UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

**Glaucia Schnoeller dos Santos**

# Um processo de representação estruturada e validação de requisitos de software para mitigar problemas semânticos

Limeira
2021

**Glaucia Schnoeller dos Santos**

# Um processo de representação estruturada e validação de requisitos de software para mitigar problemas semânticos

Tese apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutora em Tecnologia, na área de Sistemas de Informação e Comunicação.

**Orientador: Prof. Dr. Ivan Luiz Marques Ricarte**

Este trabalho corresponde à versão final da Tese defendida por Glaucia Schnoeller dos Santos e orientada pelo Prof. Dr. Ivan Luiz Marques Ricarte.

Limeira
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

Santos, Glaucia Schnoeller dos, 1992-
Sa59p Um processo de representação estruturada e validação de requisitos de software para mitigar problemas semânticos / Glaucia Schnoeller dos Santos. – Limeira, SP : [s.n.], 2021.

Orientador: Ivan Luiz Marques Ricarte.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Processamento de linguagem natural (Computação). 2. Semântica. 3. Engenharia de requisitos. I. Ricarte, Ivan Luiz Marques, 1962-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** A process for structured representation and validation of software requirements to mitigate semantic problems
**Palavras-chave em inglês:**
Natural language processing (Computer science)
Semantics
Requirements engineering
**Área de concentração:** Sistemas de Informação e Comunicação
**Titulação:** Doutora em Tecnologia
**Banca examinadora:**
Ivan Luiz Marques Ricarte [Orientador]
Ana Estela Antunes da Silva
Luiz Camolesi Júnior
Renata Pontin de Mattos Fortes
Rosana Teresinha Vaccare Braga
**Data de defesa:** 08-11-2021
**Programa de Pós-Graduação:** Tecnologia

**Identificação e informações acadêmicas do(a) aluno(a)**
- ORCID do autor: https://orcid.org/0000-0003-0198-7380
- Currículo Lattes do autor: http://lattes.cnpq.br/4828581367263582

# FOLHA DE APROVAÇÃO

Abaixo são apresentados os membros da comissão julgadora da sessão pública da defesa de tese para o Título de Doutora em Tecnologia na área de concentração de Sistemas de Informação e Comunicação, submetida pela aluna Glaucia Schnoeller dos Santos, em 08 de novembro de 2021 na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

**Prof. Dr. Ivan Luiz Marques Ricarte**
Presidente da Comissão Julgadora

**Profa. Dra. Ana Estela Antunes da Silva**
FT/UNICAMP

**Prof. Dr. Luiz Camolesi Júnior**
FT/UNICAMP

**Profa. Dra. Renata Pontin de Mattos Fortes**
ICMC/USP

**Profa. Dra. Rosana Teresinha Vaccare Braga**
ICMC/USP

Ata da defesa, assinada pelos membros da Comissão Examinadora, encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós Graduação da Faculdade de Tecnologia.

*We can only see a short distance ahead, but we can see plenty there that needs to be done.*
(Alan Turing)

# Agradecimentos

Agradeço a todos que contribuíram, de forma direta ou indireta, com a construção desta tese. Apresento minha sincera gratidão, especialmente:

Ao meu orientador, Prof. Dr. Ivan Luiz Marques Ricarte, por mais uma oportunidade de realizar a pesquisa sob sua orientação. Grata pela atenção e valiosos ensinamentos compartilhados que seguirei em toda minha trajetória profissional.

Aos meus queridos pais, Diogenes de Souza Santos e Roseli Schnoeller Santos, pelos conselhos, paciência, encorajamento e o acolhimento em todas as fases da minha vida. Obrigada por abraçarem comigo minhas escolhas.

Aos participantes da prova de conceito, que mesmo no enfrentamento da pandemia, reservaram um tempo para a avaliação da proposta desta pesquisa. Obrigada pelo comprometimento e os comentários de valor para o trabalho.

Às empresas que fui colaboradora no decorrer do curso. À equipe da empresa Mupi, sou grata pelos aprendizados e incentivos que tive para a pesquisa. À Paula Rodrigues Furtado, pela confiança em meu trabalho e por me motivar em várias etapas deste trabalho. A Rafaely Carolina da Cruz, pela sua generosidade em me apoiar em uma das tarefas da pesquisa. Ao Henrique Meira Costa, pelas discussões que me agregaram novos conhecimentos, pela prontidão e sugestões que enriqueceram este trabalho. À equipe da empresa Hacklab, em especial ao Bruno Sousa Martin, pela oportunidade oferecida e compreensão nos períodos que exigiram minha dedicação para o desenvolvimento dos experimentos da pesquisa. À equipe da empresa Diletta Solutions pelo apoio na etapa final desta tese.

Aos meus estimados amigos, que tenho profunda gratidão e respeito, pelo auxílio e companheirismo nesta jornada. Aos amigos que tive o privilégio de acompanhar desde o início do mestrado, sou grata pelos conselhos e os momentos que passamos juntos. À Franciene Duarte Gomes de Lima e Juan Fernando Galindo Jaramillo, por me ajudarem em várias etapas da pesquisa. À Dildre Georgiana Vasques e Gabriel Olivato, pela parceria nos projetos desenvolvidos e aprendizados em processamento de linguagem natural. À Carmen Pamela Rosales Sedano, por compartilhar seus conhecimentos, pela ajuda na elaboração do protótipo criado neste trabalho e sugestões de melhorias. Aos amigos que estão comigo desde meu período de estudos na graduação. À Samanta Branquinho de Lima e Virginia Trinca da Silva, minhas amigas de vida, por estarem presentes durante esses anos e me ampararem em minhas dificuldades. Ao William Toshio Nakagawa, por estar sempre disposto em ajudar e pela sua dedicação. Apresento aqui minha imensa admiração que tenho por vocês.

A todos os colaboradores da Faculdade de Tecnologia - UNICAMP, pelo suporte que contribuíram para a realização desta pesquisa. Em especial, a Danielle Emanuelle Aparecida Ribeiro, pela sua prontidão e auxílio. Aos membros da banca examinadora, pela disponibilidade em avaliar esta tese e pelos comentários relevantes para aperfeiçoar a pesquisa.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

# Resumo

Requisitos expressos em linguagem natural são imprecisos e podem apresentar deficiências como inconsistência, incompletude e ambiguidade. Essas deficiências na documentação de requisitos podem afetar o projeto, pois o procedimento para a correção desses problemas normalmente é manual e está sujeito a erros, podendo ocasionar problemas de interpretabilidade nas especificações de software como falhas na compreensão e comunicação dos requisitos. Para sanar esses problemas, este estudo apresenta um processo para a representação estruturada e validação de requisitos que visa melhorar a qualidade das especificações de software. Um *framework* foi definido como parte deste trabalho para automatizar esse processo e proporcionar sua implantação em um protótipo de uma ferramenta. Para a avaliação de suas contribuições, uma prova de conceito foi conduzida com a participação de especialistas em projetos de desenvolvimento de software. A coleta de dados foi realizada em entrevistas com a aplicação de um questionário para cada participante do estudo. Os dados obtidos das entrevistas foram avaliados pela técnica de análise de conteúdo. O desempenho do processo foi verificado por meio de experimentos com sua aplicação em requisitos obtidos de documentos públicos de software. Os resultados foram investigados por métricas de classificação e analisados pela técnica de estatística descritiva. O processo alcançou uma média de 95% de precisão para a geração do modelo conceitual e 79% para a identificação de erros nos requisitos. Os modelos conceituais gerados pela ferramenta foram mais completos e corretos. Com base nos resultados foi possível constatar que o processo automático mostrou ser eficaz para controlar problemas de interpretabilidade dos requisitos mediante a melhoria na qualidade dos requisitos, a descoberta de novas funcionalidades e a identificação de relações tácitas. A ferramenta apresentou a facilidade de uso e apoiou a tarefa de definição de requisitos com o aprimoramento na escrita dos requisitos e a modelagem a nível conceitual. Além disso, a implantação da ferramenta forneceu a integração, recuperação e agrupamento das informações extraídas dos requisitos estruturados e validados devido à criação e organização de bases de conhecimento de forma automática.

***Palavras-chave:*** *Linguagem natural; Processamento automático; Análise semântica; Qualidade de requisitos.*

# Abstract

Requirements expressed in natural language are inaccurate and may present deficiencies such as inconsistency, incompleteness and ambiguity. These deficiencies in the documentation of requirements may affect the project because the procedure required to correct them is usually manual and subject to error, which can cause issues of interpretability in the software specifications, such as failure to understand and communicate the requirements. To solve these issues, this study provides a process for the structured representation and validation of requirements that aims to improve the quality of software specifications. A framework was defined as part of this work to automate this process and provide its implementation in a tool prototype. In order to evaluate the contributions of this process, a proof of concept was conducted together with software development specialists. Data collection was done through interviews with the application of a questionnaire for each study participant. The data obtained from the interviews were evaluated using the content analysis technique. Process performance was assessed by experimenting with the project's application with requirements found in public software documentation. The results were investigated by classification metrics and analyzed using the descriptive statistics technique. The process achieved an average of 95% accuracy for generating the conceptual model and 79% for identifying errors in the requirements. The conceptual models generated by the tool were more complete and correct. Based on the results, it was possible to verify that the automatic process proved to be effective in controlling requirements interpretability problems by improving the quality of requirements, discovering new features and identifying tacit relationships. The tool was easy to use and supported the requirements definition task with the improvement in requirements writing and modeling at the conceptual level. In addition, the implementation of the tool provided the integration, retrieval and grouping of information extracted from structured and validated requirements due to the automatic creation and organization of knowledge bases.

***Keywords:*** *Natural language; Automated processing; Semantic analysis; Quality requirements.*

# Lista de Figuras

# Lista de Tabelas

# Lista de Quadros

# Siglas

**ACM** Association for Computing Machinery

**API** Application Programming Interface

**AWS** Amazon Web Services

**BPM** Business Process Models

**BPMN** Business Process Model and Notation

**CEP** Comitê de Ética em Pesquisa

**CSV** Comma Separated Values

**EC2** Elastic Compute Cloud

**IE** Information Extraction

**IEEE** Institute of Electrical and Electronics Engineers

**Json** JavaScript Object Notation

**Json-LD** JavaScript Object Notation for Linked Data

**LNC** Linguagem Natural Controlada

**LTL** Lógica Temporal Linear

**M2M** Model-to-model

**OCL** Object Constraint Language

**P** Proposições

**PLN** Processamento de Linguagem Natural

**PRISMA** Preferred Reporting Items for Systematic reviews and Meta-Analyses

**QUS** Quality User Story

**RDF** Resource Description Framework

**RPM** Remote Patient Monitoring

**SBVR** Semantics of Business Vocabulary and Rules

**subj** Subject

**SYsML** System Modeling Language

**TCLE** Termo de Consentimento Livre e Esclarecido

**Turtle** Terse RDF Triple Language

**UML** Unified Modeling Language

**URI** Uniform Resource Identifiers

**URL** Uniform Resource Locator

**vb** Verb

**VDM** Vienna Development Method

**Web** World Wide Web

**XML** Extensible Markup Language

**XP** Extreme Programming

# Sumário

# Capítulo 1

# Introdução

Uma das tarefas mais complexas na produção de sistemas de software é a definição de requisitos. Essa etapa visa identificar quais são as funcionalidades que o aplicativo deve desempenhar e quais restrições estão relacionadas ao seu desenvolvimento. Para auxiliar nessa tarefa, os engenheiros de software utilizam modelos convencionais de representação de requisitos como, por exemplo, o caso de uso (CAMPOS et al., 2010) e a estória de usuário (COHN, 2004b).

