a norma ISO/IEC/IEE 24765 (2017. p.181) uma *feature* é “uma característica que distingue um item do sistema, podendo ser uma característica funcional ou não funcional e muitas vezes se refere a uma melhoria” Além disso ela

“é uma característica mais abstrata que os usuários finais e outros *stakeholders* podem entender” ISO/IEC/IEE 24765 (ISO/IEC/IEEE, 2017, p. 181).

Para satisfazer as necessidades dos *stakeholders* e entregar valor para eles é importante que seja definido não só o que o software irá fazer, o que é funcional, mas também de que forma ele se comportará e suas limitações, não funcional. Segundo Glinz (2007, p. 25, tradução nossa) “um requisito não funcional é um atributo ou uma restrição de um sistema”, um atributo é “um requisito de desempenho ou um requisito de qualidade específico” (GLINZ, 2007, p. 24, tradução nossa) e uma restrição é “um requisito que restringe o espaço da solução além do necessário para atender aos requisitos funcionais, de desempenho e de qualidade específicos" (GLINZ, 2007, p. 24, tradução nossa).

Este trabalho usará a taxonomia do Modelo de Qualidade de Produto da norma ISO/IEC 25010 (ISO/IEC, 2010) também usada por Pohl e Rupp (2015), para se referir a atributos de Glinz (2007). Essa taxonomia utiliza o termo “atributos de qualidade” e os organiza em oito características de qualidade:

- **Adequação funcional**: capacidade de o produto de software “prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas” (ISO/IEC, 2010, p. 10, tradução nossa);
- **Desempenho**: capacidade de o produto de software apresentar “desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas” (ISO/IEC, 2010, p. 11, tradução nossa);
- **Confiabilidade**: capacidade de um “sistema, produto ou componente manter um nível de desempenho especificado ao executar as funções especificadas” (ISO/IEC, 2010, p. 13, tradução nossa);
- **Usabilidade**: capacidade de um produto ou sistema usado ser compreendido, aprendido, operado e atraente ao usuário, protegido contra erros dos usuários e acessível, quando usado sob condições especificadas (ISO/IEC, 2010).
- **Segurança**: capacidade de um “produto ou sistema proteger informações e dados para que pessoas ou outros produtos ou sistemas tenham o grau de acesso aos dados apropriado para seus tipos e níveis de autorização” (ISO/IEC, 2010, p. 13, tradução nossa);

- **Manutenibilidade**: “grau de eficácia e eficiência com que um produto ou sistema pode ser modificado pelos mantenedores” (ISO/IEC, 2010, p. 14, tradução nossa);
- **Portabilidade**: “grau de eficácia e eficiência com que um sistema, produto ou componente pode ser transferido de um hardware, software ou outro ambiente operacional ou de uso para outro” (ISO/IEC, 2010, p. 15, tradução nossa);
  Compatibilidade: “grau em que um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes e / ou executar suas funções necessárias, enquanto compartilha o mesmo ambiente de hardware ou software” (ISO/IEC, 2010, p. 11, tradução nossa).

As taxonomias de atributos de qualidade de um produto de software podem ser usadas como referência para o levantamento dos requisitos de software, além de serem úteis para a especificação de requisitos (ISO/IEC, 2010). Para que não sejam confundidos com requisitos funcionais, os requisitos não funcionais devem ser documentados separadamente, mas a relação entre os tipos de requisito deve ser explícita (POHL; RUPP, 2015).

2.2 Erros, Defeitos e Bugs

O software passa por todo o processo de desenvolvimento para ser liberado ao usuário. Nesse processo existem verificações que têm o objetivo de encontrar problemas, mas se os problemas não forem corrigidos antes da entrega do software, existe uma grande chance de eles serem encontrados pelo usuário através de um erro.

Esses problemas são conhecidos como *bug* ou defeito (*fault*) que, segundo a ISO/IEC/IEEE (2017, p. 179, tradução nossa) é “manifestação de um erro no software, *step* incorreto, processo ou definição de dados em um programa de computador”.

De uma forma geral defeitos devem ser evitados pois criam uma insatisfação com o software(PRESSMAN, 2016), a qual impacta na nota do aplicativo em lojas de apps e na obtenção de novos usuários (GU; KIM, 2015)

Quando erros são reportados pelos usuários, provavelmente existem *bugs* no software. Para endereçar esse problema, uma solicitação de mudança do software (*change request*) é criada, a qual será analisada pelo time de desenvolvimento. Essa solicitação de mudança pode ser feita ainda no processo de desenvolvimento por testadores ou, quando o software já foi entregue, por usuários (SOMMERVILLE, 2016). Caso a solicitação de mudança se refira a um requisito não existente ou casos não considerados, então será definido um novo requisito; se o pedido for mais abstrato, por exemplo uma característica do software, então pode-se falar de um pedido de novas *features*.

2.3 Processo de Engenharia de Requisitos

Durante as fases da Engenharia de Requisitos (ER), existem diversas atividades que precisam ser executadas para que os requisitos sejam suficientemente detalhados para o desenvolvimento de software. Wiegers e Beatty (2013) separam as atividades em definição de um requisito de software novo (desenvolvimento de requisitos) das atividades que são de monitoramento e mudanças de requisitos já criados anteriormente (gerenciamento de requisitos), como mostra a Figura 2.

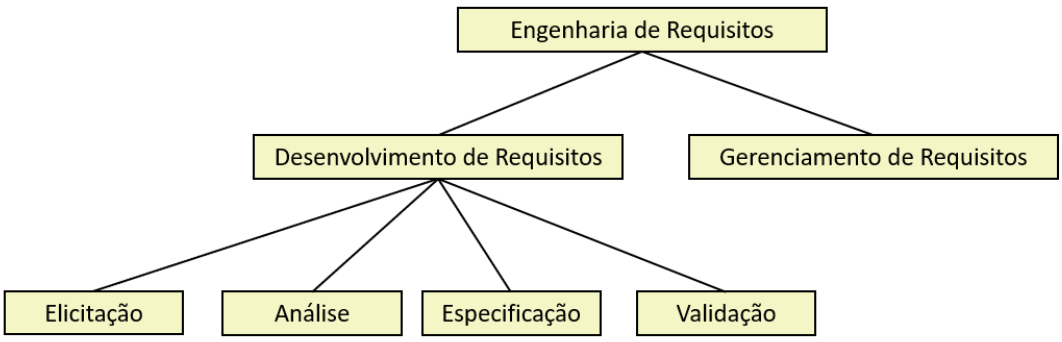

Figura 2 – Subdisciplinas da Engenharia de Requisitos de Software (2013)

Fonte:(WIEGERS; BEATTY, 2013).

No desenvolvimento de requisitos acontecem as atividades de elicitação, análise, especificação e validação. Na atividade de elicitação, acontece o levantamento dos requisitos com os *stakeholders*; na atividade de análise os requisitos elicitados serão entendidos e detalhados; na atividade de especificação os requisitos serão representados da forma mais compreensível para seus leitores; e na fase de validação

é feita a confirmação de que o conjunto de informações de requisitos permite aos desenvolvedores atingir os objetivos de negócio (WIEGERS; BEATTY, 2013).

A ordem, o nível de profundidade e o detalhe na execução de cada atividade pode variar dependendo de qual ciclo de desenvolvimento o projeto de software está seguindo: modelo cascata, iterativo, incremental, ágil ou híbrido (WIEGERS; BEATTY, 2013).

2.4 Especificações de requisitos

Uma das atividades do desenvolvimento de requisitos é a especificação de requisitos. Seu objetivo "é documentar requisitos de diferentes tipos de forma consistente, acessível e passível de revisão, que seja prontamente compreensível pelo público-alvo" (WIEGERS; BEATY, 2013, p. 51, tradução nossa).

A especificação dos requisitos é frequentemente feita em linguagem natural (português, inglês, etc.) por ter a vantagem de (supostamente) não exigir tempo de preparação para ser lida e compreendida pelos *stakeholders*, além de ser uma linguagem universal que ajuda a descrever qualquer circunstância (POHL; RUPP, 2015). Porém, a linguagem natural é inerentemente ambígua, podendo ser interpretada de diversas maneiras. Essa característica causa problemas no entendimento dos requisitos por pessoas com conhecimentos diferentes (POHL; RUPP, 2015).

Alguns formatos podem ser utilizados para escrever as especificações de requisitos, como:

- Sentenças em linguagem natural: uma sentença é uma[1] "construção sintática com sentido completo, composta por uma ou mais palavras; frase" , nesse caso, cada sentença deve expressar um requisito (SOMMERVILLE, 2016);

[1] <https://www.dicio.com.br/sobre.html>

- Linguagem natural estruturada: os requisitos são expressos na forma da linguagem natural, mas eles são escritos de maneira restrita dentro de um *template*. Esse *template* cria uma estrutura com campos de informações sobre um aspecto do requisito (SOMMERVILLE, 2016);
- Linguagens de modelagem conceitual: os requisitos são especificados por meio de representações abstratas formais, semiformais e informais, por exemplo, diagramas de caso de uso UML (POHL; RUPP, 2015).

Qualquer tipo de formato de documentação pode ser usado para documentar os requisitos (POHL; RUPP, 2015), mas é importante seguir as boas práticas da representação usada. Por exemplo, segundo a ISO/IEC/IEE (2018) para uma sentença representar um requisito ela precisa ter um sujeito e um verbo, sugere-se usar verbos como “dever” e “poder” escritos na voz ativa, além da escrita da sentença ser de forma afirmativa.

Uma boa prática de especificação de software segundo Wiegers e Beatty (2013) é adotar *templates* de documentos de requisitos pois eles orientam a escrita do requisito através de uma estrutura (WIEGERS; BEATTY, 2013, p. 51) com campos de informações para serem preenchidos sobre um aspecto do requisito (SOMMERVILLE, 2016). Um exemplo popular de técnica de representação de requisitos de usuários que propõe *template* é a história de usuário (LUCASSEN *et al.*, 2016) a qual tem o seguinte padrão de escrita “Eu como {papel do usuário}, quero {função} para que {valor de negócio}” (COHN, 2009, p. 81). Nesse padrão os elementos dentro de colchetes precisam ser preenchidos para formar cada sentença de requisito.

2.5 CrowdRE

Crowd-based Requirements Engineering (Engenharia de Requisitos baseada em multidão) ou *CrowRE* é uma área emergente de pesquisa que utiliza os conceitos do *crowdsourcing* aplicado à Engenharia de Requisitos. Segundo a definição citada por (MAO *et al.*, 2017, p. 57, tradução nossa) sobre (HOWE, 2006) *“crowdsourcing* é o ato organizado que terceiriza o seu trabalho para uma mão de obra indefinida e em rede usando uma chamada aberta para participação”. Apesar de o *crowdsourcin*g ser um conceito antigo, ele veio ganhando popularidade quando foi integrado à Internet,

permitindo ser aplicado em diversas áreas de conhecimento (KHAN *et al.*, 2019), como é o caso da área de Engenharia de Requisitos (ER).

O *crowdsourcing* não ER é importante por dois motivos principais. O primeiro é a proximidade que ele permite entre os *stakeholders* e quem desenvolve o software, o que resulta em um processo de Engenharia de Requisitos melhor e aumenta a chance de sucesso do projeto (SNIJDERS *et al.*, 2014). O segundo motivo é que as plataformas de *crowdsourcing* permitem o uso da inteligência das multidões (KHAN *et al.*, 2019), em que diferentes opiniões de muitas pessoas podem melhorar o software.

Segundo Khan et al. (2019), as pesquisas de CrowdRE tem focado mais nos aspectos: o que é a *crowd (*multidão*)*; tarefas delegadas à *crowd*; no desenvolvimento de mecanismos, como os que vão permitir a colaboração e competição; a mídia ou canal de comunicação; nos incentivos para engajar a *crowd;* e como será avaliada a qualidade das entregas vindas da *crowd*.

No estudo de *crowd* é importante definir do que é composta a multidão. Segundo Dalpiaz (2021), a *crowd* pode ser composta por usuários finais do software ou por futuros usuários (GROEN *et al.*, 2017b) ou também especialistas do domínio e/ou engenheiros de software (KHAN *et al.*, 2019) que farão parte do processo de requisitos. O trabalho de Khan et al. (2019) propõe uma classificação para *crowd* de acordo com três propriedades: escala, ou seja, se o a quantidade de pessoas é grande; papel que irão exercer; e nível de conhecimento e experiência que essa *crowd* tem.

Neste trabalho a *crowd* usada foi de usuários de aplicativos móveis. Essa multidão tem um tamanho que depende da quantidade de usuários engajados a dar *feedbacks* ao aplicativo. Esses usuários muitas vezes não têm conhecimento técnico sobre o app, mas têm conhecimento de uso e falam sobre as funcionalidades (KHALID *et al.*, 2015). Quando o usuário escreve sua opinião ou faz uma avaliação sobre as funcionalidades, diz-se que é um *feedback* explícito (MAALEJ *et al.*, 2016b). Já o *feedback* implícito é aquele gerado por interações do usuário com o software, como logs sobre cliques na interface (MAALEJ *et al.*, 2016b).

O canal ou meio em CrowdRE é a forma de se conectar à *crowd*. Esses canais podem ser de acesso público, como redes sociais, lojas de aplicativo e fóruns *online*, mas

também podem ser canais internos de envio de mensagens e fóruns que somente os usuários cadastrados em um aplicativo conseguem acesso.

Um canal muito utilizado pelos usuários de apps é a loja de aplicativos móveis (app stores), as quais são usadas como distribuidoras de apps. As lojas de aplicativos mais conhecidas são a Apple App Store, fundada pela Apple em julho de 2008 para distribuir aplicativos para o sistema operacional iOS, e a Google Play Store, que foi fundada pela Google em 2008 com o nome de Android Market para distribuir aplicativos para o sistema operacional Android.

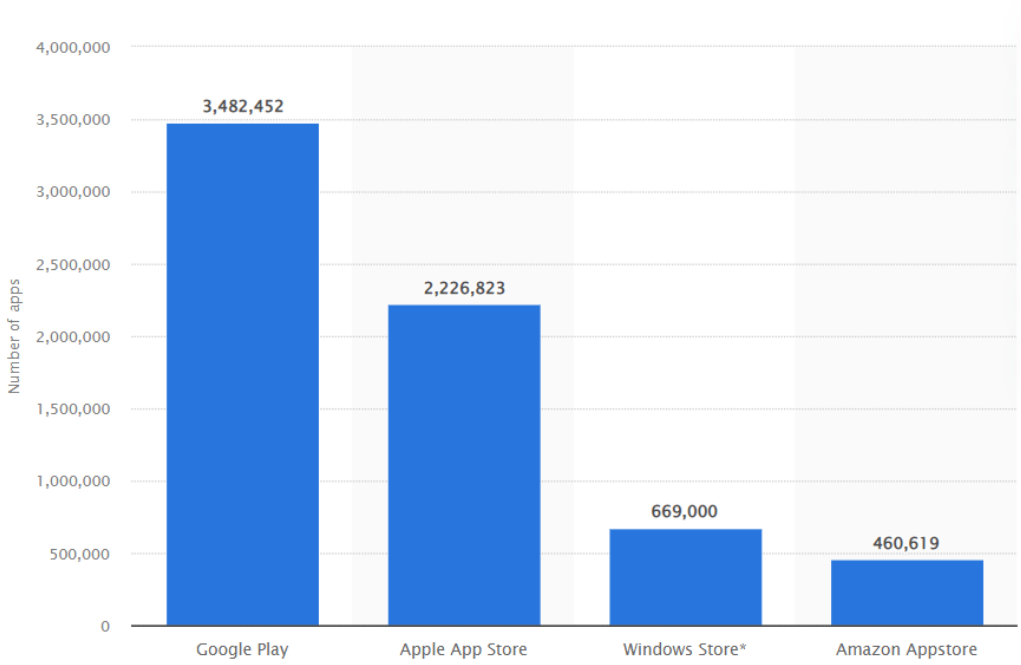

Figura 3 – Gráfico com número de aplicativos móveis disponíveis nas lojas de apps líderes no primeiro trimestre de 2021

Fonte:(Statista, 2022).

Essas lojas cresceram muito ao longo dos anos, como mostra a Figura 3. No primeiro trimestre de 2021 a Google Play chegou a ter 3,4 milhões de apps disponíveis para download e a Apple App Store 2,2 milhões de apps disponíveis (STATISTA, 2022). Esses números mostram a escala que as *crowds* vindas das plataformas podem ter.

2.6 Considerações do capítulo

Neste capítulo foi discutido o que é Engenharia de Requisitos, quais são os tipos de requisitos e sua relação com *bugs* de software. Foi falado sobre o processo de Engenharia de Requisitos e a relação com a área de pesquisa de CrowdRE, a qual tem aspectos importantes para o desenvolvimento desta pesquisa. O próximo capítulo tratará de Mineração de Dados, uma técnica de elicitação de requisitos especialmente útil para o *crowd*.

3 MINERAÇÃO DE DADOS

Para a extração de requisitos no contexto da CrowdRE, os trabalhos têm utilizado a mineração de dados (*data mining*). Segundo Tan et al.(2019, p. 2. tradução nossa) a mineração de dados (*data mining*) é “o processo de descoberta automática de informações úteis em grandes repositórios de dados”. Esse processo utiliza técnicas para descobrir padrões novos e úteis em grandes bases de dados e também pode ajudar a prever resultados de comportamentos no futuro (TAN *et al.*, 2019).

As técnicas de mineração de dados também são usadas para melhorar os sistemas de *information retrieval* (recuperação da informação), pois eles utilizam técnicas tradicionais da ciência da computação e *features* de dados muito conhecidas na área de dados para criar estruturas que indexam os dados para organizar e recuperar eficientemente a informação. Normalmente as atividades como a consulta de registros únicos em um sistema de gerenciamento de banco de dados ou consultas em um motor de busca na Internet são consideradas atividades de *information retrieval* (TAN *et al.*, 2019).

A mineração de dados tradicionalmente foi vista como um processo intermediário dentro da estrutura do *knowledge discovery in database* (KDD), mas ao longo dos anos foi ganhando mais importância no mundo acadêmico e dentro da indústria, e hoje é considerada um campo de estudo.

O KDD é um processo que tem como objetivo transformar o dado de entrada que é cru, ou sem nenhum processamento, em uma informação (TAN *et al.*, 2019), como mostra na Figura 4.

A primeira parte do processo de KDD começa com o pré-processamento dos dados de entrada. O pré-processamento tem como objetivo transformar os dados crus em um formato apropriado para a próxima análise (TAN *et al.*, 2019), como juntar dados de várias fontes, limpar dados para remover ruídos e unificar observações, selecionar registros e *features* de dados que são relevantes para a tarefa de mineração e outras atividades. Em tarefas de processamento de linguagem natural é muito comum se fazer a remoção de palavras muito frequentes em textos que não carregam informação (*stop words)* (JURAFSKY; MARTIN, 2022), como artigos e preposições.

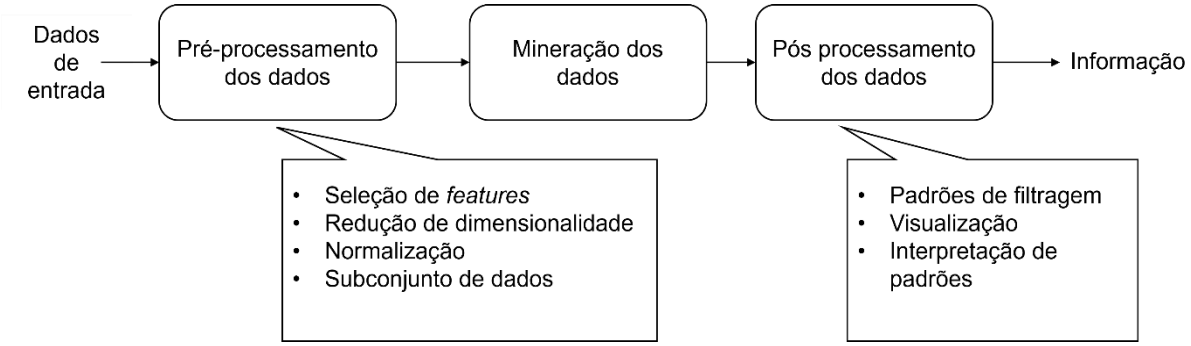

Figura 4 - O processo de KDD.

Fonte: (TAN *et al.*, 2019).

A primeira parte do processo de KDD começa com o pré-processamento dos dados de entrada. O pré-processamento tem como objetivo transformar os dados crus em um formato apropriado para a próxima análise(TAN *et al.*, 2019), como juntar dados de várias fontes, limpar dados para remover ruídos e unificar observações, selecionar registros e *features* de dados que são relevantes para a tarefa de mineração e outras atividades. Em tarefas de processamento de linguagem natural é muito comum se fazer a remoção de palavras muito frequentes em textos que não carregam informação (*stop words)* (JURAFSKY; MARTIN, 2022)*,* como artigos e preposições.

Depois do pré-processamento vem a fase em que são aplicadas as técnicas de mineração de dados (essas técnicas são detalhas na próxima seção). A última fase do KDD é a de pós processamento, em que são realizadas atividades para garantir que os dados sejam considerados válidos e úteis para serem incorporados ao sistema de suporte a decisão através de relatórios de visualização das informações ou até o envio dos dados para outros sistemas.

A mineração de dados usa como apoio áreas de estudo de banco de dados, e computação paralela e distribuída. Ela usa métodos e algoritmos que pesquisadores já utilizavam e assuntos de outras áreas de estudo como: (1) amostragem, estimativa e teste de hipóteses a partir de estatística; (2) algoritmos de busca, técnicas de modelagem e teorias de aprendizado de inteligência artificial, reconhecimento de padrões e aprendizado de máquina (*machine learning*); (3) otimização; (4) computação evolucionária; (5) teoria da informação; (6) processamento de sinais; (7) visualização; e (8) *information retrieval* (TAN *et al.*, 2019).

3.1 Técnicas de mineração

Na mineração de dados são usadas diversas técnicas que segundo Tan et al. (2019) e Murphy (2022) podem ser organizadas em duas categorias conforme seus objetivos: tarefas preditivas ou supervisionadas, e tarefas descritivas ou não supervisionadas.