As especificações de software geradas por esses modelos convencionais normalmente são escritas em linguagem natural. Alguns autores, como Ferreira e Silva (2012), consideram a utilização da linguagem natural como o melhor meio para especificar requisitos, devido à sua facilidade de uso entre a equipe do projeto e o seu nível de expressividade. Entretanto, essa linguagem é imprecisa e pode apresentar deficiências como inconsistência, incompletude e ambiguidade (SOARES; MOURA, 2015). Essas deficiências inerentes ao uso da linguagem natural podem resultar em uma fraca qualidade na documentação e apresentar problemas como a falta de compreensão e falha na comunicação (transmissão) dos requisitos.

As causas frequentes dos desafios e falhas de projetos de desenvolvimento de software estão relacionadas com problemas de interpretabilidade dos requisitos. Nos estudos de Dal Forno e Muller (2017) e Hughes et al. (2016) existem relatos de que um dos fatores críticos mais comuns em projetos de desenvolvimento de software é a especificação de requisitos. O primeiro relatório disponibilizado pelo Standish Group International (MULDER, 1994) menciona deficiências nas especificações de requisitos como requisitos incompletos, alterações nos requisitos e falta de clareza nos objetivos. Esse grupo contribui com pesquisas realizadas em projetos de desenvolvimento de software no contexto industrial. Esses

problemas foram citados há mais de 25 anos. No entanto, a taxa de projetos que enfrentaram desafios ou falharam nos últimos anos está acima de 60% (ALVES et al., 2021; REED; ANGOLIA, 2018; HASTIE; WOJEWODA, 2015). Pode-se observar que ainda é preciso uma maior atenção e investimentos para solucionar esses problemas.

O processo para reparar esses problemas normalmente é manual e, desse modo, demanda muito esforço da equipe e está propenso a falhas (FOCKEL; HOLTMANN, 2015). Uma possível solução é o processamento automático dos requisitos. Grande parte dos estudos da literatura estão focados na automatização para a avaliação da qualidade das informações, isto é, na validação dos requisitos. Existem diversos algoritmos e técnicas com esse foco, incluindo aprendizado de máquina, agrupamento de dados, análise de similaridade, entre outros. No entanto, a falta de padronização do texto pode limitar a capacidade desse processamento (CASTRO; BEZERRA; HIRATA, 2015).

Para sanar essa dificuldade, seria oportuna uma representação estruturada dos requisitos. Algumas pesquisas exploram a formalização dos requisitos com abordagens que estabelecem gramáticas e regras específicas de domínio para mapear construções predefinidas para representações formais, como a Linguagem Natural Controlada (LNC). Contudo, esse tipo de estruturação é dependente de uma formatação restrita e dicionários do domínio (DIAMANTOPOULOS et al., 2017). Além disso, a representação em modelos formais baseados em lógica matemática apresenta limitações para a aplicação, pois exige o conhecimento técnico da notação para sua compreensão (OZKAYA, 2018) e existe pouco suporte de ferramentas para sua implementação (WANG; DAMLJANOVIC; SUN, 2014).

Em contraste, técnicas associadas à análise semântica simplificam a estruturação dos requisitos, pois não requerem conhecimento prévio do domínio e processam textos livres de formatação (SANTOS, 2017). Existe pouca investigação da análise semântica em requisitos textuais (DIAMANTOPOULOS et al., 2017), mas, em outros domínios, o conhecimento obtido dessa estruturação apresentou a flexibilidade para ser representado em diversos formatos, como por exemplo tabelas semânticas e mapas conceituais (VASQUES et al., 2016). Diante disso, essa disposição contribui para o cenário atual de projetos de desenvolvimento de software, posto que a quantidade de ambientes distribuídos é crescente (VALLON et al., 2018). Neste cenário colaborativo, o modelo de representação dos requisitos deve facilitar a transferência do conhecimento entre as equipes distribuídas geograficamente.

Este trabalho apresenta um processo automático para a representação estruturada semanticamente e validação de requisitos de software, visando melhorar a eficiência na mitigação de problemas na compreensão e comunicação dos requisitos. Os conceitos desse processo foram definidos em um *framework* e implantados em um protótipo por meio de técnicas de Processamento de Linguagem Natural (PLN) e Web Semântica. Esse processo fornece uma base de conhecimento extraída dos requisitos estruturados e validados. É importante ressaltar que o *framework* não requer habilidades técnicas para sua utilização e pode ser utilizado em diversos domínios.

## 1.1 Hipóteses

Esta pesquisa se baseia nas seguintes hipóteses:

- H0: O processo automático para a representação estruturada e validação de requisitos não influencia na mitigação de problemas de interpretabilidade.
- H1: A automatização nas etapas de estruturação, validação e representação do processo de especificação de software pode aumentar a qualidade dos requisitos.
- H2: Apresentar requisitos com uma maior qualidade pode mitigar problemas na comunicação e compreensão dos requisitos escritos em linguagem natural.

## 1.2 Objetivos

O objetivo principal desta pesquisa foi melhorar a qualidade de requisitos funcionais por meio de um processo automático para sua representação estruturada e validação para mitigação de problemas de interpretabilidade, como a falta de compreensão e comunicação dos requisitos. Os objetivos específicos planejados para a elaboração deste estudo foram:

1. Identificar quais abordagens são aplicadas para reparar problemas de requisitos de software expressos em linguagem natural.
    - Identificar e mapear em quais etapas da especificação dos requisitos estão sendo aplicadas as abordagens selecionadas;
    - Identificar como estão sendo utilizadas as abordagens selecionadas.

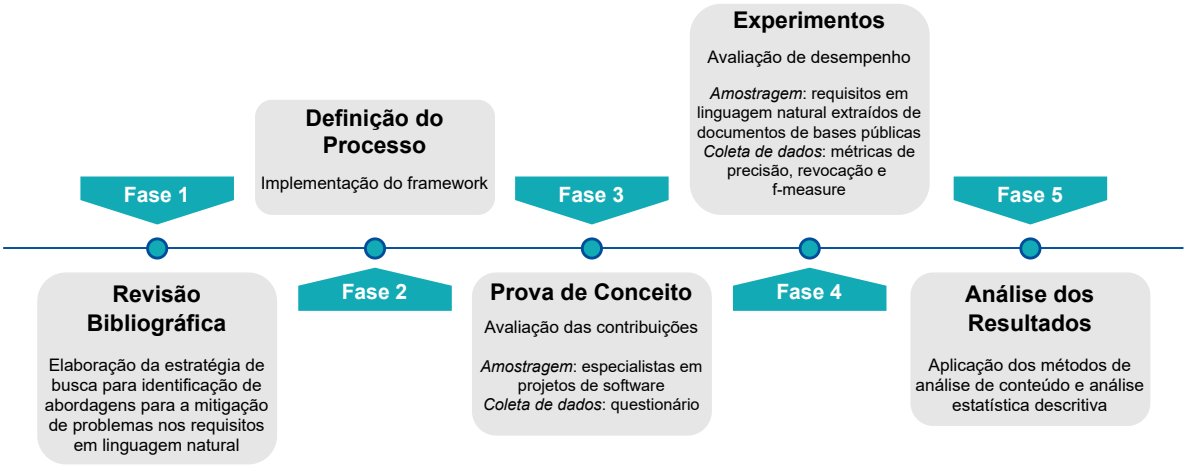
2. Definir um *framework* para tornar a informação processável por máquina e proporcionar a automatização na estruturação, validação e representação de requisitos de diferentes contextos.

3. Desenvolver um protótipo de uma ferramenta Web para a implementação do *framework*.

## 1.3 Métodos do trabalho

Para conceber os objetivos definidos para este estudo, cinco etapas foram executadas, como mostra a representação esquemática da metodologia (Figura 1.1). Inicialmente, realizou-se uma revisão bibliográfica com duas buscas de soluções para problemas em requisitos de software, afetados com o uso da linguagem natural. Após o mapeamento dessas soluções, definiu-se um processo para mitigar problemas de interpretabilidade e desenvolveu-se um *framework* para automatizar a representação estruturada e validação dos requisitos.

Figura 1.1: Estrutura esquemática da metodologia empregada na pesquisa.




Fonte: Elaborado pela autora (2021).

Para a avaliação do processo proposto, conduziu-se uma prova de conceito com a participação de especialistas em projetos de desenvolvimento de software. A questão de pesquisa que direcionou o estudo foi: "Quais as contribuições do processo automático para a representação estruturada e validação de requisitos de software na mitigação de problemas de interpretabilidade?". O desempenho do processo automático foi avaliado por meio de dois experimentos, com foco na identificação de erros e modelagem conceitual dos requisitos. Por

fim, os dados obtidos dessas avaliações foram analisados com a técnica de análise de conteúdo e estatística descritiva.

## 1.4 Organização da tese

Este trabalho é composto por seis capítulos. O Capítulo 1 situa o tema desta pesquisa com sua problematização e motivação. Para o entendimento da metodologia, é descrita a hipótese testada, os objetivos do estudo e os métodos estabelecidos no trabalho.

O Capítulo 2 introduz a revisão da literatura. Apresenta-se o embasamento teórico sobre requisitos de software e o estado da arte de abordagens para mitigar problemas em requisitos, potencialmente agravados pelo uso da linguagem natural. Essas abordagens estão categorizadas em três etapas do processo de especificação de software: estruturação, validação e representação de requisitos. Uma discussão é feita relacionada às principais descobertas.

O Capítulo 3 detalha a metodologia empregada com o tipo do estudo, os métodos de pesquisa utilizados, as amostras, os procedimentos para as coletas de dados e as técnicas aplicadas para a análise dos resultados. Esse capítulo ainda relata as ameaças à validade da pesquisa.

O Capítulo 4 revela o processo automático para a representação estruturada e validação de requisitos. Também detalha a implantação de um *framework*, criado neste trabalho, o qual incorpora os conceitos definidos nesse processo e exemplifica a sua aplicação.

O Capítulo 5 conduz a avaliação das contribuições e desempenho do processo proposto por meio de uma prova de conceito com especialistas em projetos de desenvolvimento de software e dois experimentos com requisitos extraídos de documentos públicos. São discutidos os resultados obtidos nesta avaliação.

Para finalizar, o Capítulo 6 apresenta as conclusões e limitações deste estudo. Além disso, aponta-se sugestões para pesquisas futuras.

# Capítulo 2

# Revisão da literatura

Este capítulo apresenta os conceitos envolvidos no estudo e o estado da arte de abordagens para reparar os problemas intrínsecos à linguagem natural nos requisitos de software.

## 2.1 Fundamentação teórica: requisitos de software

A engenharia de software é uma área que abrange ferramentas, métodos e processos para realizar o desenvolvimento de software com produtividade e qualidade. Nesse contexto, os processos de software estabelecem diretrizes para determinar quais atividades devem ser realizadas para cumprir os objetivos do software (HIRAMA, 2012). Esses processos podem ser compreendidos como um conjunto de tarefas que estão associadas e geram como produto um software. Apesar de diferentes processos serem aplicados em projetos de desenvolvimento de software, algumas atividades são comuns a esses processos, como a especificação de software (SOMMERVILLE, 2016). Este estudo se concentra nessa fase, na qual é realizada a apresentação e definição dos requisitos de software.

Com base no padrão 610.12-1990 estabelecido pelo IEEE (1990), requisitos são definidos como:

> (1) Uma condição ou capacidade necessária para um usuário resolver um problema ou atingir um objetivo.
> (2) Uma condição ou capacidade que deve ser satisfeita ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos.
> (3) Uma representação documentada de uma condição ou capacidade como em (1) ou (2). (p. 62, tradução da autora)

Os requisitos de software são categorizados como requisitos de usuário e requisitos de sistema. Os requisitos de usuário refletem as declarações do especialista do negócio sobre quais serviços o software em desenvolvimento deverá realizar e sob quais restrições deverá operar. Os requisitos de sistema, por sua vez, consistem na expansão dos requisitos de usuário pela equipe de desenvolvimento, para se ter informações mais específicas dos serviços que serão implementados. O Quadro 2.1 exemplifica esses tipos de requisitos em um sistema de gestão de pacientes e nota-se que a partir dos requisitos de usuários, com informações gerais, foram produzidas descrições detalhadas no nível técnico (SOMMERVILLE, 2016).

Quadro 2.1: Exemplos de requisitos de usuário e de sistema.

| **Definição de requisito de usuário** | **Especificação de requisitos de sistema** |
|---|---|
| 1. O sistema deve gerar relatórios gerenciais mensais que mostrem o custo dos medicamentos prescritos por cada clínica durante aquele mês. | 1.1 No último dia útil de cada mês deve ser gerado um resumo dos medicamentos prescritos, seus custos e as prescrições de cada clínica.<br>1.2 Após 17:30h do último dia útil do mês, o sistema deve gerar automaticamente o relatório para impressão.<br>1.3 Um relatório será criado para cada clínica, listando os nomes dos medicamentos, o número total de prescrições, o número de doses prescritas e o custo total dos medicamentos prescritos.<br>1.4 Se os medicamentos estão disponíveis em diferentes unidades de dosagem (por exemplo, 10 mg, 20 mg), devem ser criados relatórios separados para cada unidade.<br>1.5 O acesso aos relatórios de custos deve ser restrito a usuários autorizados por uma lista de controle de gerenciamento de acesso. |

Fonte: Sommerville (2016).

Requisitos de sistema descrevem os requisitos funcionais e os requisitos não funcionais. Os requisitos funcionais retratam as características ou funcionalidades do software (HIRAMA, 2012). Podem ser compreendidos como a definição dos recursos específicos que serão fornecidos pelo software (SOMMERVILLE, 2016), por exemplo:

1. O sistema deve listar todos usuários cadastrados.

2. O sistema deve buscar os usuários pelo nome.

3. O sistema exporta uma lista com o nome e o e-mail de todos os usuários.

Os requisitos não funcionais, por sua vez, descrevem as propriedades emergentes do software. Esse tipo de requisito se concentra no comportamento que é esperado pelo sistema. Em outras palavras, são restrições de qualidade que o sistema deve suportar, conforme o projeto e o ambiente. Requisitos de produto, organizacional e externo são alguns tipos desses requisitos (SOMMERVILLE, 2016; HIRAMA, 2012), como por exemplo:

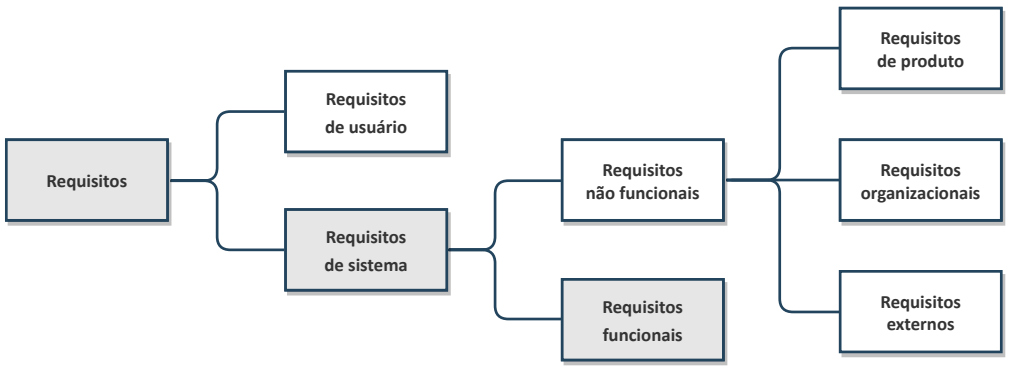
1. "*O MHC-PMS deve estar disponível para todas as clínicas durante as horas normais de trabalho (segunda a sexta-feira, 8h30 às 17h30). Períodos de não operação dentro do horário normal de trabalho não podem exceder cinco segundos em um dia*". (Requisito de produto)

2. "*Usuários do sistema MHC-PMS devem se autenticar com seus cartões de identificação da autoridade da saúde*". (Requisito organizacional)

3. "*O sistema deve implementar as disposições de privacidade dos pacientes, tal como estabelecido no HStan-03-2006-priv*". (Requisito externo)

A Figura 2.1 sintetiza a classificação dos requisitos. O foco deste trabalho está em requisitos funcionais de sistema (destacado na figura).

Figura 2.1: Classificação dos tipos de requisitos.




Fonte: Adaptado de Sommerville (2016).

### 2.1.1 Documento de requisitos de software

O conjunto de atividades e técnicas que contribuem para o entendimento e desenvolvimento dos requisitos é denominado de engenharia de requisitos (PRESSMAN; MAXIM, 2019). Essa

disciplina tem como objetivo desenvolver, analisar e manter uma documentação composta pelos requisitos (IEEE, 2010; GANESH; THANGASAMY, 2011). Para realizar essa documentação, os requisitos são especificados em alguma forma-padrão (SOMMERVILLE, 2016).

O documento de requisitos de software, também chamado de especificação de requisitos de software, é o cerne do projeto de desenvolvimento, pois envolve todas as declarações e acordos das funcionalidades que devem ser implementadas no software. São descritos tanto os requisitos gerais quanto as descrições detalhadas. Os requisitos são detalhados em diferentes níveis para apoiar todos os colaboradores do projeto como clientes, gerentes, engenheiros de software (também conhecidos como engenheiros de sistemas e analistas), bem como os engenheiros de teste e de manutenção (SOMMERVILLE, 2016; PRESSMAN; MAXIM, 2019).

Na visão de Sommerville (2016) é importante fornecer um documento de apoio para a equipe, mesmo para métodos ágeis em que os requisitos mudam rapidamente, pois deve-se ter o controle do sistema como um todo. A composição e informações do documento variam de acordo com o tipo do projeto, no entanto, o Quadro 2.2 traz uma estrutura genérica, baseada no padrão IEEE 830 (IEEE, 1998), que pode ser adaptada para diferentes projetos (SOMMERVILLE, 2016). Este documento serve de guia para as etapas subsequentes do projeto, ou seja, no planejamento, desenvolvimento, teste e manutenção do software (DUGGAL; SAXENA; GURVE, 2020).

Além dessa documentação tradicional, de acordo com o estudo de Chernak (2012) e como apontado por Barcelos e Penteado (2017), existem mais dois modelos principais para documentar os requisitos, como o caso de uso. Casos de uso expressam os comportamentos que são pretendidos para as funcionalidades do software e são utilizados para extrair os requisitos funcionais. Esse modelo se baseia em uma linguagem simples e representa, por meio de um cenário (sequência de tarefas), as interações dos atores (usuários) envolvidos nos processos do negócio e suas funções (GUEDES, 2011).

Um dos objetivos do caso de uso é facilitar a comunicação entre os membros do projeto, incluindo pessoas não técnicas. Por essa razão, a descrição do caso de uso (na forma textual) contém algumas informações fundamentais, como os atores (quem irá interagir com o sistema), o objetivo geral e as etapas sequenciais que retratam a interação (bem sucedida) dos atores com o sistema. Ademais, são incluídas as etapas alternativas, para descrever os fluxos de exceções. As pré e pós condições de estado para iniciar e encerrar a execução do caso de uso devem ser

Quadro 2.2: Estrutura da documentação de requisitos.

| Capítulo | Descrição |
|---|---|
| Prefácio | Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão. |
| Introdução | Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software. |
| Glossário | Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor. |
| Definição de requisitos de usuário | Deve descrever os serviços fornecidos ao usuário.<br>Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados. |
| Arquitetura do sistema | Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados. |
| Especificação de requisitos do sistema | Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas. |
| Modelos do sistema | Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados. |
| Evolução do sistema | Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema. |
| Apêndices | Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados. |
| Índice | Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes. |

Fonte: Sommerville (2016).

listadas. O Quadro 2.3 fornece um exemplo de caso de uso com a descrição de um projeto de uma seguradora para carros (COCKBURN, 2001).

O caso de uso é composto por cenários. No Quadro 2.3, o conjunto de etapas (1 a 5) corresponde a um cenário principal (fluxo básico). São caminhos esperados na execução de

uma determinada funcionalidade. Casos como extensões para tratamento de erros, são considerados como outro cenário (fluxo alternativo) e normalmente recebem uma identificação combinando o número relativo à etapa do cenário principal e uma letra em ordem crescente, como no exemplo do Quadro 2.3, a etapa 1a indica o caminho alternativo caso não seja executada com sucesso a etapa 1 do fluxo principal - "*O requerente apresenta a reclamação com dados comprovativos*" (COHN, 2004b).

Quadro 2.3: Exemplo de um caso de uso.

| **1 - Receber o pagamento por acidente de carro** | |
|---|---|
| Objetivo: | Receber o pagamento da seguradora "MyInsCo" |
| Nível de meta: | estratégico |
| Ator primário: | O requerente |
| Fluxo básico:<br>1. O requerente apresenta a reclamação com dados comprovativos<br>2. A seguradora verifica se o requerente possui uma apólice válida<br>3. A seguradora designa um agente para examinar o caso<br>4. O agente verifica se todos os detalhes estão de acordo com as diretrizes da política<br>5. A seguradora paga o requerente | |
| Fluxo alternativo:<br>1a. Os dados enviados estão incompletos:<br>1a1. A seguradora solicita informações ausentes<br>1a2. O requerente fornece informações em falta<br>2a. O requerente não possui uma política válida:<br>2a1. A seguradora recusa a reclamação, notifica o requerente, registra tudo isso, encerra o processo.<br>4a. O acidente viola as diretrizes básicas da política:<br>4a1. A seguradora recusa a reclamação, notifica o requerente, registra tudo isso, encerra o processo.<br>4b. O acidente viola algumas diretrizes menores da política:<br>4b1. A seguradora inicia negociação com o requerente quanto ao grau de pagamento para ser feito. | |

Fonte: Adaptado de Cockburn (2001).