A primeira, a tarefa preditiva ou supervisionada, tem o objetivo de aprender como mapear as entradas em saídas dado um conjunto de pares de entradas e saídas que são chamadas de base de treinamento. Modelos preditivos são criados para realizar tarefas preditivas e, dependendo do tipo de dado utilizado no nele, pode receber um nome diferente: se a variável de entrada for de valores discretos, o modelo é de classificação, mas se as variáveis forem contínuas, então o modelo é de regressão. Independentemente do tipo de tarefa, ambas procuram aprender modelos que minimizem o erro entre o valor real e o valor previsto (TAN *et al.*, 2019). Exemplos de técnicas para tarefas supervisionadas são: árvores de decisão, Naive Bayes, Nearest Neighbor, Support Vector Machine (SVM) e redes neurais artificiais (TAN *et al.*, 2019)

A segunda tarefa, a descritiva ou não supervisionada, tem como objetivo encontrar padrões (correlações, tendências, agrupamento *(clusters)*, trajetórias e anomalias) que resumem as relações subjacentes nos dados. Essas tarefas são muito menos definidas, uma vez que não existe uma base de treino com padrões para se basear (MURPHY, 2022; TAN *et al.*, 2019). Exemplos de técnicas para tarefas não supervisionadas são: K-means e Expectation Maximization (EM).

A mineração de dados tem apoiado o processo de requisitos de software, pois gera *insights* que auxiliam a tomada de decisão (INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS (IIBA), 2015). Suas técnicas permitem a análise de diversos tipos de dados como documentações, códigos de software, comentários de usuário e rastreamento de dados. Como pontos positivos a mineração de dados pode encontrar padrões nos dados contribuindo para o processo de detalhamento de informações e ajudar a reduzir o viés humano usando dados para descrever os fatos (INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS (IIBA), 2015). Mas se as técnicas de mineração de dados forem aplicadas sem um entendimento de como elas funcionam ou por uma pessoa que não seja especialista no assunto, pode resultar em

insights e tomada de decisão errada (INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS (IIBA), 2015).

3.2 Processamento de Linguagem Natural para Engenharia de Requisitos (NLP4RE)

Dentro do KDD costumam ser aplicadas técnicas de processamento de linguagem natural (*natural language processing*, NLP). O objetivo do NLP é usar o computador para entender a linguagem humana (JURAFSKY; MARTIN, 2022).

O processamento de linguagem natural usa o conhecimento da língua para se diferenciar dos outros tipos de processamento de dados. Esse conhecimento é composto por diversos conhecimentos, segundo Jurafsky e Martin (2013):

- Fonética e Fonologia: conhecimento sobre sons linguísticos;
- Morfologia: conhecimento das classes das palavras;
- Sintaxe: conhecimento das relações estruturais entre palavras;
- Semântica: conhecimento do significado das palavras;
- Pragmática: conhecimento da relação do significado com os objetivos e intenções do falante; e
- Discurso: conhecimento sobre unidades linguísticas maiores que um único enunciado.

A partir do conhecimento da linguagem é possível desenvolver técnicas para processar o texto de forma automatizada. Segundo Zhao et al.(2022, p. 55, tradução nossa) uma técnica de NLP é “um método, abordagem, processo ou procedimento prático para executar uma tarefa de NLP específica”. Uma técnica conhecida na área de NLP é a de POS *Tagging* (*Part of Speech Tagging*), na qual as palavras são rotuladas a partir da relação gramatical com palavras vizinhas em um texto ou regras da morfologia da palavra para remover ambiguidades (JURAFSKY; MARTIN, 2022).

Ferramentas de NLP são utilizadas para as técnicas serem executadas. Uma ferramenta de NLP segundo Zhao et al. (2022, p. 55, tradução nossa) é "um sistema de software ou uma biblioteca de software que oferece suporte a uma ou mais técnicas

de NLP". Duas ferramentas muito usadas e conhecidas na área de NLP são Stanford Core NLP[2] e NLTK [3].

Para dar suporte a técnicas ou ferramentas de NLP, foram criados recurso de dados linguísticos que podem ser um dicionário (linguagem léxica), por exemplo o WordNet[4], ou uma coleção de textos (*corpus*), por exemplo Brown Corpus[5].

Com o crescimento do interesse na evolução das técnicas de processamento de textos de Engenharia de Requisitos (especificações, códigos de software e comentários de usuários) foi criada a área de pesquisa NLP para a Engenharia de Requisitos, conhecida como NLP4RE (DALPIAZ *et al.*, 2018; ZHAO *et al.*, 2022). Apesar de a área estar em crescimento, um estudo recente (ZHAO *et al.*, 2022) mostrou que existe uma grande discrepância entre o estado da arte e o estado da prática na indústria, ou seja, foram criadas diversas ferramentas e técnicas, mas existe pouca evidência da prática delas na área de NLP (ZHAO *et al.*, 2022). Para melhorar o uso das técnicas de NLP4RE na indústria é necessário aumentar a confiança no resultado gerado por elas. Essa confiança no resultado de um método que utiliza algoritmos normalmente é baseada em métricas de avaliação das técnicas de aprendizado de máquina. O estudo de mapeamento (ZAMANI; ZOWGHI; ARORA, 2021) quis saber quais as métricas de avaliação mais usadas na área de Engenharia de Requisitos e de 65 estudos selecionados 42 (65%) utilizou pelo menos uma métrica de avaliação e dentre os estudos, as métricas mais utilizadas foram *precision* e *recall*, em 38 estudos, e na sequência a métrica F-score com 29 estudos.

As métricas de avaliação de técnicas de NLP *precision* e *recall,* e *F-score* mesmo sendo muito usadas na área de NLP4RE, foram herdadas da área de *Information Retrieval* sem serem adaptadas para o contexto de Engenharia de Requisitos (DALPIAZ *et al.*, 2018). Com isso, são necessários estudos empíricos para determinar

[2] Stanford Core NLP - https://nlp.stanford.edu/software/.

[3] NLTK - https://www.nltk.org.

[4] Word Net- https://wordnet.princeton.edu/

[5] Brown Corpus - http://www.natcorp.ox.ac.uk/.

as medidas corretas a serem usadas para avaliar as ferramentas para uma tarefa de Engenharia de Requisitos (BERRY *et al.*, 2017).

3.3 Considerações do capítulo

No contexto deste trabalho, a mineração de dados é usada para extrair requisitos de software informações dos comentários de usuários nas lojas de aplicativos. As técnicas para essa extração foram explicadas neste capítulo. Também foi apresentada uma área de pesquisa dentro da Engenharia de Requisitos, a NPL4RE que pode ajudar na evolução do tema de extração de requisitos.

Esses conceitos serão utilizados para analisar os trabalhos e responder às questões de pesquisa definidas na revisão sistemática da literatura, método de pesquisa que foi usado nesse trabalho e explicado no próximo capítulo.

4 REVISÃO SISTEMÁTICA DA LITERATURA

A revisão sistemática da literatura (RSL) é um método de pesquisa que tem o objetivo de identificar, avaliar e interpretar estudos relevantes para responder questões de pesquisa de forma objetiva e sem vieses (AMPATZOGLOU *et al.*, 2020; KITCHENHAM; CHARTERS, 2007). Ela coleta e resume o conhecimento de diversos estudos primários para fornecer uma visão geral de um determinado assunto ou problema de pesquisa, podendo ser usada para identificar lacunas na pesquisa, criar agendas de pesquisa ou simplesmente discutir um determinado assunto (SNYDER, 2019).

Para que uma RSL seja conduzida de maneira sistemática, reduzindo vieses, um conjunto de passos deve ser seguido. Existem trabalhos que provêm guias para auxiliar os pesquisadores a executar a revisão em suas áreas de pesquisa (SNYDER, 2019). Na área de Engenharia de Software o guia mais utilizado é o de Kitchenham e Charters (2007). Nele são explicadas as principais fases para a condução de uma RSL, como mostra a Figura 5: planejamento da revisão, execução e escrita da revisão. A seguir serão apresentadas cada uma dessas fases.

4.1 Planejamento da revisão

A primeira fase que um pesquisador precisa fazer em uma revisão sistemática da literatura é planejar. No planejamento é importante definir o objetivo do estudo, o qual mostra que existe uma necessidade de se criar evidências empíricas para um determinado fenômeno descrito em um conjunto de trabalhos.

Depois que o objetivo é definido, vêm a definição das questões de pesquisa. Essa é a parte mais importante de qualquer revisão sistemática da literatura (KITCHENHAM; CHARTERS, 2007) pois são as questões de pesquisa que orientam a construção do protocolo da revisão.

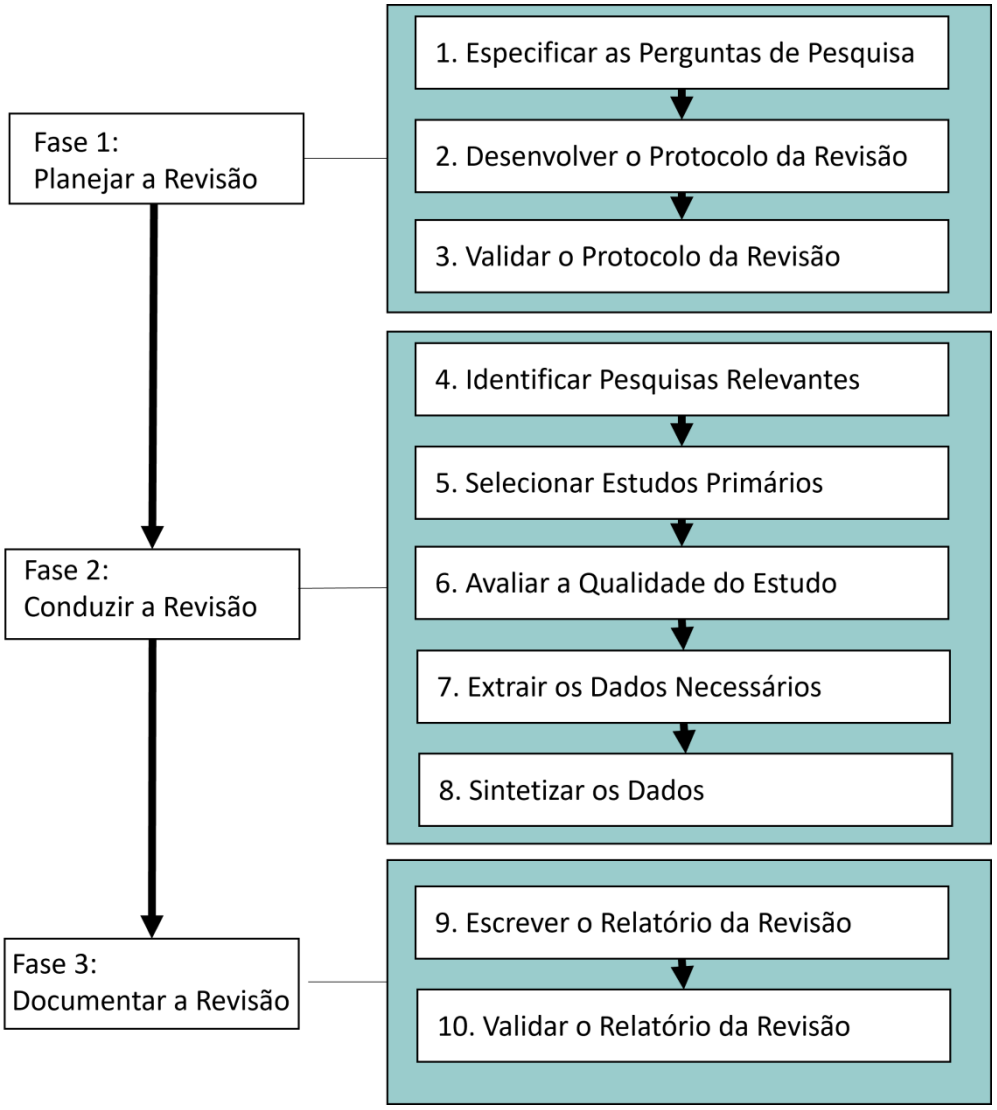

Figura 5 - Processo da revisão sistemática da literatura

Fonte: (BRERETON et al., 2007, tradução nossa)

O protocolo da revisão é um documento que contém as atividades que serão feitas na condução da revisão. Seu objetivo é minimizar os vieses do estudo (BRERETON *et al.*, 2007; KITCHENHAM; CHARTERS, 2007) e também permitir que o trabalho seja replicado. Os itens do protocolo de revisão segundo Kitchenham e Charters (2007) são:

- Justificativa para pesquisa;
- Questões de pesquisa;
- Estratégia de busca;
- Critérios de seleção de estudos;
- Procedimentos de seleção dos estudos;

- *Checklist* e procedimento de avaliação da qualidade;
- Estratégia de extração de dados;
- Síntese dos dados extraídos; e
- Estratégia de Disseminação.

A justificativa para a pesquisa e as questões de pesquisa são as primeiras informações documentadas no protocolo da revisão. Depois, a partir do tema das questões de pesquisa, define-se a estratégia de busca dos estudos primários. O objetivo da estratégia de busca é definir as bibliotecas digitais, os termos de busca que encontrem trabalhos relevantes para responder as questões de pesquisa (KITCHENHAM; CHARTERS, 2007) e como esses termos serão organizados formando a *string* de busca (*search query*).

As bibliotecas digitais indicadas por Kitchenham e Charters (2007) para a Engenharia de Software são: IEEExplore, ACM Digital Library, Google Scholar, Citeseerx, IET Digital Library, ScienceDirect, Engineering Village, SpringerLink e Scopus. Normalmente se escolhem bibliotecas que irão retornar trabalhos que foram publicados em periódicos, workshops e conferências relacionadas ao tema da revisão. Cada biblioteca de busca tem um mecanismo de pesquisa diferente da outra (BRERETON *et al.*, 2007), o que dificulta a escolha dos campos dos artigos que os termos serão buscados. Por exemplo, na biblioteca IEEExplore, quando o termo de busca é procurado no campo “All Metadata”, a busca é feita no resumo, nos termos que indexam o artigo, no título, no nome do autor etc.

Uma outra dificuldade é a construção da *string* de busca. Construí-la exige diversas tentativas de buscas nas bibliotecas digitais de forma a definir a organização lógica dos termos de busca e determinar em que casos a busca dos termos será exata ou aproximada. Assim como os campos de busca, a *string* de busca também precisa ser construída especificamente para cada biblioteca digital (BRERETON *et al.*, 2007) devido às suas diferenças.

Depois de a busca ser definida, os critérios de seleção dos estudos ou de inclusão e exclusão têm que ser criados baseados nas questões de pesquisa e nos testes da *string* de busca. O objetivo desses critérios é aumentar a assertividade dos estudos

primários pois o resultado de busca pode conter trabalhos que não respondem às questões de pesquisa ou não atendem a critérios mínimos, como o estudo não ser da área de pesquisa, a quantidade de páginas ou a data de publicação dos trabalhos.

Com a *string* de busca e os critérios de seleção de inclusão e/ou exclusão, inicia-se o planejamento da seleção de estudos. Nele é definido a ordem e onde os critérios de seleção de estudos serão aplicados. Aplicar os critérios de seleção na própria biblioteca digital pode economizar tempo e reduzir a complexidade da atividade. Já a ordem de seleção de estudos pode ajudar na economia de tempo dos pesquisadores, pois já é possível saber se o trabalho responderá as questões de pesquisa somente lendo o *abstract*.

Em alguns estudos são criados critérios de qualidade para selecionar estudos além dos critérios de inclusão e exclusão (KITCHENHAM; CHARTERS, 2007). Seu objetivo é detalhar mais os critérios de seleção. Os critérios de qualidade normalmente são avaliados através de um questionário como perguntas sobre o viés e validade do trabalho (KITCHENHAM; CHARTERS, 2007), como, por exemplo, avaliar se os efeitos observados no estudo são aplicáveis fora do estudo. Para cada estudo primário o questionário é respondido e suas respostas são comparadas a fim de escolher os trabalhos de maior qualidade para entrar na revisão.

A próxima atividade do planejamento da revisão é escrever a estratégia de extração de dados. Nela são definidas as informações que precisam ser obtidas dos estudos primários para que a questões de pesquisa possam ser respondidas e os trabalhos organizados (KITCHENHAM; CHARTERS, 2007), e também definir o processo de extração. É importante que na extração mais de uma pessoa participe para reduzir erros na execução e na interpretação da informação.

Depois de definir quais dados serão extraídos é importante determinar a forma que será feita a síntese dos dados extraídos. O objetivo dessa atividade é agregar as informações extraídas resumindo os resultados, o que pode ser feito de forma quantitativa ou qualitativa. Na área de Engenharia de Software é mais comum a síntese de dados qualitativa (KITCHENHAM; CHARTERS, 2007), ou seja, as RSL utilizam o resultado para argumentar sobre o que cada estudo primário está fazendo

e, caso muitos estudos relevantes façam a mesma coisa, podem até usar para generalizar a área de pesquisa.

Por fim, na estratégia de disseminação da revisão é importante definir onde a RSL será publicada de forma que ela atinja o público para o tema de pesquisa. Um aprendizado importante é manter o registro das decisões tomadas na execução da revisão (BRERETON *et al.*, 2007), o que pode ajudar os pesquisadores caso haja questionamentos do processo.

4.2 Conduzir a revisão

A execução do trabalho envolve praticar todas as fases do planejamento de forma sistemática, procurando gerenciar as ameaças à validade para evitar comprometer a credibilidade dos resultados da pesquisa (AMPATZOGLOU *et al.*, 2020) e com isso, aumentar a confiança nas direções futuras do trabalho.

Ampatzoglou et al. (2020) criou uma checklist que permite identificar as ameaças às validades da seleção dos estudos, dos dados e da pesquisa quando se executa uma revisão sistemática da literatura. Para cada resposta positiva à uma pergunta existe uma ação de mitigação dessa ameaça para que o pesquisador execute. É importante ter um tópico da revisão documentando as ações feitas em cada atividade de execução da revisão para mitigar os possíveis erros.

4.3 Documentar a revisão

Na escrita da revisão deve-se reportar todo o protocolo da revisão, expor os resultados e as ameaças a validade. Além disso, existem mais dois tópicos importantes para o leitor da revisão: a discussão dos resultados e direções futuras de pesquisa. O objetivo principal desses dois tópicos é voltar para o objetivo da pesquisa e discutir o que foi aprendido com o resultado obtido, como o resultado pode ajudar na evolução da área de pesquisa ou se os resultados precisam ser adaptados para a área de pesquisa com novos trabalhos (BRERETON *et al.*, 2007) e até propor problemas em aberto que novas perguntas de estudos futuros poderiam responder.

4.4 Considerações do capítulo

A revisão sistemática da literatura é um método de pesquisa útil para sintetizar resultados de diversos trabalhos que realizam estudos similares dentro do escopo definido. Seus resultados, discussões e direções futuras podem ser utilizadas como base para outros estudos. Este trabalho realiza uma revisão sistemática da literatura e seu planejamento e execução, seguindo o que foi discutido neste capítulo, foi detalhado nos próximos capítulos.

5 PROTOCOLO DA REVISÃO SISTEMÁTICA DA LITERATURA

Neste capítulo é apresentado o protocolo da revisão sistemática da literatura usado por este trabalho o qual foi baseado no guia de Kitchenham e Charters (2007). A seguir foram documentadas as atividades feitas.

5.1 Objetivo

Para atingir o objetivo deste trabalho, que é analisar a adequação para a Engenharia de Requisitos do requisito extraído automaticamente através de *feedbacks* explícitos de usuários de aplicativos móveis, foi feita uma busca por trabalhos que fazem o processo de extração de requisitos de aplicativos móveis. Essa busca encontrou uma área de pesquisa ativa na qual existem diversas propostas para extração de requisitos. Mas, para que essas propostas sejam usadas na prática, é importante que elas sigam os princípios e boas práticas da área de Engenharia de Requisitos (ER).

Então fez-se uma RSL para extrair o estado da arte dos processos de extração de requisitos praticados, para entender o quanto esses trabalhos possuem um ponto de vista da ER.

O objetivo dessa revisão sistemática é levantar informações sobre o formato, conteúdo e avaliação sobre o processo de extração de requisitos de software utilizando *feedbacks* explícitos de *crowd* (multidão) em larga escala, gerados por usuários de forma espontânea em canais de mídia e comunicação para aplicativos móveis.

5.2 Perguntas de pesquisa

Para analisar a adequação para a Engenharia de Requisitos do requisito extraído automaticamente através de *feedbacks* explícitos de usuários de aplicativos móveis a partir das informações levantadas pela RSL foram formuladas três perguntas de pesquisa:

QP1: Qual o formato de saída dos feedbacks explícitos no processo de extração de requisitos?

Dado que o resultado da extração é um texto, o objetivo desta pergunta é entender do que se trata esse texto para então discutir se o resultado da extração segue uma representação de requisitos específica (como história do usuário ou afirmação em linguagem natural estruturada) ou tem um formato próximo a uma.

QP2: Quais as classes de saída dos *feedbacks* explícitos no processo de extração de requisitos?

O objetivo desta pergunta é avaliar qual o conteúdo das classes geradas como saída do resultado da extração do processo de KDD. Nessa avaliação pretende-se analisar o que está sendo extraído pelos trabalhos e se podem ser considerados requisitos (requisitos funcionais, não funcionais, restrições ou alguma outra forma), assim como analisar o nível de refinamento deles (se são requisitos de *stakeholder*, de sistema ou de software). Dessa forma será possível entender qual o benefício que a área ER tem com a extração de requisitos no estado da arte atual.

QP3: Qual a avaliação de saída dos feedbacks explícitos no processo de extração de requisitos?

Esta pergunta tem como objetivo entender se algum trabalho que aplicou a extração recebeu um *feedback* sobre o resultado por um especialista em desenvolvimento de software e, principalmente, se essa avaliação foi sobre o formato e o conteúdo de suas saídas em complemento às perguntas de pesquisa anteriores.

5.3 Bibliotecas Digitais

Para pesquisar os trabalhos foram definidas como fontes a IEEE Xplore, ACM Digital Library e Springer Link. Essas bases de dados foram utilizadas pois são as recomendadas pelo guia de Kitchenham e Charters (2007) e ainda hoje contém trabalhos dos principais periódicos e conferências da área da Engenharia de Software e, portanto, relevantes para o objetivo desta pesquisa.

5.4 Estratégia de Busca

Com o objetivo de identificar as pesquisas relevantes para a RSL, a *string* de busca foi definida a partir do agrupamento dos três assuntos mais relevantes relacionados ao objetivo da RSL: comentários de usuários, requisitos de software e obtenção de requisitos. Dentro de cada assunto uniu-se as palavras-chave utilizando o operador OU, como apresentado na Tabela 1, e os grupos de palavras-chave dos assuntos que compõe o objetivo do trabalho foram unidos com o operador E. Uma das *strings* de busca completa usada para a pesquisa na base SpringerLink é (as *strings* completas se encontram no APÊNDICE A):

```
("user review" OR "user feedback" OR "user comment" OR "user opinion" OR "app review" OR "online review") AND ("software release" OR "feature request" OR "software feature" OR "user story" OR "requirements engineering" OR "software requirement") AND ("elicit*" OR "extract*" OR "analy*" OR "min*" OR "generat*" OR "gath*" OR "spec*")
```

Tabela 1 - String de busca por assunto

Assunto	Busca de palavras
Comentários de usuários *Feedbacks* explícitos de usuários de aplicativos	• user review • OU user *feedback* • OU user comment • OU user opinion • OU app review • OU online review
Requisitos de software	• software release • OU software feature • OU feature request • OU user story • OU requirements engineering • OU software requirement*
Obtenção de requisitos	• *elicit** • OU *extract** • OU *analy** • OU *min** • OU *generat** • OU *gath** • OU *spec**

Como estratégia da construção da *string* foram utilizados asteriscos após o começo de palavras-chave que pudessem ter variações dependendo do seu contexto. As palavras que de interesse para cada uma foram:

- *elicit**: pode recuperar trabalhos com a palavra *elicit* (elicitar em português), com a palavra *eliciting* (elicitando em português) e com a palavra *elicitation* (elicitação em português);
- *extract**: pode recuperar trabalhos com a palavra *extract* (extrair em português), com a palavra *extracting* (extrair em português) e com a palavra *extraction* (extração em português);
- *analy**: pode recuperar trabalhos com a palavra *analysyis* (análise em português) e com a palavra *analysing* (analisando em português);
- *min**: pode recuperar trabalhos com a palavra *mine* (minerar em português) e com a palavra *mining* (mineração em português);
- *generat**: pode recuperar trabalhos com a palavra generate (gerar em português) e com a palavra *generating* (geração em português);
- *gath**:pode recuperar trabalhos com a palavra *gather* (coletar em português) e com a palavra *gathering* (coleta em português); e
- *spec**: pode recuperar trabalhos com a palavra *specifying* (especificando em português) e com a palavra *specification* (especificação em português).

Apesar deste trabalho utilizar conceitos associados a *crowd* para comentários de usuários em larga escala, o termo não foi utilizado como uma palavra-chave. O motivo disso é que nos testes da *string* de busca, os estudos retornados por essa palavra estavam fora do escopo de pesquisa, indicando que, até o momento da pesquisa, não era um conceito utilizado nos principais trabalhos. Todas as buscas foram feitas em outubro de 2022.

Para a busca considerar os “títulos”, “resumos” e “palavras-chave” dos trabalhos das bases digitais, foi necessário criar uma *string* adaptada para cada base, apresentadas no APÊNDICE A, com diferentes campos na busca, como mostra a Tabela 2. O porquê dessa adaptação é explicado no capítulo sobre Revisão Sistemática da Literatura.

Seguindo um dos trabalhos relacionados (DĄBROWSKI *et al.*, 2022), os resultados foram limitados a publicações dos últimos 10 anos, portando trabalhos entre janeiro de 2012 e outubro de 2022.

Tabela 2 - Biblioteca digitais utilizadas, seus campos e data de busca

Bibliotecas digitais	Campo utilizado na busca	Data de busca
IEEE Xplore	All Metadata	08/10/2022
ACM	Anywhere	08/10/2022
Springer Link	Campo de busca simples	08/10/2022

Diferente da biblioteca digital IEEE Xplore, em que se utilizou o campo de busca avançada “All Metadata”, na ACM, não existe um campo de busca avançada para termos no “título”, “resumo” e “nas palavras-chave” de uma vez só, além de ter uma limitação de sete campos para busca (impedindo a construção manual). Assim, para a ACM foi utilizado o campo “Anywhere” que busca todas a informações do artigo. Por fim, Na Springer Link, diferente das outras duas bibliotecas digitais, a busca avançada não retornou resultados, mas a busca simples retornou, portanto, esta foi utilizada.

5.5 Seleção de estudos relevantes

Para realizar a seleção de estudos relevantes foi estruturado um processo em três fases. Na Fase 1, são organizados os dados das bases, aplicando o filtro da data de publicação e unificar a base com a remoção de artigos duplicados. Na organização dos dados, é necessário incluir todos os campos apresentados na Tabela 3. Caso o resultado da busca não tenha esse campo é possível extrair as informações no site de cada artigo. Depois o limite de data de publicação deve ser aplicado e os trabalhos duplicados devem ser removidos.

Tabela 3 – Campos extraídos das bases digitais

Campos
Fonte
Título do Artigo
Resumo do Artigo
Ano de publicação
Tipo de artigo
Local de Publicação
Autores
Link para o Artigo

Na Fase 2, os trabalhos terão seus “títulos” e “resumos” lidos e, a partir dessa leitura, poderão ser aplicados os critérios de seleção, conforme apresentado e numerado na Tabela 4 e seguindo a ordem:

1. Critério de exclusão 2 - Excluir trabalhos secundário e terciários;
2. Critério de exclusão 1- Excluir trabalhos dos seguintes tipos: índices, editoriais, *white papers*, comentários, resumos estendidos, comunicações, livros, tutoriais, artigos não revisados por pares – ou seja, os não publicados em conferências, periódicos, workshops e capítulos de livros;
3. Critério de exclusão 5 - Excluir trabalhos não escritos em inglês;
4. Critério de exclusão 4 - Excluir trabalhos fora do escopo da Engenharia de Software;
5. Critério de inclusão 1 - Incluir trabalhos em que as fontes de dados são *feedbacks* explícitos dos usuários de aplicativos móveis;
6. Critério de inclusão 2 - Incluir trabalhos em que a extração seja para obter requisitos de software.

O resultado da Fase 2 deve ser registrado em uma nova coluna - incluída na base unificada de artigos - chamada “Fase 2”. Para cada trabalho analisado, essa nova coluna deve ser preenchida com “excluído - <critério utilizado>” ou “não excluído”. Caso seja necessário ler o trabalho para avaliar com maior certeza o critério, a coluna desse trabalho deve ser preenchida com “não excluído”. No final da categorização somente os trabalhos com “não excluído” passam para a Fase 3.

Todos os critérios de seleção foram refinados e definidos com base no objetivo da revisão sistemática. Para refinar os critérios foram utilizados artigos relacionados ao tema da RSL que utilizaram os mesmos critérios como referência e foram feitos processos de seleção piloto para validar os resultados gerados. Os critérios podem ser conferidos na Tabela 4.

Tabela 4 - Critérios de inclusão e exclusão da revisão

Nº	Critério	Referência
	Inclusão	
1	Fontes de dados são *feedbacks* explícitos dos usuários de aplicativos móveis	
2	Trabalhos em que a extração seja para obter requisitos de um software	
3	Responder a pelo menos uma questão de pesquisa	
4	Trabalhos com acesso para leitura	
5	Trabalho mais recente de uma sequência do mesmo autor	
	Exclusão	
1	índices, editoriais, *white papers*, comentários, resumos estendidos, comunicações, livros, tutoriais, artigos não revisados por pares que não sejam: conferências, periódicos, workshops e capítulos de livros.	(ZHAO *et al.*, 2022)
2	O trabalho é um estudo secundário ou terciário	(ZHAO *et al.*, 2022), (DĄBROWSKI *et al.*, 2022)
3	Trabalhos que utilizam comentários de usuários que não sejam escritos em inglês ou não sejam traduzidos para o inglês.	
4	Trabalho que não estão relacionados ao escopo da Engenharia de Software	(DĄBROWSKI *et al.*, 2022)
5	Texto que não esteja escrito em inglês	(DĄBROWSKI *et al.*, 2022)
6	Trabalho que utiliza técnicas ou metodologias de extração de requisitos de outros trabalhos sem nenhuma modificação	

Por fim, na Fase 3 os trabalhos serão lidos completamente, de maneira mais rápida, com o objetivo de reaplicar os critérios de inclusão 1 e 2 para reforçar a avaliação da fase anterior, assim como aplicar os novos critérios de inclusão 3, 4 e 5, e aplicar os novos critérios de exclusão 3 e 5, conforme a ordem de itens abaixo:

- Critério de inclusão 4 - Incluir trabalhos com acesso para leitura;
- Critério de exclusão 5 - Excluir trabalhos em que o texto não esteja escrito em inglês;
- Critério de exclusão 3 - Excluir trabalhos que utilizam comentários de usuários que não sejam escritos em inglês ou não sejam traduzidos para o inglês;
- Critério de inclusão 5 - Incluir o trabalho mais recente de uma sequência do mesmo autor;
- Critério de exclusão 6 - Excluir trabalhos que utilizam técnicas ou metodologias de extração de requisitos de outros trabalhos sem nenhuma modificação;

- Critério de inclusão 1 - Incluir trabalhos em que as fontes de dados são *feedbacks* explícitos dos usuários de aplicativos móveis;
- Critério de inclusão 2 - Incluir trabalhos em que a extração seja para obter requisitos de software;
- Critério de inclusão 3 - Incluir trabalhos que respondem a pelo menos uma pergunta de pesquisa.

Para trabalhos do mesmo autor, serão lidos do mais recente para o mais antigo e, caso um trabalho seja sequência do outro, o último trabalho será escolhido. Porém, como algumas informações a serem extraídas podem ser apenas referenciadas de um trabalho anterior do autor, pode ser necessário consultar tais trabalhos.

O resultado da Fase 3 também deve ser registrado em uma nova coluna chamada “Fase 3”. Após a leitura de cada trabalho, essa nova coluna deve ser preenchida com “excluído - <critério utilizado>”, caso o trabalho passe no critério de exclusão ou não passe no critério de inclusão e “incluído”, caso contrário.

Ao final das três fases, os trabalhos com o rótulo “incluído” farão parte do estudo primário desta revisão. Uma nova base de estudos primários unificada - com remoção de artigos duplicados por título e resumo - deve ser criada.

5.6 Extração e síntese dos dados

Para a extração de informações que irão responder a cada pergunta de pesquisa os artigos são relidos com profundidade e novas colunas devem ser criadas na base unificada de artigos. Essas colunas incluem a identificação do artigo, o objetivo do trabalho, informações sobre o processo de extração e campos com informações para responder às perguntas de pesquisa, conforme indicado na Tabela 5.

.

Tabela 5 - Campos de extração para síntese

Pergunta de pesquisa	Campo de Extração
Campos padrão	ID do Artigo Objetivo trabalho
Informações sobre o processo de extração	Fonte do *feedback* explicito de usuário Forma de obtenção dos *feedbacks* Categoria de aplicativos da loja de aplicativo Técnicas utilizadas para extração de requisitos Uso da saída do processo de extração
QP1	Formato do Resultado da Extração
QP2	Conteúdo das classes do Resultado da Extração
QP3	Avaliação de saída dos *feedbacks* explícitos

Em relação às informações sobre o processo de extração, serão também obtidas a fonte do *feedback* explícito de usuário, a forma de obtenção do *feedback*, a categoria do aplicativo da loja de aplicativos, técnicas utilizadas para extração de requisitos e uso da saída do processo de extração caso o trabalho tenha. Essas informações adicionais serão usadas para análises secundárias.

A seguir, o processo de extração e síntese dos dados para cada pergunta de pesquisa será descrito.

5.6.1 QP1: Qual o formato de saída dos *feedbacks* explícitos no processo de extração de requisitos?

Para responder à pergunta sobre o formato de saída, olhou-se como os estudos primários apresentavam a saída do processo de extração. Muitos estudos usavam nomes como “tópico” quando havia o uso de técnicas de modelagem de tópicos (NOEI; ZHANG; ZOU, 2019; TUSHEV; EBRAHIMI; MAHMOUD, 2022), “rótulos” (*labels)* (AL-HAWARI; NAJADAT; SHATNAWI, 2021) e “categorias” (CHEN *et al.*, 2021; NOEI; ZHANG; ZOU, 2019; PANICHELLA *et al.*, 2016) em todos se referindo ao conteúdo do texto de saída do processo de extração e não ao formato. Esses termos ou são genéricos para análise de formato, ou são mais associados ao processo de KDD e suas técnicas do que a área de Engenharia de Software, portanto não servem para responder essa pergunta de pesquisa.

Então, buscou-se na área de Engenharia de Requisitos referências sobre escrita de requisitos dentro da atividade de Documentação ou Especificação de requisitos.

Segundo a área de ER, a linguagem natural é comumente utilizada (POHL; RUPP, 2015) para documentar requisitos de software e também pode ser estruturada em diferentes formatos.

Assim, para definir como os dados de saída do processo de extração dos estudos primários seriam sintetizados para responder a QP1, inspirou-se nos tipos de formatos de documentação explicados na Seção 2.4. Criou-se então 3 categorias para os formatos de saída, sendo 2 categorias de formatos de linguagem natural não estruturada e uma categoria de formato de linguagem natural estruturada:

- Conjunto de palavras: texto escrito em linguagem natural, com uma ou mais palavras, sem uma estrutura definida e sem sentido completo, como exemplificado na Figura 6;
- Sentença: texto escrito em linguagem natural não estruturada, mas que apresenta um sentido completo, como exemplificado na Figura 7;
- Padrão de texto: linguagem natural estruturada em formato para especificação de software, como exemplificado na Figura 8.

As três categorias foram atribuídas as saídas dos trabalhos quando estivessem no formato e foram usadas para sintetizar os resultados.

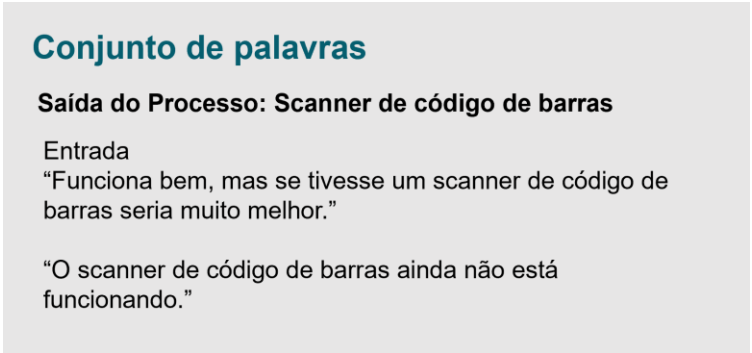

Figura 6 - Exemplo de formato de saída - conjunto de palavras

Fonte: (CARRENO; WINBLADH, 2013. tradução nossa)

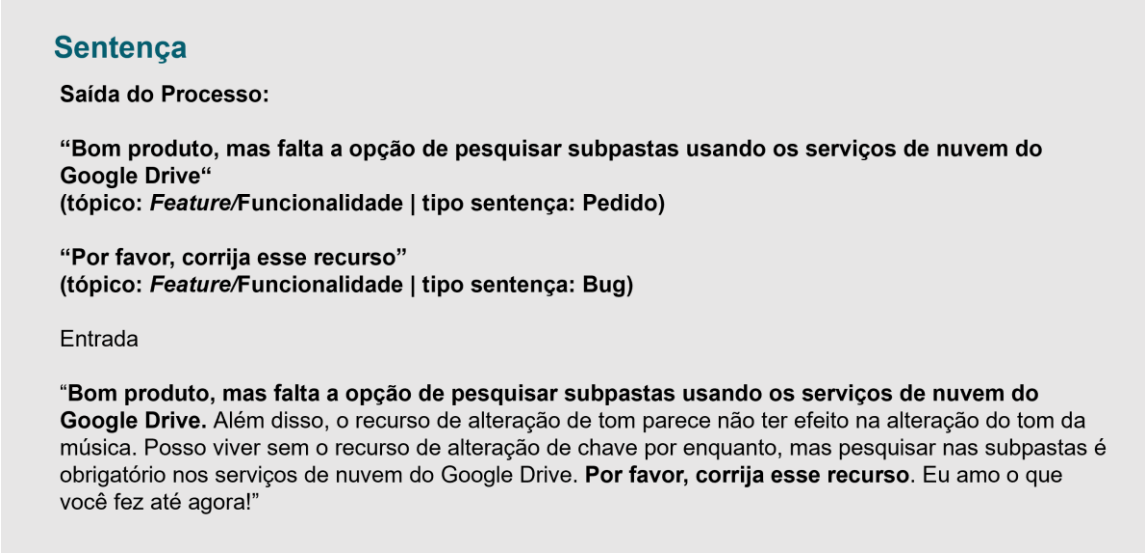

Figura 7 - Exemplo de formato de saída - sentença

Fonte: (DI SORBO *et al.*, 2017, tradução nossa)

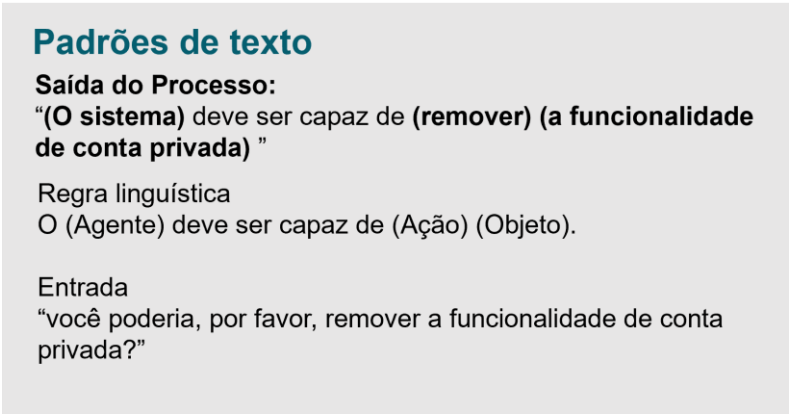

Figura 8 - Exemplo de formato de saída - padrão de texto

Fonte: (PANTHUM; SENIVONGSE, 2021, tradução nossa)

5.6.2 QP2: Quais as classes de saída dos *feedbacks* explícitos no processo de extração de requisitos?

Para a análise da saída dos *feedbacks* quanto aos tipos de requisitos e nível de refinamento utilizado pelos trabalhos, assunto dessa pergunta de pesquisa, foram utilizados os textos da saída do processo de extração que os estudos usaram para se referir aos assuntos no comentário, muitas vezes chamados pelos trabalhos de *categorias*.

Os textos sobre o conteúdo de saída dos trabalhos foram organizados com nome da classe e descrição de significado, quando presente no trabalho, em uma base de dados[6].

No exemplo da Figura 9, a saída do processo de extração do comentário específico contém quatro classes: Pedido, Interface gráfica de usuário, *Feature*/Funcionalidade e Melhoria. Dependendo do trabalho, essas classes ou categorias podem ser agrupadas por tipo, como nesse exemplo “Categoria da Intenção” e “Assunto”.

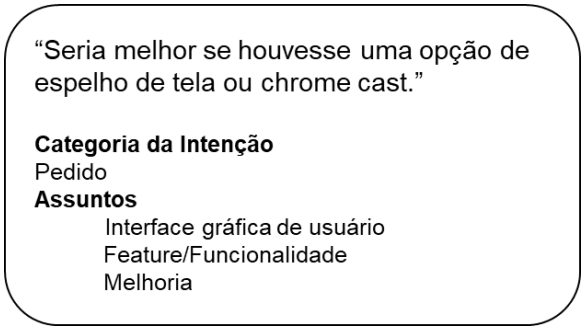

Figura 9 - Exemplo de saída do processo de extração

Fonte: (DI SORBO *et al.*, 2017, tradução nossa)

Para análise dessa QP, essas classes da saída do processo de extração foram extraídas junto da descrição de significado (quando presente no trabalho) e organizadas em uma base de dados.

Ao todo os 60 estudos primário criaram 373 textos para se referir ao conteúdo de saída do processo de extração. Esses textos chamados de classes de saída foram utilizados para responder a QP2.

Mais detalhes do processo de sintetização dos dados serão abordados no capítulo de resultados.

[6] Disponível em: <https://doi.org/10.5281/zenodo.7740478>.

5.6.3 QP3: Qual a avaliação de saída dos *feedbacks* explícitos no processo de extração de requisitos?

Para responder a essa pergunta de pesquisa foi analisado se o estudo primário aplica alguma avaliação sobre o seu resultado. Neste trabalho foi considerado que uma avaliação é qualquer opinião exposta no estudo por pessoas que não são os autores dos estudos primários e que tem domínio de Engenharia de Software.

Para a análise foram criados agrupamentos para responder como as perguntas foram feitas. Esses agrupamentos de referem aos métodos de avaliação:

- Questionário: pesquisa com questões que são apresentadas de maneira escrita.
- Entrevista: pesquisa com questões que são apresentadas de maneira oral por um entrevistador.

Esses formatos serão utilizados para a apresentação de discussão dos resultados.

5.7 Participação no trabalho

O protocolo da RSL deste trabalho foi criado a partir da evolução de uma outra revisão sistemática da literatura que também analisa o processo de extração de requisitos a partir de *feedback* de usuário. O processo dessa revisão foi feito pela pesquisadora deste trabalho com a participação do aluno de MBA do curso de pós-graduação em *Big Data,* Toshio Yassuda, pelos envolvidos terem uma motivação similar e poderem dividir tarefas para diminuir ameaças a validade na extração de dados (AMPATZOGLOU *et al.*, 2020).

Do trabalho feito em conjunto foi gerado um protocolo de revisão completo com 9 perguntas de pesquisa. Essa RSL chegou a ser conduzida até a atividade de extração de dados dos estudos primários. A partir da atividade de sintetização, os trabalhos se separaram. Toshio Yassuda adaptou o protocolo criado para responder a 6 perguntas de pesquisa (as mais relacionadas ao tema de Big Data), utilizando parte dos estudos primários selecionados e, com isso, produzindo uma revisão (YASSUDA, 2021) com

menor profundidade da análise (os resultados são apenas relatados, sem uma discussão).