O outro modelo utilizado para documentar os requisitos é a estória de usuário. As estórias de usuário são sentenças simples que representam uma funcionalidade de valor para um usuário do software. Desse modo, são escritas conforme a perspectiva do usuário que deseja a nova funcionalidade. Por esse motivo, normalmente as estórias são criadas pelos próprios usuários, visando garantir sua eficiência (COHN, 2004b). Para manter um padrão para a criação das estórias, alguns formatos foram propostos, como apresentado por Cohn (2008):

"Como <**tipo de usuário**>, eu quero <**objetivo da estória**> para que <**razão**>"

O "*tipo de usuário*" define quem irá executar a ação. O "*objetivo da estória*" indica essa ação, que é o desejo do usuário. O campo "*razão*" é opcional, ele representa qual é o resultado

esperado. Independentemente do formato de estrutura da estória, ela deve estimular a conversação do cliente e a equipe (MUNIZ et al., 2020).

As estórias devem ser descrições curtas e objetivas. Para o caso em que a estória é relativamente extensa, ela pode ser dividida em novas estórias menores. Além disso, as estórias devem atender a algumas exigências como serem independentes, negociáveis, testáveis e estimáveis (COHN, 2004b). O Quadro 2.4 apresenta alguns exemplos de estórias com descrições de funcionalidades da versão inicial da plataforma Scrum Alliance.

Quadro 2.4: Exemplos de estórias de usuário.

| **Estórias de usuário** |
|---|
| Como instrutor, quero que meu perfil liste minhas próximas aulas e inclua um link para uma página detalhada sobre cada uma, para que os participantes em potencial possam encontrar meus cursos. |
| Como membro do site, posso visualizar os perfis de outros membros para encontrar outras pessoas com as quais possa me conectar. |
| Como membro do site, posso marcar meu endereço de e-mail como privado, mesmo que o restante do meu perfil não seja para que ninguém possa entrar em contato comigo. |
| Como membro do site, posso enviar um e-mail a qualquer membro por meio de um formulário para que possamos nos conectar. |
| Como administrador do site, posso editar o perfil de qualquer membro do site para corrigir os problemas dos membros. |

Fonte: Adaptado de Cohn (2004a) (tradução da autora).

Embora os casos de uso e as estórias de usuários agreguem valor ao negócio, Cohn (2004b) aponta algumas diferenças entre esses dois modelos de documentação de requisitos. A estória de usuário é similar ao cenário do caso de uso (principal ou secundário), no entanto, devido às restrições de tamanho impostas no formato de estória de usuário, os casos de uso abrangem um escopo maior que a estória de usuário. Nesse sentido, as descrições dos casos de uso estão sujeitas a introduzir detalhes sobre a interface do usuário, ainda que haja recomendações para evitar isso (COCKBURN, 2001; ADOLPH et al., 2003). Outro ponto é que os casos de uso são estáveis, eles são mantidos durante todo o desenvolvimento e a manutenção do software. As estórias, em contrapartida, permanecem apenas até sua implementação. De maneira geral, esses dois formatos são empregados para propósitos diferentes (DAVIES, 2001). Enquanto os casos de uso servem como subsídios para outros diagramas da UML, com a documentação dos acordos feitos entre os envolvidos no projeto, as estórias têm como objetivo facilitar o planejamento do projeto e servir como instrumento para expor e discutir as necessidades dos usuários.

As representações dos requisitos devem apresentar as informações de forma clara. As descrições do software devem ser de fatos objetivos e não sobre opiniões pessoais (FERNANDES; MACHADO, 2017). Geralmente os requisitos são escritos na linguagem natural (o meio de comunicação utilizado pelos seres humanos). Esse tipo de linguagem proporciona um entendimento comum e de forma intuitiva na comunicação da equipe (PALDÊS et al., 2016; BARCELOS; PENTEADO, 2017). Todavia, sua utilização pode implicar em inconsistência, incompletude e ambiguidade (SILVA, 2014), provocando problemas de falta de comunicação e compreensão dos requisitos. Os reparos desses problemas podem ser muito dispendiosos.

### 2.1.2 Características de qualidade do documento de requisitos

O padrão IEEE 830 (IEEE, 1998) descreve algumas práticas para prover uma documentação de qualidade e define quais as características que a especificação de software deve apresentar, como ser não ambígua, ser completa e ser consistente.

***Não ambígua.*** A especificação de requisitos não é ambígua nos casos em que cada requisito tem apenas uma única interpretação. Assim, para descrever as características do software é recomendado utilizar um termo único (IEEE, 1998). A ambiguidade é classificada em diferentes níveis tais como lexical, sintática/estrutural e semântica (SANDHU; SIKKA, 2015). O tipo de ambiguidade lexical ocorre quando o termo pode ter diferentes significados, por exemplo, a palavra *pilha*, que pode indicar um objeto de fonte energética ou um conjunto de objetos sobrepostos. A ambiguidade sintática sucede de erros como a utilização de condições lógicas (“e” e “ou”) e problemas com pontuações e resultam em sentenças com diferentes sentidos (ROJAS; SLIESARIEVA, 2010; SANDHU; SIKKA, 2015; SABRIYE; ZAINON, 2017), por exemplo:

*Professores e alunos autorizados* podem acessar a plataforma.

Essa sentença traz duas possíveis interpretações: que apenas os alunos devem ser autorizados para acessar a plataforma; ou que tanto os alunos como os professores devem ser autorizados. De outro modo, a ambiguidade semântica aparece em casos que a sentença pode ter várias interpretações, pelo fato dos pronomes da sentença terem diversos antecedentes (CHIERCHIA, 2003), por exemplo:

O professor submete a prova do aluno *com sua própria identificação* no sistema.

O exemplo mostra dois entendimentos sobre o conceito de *identificação* que pode ser a identificação do aluno ou do professor.

***Completa.*** A especificação é completa se: incluir todos os requisitos do software, sejam a respeito da funcionalidade, restrições ou interfaces; definir todas as entradas e saídas do software; e descrever todas as referências, termos e medidas utilizadas no documento (IEEE, 1998). Para Eckhardt et al. (2016), a incompletude é classificada em dois níveis, sendo quando existe a ausência de algum requisito na documentação ou quando existe a ausência de informação em um requisito, por exemplo, "Exportar o relatório". Nesse exemplo, mostra-se a ausência do sujeito que deve exportar o relatório. Esse tipo de problema torna o requisito vago e dependente de informações adicionais.

***Consistente.*** A especificação é consistente quando os requisitos individuais não estão em conflito. Existem diferentes tipos de conflitos possíveis (IEEE, 1998), como a divergência na descrição dos atributos, por exemplo:

(1) As avaliações podem ser filtradas *pelo registro* do aluno.
(2) O professor pode filtrar as avaliações *pelo nome* do aluno.

Esses requisitos apresentam a contradição no atributo do aluno, no requisito (1) entende-se que as avaliações podem ser filtradas apenas pelo registro do aluno e isso se contradiz no requisito (2), que descreve que a avaliação do aluno é filtrada pelo nome. Outro tipo de conflito é com diferenças lógicas ou temporais entre as ações no software, como:

(3) O usuário remove seu perfil.
(4) O usuário desativa seu perfil.

Nesse caso, as ações "remover" e "desativar" são contradições, pois no requisito (3) o usuário tem seu perfil permanentemente excluído e no caso (4) é apenas temporário. Existem ainda conflitos nos termos utilizados nos requisitos. Devido ao fato de se utilizar diferentes terminologias para indicar o mesmo objeto, pode-se provocar requisitos redundantes, como por exemplo:

(5) O usuário visualiza o *painel de gestão.*

(6) O usuário visualiza a *área administrativa.*

Nesse contexto, o "painel de gestão" e a "área administrativa" indicam o mesmo ambiente e, assim, torna o requisito redundante.

Esses problemas estão relacionados com diferentes aspectos do significado do texto. De acordo com Loebner (2014), "semântica é o estudo dos significados das expressões linguísticas, simples ou complexas, tomadas isoladamente [...]" (tradução da autora). Nesse sentido, a pouca atenção dada a esses problemas semânticos na documentação de requisitos pode interferir na interpretação dos requisitos como falhas na comunicação e compreensão dos requisitos. Nesse contexto, os problemas relacionados com falhas nessas características de qualidade têm sido investigados nos últimos anos (FANTECHI; GNESI; SEMINI, 2017; FERRARI; DELL'ORLETTA et al., 2017).

## 2.2 Estado da arte: mitigação de problemas em requisitos

Esta seção apresenta estudos que contribuem para mitigar problemas causados pelo uso da linguagem natural. Esses estudos foram obtidos por meio de duas buscas na literatura, seguindo alguns princípios da abordagem *Preferred Reporting Items for Systematic reviews and Meta-Analyses* (PRISMA). Essa abordagem auxilia na elaboração de protocolos de revisões com a apresentação do fluxo de informações mediante cada etapa da revisão de literatura (MOHER et al., 2009).

De acordo com Gil (2002), a revisão de literatura "é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos [...]". A primeira busca consistiu em uma revisão geral para identificar abordagens que lidam com requisitos escritos em linguagem natural. O protocolo dessa revisão de literatura é apresentado no Apêndice A, com os conceitos e os termos definidos para a pesquisa, os critérios de inclusão e de exclusão dos estudos, a construção da *string* de busca, as bases de dados bibliográficos selecionadas e o diagrama do processo de seleção de estudos. Com base na primeira pesquisa da literatura foram descobertos novos termos de busca, os quais foram extraídos dos estudos analisados. Esses termos foram incrementados em uma segunda revisão para uma busca mais específica de artigos (Apêndice B).

Por meio dessa busca, foi possível identificar que as propostas de soluções estavam voltadas para três etapas: estruturação, validação e representação de requisitos. Foram identificados estudos no período de 2014 a 2021 (8 anos). Foram selecionados os estudos completos que estivessem relacionados com cada uma dessas etapas para a mitigação de problemas provocados pelo uso da linguagem natural. Além dos problemas semânticos de inconsistência, incompletude e ambiguidade, alguns estudos apresentaram soluções para outros tipos de problemas como redundância, ilegibilidade, limitação para o processamento automático e a falta de padronização do texto. A seguir, uma seção é dedicada a trabalhos correlatos de cada uma das três tarefas.

### 2.2.1 Estruturação de requisitos

Neste estudo, considerou-se a estruturação como uma tarefa que visa auxiliar na transição dos requisitos expressos em uma linguagem simples para uma especificação padronizada. Nesse sentido, os estudos estão focados em soluções para a organização dos conceitos extraídos dos requisitos.

Algumas pesquisas da literatura investigaram a LNC como uma estruturação intermediária. Essa linguagem pode ser compreendida como um subconjunto da linguagem natural (CASTRO; BEZERRA; HIRATA, 2015) e tem o objetivo de escrever requisitos que sejam sintaticamente corretos e uniformes (MARKO et al., 2015).