Na revisão atual o protocolo foi revisto, reaproveitando a *string* de busca, assim como parte dos critérios de aceite do protocolo desenvolvido em conjunto e 3 perguntas de pesquisa. As atividades do desenvolvimento da revisão foram refeitas a partir do novo protocolo e no resultado da extração apareceram 30 estudos primários em comum com a revisão anterior. Dado que 3 perguntas de pesquisa permaneceram, foi possível aproveitar campos extraídos para responder às perguntas de pesquisa. Por fim, a sintetização dos dados extraídos, análise, discussão de resultados e conclusão foram feitas pela pesquisadora deste trabalho, com a avaliação de seu orientador.

5.8 Considerações do capítulo

Este capítulo apresentou o protocolo da revisão sistemática da literatura seguindo o trabalho de Kitchenham e Charters (2007). Foram apresentadas as questões de pesquisa deste trabalho, as bibliotecas digitais utilizadas, a estratégia de busca seguida, o plano de seleção de estudos e extração e síntese de informações para a resposta às perguntas. Também se evidenciou a divisão do trabalho feita. No capítulo seguinte serão apresentados os resultados da pesquisa.

6 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos pela revisão sistemática da literatura, seguindo o protocolo descrito no capítulo anterior.

6.1 Seleção de estudos primários

A busca de estudos foi realizada usando as *strings* indicadas no APÊNDICE A, sem aplicar nenhum filtro nas bases, e então a lista de resultado foi baixada. Cada base possui algumas diferenças: na IEEE Xplore foi possível baixar os dados bibliográficos dos trabalhos no formato ".csv"; na Springer Link, como há uma limitação de 1000 referências para download, foram feitas buscas considerando intervalos de anos; e na ACM os dados dos artigos foram extraídos como citação do tipo bibtext e depois convertidos para ".csv" usando um site de conversão *BibTeX to CSV converter*[7]. Depois, foi criado um arquivo Excel com as três bases ".csv" em cada planilha.

A busca nas bibliotecas digitais foi feita em outubro de 2022 e resultou em 2.810 artigos no total, distribuídos em três bases conforme a Figura 10.

Com esses trabalhos iniciou-se o processo de seleção dos estudos primários. Como discutido na Seção 5.4, esse processo foi feito em três fases e foi descrito no fluxograma do processo de seleção de estudos da Figura 11.

Na primeira fase, os dados bibliográficos extraídos de cada base foram organizados, seguindo o protocolo. De todas as bases, a SpringerLink foi a única que não continha o resumo dos artigos na extração. Como a falta dessas informações poderia prejudicar o processo de seleção dos estudos, foi criado um programa[8] para extrair o resumo do site do artigo para os estudos nessa base.

---

[7] BibTeX to CSV converter - https://www.bibtex.com/c/bibtex-to-csv-converter/

[8] O programa está disponível em: https://colab.research.google.com/drive/1VblZ__mpyjeBfxC_VX6jH8gh_v-MuxGA?usp=share_link

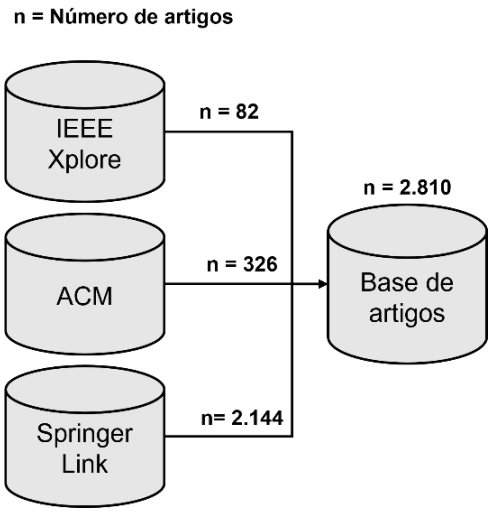

Figura 10 – Resultado da busca nas bibliotecas digitais

Ainda na mesma fase, foi aplicado o filtro de trabalhos nos últimos 10 anos, resultando em estudos entre janeiro de 2012 e outubro de 2022. Depois as bases da IEEE Xplore, ACM e SpringerLink foram unificadas e os trabalhos duplicados foram removidos, resultando em 2.066 trabalhos.

Iniciou-se a Fase 2, na qual os estudos foram selecionados com base na leitura do título e resumo. Depois, os 198 estudos que passaram para a Fase 3 foram selecionados baseado na leitura completa.

Tanto para a Fase 2 quando para a Fase 3, os trabalhos que foram excluídos por estarem fora do escopo do trabalho (na Figura 11), não passaram pelos critérios de aceite: fontes de dados são *feedbacks* explícitos dos usuários de aplicativos móveis (critério de inclusão 1 da Tabela 4); trabalhos em que a extração seja para obter requisitos de software (critério de inclusão 2 da Tabela 4); e trabalho que estão relacionados ao escopo da Engenharia de Software (critério de exclusão 3 da Tabela 4).

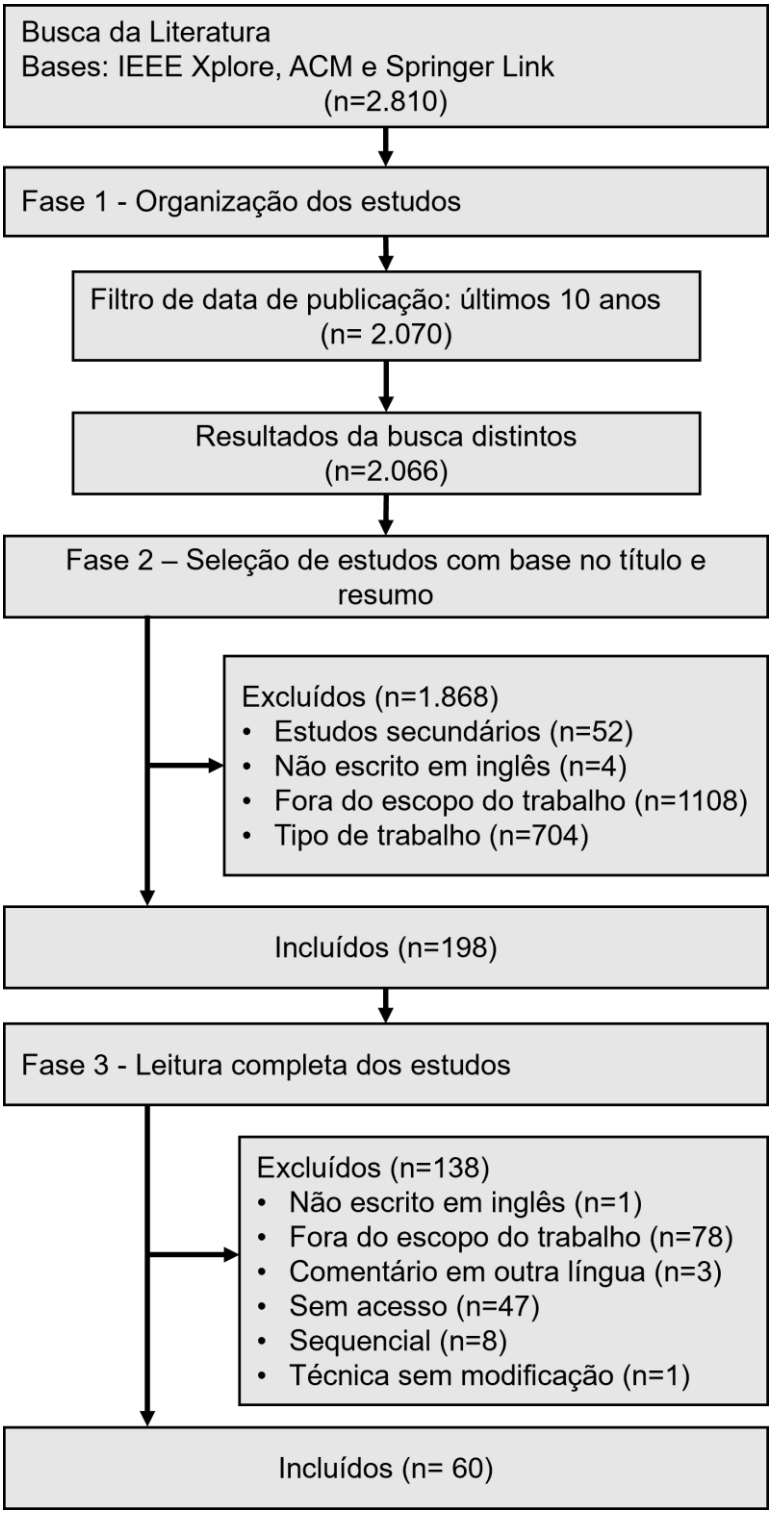

Figura 11- Processo de seleção de estudos, baseado no trabalho de Liberati et al. (2009).

Os trabalhos que foram excluídos por serem sequenciais não passaram pelos critérios de aceite: trabalho que utiliza técnicas ou metodologias de extração de requisitos de outros trabalhos sem nenhuma modificação (critério de exclusão 6 da Tabela 4) e trabalho mais recente de uma sequência do mesmo autor (critério de inclusão 5 da Tabela 4).

Para a leitura dos trabalhos na Fase 3 foi necessário o acesso aos textos na íntegra, o que não foi possível para 47 trabalhos da Springer Link devido a limitação de bases digitais disponíveis para a CAPES e a Universidade de São Paulo. Assim, esses trabalhos foram excluídos por não passarem no critério de aceite de acesso para leitura (critério de inclusão 4 da Tabela 4).

No final da seleção de estudos, chegou-se em 60 estudos primários relevantes para a revisão sistemática da literatura. Como mostra a Figura 12, os estudos primários foram publicados de 2013 a 2022 em conferências, periódicos, seminários e simpósios. Em 2021 ocorreu o maior número de publicações - 12 no total - e em 2015 houve somente uma publicação. No geral, os trabalhos selecionados foram mais publicados em conferências - 38 trabalhos (63%) - do que os outros locais. A lista com os 60 trabalhos pode ser encontrada no APÊNDICE B.

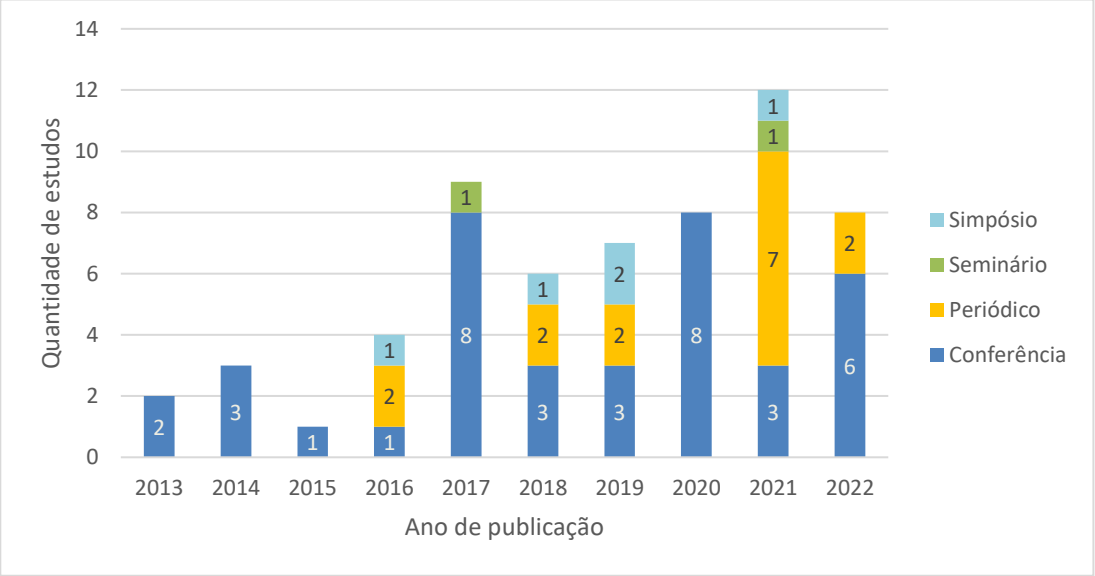

Figura 12 - Estudos primários por tipo de local de publicação

Os locais com mais publicações foram as conferências *International Conference on Software Engineering* (ICSE) com 12 artigos e a *International Requirements Engineering Conference* (RE) com 9 artigos. Esses locais são da área de software. Os outros lugares de publicações tiveram menos que três artigos publicados. Mais detalhes são apresentados no APÊNDICE C.

Os dados da RSL foram disponibilizados para a consulta em uma base de forma aberta[9] . A seguir, as respostas as perguntas de pesquisa são apresentadas.

6.2 QP1: Qual o formato de saída dos *feedbacks* explícitos no processo de extração de requisitos?

O formato de saída do processo de extração de requisitos mais presente nos trabalhos é o de “sentença”, sendo utilizado por 50 estudos primários (83%), o formato “conjunto de palavras” é usado por 18 estudos (30%) e 2 estudos (3%) usam o formato “padrão de texto” na saída, conforme a Figura 13. Alguns trabalhos usam mais de um formato.

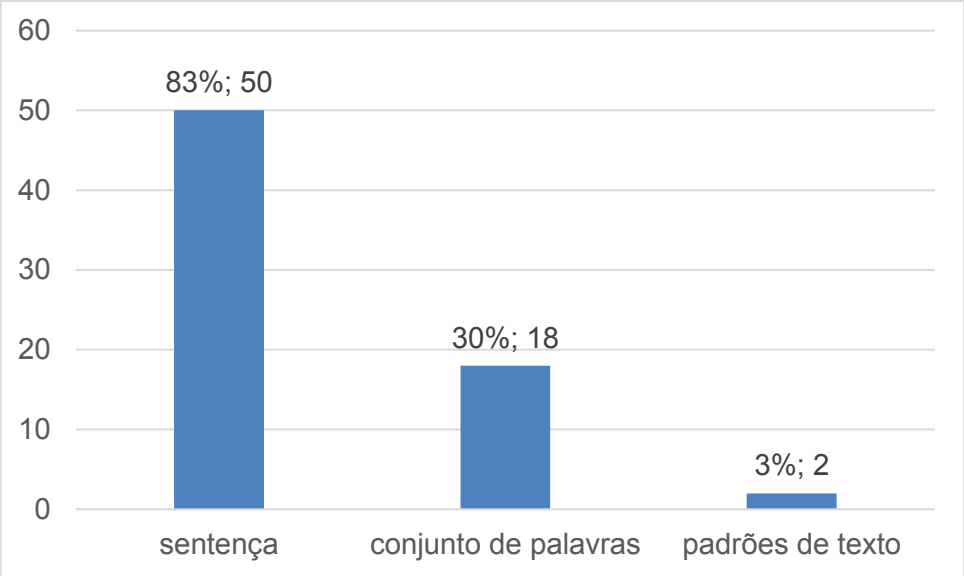

Figura 13 - Gráfico com formatos mais utilizados pelos estudos

O formato “sentença”, que é o mais presente nos estudos primários, é similar ao comentário original do usuário, usado como entrada do processo de extração, portando ainda é possível existir ambiguidade, dificultando a interpretação por pessoas com conhecimentos diferentes, um problema conhecido na área de ER (POHL; RUPP, 2015).

A única modificação feita em alguns trabalhos em relação à entrada, é a quebra dos comentários que tem mais de uma sentença. Por exemplo, no trabalho de Di Sorbo et

[9] Base de dados da revisão sistemática da literatura - https://doi.org/10.5281/zenodo.7740478

al. (2017) um comentário de usuário tinha cinco sentenças que foram separadas, então a saída ficou menor que a entrada, como pode-se ver na Figura 14. Apesar da quebra dos comentários, não houve modificação nas sentenças para um formato mais próximo a um requisito como o uso de determinados verbos como “dever” e “poder” escritos na voz ativa e da escrita da sentença ser de forma afirmativa.

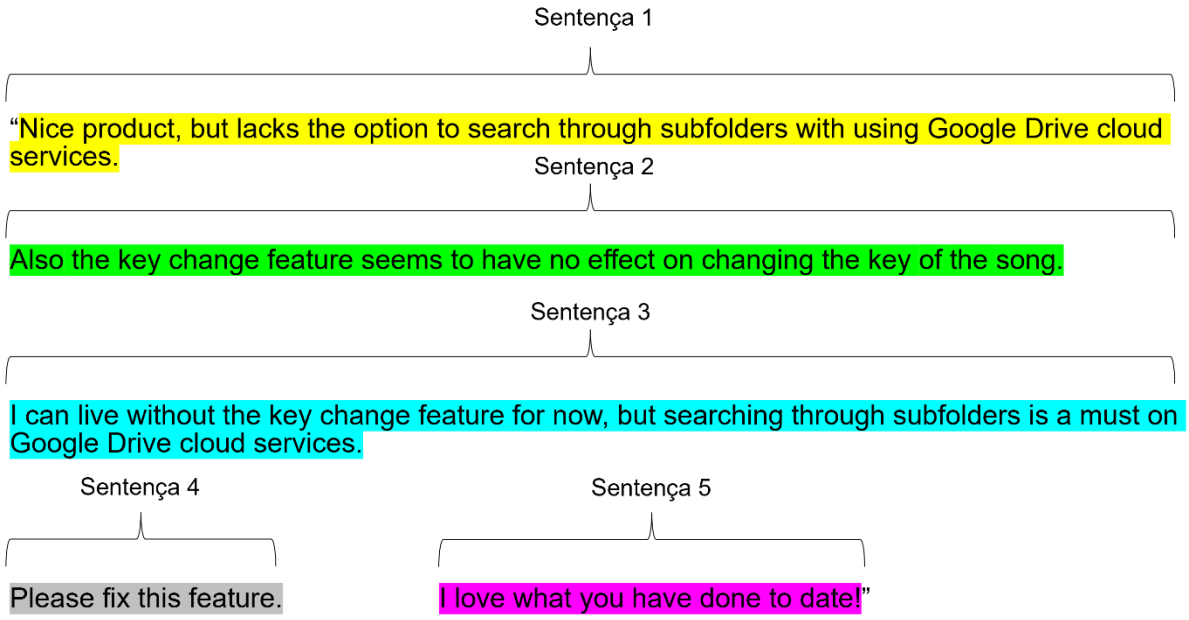

Figura 14 - Exemplo de comentário de usuário mais de uma sentença

Fonte: Base de demonstração do trabalho (DI SORBO et al., 2017)

Outra característica desfavorável a esse formato é que, quando há quebra do comentário, pode haver falta de contexto na leitura de uma sentença só. No exemplo da Figura 14, ao ler as sentenças separadas o assunto principal do comentário não é entendido.

Por fim, a saída do processo de extração em sentenças não pode ser definida como um requisito pois a sentença do *feedback* da forma que está, não se refere ao sistema ou software como um requisito. A sentenças estão mais adequadas para serem usadas como necessidades do usuário que precisam de refinamento, análise e transformação para um formato mais próximo de uma especificação de requisitos.

O “conjunto de palavras” é um formato que pode vir com uma palavra única, por exemplo, a saída: “melhor, melhorar, executar, adicionar, *feature*” (NOEI; ZHANG;

ZOU, 2021, tradução nossa) ou, pode vir com mais de uma palavra como “baixar_foto” ou “mudar_nome” (GUZMAN; MAALEJ, 2014, tradução nossa).

Diferentemente do formato “sentença”, o “conjunto de palavras” é um texto bem mais curto formado por verbos e substantivos. No entanto, ler essa saída curta sem mais informações pode gerar dúvidas sobre o contexto e significado das palavras, o que pode dificultar a atividade de Engenharia de Requisitos.

Se observa que os trabalhos têm agrupado o formato “conjunto de palavras” de diferentes maneiras. Por exemplo, o trabalho de Carreno e Winbladh (2013) extraiu as palavras “Atualizações, *Features*, Desenvolvedores, Mensagens, Fotos, Amigos, Chat Notificações e Sony Ericson”. Sozinhas essas palavras ficam sem contexto sobre qual mensagem precisa transmitida ao leitor, mas ao saber que são palavras sobre comentários do aplicativo Facebook, é mais fácil entendê-las. O mesmo acontece no agrupamento por categorias de aplicativos, por exemplo, as palavras “aplicativo, ordem, tempo, segundo, travar, tentar, duas vezes, abrir, usar, a cada” que são sobre aplicativos dentro do mesmo contexto de entrega de comida (TUSHEV; EBRAHIMI; MAHMOUD, 2022).

Esses agrupamentos podem ajudar a complementar o entendimento sobre o conjunto de palavras extraído, mas não desqualifica a necessidade de conhecer os assuntos relacionados ao aplicativo ou às categorias do aplicativo.

Observou-se que dos 60 trabalhos, 10 trabalhos (17%) apresentam a combinação dos formatos “conjunto de palavras” e “sentença”, e poucos trabalhos possuem somente o formato “conjunto de palavras”, como mostra a Tabela 6. Talvez a necessidade de entender o contexto da saída no formato “conjunto de palavras” pode ter gerado a inclusão do formato “sentença”, para trazer o entendimento completo da saída. Por exemplo, no trabalho de Assi et al. (2021, tradução nossa) ao ler o conjunto de palavras “inscrever-se, pagar, vitalício, compra, interrupção da assinatura” com a sentença junto “Depois que atualizei para premium, a função de registro de chamadas desapareceu”, o entendimento do conjunto de palavras fica mais completo.

Apesar das tentativas de melhorar a apresentação dos dados para o formato de “conjunto de palavras”, ainda não é um dos melhores formatos, além disso não é uma

forma da escrever as especificações de requisitos e não tem informação suficiente para ser um requisito.

Tabela 6 - Formatos presentes nos trabalhos

Formatos nos estudos primários	Quantidade de trabalhos
sentença	40 (67%)
conjunto de palavras e sentença	10 (17%)
conjunto de palavras	8 (13%)
padrões de texto	2 (3%)
Total Geral	60

Já para o formato “padrões de texto”, foram encontrados somente 2 trabalhos (3%) que geram esse formato na saída. O trabalho de Panthum e Senivongse (2021) gerou a saída com seis variações de padrão de texto ou sintaxe, seguindo o padrão da norma ISO/IEC/IEEE 29148 (2018). Quando se compara os padrões da norma e o proposto pelo trabalho, conforme a Tabela 7, é possível ver que os trabalhos têm componentes com nomes similares, como <Objeto> e <Ação>, mas que tem algumas diferenças:

- no padrão do estudo primário existem termos fixos na escrita como a expressão “dever ser capaz de”, diferente do padrão da ISO/IEC/IEEE 29148 (2018) que fala para evitar esse tipo de combinação de termos e sugere usar verbos como “dever” e “poder”;
- existem componentes com nomes diferentes, mas que significam a mesma informação, como <Sujeito> e <Agente> que são quem pratica a <Ação> na sentença e, portanto, o mesmo componente;
- os componentes <Condição> e a <Limitação de uma ação> da norma não foi replicado no estudo primário; e
- o componente <Como>, que é algo proposto pelo estudo primário, é escrito com a palavra “*Way”* que significa caminho, mas deveria ter sido usado o termo “*How”,* que indica a forma com que determinada ação foi feita.

Já o trabalho de Guo e Singh (2020) gerou uma saída com um padrão seguindo uma definição própria do que seria uma história de usuário: uma sequência de eventos ordenados em que um usuário relata sua interação com um aplicativo. Esses eventos, segundo o trabalho de Guo e Singh (2020), têm dois tipos: “Ação de usuário”, relatada

em comentário que pode indicar expectativas do usuário, ou um “Problema no aplicativo”, que é o relato de um comportamento indesejável que o app teve após uma ação do usuário que violou a sua expectativa.

Tabela 7 - Comparação do padrões do estudo primário ao da ISO/IEC/IEEE 29148

ISO/IEC/IEEE 29148	Padrão do estudo primário
<Condição><Sujeito><Ação><Objeto><Limitação de uma ação> <Sujeito><Ação><Limitação de uma ação>	O <Agente> deve ser capaz de <Ação> <Objeto>. O sistema deve permitir que < Objeto> esteja em <Agente>. O <Agente> deverá ser capaz de <Ação> <Objeto> com <Como>. O < Agente> deve permitir que um usuário seja capaz de <Ação> < Objeto >. O <Agente> deve ter < Objeto>. O <Agente> deve < Ação > < Objeto >.

Fonte: (ISO/IEC/IEEE, 2018; PANTHUM; SENIVONGSE, 2021, adaptação e tradução nossa)

Ao comparar a história de usuário do estudo primário (GUO; SINGH, 2020) e o modelo de história de usuário proposto por Wautelet et al. (2014), pode-se observar que, apesar de o estudo primário usar o nome história de usuário, ele não segue o padrão da história de usuário, descrito na Tabela 8.

Esse estudo usa um padrão próprio, não citando uma referência sobre histórias de usuário da área de Engenharia de Requisitos de software, como o trabalho de Wautelet et al. (2014).

Tabela 8 - Comparação do padrão do estudo primário ao história de usuário

História de usuário	Padrão do estudo primário
Eu como: <papel> quer/quer/precisa/pode/gostaria: <tarefa> para que <objetivo>	Ação de usuário: < Ação de usuário > Problema no aplicativo: <Problema no aplicativo>

Fonte: (GUO; SINGH, 2020; WAUTELET *et al.*, 2014)

O formato mais adequado para a Engenharia de Requisitos depende de qual atividade usará a saída do processo. Para o uso da saída do processo de extração na atividade de elicitação de requisitos, o formato de sentença é adequado desde que tenha o conteúdo com informações sobre as necessidades do usuário com informações sobre o contexto do software de forma a reduzir ambiguidades. Já para a atividade de documentação de requisitos, o formato mais adequado depende do formato usado pelo projeto, como, por exemplo, a sintaxe da ISO/IEC/IEEE 29148 (2018) para requisitos funcionais ou a história de usuário (COHN, 2009).

6.3 QP2: Quais as classes de saída dos *feedbacks* explícitos no processo de extração de requisitos?

Para responder essa QP foi analisado o conteúdo das classes previamente organizadas na base de dados conforme explicação no capítulo de Extração e Síntese dos dados. Ao todo foram encontradas 373 classes presentes nos estudo primários.

A análise do conteúdo tem o objetivo de entender as seguintes perguntas:

1. Quantas classes de saída de texto únicas existem nos estudos primários?
2. As mesmas classes de saída de texto são utilizadas em estudos diferentes e quais as classes mais utilizadas?
3. Qual o nível de detalhe ou de refinamento, o tipo do requisito e se existem outros conteúdos?

A seguir, para cada pergunta será explicado a análise feita e dada a resposta.

6.3.1 Quantas classes de saída de texto únicas existem nos estudos primários?

Ao analisar as classes para responder a essa primeira pergunta, observou-se que muitas classes de saída eram parecidas em significado, então elas foram normalizadas de duas formas antes de serem analisadas.

A primeira normalização foi transformar as palavras para o singular, por exemplo, classes como *features* ficaram como *feature*. Já a segunda adaptação foi remover inflexões nas palavras de forma que não houvesse alteração no significado das classes para o autor do estudo primário, por exemplo em *recommendation* (PAGANO; MAALEJ, 2013) e *recommending* (KUNAEFI; ARITSUGI, 2021), foi possível normalizar para *recommend,* mas em classes como *recoverability* (GROEN *et al.*, 2017a) e *recovery* (DINIZ; DE SOUZA FILHO; CARVALHO, 2022) não foi possível normalizar, pois no primeiro caso o estudo primário se refere a um atributo de qualidade de produto segundo a ISO/IEC 25010 e o segundo se refere a uma heurística de usabilidade (sobre o usuário se recuperar de erros).

A normalização das 373 classes resultou em 198 classes únicas saindo do processo de extração de requisitos dos estudos primários. Em outras palavras, uma redução de 47% de classes com a normalização.

Olhando a proporção de classes únicas e quantidade de trabalhos, é possível dizer, de modo geral, que em média existem 3 classes por estudo primário (198 dividido por 60). Para entender melhor, a próxima pergunta analisará como os estudos primários têm usado essas 198 classes.

6.3.2 As mesmas classes de saída de texto são utilizadas em trabalhos diferentes e quais as classes mais utilizadas?

Para responder a essa pergunta, as referências e explicação do uso das classes nos trabalhos foram analisadas. Uma questão é saber se muitos trabalhos estão usando uma determinada classe, então contou-se em quantos estudos primários uma classe aparecia. Essa informação é apresentada na Tabela 9.

Tabela 9 - Principais classes usadas pelos trabalhos.

Classe	Quantidade de trabalhos
feature request	27
<varia>	19
bug report	16
other	16
usability	8
problem discovery	7
rating	6
reliability	5
user experience	5
information seeking	5
information giving	5
functional requirement	5
performance	5
improvement request	4
portability	4
security	4
user interface	3
problem report	3
supportability	3
inquiry	3
dependability	3
praise	3
compatibility	3

Olhando os dados da Tabela 9, observa-se que a classe de nome *'feature request'* está presente em 27 trabalhos (45%) - quase a metade dos estudos primários. Já a classe de nome "<varia>", está aparece em 19 trabalhos (31%) e as classes de nome "*bug report"* e *"other"* estão presentes em 16 trabalhos (27%) cada.

A classe de nome '<varia>' se refere a classes que não tem padrão de nome pois ele depende do conteúdo no momento do processamento. Por exemplo os assuntos dos tópicos extraídos no trabalho de Zhou et al. (2022) variam conforme os *feedbacks* sobre a última versão do aplicativo que muda com uma certa frequência.

Dos 19 estudos que a classe '<varia>' aparece, segundo Tabela 9, em 8 estudos (ASSI et al., 2021; CARRENO; WINBLADH, 2013; DĄBROWSKI et al., 2020; GUZMAN; MAALEJ, 2014; MALGAONKAR; LICORISH; SAVARIMUTHU, 2022; NAKAMURA et al., 2021; TUSHEV; EBRAHIMI; MAHMOUD, 2022; ZHOU et al., 2022) só existe essa classe sozinha, então é difícil saber qual o conteúdo que esses trabalhos extraem pois podem variar dependendo da entrada do processo de extração.

Analisando as classes mais usadas foi possível identificar que uma parte dos trabalhos seguem as mesmas referências. O primeiro trabalho encontrado, de Panichella et al. (2016) o qual teve suas classes de saída (*information giving, information seeking, feature request, problem discovery, other*) utilizadas por 5 trabalhos (AL-HAWARI; NAJADAT; SHATNAWI, 2021; ALI; JOORABCHI; MESBAH, 2017; DI SORBO *et al.*, 2017; PALOMBA *et al.*, 2017; SUN *et al.*, 2021). Tanto o trabalho de Panichella et al. (2016) quando o trabalho de Maalej et al. (2016) usam como referência para criar suas classes de saída com as classes encontradas nos *feedbacks* pelo trabalho de Pagano e Maalej (2013).

Outro trabalho que usa referências é trabalho de Pandey; Litoriya e Pandey (2019), esse identifica as classes dos trabalhos de Maalej et al. (2016) e McIlroy et al. (2016) em seus comentários. Já o trabalho de Xiao et al. (2021) usa o padrão do trabalho de Di Sorbo et al. (2017), o qual usa o padrão do Panichella et al. (2016).

Ao analisar o trabalho mais antigo utilizado como referência inicial para os outros trabalhos Pagano e Maalej (2013) não se encontra referências de padrões para a ER para requisitos não funcionais (ISO/IEC, 2010) ou requisitos funcionais (ISO/IEC 29148, 2018).

Esse comportamento de reutilizar classes como referência pode estar relacionado ao foco dos trabalhos em melhorar a técnica de extração e não o conteúdo que está sendo extraído, ou foco em procurar informações em uma nova fonte de *feedback* utilizando técnicas de outro trabalho, como o trabalho de Nayebi, Cho e Ruhe (2018) que utiliza as técnicas e os padrões de texto de Di Sorbo et al. (2017) para explorar dados do Twitter.

Além dos resultados de palavras que mais aparecem apresentados, foi possível identificar padrões relacionado ao uso das classes e assuntos para a ER conforme seu significado para o trabalho. Essa descoberta foi usada para responder a próxima pergunta.

6.3.3 Qual o nível de detalhe ou de refinamento, o tipo do requisito e se existem outros conteúdos?

A fim de continuar a análise das classes de saída para responder QP2, as classes foram agrupadas e analisadas conforme pedido de *feature* e bugs; e tipos de requisito (Requisitos não funcionais, Requisitos funcionais, *Features*) Pedido de *feature* e bug/defeito.

6.3.3.1 Pedido de feature e bugs

Existiam classes com o mesmo significado que '*feature request' (*pedido de *feature*) e '*bug request*' (pedido de bug), mas que tinham nomes diferentes. Com isso foram propostas três novas categorias para agrupar os nomes de conteúdo com base nas explicações dada pelos trabalhos:

- Bug/Defeito: se refere a termos *como* bug, defeito, problema de software, e podem ter sido definidos como problemas no aplicativo, comportamentos inesperados no aplicativo;
- Pedido de *feature*: se refere a termos relacionados ao pedido de novas *features* presente na saída do processo de extração de requisitos;
- Bug/Defeito e Pedido de *feature*: se refere a texto de saída que não diferencia entre pedido de *feature* e bug/defeito, mas deixa claro que sua saída tem esse conteúdo; e
- Não definido - se refere a classificação de um estudo que não tem relação com os grupos anteriores ou não fazem a separação de um ou outro.

As classes foram agrupadas gerando uma nova categoria conforme a Tabela 10.

Pode-se ver que tanto "Pedido de Feature" quando "Bug/Defeito" estão presentes em metade dos estudos primários. A diferença é que para o primeiro existem 7 nomes classe de saída diferentes e para o segundo existem 13 nomes classe de saída diferentes. A presença dessas duas classificações em metade dos trabalhos, mostra uma preocupação dos pesquisadores em se gerar o que é pedido de requisito e é bug/defeito do software.

Esses dois nomes de classes podem ser associados a uma solicitação de mudança de software. Se nessa solicitação existir um pedido de requisito novo, esse irá passar pelo processo de Engenharia de Requisitos em ele será refinado até uma especificação para então ser desenvolvido. Já se for um relato de um bug, ele precisará passar pelo processo de Gestão de Mudança.

Um fator importante para saber se é um *bug* ou requisito é saber qual a versão do app que o comentário se refere, para saber a qual momento do software. Somente 4 trabalhos (CHEN *et al.*, 2014; GUZMAN; MAALEJ, 2014; PHETRUNGNAPHA; SENIVONGSE, 2019; ZHOU *et al.*, 2022) citam que extraem a informação da versão do aplicativo a qual o usuário se referia ao mandar o *feedback*.

Tabela 10 - Nomes de conteúdo por categoria de bug/defeito e feature request

Categoria	Classe de Saída	Quantidade estudos **(% dos estudos primários)**
Bug/Defeito	App Problem, Bug, Bug Report, Compatibility Issue, Crashing, Defect report, Functional Complaint, Installation issue, Network connection issue, Network Problem, Problem Discovery, Problem Report, Severe Problem (SP)	30 (50%)
Pedido de feature	Feature Request, Improvement, Improvement Request, Inquiry, Mild Suggestion (MS), Request, FR (feature request)	32 (53%)
Bug/Defeito e Pedido de feature	Compatibility, Core feature of app (CFP), Functional requirement, Functional suitability, Improvement Request, Informative, Maintainability, Performance efficiency, Portability, Reliability, requirement, Security, Usability	7 (12%)
Não definido	<verificar na base do estudo>	56 (93%)

A classe de saída que apresenta “Bug/Defeito e Pedido de feature” juntos, aparece em 7 trabalhos (CHEN *et al.*, 2014; GUZMAN; IBRAHIM; GLINZ, 2017; LICORISH; SAVARIMUTHU; KEERTIPATI, 2017; LU; LIANG, 2017; PAGANO; MAALEJ, 2013; SRISOPHA; ALFAYEZ, 2018; WANG *et al.*, 2018). Quando se analisa essa classe não parece ter tanta utilidade pelo nível de detalhe apresentado, mas esses

pesquisadores extraem também classes com mais detalhes sobre o que é a *feature* ou defeito como, por exemplo o estudo (SRISOPHA; ALFAYEZ, 2018) que identifica atributos de qualidade nos comentários.

6.3.3.2 Requisitos Não Funcionais

Ao analisar outros nomes de classe, nota-se que, das classes de saída que aparecem mais vezes, aparecem nomes de atributos de qualidade segundo a ISO/IEC (2010) ou relacionados à qualidade do software como *'usability'* que aparece em 8 (13%) estudos, *'reliability'* e *'performance'* que aparecem em 5 (8%) estudos cada. Porém, de forma semelhante a *'feature request'* e *'bug report'*, os trabalhos usam termos diferentes para se referir a um mesmo atributo de qualidade.

Então, para investigar a existência de outros nomes de classe que podiam estar relacionados à qualidade de software, agrupou-se os nomes de classe conforme as oito características do modelo de qualidade de produto (ISO/IEC, 2010) e suas subcaracterísticas, quando eles têm o mesmo nome ou o estudo explica como sendo relacionado. Quando não é possível identificar as características da ISO/IEC (2010) mas o trabalho utiliza uma referência como padrão, os nomes de classe foram agrupados em "Outro padrão". Por fim, quando a classe de saída for sobre qualidade de software, mas não identificar qual a características de qualidade foi agrupado como "Não especifica a característica".

Após criar os grupos de características qualidade, conforme a Tabela 11, foi possível identificar que existem 105 classes sobre qualidade nos trabalhos, o que representa 53% das classes existentes. Ao analisar onde as classes aparecem, a categoria com classes de *Usability* (Usabilidade) está presente em 16 estudos primários (27%) e *Performance efficiency* (Eficiência de desempenho) aparece em 14 estudos primários (23%).

Tabela 11 - Análise de textos sobre o assunto de qualidade

Categoria relacionada a Requisito não funcional	Textos de Saída	Quantidade estudos (% dos estudos primários)
Usabilidade	<varia>, accessibility, advertisement, appearance complaint, audio, audio/video, comparative review, complaint ui generic review, customization, design, design specification, dynnamic content, editorial, error prevention, experience complaint, flexibility and efficiency, focus, form, gesture, gui, iconography, image, interaction complaint, language support, layout, learnability, legibility and color, links, motion, navigation, notification, operability, other, principle, recognition rather than recall, redundancy, structure, system visibility, tangible interaction, text, text equivalent, typography and font, ui aesthetic, ui element, usability, user control, user interface, video;	16 (27%)
Eficiência de desempenho	battery consumption, *feedback*, performance, performance efficiency, resource, resource heavy, resource usage, resource utilization, response time, speed, storage, time behavior, traffic wasting;	14 (23%)
Outro padrão	dependability, searchability, supportability, h1 - visibility, h10 - help, h2 - match, h3 - control, h4 - consistency, h5 - prevention, h6 - recognition, h7 - efficiency, h8 - aesthetic, h9 - recovery;	5 (8%)
Segurança	<varia>, permission, personal information, privacy, privacy and ethical issue, privacy control, security, security complaint, selling data, tracking;	8 (13%)
Confiabilidade	availability, crashing, fault tolerance, recoverability, reliability;	7 (12%)
Compatibilidade	co-existence, compatibility, compatibility issue, device compatibility, interoperability, specific android version, specific device or platform;	6 (10%)
Portabilidade	adaptability, installability, installation issue, portability, replaceability;	5 (8%)
Manutenibilidade	extensibility, maintainability;	2 (3%)
Adequação funcional	functional correctness, functional suitability;	2 (3%)
Não especifica a característica	non-functional, quality.	2 (3%)
Total		21 (35%)

O envio de *feedbacks* que contém características de qualidade de Usabilidade e Eficiência de desempenho provavelmente ocorre mais que as outras características pois esses atributos são mais fáceis de identificar pelo usuário, por exemplo, quando o aplicativo demora para carregar as informações, ele está com um desempenho ruim.

Oito trabalhos fizeram estudos específicos sobre requisitos não funcionais, usando a extração de requisitos do tipo características de qualidade:

- o trabalho de Groen et al. (2017) analisou os *feedbacks* de loja de aplicativos para entender se é uma fonte útil de requisitos não funcionais e apoiar a elicitação ou priorização de requisitos;
- o trabalho de Eler, Orlandini e Oliveira (2019), fez uma análise de *feedbacks* para entender o que os usuários de aplicativos Android falam sobre Acessibilidade - uma subcaracterística de Usabilidade;
- o trabalho de Jha e Mahmaoud (2019) fez um análise qualitativa dos tipos de requisitos não funcionais em *feedbacks* de usuários em diferentes categorias de aplicativos;
- o trabalho de Mukherjee e Ruhe (2020) estudou a Compatibilidade para o sistema operacional Android;
- o trabalho de Araújo et al. (2022) propôs uma técnica para extração de requisitos não funcionais de maneira automática;
- o trabalho de Diniz, De Souza Filho e Carvalho (2022) investigou a presença de problemas relacionados a heurísticas de Usabilidade;
- o trabalho de Reyes et al. (2022) foi sobre Acessibilidade e utilizou os aprendizados de outro trabalho (ELER; ORLANDIN; OLIVEIRA, 2019) para identificar os tipos de Acessibilidade nos *feedbacks* e evoluir diretrizes de acessibilidade da W3C *Accessibility Guidelines* (WCAG 2.1)[10]; e
- o trabalho de Nema et al. (2022) estudou preocupações que usuários de aplicativos móveis têm sobre Privacidade, um assunto relacionado ao atributo de Segurança.

[10] W3C *Accessibility Guidelines* (WCAG 2.1) - https://www.w3.org/TR/WCAG21/

Quatro estudos foram encontrados no grupo “Outro padrão”. Apesar de não estarem seguindo o padrão de qualidade de produto da ISO/IEC 25010 (2010), usam padrões relacionados a requisitos não funcionais. O trabalho de Abad et al. (2017) utiliza o termo *Searchability (*Pesquisabilidade) mas não explica qual a definição usada, se é relacionado a capacidade de pesquisado ou buscado relacionado a uma funcionalidade de busca ou capacidade de ser fácil de ser encontrado.

Já os trabalhos (HIDAYAT; ROCHIMAH, 2021; KUNAEFI; ARITSUGI, 2021) utilizam os termos *Dependability* (dependabilidade) e *Supportability* (suportabilidade), ambos usando como referência outro estudo primário (JHA; MAHMOUD, 2019). O estudo de Jha e Mahmoud (2019) classifica o *feedback* de usuário com dependabilidade quando contém preocupações sobre confiabilidade, disponibilidade e segurança; e suportabilidade quando o *feedback* tem assuntos sobre a compatibilidade, interoperabilidade, compatibilidade, adaptabilidade, portabilidade, instalabilidade, testabilidade e modificabilidade do aplicativo. Em ambos os usos o autor diz utilizar a classificação proposta pelo estudo primário de Kurtanović e Maalej (2016) mas que na prática tem diferenças, como por exemplo a ausência do termo dependabilidade.

Por fim, o trabalho de Diniz, De Souza Filho e Carvalho (2022) utiliza as 10 Heurísticas de Nielsen (1993) que estão relacionadas a usabilidade, uma característica que qualidade.

6.3.3.3 Requisitos Funcionais

Analisando se a saída do processo de extração identifica requisitos funcionais, pode-se ver que, dos 60 estudos primários, somente 6 estudos (ABAD *et al.*, 2017; KUNAEFI; ARITSUGI, 2021; LU; LIANG, 2017; PAGANO; MAALEJ, 2013; PANTHUM; SENIVONGSE, 2021; WANG *et al.*, 2018; WANG; ZHENG; LI, 2020; YIN; PFAHL, 2021) usam as classes: “*functional requirement”*, “*functional*” ou “*functionality*” como saídas.

O estudo de Abad et al. (2017) encontra tópicos com as palavras de exemplo: *make*, *deck*, *like*, *time*, *set;* e diz que essas palavras são “functional requirement”, mas não explica o porquê dessas e nem a definição usada para requisitos funcionais. O estudo de Lu e Liang (2017) é um trabalho que utiliza o termo “*functional requirement”* como saída de seu processo e descreve como “Coisas que um sistema/produto deve fazer”,

o que não é claro; além disso não cita uma fonte que define requisitos funcionais, como faz com os requisitos não funcionais – apenas diz que estudantes da área de Engenharia de Requisitos seguiram a norma de requisitos não funcionais. Diferente do trabalho anterior, o trabalho de Wang et al. (2018), não define o termo usado *"funcional requirement"*, mas quando usam caracteristicas de qualidade, os pesquisadores definem os nome usando os conceitos da ISO/IEC 25010

O trabalho de Wang, Zheng e Li (2020, p. 28, tradução nossa)[11] define requisitos funcionais como: "as coisas de software específicas para fazer podem ser descritas por um conjunto de requisitos, que consiste em funções e pontos de comportamento", um conceito dificil de entender. Ele utiliza essa definição para marcar manualmente os *feedbacks* no processo de extração gerando uma base que é usada para o proceso automático de extração de requisitos.