A aplicação da LNC facilita processos automáticos. Castro, Bezerra e Hirata (2015) aplicaram a LNC para um processo automatizado. Os autores representaram os requisitos estruturados com LNC em descrições no formato *Extensible Markup Language* (XML) - uma linguagem de marcação. A implementação teve foco em projetos de desenvolvimento de software críticos (setor aeronáutico). Sua avaliação em quarenta e seis requisitos de software reais mostrou que a abordagem possibilitou uma redução de esforço humano e tempo para a especificação dos requisitos. Marko et al. (2015) utilizaram a LNC em uma abordagem integrada com outras ferramentas. Os autores executaram uma prova de conceito para investigar a proposta e concluíram que as descrições extraídas dos requisitos suportaram o processamento automático. Skersys, Danenas e Butleris (2018) aplicaram a LNC em um modelo visando manter a expressividade da linguagem natural e ser interpretável por computadores. Os autores aplicaram a transformação de modelo a modelo, em inglês - *Model-to-model* (M2M), em conceitos extraídos de cenários de diagramas de casos de uso de

forma automática e semiautomática e avaliaram a precisão da transformação em um experimento. As transformações automáticas apresentaram uma redução de tempo, no entanto, a precisão foi menor em relação às semiautomáticas. Selway, Grossmann et al. (2013) estruturaram os requisitos em LNC para a representação em um modelo conceitual. Os autores mostraram a aplicação dessa abordagem e relataram melhorias no nível de automatização.

A estruturação da LNC fornece melhorias em problemas, como a ambiguidade. Lebeaupin, Rauzy e Roussel (2017) empregaram a LNC fundamentada em lógica e a parte de sintaxe do texto para reduzir a ambiguidade nos requisitos. Contudo, os autores não a avaliaram na prática. Bajwa, Bordbar e Lee (2014) exploraram a LNC para resolver a sintaxe complexa das declarações da *Object Constraint Language* (OCL), a qual consiste em uma das linguagens da *Unified Modeling Language* (UML). Os autores automatizaram essa aplicação. Foi analisada a precisão e a eficácia dessa proposta. A pesquisa revelou que os resultados da eficácia eram melhores em comparação com outras técnicas relacionadas. Apaza et al. (2018) apresentaram uma abordagem baseada em regras para descrever os requisitos utilizando como base uma sintaxe controlada. Os autores adaptaram sua aplicação para o idioma espanhol e implantaram essa sintaxe em uma ferramenta, com o intuito de guiar o processo de escrita dos requisitos. A abordagem foi avaliada com a execução de estudos de casos de domínios diferentes. Os participantes da pesquisa (clientes, analistas e desenvolvedores) avaliaram as descrições dos requisitos criados. O principal resultado obtido foi a redução de ambiguidades. Além disso, a LNC mostrou-se capaz de lidar com a escrita dos requisitos em diferentes idiomas.

Ashfaq e Bajwa (2021) lidaram com o problema de ambiguidade aplicando a LNC independente de domínio. A LNC empregada foi baseada em *Semantics of Business Vocabulary and Rules* (SBVR), isto é, em um vocabulário e regras do negócio. Os autores buscaram ter requisitos sintaticamente claros e semanticamente consistentes, os quais são representados no formato XML. Como relatado pelos os autores, esse formato é flexível para ser convertido em outros modelos, devido à sua facilidade no processamento por máquinas. Nos experimentos realizados, a abordagem proposta atingiu 96,6% de precisão e 96% de revocação. Os resultados confirmam o desempenho eficiente da ferramenta em termos de correção de ambiguidade. Veizaga et al. (2021) exploraram a LNC no domínio financeiro e descobriram em um estudo de caso que 88% das declarações de requisitos funcionais foram capturadas pela ferramenta proposta.

A aplicação de padrões nos requisitos obtém resultados satisfatórios para a padronização de requisitos e contribuiu para a mitigação de problemas relacionados com inconsistência, incompletude e redundância. Um dos padrões aplicados como solução foi o *boilerplate*, o qual pode ser compreendido como um *template*, isto é, elementos de sintaxe fixos (FARFELEDER et al., 2011). Para Fraga et al. (2015) esses padrões são áreas predefinidas e sequenciais que devem ser utilizadas com termos e valores específicos. Mahmud, Seceleanu e Ljungkrantz (2016) utilizaram *boilerplates* em um conjunto de ferramentas para o contexto de sistemas embarcados automotivos. Os autores propuseram uma função automática de verificação de inconsistência dos requisitos. A verificação de requisitos contribuiu com a identificação de possíveis erros lógicos. Esse conjunto de ferramentas foi avaliado em referência ao seu uso em uma função para a segurança automotiva. Alguns engenheiros participaram dessa avaliação, por meio de questionários. A linguagem aplicada nos requisitos mostrou ser expressiva.

Alguns estudos desenvolveram novos formatos de padrões. Barcelos e Penteado (2017) contribuíram com um catálogo de padrões e um suporte computacional. Para sua avaliação, alguns estudantes responderam um questionário a respeito da aplicação desses padrões em requisitos de estudos de casos. Os resultados mostraram que os requisitos foram completos. Ainda, os estudantes apontaram a facilidade em utilizar a ferramenta. Todavia, o trabalho não se estende a requisitos não funcionais. Entretanto, outros autores colaboraram para reparar essa dificuldade. Eckhardt et al. (2016) desenvolveram alguns padrões para serem utilizados com atributos de desempenho. Os autores avaliaram a aplicabilidade e completude dos requisitos com a implantação desses padrões em um estudo de caso. O estudo mostrou que o *framework* pode ser utilizado para detectar incompletude e mostram que sua aplicação é viável na prática. Jue et al. (2019) utilizaram a sintaxe do *JavaScript Object Notation* (Json) para a transformação automática dos requisitos em modelos estruturados. O método foi capaz de processar os requisitos em um estudo de caso, porém, foi avaliado em um contexto específico de sistemas ciberfísicos.

Outro estudo apresenta novos padrões voltados para casos de uso. Tiwari, Ameta e Banerjee (2019) propuseram alguns padrões e técnicas de PLN para a geração de casos de uso. A abordagem é composta pela análise de qualidade com base em uma lista de verificação. A aplicação desses padrões revelou casos de uso consistentes e não redundantes.

Além do uso de *templates*, uma solução é a padronização dos termos usados nos requisitos. Sneed e Verhoef (2013) lidaram com o comportamento de serviços com base nos

termos principais do domínio. A abordagem desse estudo visa fornecer um documento que mantém a expressividade presente na linguagem natural para os especialistas do negócio, mas que também pode ser submetida a um processamento automático. Um estudo de caso foi conduzido com especificações de serviços no ambiente *World Wide Web* (Web). Os resultados principais indicaram que a abordagem identificou problemas que não satisfaziam os requisitos. Dorigan e Barros (2014) apresentaram um processo que visa padronizar os termos com a reutilização de palavras já validadas no contexto da aplicação. O estudo conduz uma discussão com a comparação do modelo com outros trabalhos identificados na literatura e apresenta os benefícios esperados pela validação dos requisitos como aumento na qualidade, rastreabilidade, redução de redundâncias e inconsistências.

A análise semântica pode alcançar bom desempenho para a estruturação dos requisitos e foi aplicada para sanar problemas de ilegibilidade nos requisitos. Wang e Zhang (2016) estruturaram as informações obtidas da extração dos requisitos com a análise semântica. A abordagem automática foi aplicada em requisitos e apresentou uma boa precisão, mas não se estende a requisitos não funcionais. Um estudo que contribui com essa limitação foi realizado por Selway, Mayer e Stumptner (2014). Nessa pesquisa foram considerados na análise semântica tanto os requisitos funcionais como também os não funcionais.

### 2.2.2 Validação de requisitos

Este estudo empregou a mesma definição de validação de requisitos adotada por Silva (2015) e sugerida por Pohl (2010), que afirma que a validação é uma tarefa com o objetivo de detectar erros, como exemplo, inconsistências, ambiguidades e incompletude. Nesse contexto, nesta subseção são apresentadas estratégias para a detecção de problemas.

A maioria das soluções propostas na literatura para a detecção de inconsistência, incompletude e ambiguidade nos requisitos de forma automática faz o uso de técnicas de inteligência artificial, como o aprendizado de máquina. Hayrapetian e Raje (2018) utilizaram a rede neural para realizar a classificação semântica na avaliação de incompletude e ambiguidade nos requisitos não funcionais. O modelo treinado mostrou que os melhores resultados de predição foram para analisar o nível de completude dos requisitos. Gramajo, Ballejos e Ale (2021) exploraram a rede neural para avaliar os requisitos com base nas propriedades de qualidade apresentadas no padrão IEEE (2018). Os resultados da classificação mostraram uma precisão de 75%. Atoum (2019) combinou os métodos de aprendizado de

máquina e um modelo de similaridade semântica profundo, com o objetivo de identificar semelhanças e defeitos entre os requisitos. Entretanto, o desempenho do método é limitado pela falta de disponibilidade de conjuntos de dados para o treinamento do modelo. Em outro estudo, a precisão obtida de uma abordagem baseada na análise de regressão foi suficiente para a aplicação prática de revisões de requisitos (ANTINYAN; STARON, 2017). Essa abordagem apresentou um valor entre 73% e 80% de concordância com uma classificação manual realizada por engenheiros de software.

Alguns modelos de classificação foram baseados em critérios de qualidades realizados por especialistas. Moreno et al. (2020) apresentaram um método automático para avaliar a qualidade de requisitos individuais. O diferencial do método consiste na obtenção de critérios de qualidade feitos por especialistas para o treinamento do modelo de classificação. O resultado da avaliação do modelo mostrou a capacidade para a avaliação de requisitos no domínio aeroespacial. Parra et al. (2015) apresentaram uma abordagem automática utilizando aprendizado de máquina. Em um estudo de caso, a abordagem mostrou ser capaz de avaliar a qualidade do requisito com uma média de 86% de precisão.

Osman e Zaharin (2018) combinaram técnicas como aprendizado de máquina e mineração de texto em uma ferramenta. Os autores avaliaram diferentes algoritmos para a classificação e desenvolveram um protótipo com a aplicação do algoritmo *Random Forest*. Engenheiros e analistas de software avaliaram esse protótipo e descreveram suas percepções. No geral, os participantes afirmaram que a ferramenta fornece informações relevantes e é útil para a detecção de especificações ambíguas em requisitos no idioma malaio.

A técnica de agrupamento proporciona a verificação de deficiências como redundância nos requisitos. Mezghani, Kang e Sèdes (2018) utilizaram o agrupamento com o objetivo de encontrar requisitos redundantes e inconsistentes. Especificamente, os autores aplicaram o algoritmo k-*means* e validaram o seu uso em especificações industriais. Os autores declararam que obtiveram a detecção de problemas nos requisitos. Em um estudo semelhante, Mokammel et al. (2018) implantaram o agrupamento e análise de similaridade de requisitos em uma ferramenta baseada em regras, a qual gera um gráfico que representa as relações semânticas entre os requisitos. Os gráficos formados ajudaram a identificar contradições nos requisitos, como observado em estudos de caso. Os resultados da avaliação da ferramenta apontaram a descoberta de contradições nos requisitos. Misra (2016) aplicou a análise de agrupamento para a desambiguação de termos relacionados com entidades em

requisitos. Essa abordagem forneceu uma unificação de termos. A utilização dessa técnica foi considerada viável por meio de sua implantação em um protótipo.

Dalpiaz, Schalk et al. (2019) combinaram técnicas para a extração de conceitos e similaridade semântica para a construção de um diagrama, semelhante ao Venn, para verificação de ambiguidade. Os autores construíram uma ferramenta para combinar essas técnicas e conduziram um quase-experimento com estudantes para avaliar sua eficácia em comparação com a revisão de ambiguidade de forma manual. Os resultados obtidos por meio de uma ferramenta indicam que o diagrama apoia a descoberta de termos ambíguos.

Algumas pesquisas abordam os recursos de PLN, combinadas com outras técnicas. Bhatia, Kumar e Beniwal (2016) usaram técnicas de PLN e Web Semântica para o desenvolvimento de uma ontologia para detectar ambiguidades. Os autores definiram um *framework* baseado nessa ontologia, a qual visa representar o conhecimento do domínio extraído das especificações. Com o mesmo intuito, Sabriye e Zainon (2017) construíram um *framework* fundamentado na técnica de classes gramaticais. Para avaliar a proposta, os autores construíram um protótipo baseado em regras e técnicas de PLN. Silva (2015) propôs uma nova ferramenta automática que consiste em um *framework* genérico nomeado como SpecQua. O autor utilizou recursos lexicais. O autor relata que os resultados preliminares mostraram que o *framework* pode prever problemas nos requisitos. Essas abordagens ainda requerem a avaliação em um contexto real.

Lucassen, Dalpiaz et al. (2016) contribuíram com uma ferramenta para avaliar e melhorar a qualidade em estórias de usuário. Esse trabalho é uma extensão de um trabalho anterior dos autores, no qual foi proposto o *framework Quality User Story* (QUS). O QUS consiste em um conjunto de treze critérios que avaliam a qualidade das descrições nas estórias de usuário. Em alguns estudos de caso, a ferramenta atingiu uma média de 92,1% de revocação e 77,4% de precisão para os conjuntos maiores de dados e revelou ajustes que contribuem para o aumento desse resultado. Singh (2019) empregou o PLN, análise semântica e mineração baseada em teoria dos grafos nos requisitos para verificar falhas, como redundância. Em um experimento com a implementação dessa abordagem, os autores conseguiram obter a detecção de falhas nos requisitos testados.

Técnicas utilizam indicadores como instrumentos para analisar o nível de ilegibilidade dos requisitos. Nesse contexto, Thitisathienkul e Prompoon (2015) propuseram um método de avaliação abrangendo as características e estrutura do documento e observaram que os valores obtidos de um experimento se mostraram próximos aos resultados esperados pelos

especialistas. Nesse método, as decisões humanas ainda são empregadas para a avaliação da qualidade. Rine e Fraga (2015) definiram algumas métricas de complexidade e constataram em um estudo de caso que é possível medir as propriedades no conteúdo das especificações de software. As métricas são fundamentadas em sintagmas (unidades sintáticas) nominais e não verbais. Femmer et al. (2017) apresentaram uma abordagem que é constituída por indicadores de violações nas propriedades de qualidade. A detecção automática mostrou nos estudos de casos uma média de 59% em termos de precisão e 82% em revocação com variação. Véras et al. (2015) definiram um *benchmark* para especificações de requisitos no contexto de aplicações espaciais. Em sua avaliação, foi realizado um estudo de caso, o qual apresentou resultados que indicaram a contribuição na detecção de pontos fracos da documentação de requisitos. Laue, Koop e Gruhn (2016) apresentaram um catálogo de indicadores de alguns problemas na tarefa de modelagem de *Business Process Models* (BPM) com técnicas de PLN. Os autores desenvolveram um protótipo e aplicaram em modelos no idioma alemão. Foram apontados os problemas nesses modelos, como por exemplo testes e verbos vagos.

A literatura dispõe de alguns algoritmos específicos para a revisão de problemas do tipo semântico. Sanne et al. (2016) criaram um algoritmo com o objetivo de identificar atores em processos de negócios. Os resultados de sua execução relataram uma precisão de 56% e 87% de revocação em sua execução, no entanto, as sentenças em voz passiva podem interferir em sua aplicação. Pittke, Leopold e Mendling (2015) desenvolveram algoritmos para a detecção de ambiguidades de característica lexical e verificaram uma redução de ambiguidade nos requisitos. Khtira, Benlarabi e Asri (2015) aplicaram um algoritmo para a identificação de duplicação de recursos no contexto de linha de produtos, no entanto, os requisitos devem estar no formato XML. A solução proposta foi capaz de detectar recursos duplicados em um teste. Mu et al. (2013) desenvolveram uma estrutura lógica para avaliar a qualidade dos requisitos, com a identificação e gerenciamento de requisitos que sejam inconsistentes, incompletos e redundantes. Os autores elaboraram três algoritmos para lidar com esse tipo de requisitos.

Osama et al. (2020) buscaram uma estratégia para detectar a ambiguidade do tipo sintática. Os autores implantaram algoritmos baseados em uma classificação de pontuação obtida de um PLN. Os resultados apontaram uma precisão de 65% e 99% de revocação para a identificação de ambiguidade. Guo, Zhang e Lian (2021) projetaram um algoritmo embasado em análise semântica para a descoberta de conflitos em requisitos funcionais. O algoritmo alcançou uma

média de 83,8% na identificação de conflitos. Uma das razões desse resultado é a análise do requisito em composições semânticas mais refinadas, pois o algoritmo lida com os requisitos no formato de tuplas.

Outra estratégia para detectar ambiguidade e inconsistência nos requisitos é a criação de heurísticas. Ferrari, Spagnolo et al. (2019) criaram uma ferramenta Web com a implantação de um conjunto de regras linguísticas. A ferramenta ainda requer uma avaliação. Phanthanithilerd e Prompoon (2015) desenvolveram um método de verificação de alguns atributos de qualidade extraídos do padrão IEEE 830 (IEEE, 1998). Os autores definiram algumas regras a partir dos relacionamentos entre os componentes dos requisitos. Essas regras foram implementadas em uma ferramenta automatizada visando avaliar sua aplicação em diferentes tipos de requisitos e descrições extraídas de modelos. Os resultados dessa execução revelaram que a verificação pode identificar defeitos de qualidade nos requisitos. Em outro estudo, Phanthanithilerd e Prompoon (2016) propuseram regras a partir dos relacionamentos entre os componentes dos requisitos e modelos da UML. Pi et al. (2019) investigaram a heurística hierárquica para identificar inconsistências em propriedades temporais e obtiveram a melhoria na precisão em experimentos. Para isso, os autores formalizaram os requisitos em fórmulas de Lógica Temporal Linear (LTL) utilizando PLN e regras de extração. Essa abordagem foi implementada em um protótipo. Em uma pesquisa similar, Yan, Cheng e Chai (2015) forneceram algumas heurísticas para garantir a consistência dos requisitos expressos em diferentes representações. Embora os resultados de três estudos de caso com a implementação da abordagem sejam positivos, os autores apontam, porém, a necessidade de uma análise semântica para a validação.

O estudo de Escalona et al. (2013) confirma essa deficiência para abranger a detecção de falhas a nível semântico. Os autores definiram uma técnica baseada em regras para a identificação de inconsistência fundamentada em modelos de requisitos no contexto de aplicativos da Web. Entretanto, não foram consideradas inconsistências semânticas. Foi conduzido um estudo de caso e obteve-se os mesmos resultados que a análise original, porém, com uma redução de tempo. Os participantes do estudo revelaram que essa técnica pode ser difícil e complexa.

Métodos de avaliação fundamentados nos atributos de qualidade fornecem o apoio para a detecção de falhas. Matsumoto, Shirai e Ohnishi (2017) utilizaram um modelo de quadro de requisitos não funcionais e contribuíram com a verificação de problemas relacionados a

alguns atributos de qualidade. O foco do estudo foi para requisitos associados ao tempo de resposta e usabilidade, visando proporcionar a detecção de redundância, ambiguidade, inconsistência e incompletude. Os autores utilizaram um quadro de requisitos estendido e desenvolveram um protótipo para a detecção desses erros. Em outro estudo, Yamada, Omori e Ohnishi (2019) adaptaram esse quadro de requisitos estendido para avaliar os requisitos de confiabilidade em diferentes idiomas. O resultado dessa aplicação é um relatório com a descrição dos erros detectados. O método apontou os requisitos redundantes em documentos fornecidos pelo governo japonês. Entretanto, foi possível observar a limitação para analisar requisitos compostos com frases múltiplas. Sarmiento, Leite e Almentero (2015) definiram um modelo de catálogo de padrões considerando as propriedades de correção, consistência e integridade pela estrutura de requisitos não funcionais. Uma exemplificação foi realizada dessa abordagem, para a qual os requisitos devem estar no formato adequado, ou seja, aplicar as regras estabelecidas. Kamalrudin, Hosking e Grundy (2017) também usaram um modelo de padrões e apresentaram a ferramenta MaramaAIC para fornecer o auxílio visual com o destaque de inconsistência, incorreção e incompletude dos requisitos. Três estudos de casos foram conduzidos e verificou-se que a precisão e o desempenho foram melhores quando comparados com a extração manual. Foi possível observar que a falta de estruturação dos requisitos interferiu nos resultados. Além disso, como citado pelos autores, as deficiências na gramática podem resultar em interpretações inconsistentes. A ferramenta é limitada para lidar com vários requisitos.

### 2.2.3 Representação de requisitos

Como solução para reparar problemas referentes ao uso da linguagem natural, algumas pesquisas propuseram novas formas de apresentação dos requisitos. Além disso, outras pesquisas adaptaram modelos existentes, como os casos de uso.

A representação de requisitos em modelos formais contribui para reduzir erros de inconsistência, ambiguidade, incompletude e melhorar a elegibilidade dos requisitos. Naumchev e Meyer (2017) criaram a representação *Seamless requirements*, a qual é composta por uma linguagem natural e componentes formais. Madala, Do e Aceituna (2018) apresentaram outro modelo em componentes para reduzir a ambiguidade nos requisitos. Em um estudo de caso, o modelo mostrou que pode diminuir os esforços manuais e de tempo.

Wan et al. (2016) propuseram a representação em modelos formais de System Modeling Language (SYsML) (FRIEDENTHAL; MOORE; STEINER, 2014). A aplicação dessa abordagem foi comparada com trabalhos relacionados e se destacou por lidar com comportamentos dinâmicos e suportar o processamento de um texto não estruturado.

Para formalizar e padronizar os requisitos, algumas pesquisas aplicam modelos lógicos. Sharma e Biswas (2015) formalizaram os requisitos em representações lógicas, por meio de um modelo nomeado como Lógica Cortês. A solução proposta incorpora a padronização dos requisitos para serem formalizados de forma automática. Os resultados de um estudo de caso mostraram que a formalização pode melhorar a tarefa de análise de requisitos. No entanto, o processo implantado não considera os problemas que pode haver, como a ambiguidade.