O trabalho de Kunaefi e Aritsugi (2021) descreve o requisito funcional como sendo "aspectos relacionados a *features* e funcionalidades específicas do aplicativo" (2021, p. 45085, tradução nossa) e explica que eles são "expressos explicitamente" (2021, p. 45084, tradução nossa). Ambas as explicações não apresentam referências que expliquem a definição.

O estudo de Yin e Pfahl (2021) utiliza a classe "*functional"* para identificar sentenças que contém uma *feature* e classificar essas sentenças vindas do *feedback* de usuário de acordo com os princípios do modelo de Kano (Kano, 1984). Este modelo é usado para relacionar a satisfação do usuário a *feature*. Segundo Yin e Pfahl (2021), o modelo Kano define a relação entre a satisfação do usuário e as *features* do produto.

O único estudo que cita a norma ISO/IEC 29148 (2018) é o de Panthum e Senivongse (2021), o qual tem o objetivo especifico de extrair comentários que "abordam funções que o aplicativo já executa ou deveria executar" (PANTHUM; SENIVONGSE, 2021, p. 15, tradução nossa) através de *boilerplates* de requisitos, o que segundo Dick, Hull e

[11] 'The things of software specific to do can be described by a set of requirements consisting of functions and behavior points."

Jackson (2019) são uma boa maneira de padronizar a linguagem usada para requisitos.

O trabalho de Jiang et al. (2014) usa o texto "*functionality"* para se referir a requisito funcional*,* mas não explica o que quer dizer ou qual a relação com o objetivo do trabalho de extrair "*evolutionary requirements"*.

Uma hipótese para os trabalhos não apresentarem definições precisas é o fato de o requisito funcional não ter padrões ou delimitação de funções para orientar o desenvolvimento, como existem para requisitos não funcionais através dos atributos de qualidade de produto na norma da ISO/IEC 25010 (2010).

6.3.3.4 Feature

O texto de saída com "*feature"* é muito utilizado entre os trabalhos. Ao analisar seu uso observa-se que o termo aparece estar sozinho (DI SORBO *et al.*, 2017; NAYEBI; CHO; RUHE, 2018) ou acompanhado de diferentes palavras como "*request*" - 27 trabalhos, vide Tabela 9, "*removal"* (MCILROY *et al.*, 2016; PANDEY; LITORIYA; PANDEY, 2019), "*praise"* (NOEI; ZHANG; ZOU, 2019), *"evaluation"* (SHAH; SIRTS; PFAHL, 2019) de *"Core < feature> of app"* (LICORISH; SAVARIMUTHU; KEERTIPATI, 2017) e *"information"* (PAGANO; MAALEJ, 2013).

A maioria dos trabalhos não explica o que é *feature* nem como o texto foi atribuido ao *feedback* de usuário e acaba usando o recurso de dar alguns exemplos de *review* de aplicativo que se encaixam nesse conteúdo. Esse tipo de prática dificulta a reprodução por outros trabalhos.

Outro problema é que o uso da palavra *Feature* se refere a requisitos funcionais e não funcionais de sistema (ISO/IEC/IEEE, 2017). No contexto de *feedbacks* de usuários, as *features* vem de usuários, então o detalhe do assunto ainda está em alto nível (requisitos de *stakeholder*).

Em resumo, as classes de saída do processo de extração de feedbacks não seguem padrões da área de ER e não usam tipicamente termos de Engenharia de Requisitos adequados, até mesmo os trabalhos que têm suas classes reutilizadas. A falta de padrão foi confirmada pela quantidade de muitas classes diferentes (198) e por trabalhos terem padrões próprios.

Mas pode-se observar que algumas classes, por aparecerem em mais da metade dos trabalhos, poderiam se transformar em um padrão, como classes relacionadas a gestão de mudança, em que é destacada a separação entre pedido de *feature* em 53% dos trabalhos e correção de bugs em 50% dos trabalhos. Além dessas classes, se destaca a presença de classes relacionadas a requisitos não funcionais que aparecem 35% dos estudos primários.

6.4 QP3: Qual a avaliação de saída dos *feedbacks* explícitos no processo de extração de requisitos?

A Figura 15 apresenta as avaliações feitas pelos autores dos estudos. Observa-se que a maioria - 51 estudos primários (83%) - não tiveram uma avaliação de seu resultado. Dos 9 estudos (15%) que tiveram uma avaliação da extração, em 7 trabalhos (ASSI *et al.*, 2021; CARRENO; WINBLADH, 2013; DI SORBO *et al.*, 2017; GU; KIM, 2015; NAKAMURA *et al.*, 2021; PALOMBA *et al.*, 2017) foram pesquisas com questionários com perguntas fechadas e em 3 trabalhos (MAALEJ *et al.*, 2016; VILLARROEL *et al.*, 2016; YIN; PFAHL, 2021) foram entrevistas. Em questionários, o processo pode ter vieses no entendimento do resultado, assim como no entendimento das perguntas. Já em entrevistas é possível uma melhor compreensão da avaliação por haver oportunidade de confirmação de entendimento entre o entrevistado e o entrevistador.

Dos estudos que fazem avaliações da saída, os trabalhos realizaram perguntas gerais sobre:

- a utilidade das informações geradas por seus estudos (CARRENO; WINBLADH, 2013; DI SORBO *et al.*, 2017; GU; KIM, 2015; MAALEJ *et al.*, 2016; NAKAMURA *et al.*, 2021; VILLARROEL *et al.*, 2016; YIN; PFAHL, 2021);
- o benefício de economia de tempo (DI SORBO *et al.*, 2017; NAKAMURA *et al.*, 2021; YIN; PFAHL, 2021);
- a economia de dinheiro (YIN; PFAHL, 2021);
- a redução de esforço para extrair requisitos (NAKAMURA *et al.*, 2021; VILLARROEL *et al.*, 2016); e

- desenvolver melhor o software (MAALEJ *et al.*, 2016; VILLARROEL *et al.*, 2016; YIN; PFAHL, 2021).

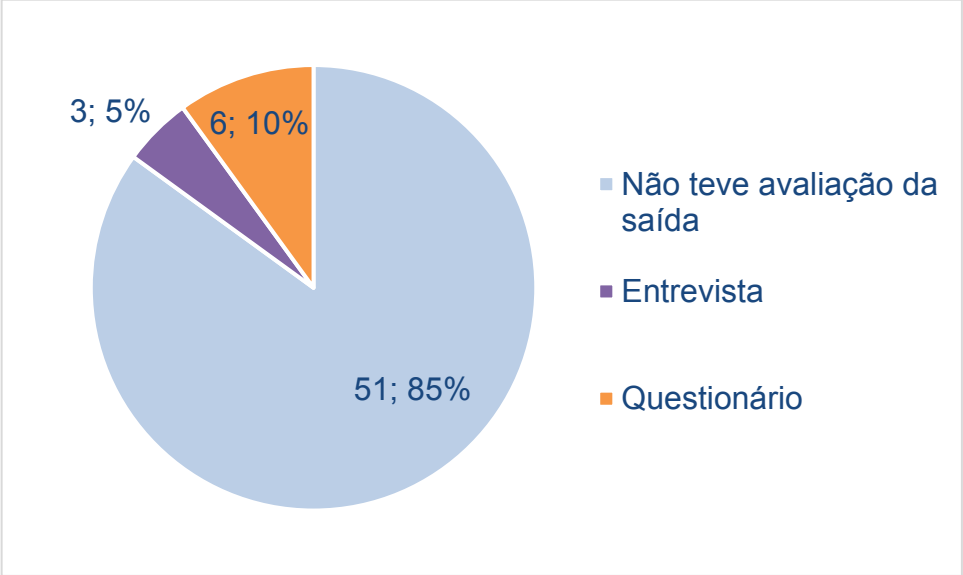

Figura 15 - Avaliação do resultado dos estudos primários

Além disso, alguns trabalhos realizam perguntas especificas sobre o uso e o resultado dos seus trabalhos, sobre utilizar a visualização de opinião de usuários para entender as preferências dos usuários (GU; KIM, 2015), identificar os componentes do código-fonte que precisam ser alterados conforme o resultado de extração de "*feature request*" e "*bug report*" (PALOMBA *et al.*, 2017); promover a análise dos *feedbacks*. (VILLARROEL *et al.*, 2016); e usar a extração de requisitos para a análise competitiva de funcionalidades de aplicativos (ASSI *et al.*, 2021).

Em trabalhos que fizeram perguntas mais especificas sobre o conteúdo gerado (MAALEJ *et al.*, 2016; VILLARROEL *et al.*, 2016), destaca-se a necessidade do conteúdo gerado trazer mais contexto a qual *feedback* do usuário se refere através do nome do aplicativo, versão do aplicativo e, indo além das soluções vistas, integração de dados com *feedbacks* implícitos com logs de falhas no aplicativo.

A falta de contexto também impacta tanto na análise dos comentários para então pensarem em requisitos (NAKAMURA *et al.*, 2021) quanto na confiança do resultado gerado a ponto de tirar o especialista ER do processo (YIN; PFAHL, 2021).

Por fim, seria interessante que a avaliação do resultado de saída seja padronizada para que seja possível comparar os resultados entre estudos, e leve em consideração

características individuais de requisitos (ISO/IEC/IEEE, 2018) que podem ser usados para validação de requisitos para poder gerar mais confiança no resultado do processo.

6.5 Fontes do *Feedback* de usuário

Levantou-se como informação complementar a fonte do *feeback* do usuário, ou seja, onde o usuário deixou o *feedback* sobre o aplicativo. Esses dados podem ser vistos na Tabela 12.

Tabela 12 - Fontes de *feedback* de aplicativo móvel mais usadas

Origem do *Feedback*	Quantidade trabalhos que aparecem **(% dos trabalhos)**
Google Play Store	43 (72%)
Apple App Store	29 (48%)
Twitter	5 (8%)
Não informa	2 (3%)
Google +	1 (2%)
Windows Phone Store	1 (2%)
BlackBerry App Store	1 (2%)
GitHub	1 (2%)
360 mobile assistant	1 (2%)
F-droid	1 (2%)

A partir da tabela é possível ver que a Google Play é a mais usada pelos trabalhos com 42 (72%) estudos, depois a Apple App Store com 29 (48%) dos estudos, as outras fontes de *feedback* estão menos presentes.

Apesar da Google Play e Apple App Store serem fontes de *feedback* muito usadas não quer dizer que cada trabalho extraiu o *feedback* de cada fonte. Foi levantado que em 48% dos trabalhos o próprio trabalho extraiu os dados da fonte e em 33% dos trabalhos os dados de outro trabalho foram utilizados, ou seja, o estudo não extraiu e sim usou uma base de dados já pronta.

6.6 Considerações do capítulo

Nesse capítulo foram apresentados os resultados da revisão sistemática da literatura assim como a discussão de seus resultados. A seguir, serão apresentadas as ameaças à validade da RSL.

7 AMEAÇAS À VALIDADE

Esta Revisão Sistemática da Literatura possui algumas ameaças à validade, as quais foram avaliadas utilizando o trabalho de Ampatzoglou et al. (2020), que discute as ameaças à validade da seleção dos estudos, à validade de dados e à validade de pesquisa e aponta ações de mitigação para cada uma.

Em relação à validade da seleção dos estudos selecionados na RSL, não foi feito o *snowballing* no processo de seleção de estudos. Isso pode ter reduzido o número de estudos relevantes incluídos. Para mitigar essa ameaça foram escolhidas bases digitais de artigos de relevância segundo o guia de RSL (KITCHENHAM; CHARTERS, 2007). Também a *string* de busca construída foi revisada pelo orientador do trabalho e foi avaliada em pesquisas piloto. Além disso, foram usadas ferramentas para facilitar o processo da RSL e os resultados da pesquisa foram validados e documentados.

Uma outra ameaça é que na Fase 3 do processo de seleção, 47 estudos não estavam disponíveis para baixar. Essa quantidade de estudos, mesmo que parte não passaria pelos critérios de aceite após leitura, representa 24% dos estudos que passaram pela Fase 3, sendo assim um número relevante.

Os critérios de inclusão e exclusão foram avaliados pelo orientador deste trabalho para mitigar possíveis problemas em sua aplicação. Por esse mesmo motivo, foi criado um processo com fases que explicam a decisão para inclusão e exclusão de estudos. Além disso, a decisão de incluir os tipos de literatura foi tomada de forma a cumprir o objetivo do estudo e a disponibilidade de trabalhos sobre o assunto.

Em relação à validade de dados, a quantidade de estudo primários selecionados pode ser justificada devido a possibilidade de tirar conclusões a partir das tendências dos dados. A escolha dos campos de extração foi discutida entre os autores, para garantir que as questões de pesquisa pudessem ser respondidas.

A validade dos dados extraídos não pode ser garantida pois o processo de pesquisa não foi concentrado em apenas locais de qualidade, não foi feita uma avaliação de qualidade no processo de seleção e nem uma avaliação de impacto dos estudos primários usando estatísticas.

Para mitigar vieses de extração e intepretação de resultados, foi executada uma triagem dos artigos para verificar a extração e foram executadas análises e interpretação dos dados.

Para sintetizar os resultados, foram criadas classificações para os estudos primários sem seguir esquemas de classificações existentes. Assim, para migar vieses de extração, o processo de criação dessas classificações foi documentado.

Quanto à mitigação da validade de pesquisa, foi feita uma análise de objetivo e propósito do método foi feita e depois a decisão de usar a RSL foi tomada após discussão com o orientador, o autor desse estudo e participantes a RSL que originou esse trabalho. Além disso, para que os resultados deste trabalho sejam generalizáveis, seus resultados foram validados com estudos existentes e mostram-se alinhados.

Por fim, para que o processo fosse confiável, os dados foram disponibilizados de forma pública [12] para consulta e todas suas as atividades foram documentadas em um protocolo de revisão sistemática da literatura seguindo boas práticas de pesquisa.

7.1 Considerações do capítulo

Nesse capítulo foram apresentados os resultados da análise e discussão sobre de dados de formato e classes de saída, avaliação da saída dos estudos primários. Foram apresentadas também as ameaças à validade da RSL. No próximo capítulo são apresentados os trabalhos relacionados a esse.

[12] Base de dados da revisão sistemática da literatura - https://doi.org/10.5281/zenodo.7740478

8 TRABALHOS RELACIONADOS

Foram encontrados 9 estudos relacionados ao tema deste trabalho, como mostra a Tabela 13. Esses estudos abordam o processo de extração de requisitos sob diferentes perspectivas mudando o escopo de análise em três variáveis:

- Fonte de dados feedback *crowd* - que podem estar associadas ao canal de comunicação com a *crowd ou ao* tipo de software que a *crowd* utiliza. Exemplo: Lojas de aplicativos móveis;
- Formas de extração de dados - que podem ser manuais ou automatizadas. Se automatizada, usar técnicas de diferentes áreas de estudo como *Data Mining*, *Machine Learning* (ML) e NLP;
- Aplicação do processo - que pode ser de escopo mais amplo aplicado na Engenharia de Software (ES), pode ser aplicado na ER (dentro da ES) ou em atividades dentro de processos da ER, por exemplo a elicitação de requisitos.

Uma árvore para ilustrar as diferentes perspectivas do processo de extração de requisitos foi feita, conforme Figura 16.

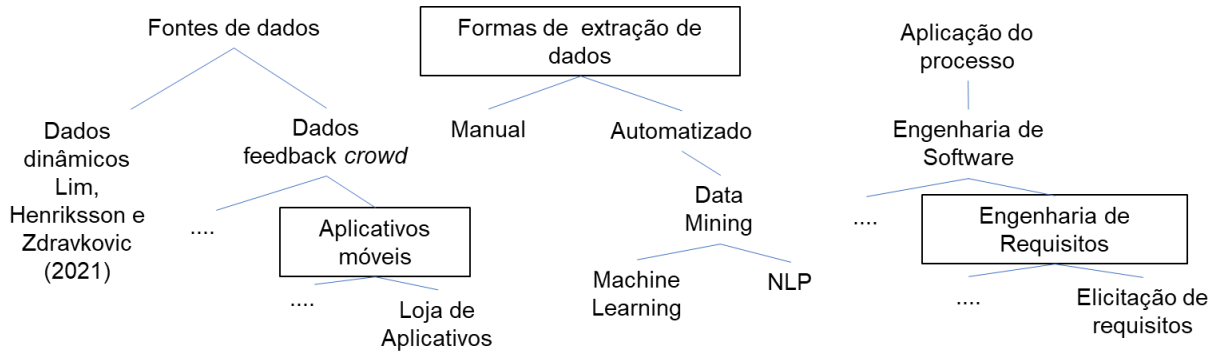

Figura 16 - Variáveis de escopo de análise do processo de extração de requisitos no estudo

Dos 9 trabalhos relacionados, 3 tiveram um foco em dados de lojas de aplicativo como fonte de dados de *feedback crowd* e trataram o assunto de Engenharia de Requisitos e não delimita a forma de extração de requisitos. A revisão de Genc-Nayebi e Abran (2016) apresentou os desafios quanto às técnicas de mineração de dados utilizadas e a dependência de domínio dos dados, o que faz um comentário de usuário de app ser útil, métodos de detecção de comentários *spam* e análise inicial do conteúdo dos comentários. Já Tavakoli et al. (2018) teve o foco nas técnicas mineração de dados e

ferramentas de software criadas para suportar as técnicas e quais os principais assuntos nos comentários com foco em desenvolvimento de software usados.

Ainda com foco em dados de lojas de aplicativo e Engenharia de Requisitos extração de requisitos de forma mais genérica, a RSL de Dabrowski et al. (2022) criou categorias de tipos de análise de dados feitas nos comentários de usuário - classificação, agrupamento, extração de informações, busca e recuperação de informações, análise de sentimentos, recomendação e sumarização. Para cada tipo de análise avaliou: quais técnicas de mineração foram usadas; para quais atividades de ES os trabalhos dizem que a análise suporta e quão bem suporta; se foram feitas avaliações empíricas nos estudo e qual seu resultado.

Duas revisões já se aprofundaram mais nos conceitos de *crowd*, abrindo o leque para todos os tipos de fonte de *feedback crowd* e para as formas de extração, mas com foco em aplicação na Engenharia de Requisitos. O trabalho de Wang et al. (2019), que teve o objetivo de estudar o uso dos dados *feedbacks* de *crowd* para a Engenharia de Requisitos e focou em mapear quais os dados e as origens de informações, as atividades de ER auxiliadas e como está a área de pesquisa. De forma similar, Khan et al. (2019) fez um estudo para entender como as atividades de ER podem ser melhoradas pelo *crowd* e propôs um mapa de pesquisa que sugere papéis futuros na área de ER para *crowd.*

Outras duas RSL tem um foco maior em explorar a extração de requisitos, o trabalho de Zhao et al. (2022)teve um objetivo em entender o estado da arte da aplicação da NLP na ER (NLP4RE), quais as tecnologias de NLP utilizadas e em quais atividades de ER. Enquanto o estudo relacionado anterior tem um foco na área de NLP, Zamani, Zowghi e Arora (2021) tiveram um foco na área de ML para o uso em ER, o qual avalia quais as técnicas, dados, a métricas de avaliação da área de ML utilizados pelos estudos assim como os desafios do uso em ER.

Um trabalho (LIM; HENRIKSSON; ZDRAVKOVIC, 2021) focou mais aplicação da extração automatizada de dados para atividade de elicitação de requisitos de forma não se limitando a fonte da dados de feedbacks *crowd*, e sim aos dados dinâmicos, segundo os autores são dados gerados por humanos, por processos e eventos de negócio e dados gerados por máquina. Esse trabalho olhou os tipos de dados,

tecnologias usadas para automatizar a extração e os resultados gerados para a elicitação de requisitos.

E por fim, o trabalho de Santos, Groen e Villela (2019) teve foco em dados de *feedback* do usuário *crowd*, mas não especifica se o escopo do estudo foi de aplicativos móveis, além disso analisa especificamente a extração com tecnologias de NLP e apesar de ampla a *string* de busca dentro de ER, pode-se dizer que tem aplicação especifica na atividade de elicitação de requisitos pois faz parte de seu objetivo.

Esta pesquisa se diferencia das demais ao focar na aplicação do processo para Engenharia de Requisitos com fontes de dados de *feedback crowd* de aplicativos móveis independente da forma de extração de dados, conforme destacado na árvore de variáveis da Figura 16.

Tabela 13 - Mapeamento de respostas as questões de pesquisa

	Questões de Pesquisa		
Trabalho	QP1	QP2	QP3
Este estudo	**Responde**	**Responde**	**Responde**
(GENC-NAYEBI; ABRAN, 2017)	Não responde	Responde parcialmente	Não responde
(TAVAKOLI et al., 2018)	Não responde	Responde parcialmente	Não responde
(WANG et al., 2019)	Não responde	Não responde	Não responde
(KHAN et al., 2019)	Não responde	Não responde	Não responde
(SANTOS; GROEN; VILLELA, 2019)	Não responde	Responde parcialmente	Não responde
(ZHAO *et al.*, 2022)	Não responde	Não responde	Não responde
(LIM; HENRIKSSON; ZDRAVKOVIC, 2021)	Não responde	Não responde	Não responde
(ZAMANI; ZOWGHI; ARORA, 2021)	Não responde	Não responde	Não responde
(DĄBROWSKI et al., 2022)	Não responde	Não responde	Responde parcialmente

Além de não ter trabalhos no mesmo escopo de análise do processo de extração de requisitos, dos trabalhos relacionados, nenhum deles respondeu a todas as questões de pesquisa propostas por este trabalho. Como mostra a Tabela 13, nenhum dos trabalhos relacionados a esta pesquisa fala sobre o formato da saída do processo de

extração de requisitos (QP1), três trabalhos respondem parcialmente a (QP2) e um trabalho responde parcialmente a (QP3).

O detalhe do conteúdo de saída do processo de extração de requisitos (QP2) é abordado de diferentes maneiras pelos estudos relacionados. Genc-Nayebi e Abran (2017), tem um foco em utilidade dos comentários de aplicativo para os próprios usuários do app e por isso quando se falar do ponto de vista para a engenharia de requisitos só compila de forma geral as classificações feitas pelos trabalhos e fala que o desenvolvedor de software tem um ponto de vista diferente do usuário de app quando usa os comentários de loja de aplicativos para extrair informações de usabilidade e experiência do usuário; elicitar requisitos ausentes e definir as *features* solicitadas; e melhorar a qualidade do software. Já o trabalho de Tavakoli et al. (2018), trata a saída do processo de forma mais geral agrupando os assuntos mais usados nos trabalhos pois tem um foco maior nas técnicas e ferramentas usadas no processo de extração. E o trabalho de Santos, Groen e Vilella (2019) teve o objetivo mais relacionado a QP2 em que identifica as categorias de conteúdo relevantes extraídas do *feedback* de usuários e propõe uma taxonomia inicial agrupando os principais assuntos, mas é diferente desse estudo pois se limita a técnicas de NLP e a atividade de elicitação de requisitos, além de não discutir aplicação dessas taxonomias no processo de ER.

Por fim com relação a (QP3), a maioria dos trabalhos relacionados (DĄBROWSKI *et al.*, 2022; GENC-NAYEBI; ABRAN, 2017; LIM; HENRIKSSON; ZDRAVKOVIC, 2021; ZAMANI; ZOWGHI; ARORA, 2021; ZHAO *et al.*, 2022) usou o termo avaliação para se referir a avaliação das técnicas de mineração de dados utilizadas pelos trabalhos e não quanto a adequação das necessidades do engenheiro de requisitos, o qual é o foco deste trabalho. E somente o trabalho de Dabrowski et al. (2022) se assemelha a esse trabalho pois levanta a quantidade de trabalhos que faz a avaliação da qualidade percebida pelos usuários do processo de extração. O trabalho criou categorias para agrupar as avaliações (utilidade, precisão do resultado, usabilidade do processo, eficiência para extrair com base no esforço humano e de ser informativo e instrutivo para o usuário) mas não explicou o processo de categorização dos trabalhos além do agrupamento de resultados ser específicos para as categorias de tipos de análise - classificação, agrupamento, extração de informações, busca e recuperação de

informações, análise de sentimentos, recomendação e sumarização. Ele também analisa estatisticamente a quantidade de participantes no estudo e seu tipo de participante e o setor que atua.

8.1 Considerações do capítulo

Nesse capítulo foram apresentados os escopos de análise do processo de extração de requisitos conforme as fontes de dados *feedback crowd,* formas de extração de dados e aplicação do processo dos estudos relacionados; o escopo de análise desse trabalho; a relação das perguntas de pesquisa deste trabalho com os estudos relacionados e suas principais diferenças.

9 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma análise da adequação para a Engenharia de Requisitos do requisito extraído automaticamente através de *feedbacks* explícitos de usuários de aplicativos móveis. Essa análise consistiu em avaliar o formato e o conteúdo de saída do processo de extração de trabalhos de pesquisa publicados nos últimos dez anos.

Com relação ao formato da saída, a maioria dos trabalhos não se preocupa em criar um formato próximo a requisito e sim em reduzir o tamanho do texto. Porém, isso ao invés de reduzir o tempo de entendimento da informação, causa problemas para o entendimento do texto.

Ambos os trabalhos que utilizam “padrões de texto” se aproximam mais de uma especificação de software do que os outros trabalhos, pois começam a criar um formato de saída independente da entrada. Porém, em ambos os padrões ou há falta de referência a ER e de explicação clara de como ele é formado; ou, quando há referência, ela não segue as boas práticas tais como o padrão de sintaxe de requisito funcional da ISO/IEC/IEEE 29148 (2018).

Com relação ao conteúdo que é extraído, os trabalhos buscam separar os defeitos de requisitos novos: 53% dos trabalhos extraem “pedidos de *features*” dos *feedbacks* e 50% extraem “*bug requests*”. Para que o uso dessas informações ganhe mais utilidade para a ER, esses novos pedidos de *feature* poderiam ser mais detalhados.

35% dos trabalhos identificam requisito não funcional, em diferentes níveis de detalhe no *feedback,* mas não necessariamente os trabalhos que identificam requisitos não funcionais também identificam pedidos de *feature*.

Apesar da grande presença de “*feature”* nos estudos, esse termo se refere a dois tipos de requisitos de software (funcional e não funcional), portanto é necessário o trabalho de refinamento pelo engenheiro de requisitos para chegar em funções ou atributos de qualidade do software. Um outro problema é que muitos trabalhos não seguem uma definição do que é *feature* para a ER e também não separam os requisitos de *stakeholder*, requisitos de sistema e requisitos de software abrindo margem para dúvidas na interpretação do resultado de extração.

Para o uso do requisito ser adequado para a ER é importante que os especialistas de ER estejam envolvidos no processo de definição do *KDD*. Nesse processo é importante separar defeitos do software de novos requisitos. Dentro dessa separação seria importante definir se as informações contidas no comentário já podem ser consideradas requisitos de *stakeholder* ou se ainda são somente necessidades e objetivos que precisam ser refinados.

Por fim, apenas 15% dos trabalhos têm uma avaliação da saída por especialista de requisitos ou desenvolvedor de software. Mais que isso, a avaliação se mostrou muito específica para o processo criado por cada estudo. Além disso, apesar das respostas serem positivas sobre a utilidade dos processos, falta contexto para os respondentes opinarem, o que pode ter gerado um resultado positivo enviesado.

Os resultados da revisão ainda confirmam a afirmação de Zhao et al. (2022) sobre a necessidade de se ter mais estudos de caso ou aplicações das técnicas propostas de forma a avaliar o seu uso dentro de contextos reais com times de desenvolvimento do próprio aplicativo.

Como trabalhos futuros, são necessários estudos de técnicas de NLP e *data mining* que possam ajudar na separação da informação contida no *feedback* entre requisito e necessidade ou objetivo do usuário. Também podem ser feitos estudos de quais *templates* poderiam ser mais utilizados dependendo da informação contida no *feedback*. Outra ideia é realizar estudos de técnicas de NLP e *data mining* que permitam evoluir o formato de saída com uma estrutura bem definida pela ER. Pode ser estudado como aproximar a atividade de validação de requisitos com a saída do processo de extração. Por fim, pode ser feito um estudo para determinar quais recursos específicos, como dicionários e conjuntos de textos, no domínio da Engenharia de Requisitos poderiam ser criados para uso da NLP4RE.

REFERÊNCIAS

ABAD, Z. S. H.; SIMS, S. D. V.; CHEEMA, A.; NASIR, M. B.; HARISINGHANI, P. Learn More, Pay Less! Lessons Learned from Applying the Wizard-of-Oz Technique for Exploring Mobile App Requirements. In: INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE WORKSHOPS (REW), 25. 2017, **Proceedings**[...] IEEE, 2017. p. 132–138.

ACTIVATE CONSULTING. **Activate Tech & Media Outlook 2022**. Disponível em: <https://activate.com/outlook/2022/>. Acesso em: 8 mar. 2022.

AL-HAWARI, A.; NAJADAT, H.; SHATNAWI, R. Classification of application reviews into software maintenance tasks using data mining techniques. **Software Quality Journal**, v. 29, n. 3, p. 667–703, 2021.

ALI, M.; JOORABCHI, M. E.; MESBAH, A. Same App, Different App Stores: A Comparative Study. In: INTERNATIONAL CONFERENCE ON MOBILE SOFTWARE ENGINEERING AND SYSTEMS, 4. 2017, **Proceedings**[...]. IEEE. p. 79–90, 2017.

AMPATZOGLOU, A.; BIBI, S.; AVGERIOU, P.; CHATZIGEORGIOU, A. Guidelines for Managing Threats to Validity of Secondary Studies in Software Engineering. **Contemporary Empirical Methods in Software Engineering**, n. August, 2020.

APP ANNIE. **Reviews Report**. Disponível em: <https://helpcenter.data.ai/community/s/article/Reviews-Report-Guide>. Acesso em: 5 mar. 2022.

ARAUJO, A. F.; GÔLO, M. P. S. S.; MARCACINI, R. M. Opinion mining for app reviews: an analysis of textual representation and predictive models. **Automated Software Engineering**, v. 29, n. 1, p. 5, 6 maio 2022. Disponível em: <https://doi.org/10.1007/s10515-021-00301-1>.

ASSI, M.; HASSAN, S.; TIAN, Y.; ZOU, Y. FeatCompare: Feature comparison for competing mobile apps leveraging user reviews. **Empirical Software Engineering**, v. 26, n. 5, 2021.

BENNACEUR, A.; TUN, T. T.; YU, Y.; NUSEIBEH, B. Requirements Engineering. In: **Handbook of Software Engineering,**. [s.l: s.n.]p. 2–45.

BOURQUE, P.; FAIRLEY, R. E.; EDS. **Guide to the Software Engineering Body of Knowledge,Version 3.0**. [s.l.] IEEE Computer Society, 2014. 335 p.

BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. **Journal of Systems and Software**, v. 80, n. 4, p. 571–583, 2007. Disponível em: <http://dx.doi.org/10.1016/j.jss.2006.07.009>.

CARRENO, L. V. G. G.; WINBLADH, K. Analysis of user comments: An approach for software requirements evolution. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2013, **Proceedings**[...]. 2013. p. 582–591.

CHEN, N.; LIN, J.; HOI, S. C. H. H.; XIAO, X.; ZHANG, B. AR-miner: mining informative reviews for developers from mobile app marketplace. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 36. 2014, New York, NY, USA. **Proceedings**[...]. New York, NY, USA: ACM, 2014. p. 767–778.

CHEN, Q.; CHEN, C.; HASSAN, S.; XING, Z.; XIA, X.; HASSAN, A. E. How Should I Improve the UI of My App? **ACM Transactions on Software Engineering and Methodology**, v. 30, n. 3, p. 1–38, 31 jul. 2021. Disponível em: <https://doi.org/10.1145/3447808>.

COHN, M. **User stories applied: For Agile Software Development**. [s.l.] Addison-Wesley Professional, 2004.

DĄBROWSKI, J.; LETIER, E.; PERINI, A.; SUSI, A. Mining User Opinions to Support Requirement Engineering: An Empirical Study. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [s.l: s.n.]p. 401–416.

DĄBROWSKI, J.; LETIER, E.; PERINI, A.; SUSI, A. Analysing app reviews for software engineering: a systematic literature review. **Empirical Software Engineering**, v. 27, n. 2, 2022.

DALPIAZ, F. On the Value of CrowdRE in Research and Practice. In: INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE WORKSHOPS (REW), 29., 2021, **Proceedings**[...]. IEEE, 2021. v. 2021- Septe, p. 291–292.

DALPIAZ, F.; FERRARI, A.; FRANCH, X.; PALOMARES, C. Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. **IEEE Software**, v. 35, n. 5, p. 115–119, set. 2018. Disponível em: <https://ieeexplore.ieee.org/document/8474507/>.

DI SORBO, A.; PANICHELLA, S.; ALEXANDRU, C. V.; VISAGGIO, C. A.; CANFORA, G. SURF: Summarizer of User Reviews Feedback. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING COMPANION (ICSE-C), 39., 2017, **Proceedings**[...]. IEEE, 2017. p. 55–58.

Dicio-Dicionário Online de Português. Disponível em: <https://www.dicio.com.br/>. Acesso em: 19 mar. 2023.

DICK, J.; HULL, E.; JACKSON, K. **Requirements Engineering**. Cham: Springer International Publishing, 2017.

DINIZ, L. D. N.; DE SOUZA FILHO, J. C.; CARVALHO, R. M. Can User Reviews Indicate Usability Heuristic Issues? In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS EXTENDED ABSTRACTS, 2022, New York, NY, USA. **Proceedings**[...]. New York, NY, USA: ACM, 2022. p. 1–6.

ELER, M. M.; ORLANDIN, L.; OLIVEIRA, A. D. A. Do Android app users care about accessibility? In: BRAZILIAN SYMPOSIUM ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2019, New York, NY, USA. **Proceedings**[...]. New York, NY, USA: ACM, 2019. p. 1–11.

GENC-NAYEBI, N.; ABRAN, A. A systematic literature review: Opinion mining studies from mobile app store user reviews. **Journal of Systems and Software**, v. 125, p. 207–219, mar. 2017. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85003875563&doi=10.1016%2Fj.jss.2016.11.027&partnerID=40&md5=320e0bd738a694513c6444018a983816>.

GLINZ, M. On Non-Functional Requirements. In: IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE 2007), 15., 2007, **Proceedings**[...]. IEEE, 2007. p. 21–26.

GROEN, E. C.; KOPCZYNSKA, S.; HAUER, M. P.; KRAFFT, T. D.; DOERR, J. Users - The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews. In: Proceedings INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE, RE,25. 2017, 2017a, **Proceedings**[...]. 2017. p. 80–89.

GROEN, E. C.; SEYFF, N.; ALI, R.; DALPIAZ, F.; DOERR, J.; GUZMAN, E.; HOSSEINI, M.; MARCO, J.; ORIOL, M.; PERINI, A.; STADE, M. The Crowd in Requirements Engineering: The Landscape and Challenges. **IEEE Software**, v. 34, n. 2, p. 44–52, mar. 2017b. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85017173259&doi=10.1109%2FMS.2017.33&partnerID=40&md5=b02967d5a72c38a9f700fbcb0616e6ae>.

GU, X.; KIM, S. “What Parts of Your Apps are Loved by Users?” (T). In: IEEE/ACM INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING (ASE).30, 2015, **Proceedings**[...]. IEEE, 2015. p. 760–770.

GUO, H.; SINGH, M. P. Caspar. In: ACM/IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 42. 2020, New York, NY, USA. **Proceedings**[...]. New York, NY, USA: ACM, 2020. p. 628–640.

GUZMAN, E.; IBRAHIM, M.; GLINZ, M. A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), 3., 2017, **Proceedings**[...]. IEEE, 2017. p. 11–20.

GUZMAN, E.; MAALEJ, W. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In: INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE), 22., 2014, **Proceedings**[...]. IEEE, 2014. p. 153–162.

GUZMAN, E.; OLIVEIRA, L.; STEINER, Y.; WAGNER, L. C.; GLINZ, M. User feedback in the app store. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING IN SOCIETY, 40., 2018, New York, NY, USA. **Proceedings**[...]. New York, NY, USA: ACM, 2018. p. 13–22.

HARMAN, M.; JIA, Y.; ZHANG, Y. App store mining and analysis: MSR for app stores. In: WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), 9., 2012, **Proceedings**[...]. IEEE, 2012. p. 108–111.

HIDAYAT, T.; ROCHIMAH, S. NFR Classification using Keyword Extraction and CNN on App Reviews. In: INTERNATIONAL SEMINAR ON RESEARCH OF INFORMATION TECHNOLOGY AND INTELLIGENT SYSTEMS (ISRITI), 4., 2021, **Proceedings**[...]. 2021. p. 211–216.

HOWE, J. The Rise of Crowdsourcing. **Wired Magazine**, n. 14.06, p. 1–5, 2006. Disponível em: <https://www.wired.com/2006/06/crowds/>.

INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS (IIBA). **BABOK - A a guide to the business analysis body of knowledge**.. Toronto, Ontario, Canada: International Institute of Business Analysis, 2015. 514 p.

ISO/IEC/IEEE. Systems and software engineering — Vocabulary 24765. p. 536, 2017. Disponível em: <https://ieeexplore.ieee.org/document/8016712>.

ISO/IEC/IEEE. Systems and software engineering — Life cycle processes — Requirements engineering 29148. v. 2018, 2018.

ISO/IEC. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models 25010. v. 2010, p. 44, 2010. Disponível em: <papers2://publication/uuid/4E7AB6E6-6F61-4D11-A7BA-9644832A7308>.

JHA, N.; MAHMOUD, A. Mining non-functional requirements from App store reviews. **Empirical Software Engineering**, v. 24, n. 6, p. 3659–3695, 7 dez. 2019. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067234356&doi=10.1007%2Fs10664-019-09716-7&partnerID=40&md5=50723ae4ceae7c455ad0d6460a088452>.

JIANG, W.; RUAN, H.; ZHANG, L.; LEW, P.; JIANG, J. For user-driven software evolution: Requirements elicitation derived from mining online reviews. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 8444 LNAI, n. PART 2, p. 584–595, 2014.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. Disponível em: <http://web.stanford.edu/~jurafsky/slp3/>. Acesso em: 6 mar. 2022.

KANO, N. Attractive quality and must-be quality. **Hinshitsu (Quality, the Journal of Japanese Society for Quality Control)**. 14, 39–48, 1984.

KHALID, H.; SHIHAB, E.; NAGAPPAN, M.; HASSAN, A. E. What Do Mobile App Users Complain About? **IEEE Software**, v. 32, n. 3, p. 70–77, maio 2015. Disponível em: <http://ieeexplore.ieee.org/document/6762802/>.

KHAN, J. A.; LIU, L.; WEN, L.; ALI, R. Crowd Intelligence in Requirements Engineering: Current Status and Future Directions. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [s.l: s.n.]p. 245–261.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. In: **EBSE Technical Report**. New York, NY, USA: ACM, 2007.

KUNAEFI, A.; ARITSUGI, M. Extracting Arguments Based on User Decisions in App Reviews. **IEEE Access**, v. 9, p. 45078–45094, 2021. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85103757570&doi=10.1109%2FACCESS.2021.3067000&partnerID=40&md5=66f969cacfcc5d6160440e90ed519df4>.

LIBERATI, A.; ALTMAN, D. G.; TETZLAFF, J.; MULROW, C.; GOTZSCHE, P. C.; IOANNIDIS, J. P. A.; CLARKE, M.; DEVEREAUX, P. J.; KLEIJNEN, J.; MOHER, D. The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate healthcare interventions: explanation and elaboration. **BMJ**, v. 339, n. jul21 1, p. b2700–b2700, 4 dez. 2009. Disponível em: <https://www.bmj.com/lookup/doi/10.1136/bmj.b2700>.

LICORISH, S. A.; SAVARIMUTHU, B. T. R.; KEERTIPATI, S. Attributes that Predict which Features to Fix. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 21., 2017, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2017. v. Part F1286, p. 108–117.

LIM, S.; HENRIKSSON, A.; ZDRAVKOVIC, J. Data-Driven Requirements Elicitation: A Systematic Literature Review. **SN Computer Science**, v. 2, n. 1, p. 1–35, 2021. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85107230737&doi=10.1007%2Fs42979-020-00416-4&partnerID=40&md5=0ba20bb6e3cd4a891c83cd0c13f88674>.

LU, M.; LIANG, P. Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 21., 2017, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2017. v. Part F1286, p. 344–353.

LUCASSEN, G.; DALPIAZ, F.; WERF, J. M. E. M. van der; BRINKKEMPER, S. The Use and Effectiveness of User Stories in Practice. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [s.l: s.n.]p. 205–222.

MAALEJ, W.; KURTANOVIĆ, Z.; NABIL, H.; STANIK, C. On the automatic classification of app reviews. **Requirements Engineering**, v. 21, n. 3, p. 311–331, 14 set. 2016. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84968538213&doi=10.1007%2Fs00766-016-0251-9&partnerID=40&md5=f44f36d563f85f8592ade371cd02ec42>.

MALGAONKAR, S.; LICORISH, S. A.; SAVARIMUTHU, B. T. R. Automatically generating taxonomy for grouping app reviews — a study of three apps. **Software Quality Journal**, v. 30, n. 2, p. 483–512, 2022. Disponível em: <https://doi.org/10.1007/s11219-021-09570-1>.

MAO, K.; CAPRA, L.; HARMAN, M.; JIA, Y. A survey of the use of crowdsourcing in software engineering. **Journal of Systems and Software**, v. 126, p. 57–84, 2017.

MCILROY, S.; ALI, N.; KHALID, H.; E. HASSAN, A. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. **Empirical Software Engineering**, v. 21, n. 3, p. 1067–1106, 24 jun. 2016.

MUKHERJEE, D.; RUHE, G. Analysis of Compatibility in Open Source Android Mobile Apps. In: IEEE INTERNATIONAL WORKSHOP ON ARTIFICIAL INTELLIGENCE FOR REQUIREMENTS (AIRE),70., 2020, **Proceedings** [...]. IEEE, 2020. p. 70–78.

NAKAMURA, W. T.; DE SOUZA, J. C.; TEIXEIRA, L. M.; SILVA, A.; DA SILVA, R.; GADELHA, B.; CONTE, T. Requirements Behind Reviews: How do Software Practitioners See App User Reviews to Think of Requirements? In: BRAZILIAN SYMPOSIUM ON SOFTWARE QUALITY, 20., 2021, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2021. p. 1–9.

NAYEBI, M.; CHO, H.; RUHE, G. **App store mining is not enough for app improvement**. [s.l.] Empirical Software Engineering, 2018. v. 232764–2794 p.

NEMA, P.; ANTHONYSAMY, P.; TAFT, N.; PEDDINTI, S. T. Analyzing User Perspectives on Mobile App Privacy at Scale. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 44. 2022, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2022. p. 112–124.

NIELSEN, J. **Usability Enginnering**. Cambridge, MA: Academic Press, Inc., 1993. 392 p.

NOEI, E.; ZHANG, F.; ZOU, Y. Too Many User-Reviews! What Should App Developers Look at First? **IEEE Transactions on Software Engineering**, v. X, n. X, p. 367–378, 2019. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85100902334&doi=10.1109%2FTSE.2019.2893171&partnerID=40&md5=06a6a820a89998d26d683d4c1a980a99>.

NUSEIBEH, B.; EASTERBROOK, S. Requirements Engineering: A Roadmap. In: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING (ICSE'00), **Proceedings** [...]. v. 1, p. 35–46, 2000.

PAGANO, D.; MAALEJ, W. User feedback in the appstore: An empirical study. In: INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE), 21., 2013, **Proceedings** [...]. IEEE, 2013. p. 125–134.

PALOMBA, F.; LINARES-VASQUEZ, M.; BAVOTA, G.; OLIVETO, R.; DI PENTA, M.; POSHYVANYK, D.; DE LUCIA, A. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME), 2015, **Proceedings** [...]. IEEE, 2015. p. 291–300.

PALOMBA, F.; SALZA, P.; CIURUMELEA, A.; PANICHELLA, S.; GALL, H.; FERRUCCI, F.; DE LUCIA, A. Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 39., 2017, **Proceedings** [...]. IEEE, 2017. p. 106–117.

PANDEY, M.; LITORIYA, R.; PANDEY, P. Application of Fuzzy DEMATEL Approach in Analyzing Mobile App Issues. **Empirical Software Engineering**, v. 21, n. 3, p. 1067–1106, 2019.