Zaman, Nadeem e Sindhu (2020) utilizaram a estrutura Kripke (modelo de representação) e especificações de Lógica Temporal Linear (LTL) em descrições de casos de uso. LTL proporciona a formalização de requisitos temporais. Para gerar essas representações, os analistas devem usar um modelo para a organização das informações. Os autores validaram essa proposta em uma exemplificação com a implementação de uma ferramenta. Os modelos gerados validaram a formalização empregada nos requisitos. Li, Hayes e Truszczyński (2015) construíram um método para a transformação da linguagem natural para uma formalização utilizando teorias de LTL e avaliaram sua qualidade por métricas de precisão e pelas declarações dos participantes (estudantes) envolvidos no estudo. Os resultados dessas aplicações indicaram melhorias na geração de requisitos temporais.

Outro formato para a formalização é a ontologia. Diamantopoulos et al. (2017) estabeleceram uma nova forma para a anotação semântica de requisitos, visando à classificação dos conceitos para serem representados em uma ontologia fundamentada em casos de uso. Foi avaliado o desempenho da abordagem e observou-se que o módulo é eficaz e contribui para a automatização do processo na engenharia de software. Djilani e Khouri (2015) utilizaram uma ontologia multilíngue e a unificação do vocabulário em um cenário de desenvolvimento global. Essa abordagem foi avaliada com a aplicação de *benchmark* de ontologia, redes semânticas e uma documentação de requisitos. Os resultados provaram que a abordagem identificou mais relações entre os requisitos em comparação com outras abordagens alternativas.

Modelos de formalização estendem a aplicação para casos de uso. Couto, Ribeiro e Campos (2014) basearam-se em uma ontologia para a formalização das descrições de casos de

uso. Foi avaliada a capacidade da abordagem em um estudo de caso e obteve-se a formalização dos casos de uso. No entanto, não foram reparados outros problemas da linguagem natural, como a ausência de informações. Wong, Mit e Sidi (2016) formalizaram e estruturaram o modelo de casos de uso. Para isso, os autores utilizaram uma versão estendida do Método de Desenvolvimento de Viena (em inglês, *Vienna Development Method* - VDM) (BJØRNER, 1979). Embora o modelo possa ser facilmente formalizado, é necessária uma avaliação dessa proposta. Oliveira et al. (2015) descreveram um processo de transformação de grafos para a formalização dos requisitos em casos de uso, mantendo o significado do requisito expresso em linguagem natural. A partir dos resultados do experimento realizado, observou-se que o processo pode fornecer qualidade aos casos de uso.

Embora o formalismo incremente o nível de precisão dos requisitos, a sua interpretação é difícil. Para superar essa dificuldade, alguns estudos buscaram reparar problemas de interpretabilidade com modelos conceituais. Sagar e Abirami (2014) propuseram a automatização de modelos conceituais, a qual atingiu uma precisão superior a 81% em alguns estudos de caso. Lucassen, Robeer et al. (2017) avaliaram a ferramenta *Visual Narrator* para a geração automática de modelos extraídos de estórias de usuário. Os autores declararam que a proposta pode melhorar a precisão e comunicação dos requisitos e apoiar a detecção de problemas como dependências, inconsistências e ambiguidade nos requisitos. Quatro estudos de casos investigaram a precisão, revocação e acurácia da ferramenta em relação à aplicação manual. Os resultados indicam melhorias na precisão. No entanto, a ferramenta tem algumas limitações. Devido à falta de estruturação de requisitos, a ferramenta não suporta declarações de requisito complexas, por exemplo com conjunções coordenadas. Outra deficiência é para processar requisito com o sujeito indeterminado, nessa situação, ele não é atendido e perde-se informações. Além disso, a ferramenta converte os requisitos em formatos de triplas (sujeito, verbo, objeto) e não abrange alguns tipos de atributos.

Outra estratégia para facilitar o entendimento dos requisitos é a implantação de *mockups*, que consistem em protótipos gráficos. Rivero et al. (2019) descreveram o DataMock, o qual visa representar requisitos em *mockups* e diagramas de classes da UML, visando o reparo de inconsistências nos modelos de dados. As tarefas dessa abordagem são baseadas em técnicas obtidas da engenharia orientadas a modelos. A abordagem gerou modelos com melhor qualidade, rastreabilidade e facilitou a comunicação entre os integrantes da equipe de desenvolvimento em um experimento controlado com alunos. Nesse mesmo cenário, Hoyos e

Restrepo-Calle (2018) determinaram um esquema de prototipagem em uma linguagem natural restrita para a geração automática do código-fonte de uma aplicação Web. A solução implanta as características do Modelo de Processo de Negócios e Notação (em inglês, *Business Process Model and Notation* - BPMN) (WHITE, 2004). Os resultados em dois estudos de casos foram positivos e mostraram o rápido processamento para a geração do protótipo. Entretanto, de acordo com os autores, sua utilização pode ser complexa para a análise e difícil de compreender para usuários não-técnicos. Medeiros et al. (2020) criaram uma abordagem de especificação de requisitos composta por diferentes representações como a modelagem conceitual, *mockups* e restrições técnicas, proporcionando uma visão integrada dos requisitos. Alguns engenheiros de software avaliaram a aplicação da abordagem em projetos ágeis e constataram que a abordagem contribuiu com especificações mais legíveis e completas.

A representação por animação também pode facilitar na compreensão dos requisitos e reparar problemas de ambiguidade. Um modelo de animação foi realizado por Zainuddin e Liu (2015) para facilitar a comunicação dos requisitos. Os autores desenvolveram um protótipo para apoiar especificações informais. Para a avaliação da eficiência do protótipo, um estudo controlado foi realizado e mostrou-se que auxilia no processo de interpretação dos requisitos. No entanto, o modelo não realiza o tratamento do texto para obter uma melhor qualidade dos requisitos.

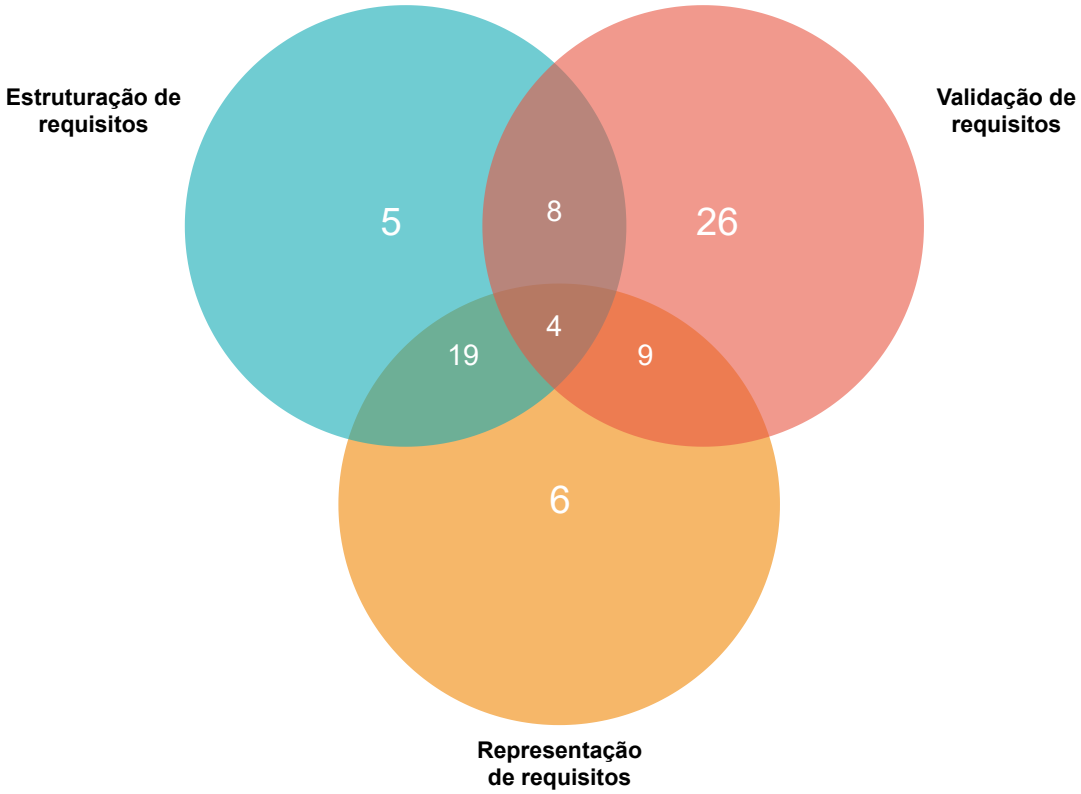
Requisitos de serviços também apresentam problemas com a aplicação de linguagem natural. Wang, Jiang e Wang (2016) detalharam uma nova modelagem centrada no fluxo de trabalho e validação de especificações de requisitos de serviços, visando à redução de problemas como inconsistência, incorreção e incompletude. Todavia, os requisitos devem seguir uma estruturação que não é fornecida pela proposta.

### 2.2.4 Combinação das etapas: estruturação, validação e representação de requisitos

A maioria das soluções propostas para mitigar problemas nas especificações de software está voltada para a automatização da etapa de validação, como observado no Apêndice C e no diagrama de Venn ilustrado na Figura 2.2, o qual representa os conjuntos dos estudos obtidos da revisão bibliográfica. Os estudos buscaram a detecção e avaliação de problemas, por meio de algoritmos, heurísticas, entre outros (Figura 2.3). No entanto, dentre os 47 estudos

associados à validação, apenas 12 realizaram a padronização dos requisitos. A falta de padronização do texto em linguagem natural pode comprometer a execução dessas soluções. Por outro lado, alguns estudos visaram a estruturação dos requisitos aplicando a LNC, padrões específicos e análise semântica. Entretanto, identificou-se a necessidade de mais investigações da aplicação da análise semântica, já que muitas abordagens estavam voltadas para a análise em nível sintático. Além disso, os resultados dessas soluções são afetados por problemas gramaticais, como o uso da voz passiva, no texto. Uma quantidade menor de estudos focaram na representação de requisitos. A formalização dos requisitos foi introduzida no formato de componentes e modelos. Outras abordagens de representação visaram facilitar a compreensão e comunicação de requisitos por meio de modelos conceituais, fluxos e protótipos gráficos.