PANICHELLA, S.; DI SORBO, A.; GUZMAN, E.; VISAGGIO, C. A.; CANFORA, G.; GALL, H. ARdoc: App reviews development oriented classifier. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2016, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2016. v. 13-18- Nove, p. 1023–1027.

PANTHUM, T.; SENIVONGSE, T. Generating Functional Requirements Based on Classification of Mobile Application User Reviews. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING RESEARCH, MANAGEMENT AND APPLICATIONS (SERA), 19. Research, Management and Applications (SERA), 2021, **Proceedings** [...]. 2021. p. 15–20.

PHETRUNGNAPHA, K.; SENIVONGSE, T. Classification of Mobile Application User Reviews for Generating Tickets on Issue Tracking System. In: INTERNATIONAL CONFERENCE ON INFORMATION & COMMUNICATION TECHNOLOGY AND SYSTEM (ICTS), 12., 2019, **Proceedings** [...]. IEEE, 2019. p. 229–234.

POHL, K.; RUPP, C. **A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level – IREB compliant**. Cambridge: Rocky Nook Inc, 2015. 1–30 p.

PRESSMAN, R. S. **Software engineering: A practitioner's approach. 8th Edition**. 8. ed. Porto Alegre: AMGH Editora ltda., 2016. 940 p.

REYES ARIAS, J. E.; KURTZHALL, K.; PHAM, D.; MKAOUER, M. W.; ELGLALY, Y. N. Accessibility Feedback in Mobile Application Reviews: A Dataset of Reviews and Accessibility Guidelines. In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS EXTENDED ABSTRACTS, 2022, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2022. p. 1–7.

SANTOS, R. I.; GROEN, E. C.; VILLELA, K. A taxonomy for user feedback classifications. In: JOINT OF INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY WORKSHOPS, DOCTORAL SYMPOSIUM, LIVE STUDIES TRACK, AND POSTER TRACK, REFSQ-JP, 2019, **Proceedings** [...]. CEUR Workshop Proceedings, 2019. v. 2376

SHAH, F. A.; SIRTS, K.; PFAHL, D. Using app reviews for competitive analysis: tool support. In: ACM SIGSOFT INTERNATIONAL WORKSHOP ON APP MARKET ANALYTICS, 3., 2019, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2019. p. 40–46.

SNIJDERS, R.; DALPIAZ, F.; HOSSEINI, M.; SHAHRI, A.; ALI, R. Crowd-centric requirements engineering. In: IEEE/ACM INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING, 7., UCC 2014, August 2016., 2014, **Proceedings** [...]. IEEE, 2014. p. 614–615.

SOMMERVILLE, I. **Software Engineering - 10th ed**. [s.l: s.n.]810 p.

SRISOPHA, K.; ALFAYEZ, R. Software quality through the eyes of the end-user and static analysis tools. In: INTERNATIONAL WORKSHOP ON SOFTWARE QUALITIES

AND THEIR DEPENDENCIES, 1. 2018, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2018. p. 1–4.

STATISTA. **Number of apps available in leading app stores as of 1st quarter 2021**. Disponível em: <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>. Acesso em: 18 mar. 2022.

SUN, X.; LI, L.; MERCALDO, F.; YANG, Y.; SANTONE, A.; MARTINELLI, F. Automated Intention Mining with Comparatively Fine-tuning BERT. In: INTERNATIONAL CONFERENCE ON NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL (NLPIR),5., 2021, New York, NY, USA. **Proceedings** [...] New York, NY, USA: ACM, 2021. p. 157–162.

TAN, P.-N.; STEINBACH, M.; KARPATNE, A.; KUMAR, V. **Introduction to Data Mining**. 2. ed. [s.l.] Pearson Addison Wesley, 2019. 866 p.

TUSHEV, M.; EBRAHIMI, F.; MAHMOUD, A. Domain-specific analysis of mobile app reviews using keyword-assisted topic models. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 44., 2022, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2022. v. 2022- May, p. 762–773.

VAN LAMSWEERDE, A. Goal-oriented requirements engineering: a guided tour. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 50., 2001, **Proceedings** [...]. IEEE Comput. Soc, 2001. p. 249–262.

VILLARROEL, L.; BAVOTA, G.; RUSSO, B.; OLIVETO, R.; DI PENTA, M. Release planning of mobile apps based on user reviews. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 38. 2016, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2016. v. 14-22- May-, p. 14–24.

WANG, C.; ZHANG, F.; LIANG, P.; DANEVA, M.; VAN SINDEREN, M. Can app changelogs improve requirements classification from app reviews? In: ACM/IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 2018, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2018. p. 1–4.

WANG, Y.; ZHENG, L.; LI, N. ROM: A requirement opinions mining method preliminary try based on software review data. **ACM International Conference Proceeding Series**, p. 26–33, 2020. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085937715&doi=10.1145%2F3380625.3380665&partnerID=40&md5=cc6fa478afcd298a2fa7d60d7b771037>.

WAUTELET, Y.; HENG, S.; KOLP, M.; MIRBEL, I. Unifying and Extending User Story Models. In: **Advanced Information Systems Engineering. CAiSE 2014. Lecture Notes in Computer Science**. [s.l.] Springer, 2014. p. 211–225.

WIEGERS, K.; BEATTY, J. **Software Requirements, Third Edition**. [s.l.] Microsoft Press, 2013.

XIAO, J.; ZENG, J.; YAO, S.; CAO, Y.; JIANG, Y.; WANG, W. Listening to the Crowd for the Change File Localization of Mobile Apps. In: ACM INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING AND ITS EMERGING APPLICATIONS, 1., 2021, New York, NY, USA. **Proceedings** [...]. New York, NY, USA: ACM, 2021. v. 1, p. 71–76.

YASSUDA, T. **Revisão Sistemática da Literatura sobre processos de extração e análisa de requisitos de software a partir de comentários de usuários em aplicativos móveis**. 2021. Escola Politécnica da Universidade de São Paulo, 2021.

YIN, H.; PFAHL, D. Application of the OIRE method—tool support and initial feedback from two chinese companies. **Software Quality Journal**, v. 29, n. 4, p. 783–815, 2021. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85108316487&doi=10.1007%2Fs11219-021-09562-1&partnerID=40&md5=cd379cb4c56dc4299c6fb54d19043c04>.

ZAMANI, K.; ZOWGHI, D.; ARORA, C. Machine Learning in Requirements Engineering: A Mapping Study. In: INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE WORKSHOPS (REW), 29., 2021, **Proceedings** [...]. IEEE, 2021. v. 2021- Septe, p. 116–125.

ZHAO, L.; ALHOSHAN, W.; FERRARI, A.; LETSHOLO, K. J.; AJAGBE, M. A.; CHIOASCA, E.-V.; BATISTA-NAVARRO, R. T. Natural Language Processing for Requirements Engineering. **ACM Computing Surveys**, v. 54, n. 3, p. 1–41, 30 abr. 2022. Disponível em: <https://dl.acm.org/doi/10.1145/3444689>.

ZHOU, W.; WANG, Y.; GAO, C.; YANG, F. Emerging topic identification from app reviews via adaptive online biterm topic modeling. **Frontiers of Information Technology & Electronic Engineering**, v. 23, n. 5, p. 678–691, 11 maio 2022. Disponível em: <https://link.springer.com/10.1631/FITEE.2100465>.

APÊNDICE A – *STRINGS* DE BUSCA

Nome da Base	String de busca
IEEE	((((("All Metadata": "user review" OR "All Metadata": "user feedback" OR "All Metadata": "user comment" OR "All Metadata": "user opinion" OR "All Metadata": "app review" OR "All Metadata": "online review") AND ("All Metadata": "software release" OR "All Metadata": "feature request" OR "All Metadata": "software feature" OR "All Metadata": "user story" OR "All Metadata": "requirements engineering" OR "All Metadata": "software requirement*") AND ("All Metadata": "elicit*" OR "All Metadata": "extract*" OR "All Metadata": "analy*" OR "All Metadata": "min*" OR "All Metadata": "generate" OR "All Metadata": "specif*" OR "All Metadata": "gath*"))
SpringerLink	("user review" OR "user feedback" OR "user comment" OR "user opinion" OR "app review" OR "online review") AND ("software release" OR "feature request" OR "software feature" OR "user story" OR "requirements engineering" OR "software requirement") AND ("elicit*" OR "extract*" OR "analy*" OR "min*" OR "generat*" OR "gath*" OR "spec*")
ACM	**Anywhere** "user review" OR "user feedback" OR "user comment" OR "user opinion" OR "app review" OR "online review" **Anywhere** "software release" OR "feature request" OR "software feature" OR "user story" OR "requirements engineering" OR "software requirement*" **Anywhere** elicit* OR extract* OR analy* OR "min*" OR "generat*" OR "gath*" OR "spec*" [[All: "user review"] OR [All: "user feedback"] OR [All: "user comment"] OR [All: "user opinion"] OR [All: "app review"] OR [All: "online review"]] AND [[All: "software release"] OR [All: "feature request"] OR [All: "software feature"] OR [All: "user story"] OR [All: "requirements engineering"] OR [All: "software requirement*"]] AND [[All: "elicit*"] OR [All: "extract*"] OR [All: "analy*"] OR [All: "min*"] OR [All: "generat*"] OR [All: "gath*"] OR [All: "spec*"]]

APÊNDICE B - ESTUDOS SELECIONADOS NA RSL

ID	Título	Autores	Publicado em	Ano de Publicação
A01	Analysis of user comments: An approach for software requirements evolution	Carreno L.V.G., Winbladh K.	International Conference on Software Engineering (ICSE)	2013
A02	User feedback in the appstore: An empirical study	Pagano D., Maalej W.	International Requirements Engineering Conference (RE)	2013
A03	AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace	Chen N,Lin J,Hoi SC,Xiao X,Zhang B	International Conference on Software Engineering (ICSE)	2014
A04	For user-driven software evolution: Requirements elicitation derived from mining online reviews	Jiang W., Ruan H., Zhang L., Lew P., Jiang J.	Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)	2014
A05	How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews	Guzman E., & Maalej W.	International Requirements Engineering Conference (RE)	2014
A06	What Parts of Your Apps Are Loved by Users?	Gu X,Kim S	International Conference on Automated Software Engineering (ASE)	2015
A07	Release Planning of Mobile Apps Based on User Reviews	Villarroel L,Bavota G,Russo B,Oliveto R,Di Penta M	International Conference on Software Engineering (ICSE)	2016
A08	Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews	McIlroy S, Ali N, Khalid H, Hassan AE.	Empirical Software Engineering	2016
A09	On the automatic classification of app reviews	Maalej W., Kurtanović Z., Nabil H., Stanik C.	Requirements Engineering	2016
A10	ARdoc: App Reviews Development Oriented Classifier	Panichella S, Di Sorbo A,Guzman E,Visaggio CA,Canfora G,Gall HC	International Symposium on Foundations of Software Engineering (SIGSOFT)	2016
A11	Same App, Different App Stores: A Comparative Study	Ali M, Joorabchi ME, Mesbah A.	International Conference on Mobile Software Engineering and Systems (MOBILESoft)	2017

ID	Título	Autores	Publicado em	Ano de Publicação
A12	Learn more, pay less! Lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements	Abad ZS, Sims SD, Cheema A, Nasir MB, Harisinghani P.	International Requirements Engineering Conference (RE)	2017
A13	Software feature extraction using infrequent feature extraction	Putri D.G.P., Siahaan D.O.	International Annual Engineering Seminar (InAES)	2017
A14	Automatic Classification of Non-Functional Requirements from Augmented App User Reviews	Lu, M., & Liang, P.	International Conference on Evaluation and Assessment in Software Engineering (EASE)	2017
A15	Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews	Palomba F., Salza P., Ciurumelea A., Panichella S., Gall, H., Ferrucci F., & De Lucia A.	International Conference on Software Engineering (ICSE)	2017
A16	SURF: Summarizer of User Reviews Feedback	Di Sorbo A, Panichella S,Alexandru CV,Visaggio CA,Canfora G	International Conference on Software Engineering (ICSE)	2017
A17	Users — The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews	Groen EC, Kopczyńska S, Hauer MP, Krafft TD, Doerr J.	International Requirements Engineering Conference (RE)	2017
A18	A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution	Guzman E., Ibrahim M., Glinz M.	International Requirements Engineering Conference (RE)	2017
A19	Attributes that predict which features to fix: Lessons for app store mining	Licorish, S. A., Savarimuthu, B. T. R., & Keertipati, S.	International Conference on Evaluation and Assessment in Software Engineering (EASE)	2017
A20	App review analysis via active learning: Reducing supervision effort without compromising classification accuracy	Dhinakaran V.T., Pulle R., Ajmeri N., Murukannaiah P.K.	International Requirements Engineering Conference (RE)	2018
A21	App store mining is not enough for app improvement	Nayebi M., Cho H., Ruhe G.	Empirical Software Engineering	2018
A22	Using frame semantics for classifying and summarizing application store reviews	Jha N., Mahmoud A.	Empirical Software Engineering	2018

ID	Título	Autores	Publicado em	Ano de Publicação
A23	Can app changelogs improve requirements classification from app reviews?: An exploratory study	Wang Chong, Zhang Fan, Liang P., Daneva M., Van Sinderen M.	International Symposium on Empirical Software Engineering and Measurement (ESEM)	2018
A24	Software Quality through the Eyes of the End-User and Static Analysis Tools: A Study on Android OSS Applications	Srisopha K, Alfayez R	International Conference on Software Engineering (ICSE)	2018
A25	User Feedback in the App Store: A Cross-Cultural Study	Guzman E, Oliveira L,Steiner Y,Wagner LC,Glinz M	International Conference on Software Engineering (ICSE)	2018
A26	Do Android App Users Care about Accessibility? An Analysis of User Reviews on the Google Play Store	Eler MM, Orlandin L, Oliveira AD.	Brazilian Symposium on Human Factors in Computing Systems (IHC)	2019
A27	Preprocessing Role in Analyzing Tweets Towards Requirement Engineering	Ebrahimi, A. M., & Barforoush, A. A.	Iranian Conference on Electrical Engineering (ICEE)	2019
A28	Classification of mobile application user reviews for generating tickets on issue tracking system	Phetrungnapha K., Senivongse T.	International Conference on Information and Communication Technology and Systems (ICTS)	2019
A29	Classifying multilingual user feedback using traditional machine learning and deep learning	Stanik C., Haering M., Maalej W.	International Requirements Engineering Conference (RE)	2019
A30	Mining non-functional requirements from App store reviews	Jha N., Mahmoud A.	Empirical Software Engineering	2019
A31	Using app reviews for competitive analysis: Tool support	Shah F.A., Sirts K., Pfahl D.	International Symposium on Foundations of Software Engineering (SIGSOFT)	2019
A32	Application of Fuzzy DEMATEL Approach in Analyzing Mobile App Issues	Pandey, M., Litoriya, R. & Pandey, P.	Programming and Computer Software	2019
A33	Automatic Requirements Elicitation from Social Media (ARESM)	Kengphanphanit, N., & Muenchaisri, P.	International Conference on Computer Communication and Information Systems (CCCIS)	2020

ID	Título	Autores	Publicado em	Ano de Publicação
A34	Mining User Opinions to Support Requirement Engineering: An Empirical Study	Dąbrowski J., Letier E., Perini A., Susi A.	International Conference on Advanced Information Systems Engineering (CAiSE)	2020
A35	Pattern Learning for Detecting Defect Reports and Improvement Requests in App Reviews	Gino V. H.,Truşcă M. M., Frasincar F.	International Conference on Applications of Natural Language to Information Systems (NLDB)	2020
A36	ROM: A requirement opinions mining method preliminary try based on software review data	Wang Ying, Zheng L., Li N.	International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)	2020
A37	On Building an Automatic Identification of Country-Specific Feature Requests in Mobile App Reviews: Possibilities and Challenges	Srisopha K., Phonsom C., Li M., Link D., Boehm B.	International Conference on Software Engineering (ICSE)	2020
A38	Automatically identifying requirements-oriented reviews using a top-down feature extraction approach	Song R., Li T., Ding Z.	Asia-Pacific Software Engineering Conference (APSEC)	2020
A39	Caspar: Extracting and synthesizing user stories of problems from app reviews	Guo H., Singh M.P.	International Conference on Software Engineering (ICSE)	2020
A40	Analysis of Compatibility in Open Source Android Mobile Apps	Mukherjee D., Ruhe G.	International Requirements Engineering Conference (RE)	2020
A41	Classification of application reviews into software maintenance tasks using data mining techniques	Al-Hawari A, Najadat H, Shatnawi R.	Software Quality Journal	2021
A42	How Should I Improve the UI of My App? A Study of User Reviews of Popular Apps in the Google Play	Chen Q, Chen Chunyang, Hassan S, Xing Z, Xia X, Hassan AE.	ACM Transactions on Software Engineering and Methodology	2021
A43	NFR Classification using Keyword Extraction and CNN on App Reviews	Hidayat T., Rochimah S.	International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)	2021

ID	Título	Autores	Publicado em	Ano de Publicação
A44	Opinion mining for app reviews: an analysis of textual representation and predictive models	De Araújo, A.F., Gôlo, M.P.S. & Marcacini, R.M	Automated Software Engineering	2021
A45	Extracting Arguments Based on User Decisions in App Reviews	Kunaefi A., Aritsugi M.	IEEE Access	2021
A46	Requirements Behind Reviews: How Do Software Practitioners See App User Reviews to Think of Requirements?	Nakamura, W. T., de Souza, J. C., Teixeira, L. M., Silva, A., da Silva, R., Gadelha, B., & Conte, T.	Brazilian Symposium on Software Quality (SBQS)	2021
A47	Application of the OIRE method—tool support and initial feedback from two chinese companies	Yin H., Pfahl D.	Software Quality Journal	2021
A48	Automated Intention Mining with Comparatively Fine-Tuning BERT	Sun X,Li L,Mercaldo F,Yang Y,Santone A,Martinelli F	International Conference on Natural Language Processing and Information Retrieval (NLPIR)	2021
A49	Classifying User Requirements from Online Feedback in Small Dataset Environments using Deep Learning	Mekala RR, Irfan A, Groen EC, Porter A, Lindvall M.	International Requirements Engineering Conference (RE)	2021
A50	Generating functional requirements based on classification of mobile application user reviews	Panthum T., Senivongse T.	International Conference on Software Engineering Research, Management and Applications (SERA)	2021
A51	FeatCompare: Feature comparison for competing mobile apps leveraging user reviews	Assi M, Hassan S, Tian Y, Zou Y.	Empirical Software Engineering	2021
A52	Too Many User-Reviews! What Should App Developers Look at First?	Noei E., Zhang Feng, Zou Y.	IEEE Transactions on Software Engineering	2021
A53	Listening to the Crowd for the Change File Localization of Mobile Apps	Xiao J, Zeng J, Yao S, Cao Y, Jiang Y, Wang W	International Conference on Intelligent Computing and Its Emerging Applications (ICEA)	2022
A54	Finding Appropriate User Feedback Analysis Techniques for Multiple Data Domains	Devine P.	International Conference on Software Engineering (ICSE)	2022

ID	Título	Autores	Publicado em	Ano de Publicação
A55	Domain-Specific Analysis of Mobile App Reviews Using Keyword-Assisted Topic Models	Tushev M,Ebrahimi F,Mahmoud A	International Conference on Software Engineering (ICSE)	2022
A56	Automatically generating taxonomy for grouping app reviews — a study of three apps	Malgaonkar S, Licorish SA, Savarimuthu BT	Software Quality Journal	2022
A57	Can User Reviews Indicate Usability Heuristic Issues?	Diniz, L. D. N., de Souza Filho, J. C., & Carvalho, R. M.	Conference on Human Factors in Computing Systems (CHI)	2022
A58	Accessibility Feedback in Mobile Application Reviews: A Dataset of Reviews and Accessibility Guidelines	Reyes Arias, J. E., Kurtzhall, K., Pham, D., Mkaouer, M. W., & Elglaly, Y. N.	Conference on Human Factors in Computing Systems (CHI)	2022
A59	Analyzing User Perspectives on Mobile App Privacy at Scale	Nema P,Anthonysamy P,Taft N,Peddinti ST	International Conference on Software Engineering (ICSE)	2022
A60	Emerging topic identification from app reviews via adaptive online biterm topic modeling	Zhou W, Wang Yong, Gao C, Yang F.	Frontiers of Information Technology & Electronic Engineering - Journal of Zhejiang University SCIENCE C (Computers & Electronics)	2022

APÊNDICE C - CONFERÊNCIAS DOS TRABALHOS DA REVISÃO

Local de Publicação	Quantidade de trabalhos
International Conference on Software Engineering (ICSE)	12
International Requirements Engineering Conference (RE)	9
Empirical Software Engineering	5
Software Quality Journal	3
International Symposium on Foundations of Software Engineering (SIGSOFT)	2
Conference on Human Factors in Computing Systems (CHI)	2
International Conference on Evaluation and Assessment in Software Engineering (EASE)	2
International Conference on Software Engineering Research, Management and Applications (SERA)	1
Requirements Engineering	1
Asia-Pacific Software Engineering Conference (APSEC)	1
IEEE Access	1
International Conference on Natural Language Processing and Information Retrieval (NLPIR)	1
IEEE Transactions on Software Engineering	1
International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)	1
International Annual Engineering Seminar (InAES)	1
Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)	1
International Conference on Advanced Information Systems Engineering (CAiSE)	1
Frontiers of Information Technology & Electronic Engineering	1
International Conference on Applications of Natural Language to Information Systems (NLDB)	1
Brazilian Symposium on Human Factors in Computing Systems (IHC)	1
International Conference on Automated Software Engineering (ASE)	1
Brazilian Symposium on Software Quality (SBQS)	1
International Conference on Computer Communication and Information Systems (CCCIS)	1
International Symposium on Empirical Software Engineering and Measurement (ESEM)	1
ACM Transactions on Software Engineering and Methodology	1
Iranian Conference on Electrical Engineering (ICEE)	1
International Conference on Information and Communication Technology and Systems (ICTS)	1
Programming and Computer Software	1
International Conference on Intelligent Computing and Its Emerging Applications (ICEA)	1
Automated Software Engineering	1
International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)	1
International Conference on Mobile Software Engineering and Systems (MOBILESoft)	1
Total Geral	60