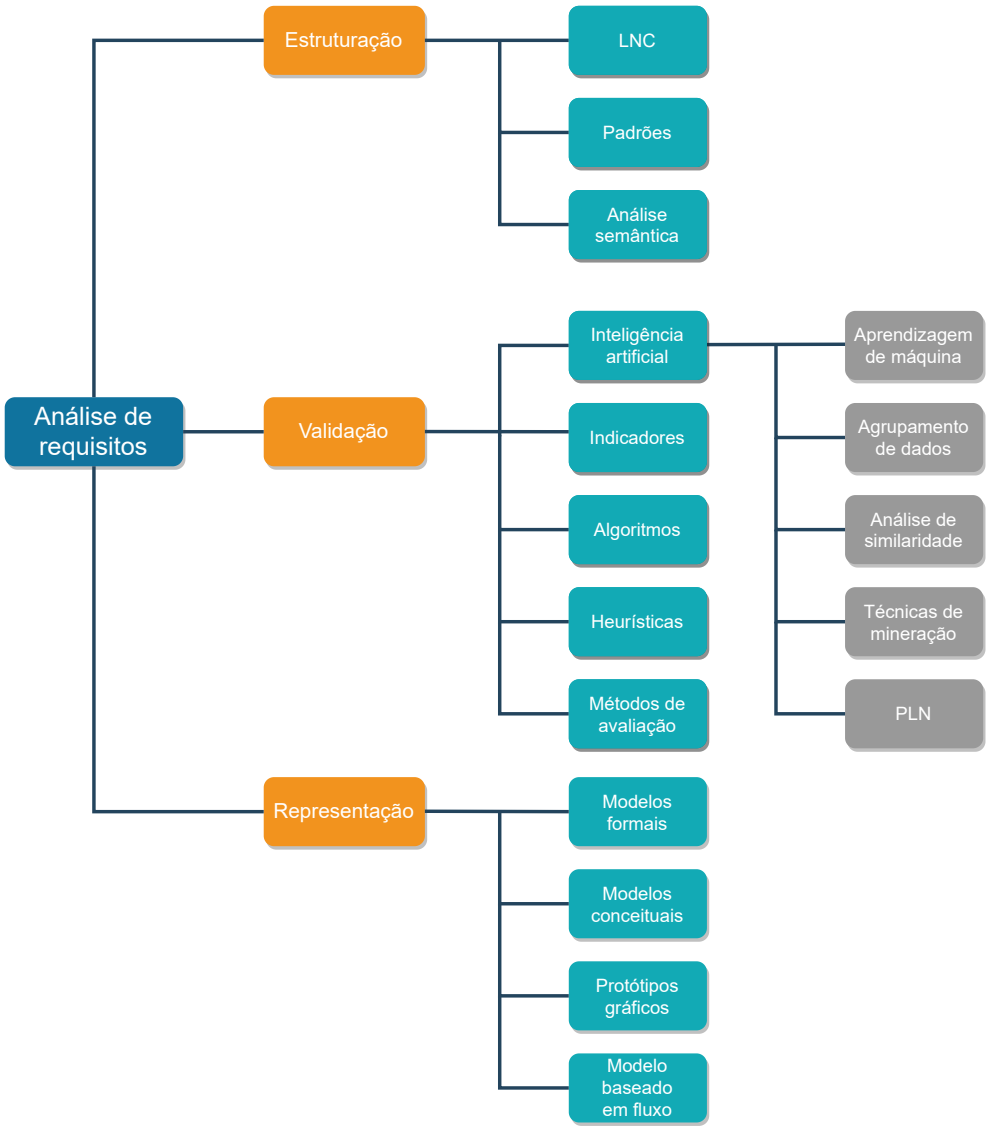
Figura 2.2: Diagrama de Venn dos estudos obtidos na revisão da literatura.




Fonte: Elaborado pela autora (2021).

Com base nessas descobertas da literatura, pode-se concluir que para obter melhores resultados na mitigação de problemas inerentes à língua natural é preciso a representação estruturada e validação de requisitos. Por causa da falta da combinação das três etapas do

Figura 2.3: Mapeamento das soluções propostas na literatura.




Fonte: Elaborado pela autora (2021).

processo de especificação de software (estruturação, validação e representação), os requisitos ainda são imprecisos.

Poucos estudos exploraram essa combinação das etapas, entretanto, os resultados são promissores. A validação em requisitos obtidos de uma representação estruturada tende a ser mais eficaz. Stachtiari et al. (2018) relataram que a combinação reduziu os esforços para os testes de validação no estágio final do ciclo do processo. Os autores automatizaram a representação dos requisitos em um modelo formal, o qual foi estruturado com *boilerplates* e com a validação antecipada baseada em um modelo de design. Soares e Moura (2015) introduziram uma nova metodologia para estruturação e validação baseada no padrão IEEE,

recursos de mineração de textos, padrões de requisitos e linguísticos. Ainda, o modelo de representação suporta diferentes formatos, como *mockups*. Sua aplicação contribui com a geração de requisitos mais completos, consistentes e a redução de ambiguidades. Reggio et al. (2018) definiram um método para especificações em casos de uso. O método é constituído por um conjunto de restrições para a estruturação do texto, uma lista de verificação para a detecção de problemas e a modelagem baseada em *mockups*. Os resultados da avaliação desse método confirmaram sua eficácia na melhoria da qualidade e compreensão dos requisitos. Walter et al. (2017) formalizaram os requisitos estruturados para expressões em LTL e validaram com o objetivo de detectar redundâncias em requisitos no contexto da indústria automotiva. A avaliação dos requisitos formalizados mostrou melhoria na qualidade e apresentou que a validação foi melhor do que feita manualmente.

Embora os resultados sejam positivos com essas abordagens, ainda existem algumas dificuldades em relação à aplicabilidade. Uma limitação é a estruturação fundamentada em padrões que exigem a utilização de glossário ou dicionário de termos e pode ocasionar um esforço manual para a equipe, gerando um maior custo de tempo. Ademais, alguns estudos empregaram a formalização dos requisitos. Esse tipo de representação embasado em lógica matemática implica em problemas na usabilidade, já que para sua aplicação e interpretação é necessário ter o conhecimento técnico. Nota-se que pode ocorrer a falta de expressividade de representação que está naturalmente presente na linguagem natural. Além do mais, poucas ferramentas suportam esse tipo de notação. Desse modo, ela restringe tarefas como a criação de bases de conhecimento e manipulação das informações. Portanto, observa-se a necessidade de uma representação que seja estruturada e validação dos requisitos, mas com estratégias que superem essas deficiências para proporcionar a aplicação em múltiplos domínios.

O Quadro 2.5 mostra um resumo das contribuições desses trabalhos relacionados para sanar essas limitações relatadas. Em comparação com esses trabalhos, esta pesquisa visa abranger todas essas contribuições mapeadas, e, ainda, realizar o tratamento textual dos requisitos, facilitar a aplicação dessas etapas e proporcionar a identificação de causa-efeito entre os conceitos extraídos dos requisitos.

Quadro 2.5: Avaliação das contribuições dos trabalhos relacionados.

| **Critérios** | Soares e Moura (2015) | Walter et al. (2017) | Reggio et al. (2018) | Stachtiari et al. (2018) |
|---|---|---|---|---|
| Aplicação em múltiplos domínios | ✓ | | ✓ | |
| Formatação livre | | | ✓ | |
| Dispensa dicionário/glossário de termos | | | | ✓ |
| Análise semântica | ✓ | | | |
| Criação de base de conhecimento | ✓ | | | ✓ |
| Suporte de ferramenta | ✓ | ✓ | | ✓ |
| Permite manipulação dos dados | | | | ✓ |

Fonte: Elaborado pela autora (2021).

### 2.2.5 Considerações finais

Este capítulo apresentou o embasamento teórico da pesquisa com as definições e os tipos de requisitos, documento de software e suas características de qualidade. Também foi apresentado o estado da arte de abordagens propostas como solução para lidar com os problemas inerentes ao uso da linguagem natural no processo de especificação de software. Os estudos descritos foram obtidos em duas pesquisas bibliográficas e foram mapeados para as etapas de estruturação, validação e representação de requisitos. Observou-se que as abordagens que foram propostas ainda são imprecisas e limitadas para o processamento automatizado.

Entretanto, foi evidenciada a melhoria com a combinação dessas três etapas. Há poucos estudos que exploraram essa solução e, assim, este capítulo contribui com uma discussão dos trabalhos relacionados a esta pesquisa, isto é, com a automatização para uma representação estruturada e validação dos requisitos.

# Capítulo 3

# Metodologia

Este capítulo apresenta esses métodos e descreve os esquemas considerados para a determinação da validade e ameaças da pesquisa. Para atender o objetivo principal desta pesquisa, o qual consiste em melhorar a qualidade de requisitos funcionais por meio de um processo automático para sua representação estruturada e validação, visando a mitigação de problemas de interpretabilidade, como a falta de compreensão e comunicação dos requisitos, utilizou-se métodos de pesquisa qualitativa e quantitativa.

Uma prova de conceito e dois experimentos foram realizados para testar as seguintes hipóteses formuladas para a pesquisa:

- H1: A automatização nas etapas de estruturação, validação e representação do processo de especificação de software pode aumentar a qualidade dos requisitos.
- H2: Apresentar requisitos com uma maior qualidade pode mitigar problemas na comunicação e compreensão dos requisitos escritos em linguagem natural.

A hipótese nula nesta pesquisa é dada por:

- H0: O processo automático para a representação estruturada e validação de requisitos não influencia na mitigação de problemas de interpretabilidade.

Para a avaliação qualitativa do processo proposto neste estudo, buscou-se responder à seguinte questão de pesquisa: **“Quais as contribuições do processo automático para a representação estruturada e validação de requisitos de software na mitigação de problemas de interpretabilidade?”**

## 3.1 Experimentos

Os experimentos seguiram os princípios da engenharia de software experimental. Segundo Travassos, Gurov e Amaral (2002), "os objetivos relacionados à execução de experimentos em Engenharia de Software são a caracterização, avaliação, previsão, controle e melhoria a respeito de produtos, processos, recursos, modelos, teorias entre outros". Esse tipo de pesquisa pode ser aplicado para investigar se a precisão do modelo é a esperada. Em um experimento deve-se identificar o contexto que é de interesse, as variáveis relacionadas e a amostra (WOHLIN et al., 2012).

Os experimentos objetivaram avaliar o desempenho do processo automático para a representação estruturada e validação de requisitos. A amostragem é composta por requisitos expressos em linguagem natural e escritos no idioma inglês obtidos de bases de projetos públicos. O desempenho do *framework* foi avaliado por métricas de avaliação, especificamente a precisão (Equação 3.1) e a revocação (Equação 3.2). A média harmônica (f-*measure*) da precisão e revocação foi obtida pela Equação 3.3.

$$\text{Precisão} = \frac{\text{verdadeiro positivo}}{\text{verdadeiro positivo + falso positivo}} \tag{3.1}$$

$$\text{Revocação} = \frac{\text{verdadeiro positivo}}{\text{verdadeiro positivo + falso negativo}} \tag{3.2}$$

$$\text{F-}measure = 2x\frac{\text{precisão x revocação}}{\text{precisão + revocação}} \tag{3.3}$$

Onde:

**Verdadeiro positivo** - número de classes identificadas manualmente e na automatização;

**Falso positivo** - número de classes identificadas somente na automatização;

**Falso negativo** - número de classes que foram extraídas manualmente e não foram identificadas na automatização.

Para avaliar a geração do modelo de representação, o experimento foi baseado no estudo de Sagar e Abirami (2014). A ideia foi aplicar métricas de avaliação como a precisão e revocação para comparar os resultados obtidos da automatização com a aplicação manual. Como citado pelos autores acima mencionados, é um desafio a comparação dos resultados,

porque existem poucas pesquisas que trabalham com representações de requisitos geradas a partir de textos em linguagem natural. No entanto, podem-se comparar os modelos gerados com outras representações de requisitos existentes. Desse modo, nesta pesquisa, foi avaliada a precisão que consiste na exatidão para a identificação e representação de classes e relacionamentos. Neste caso, as classes são sujeitos (usuário/ator) e objetos (atributo) e os relacionamentos são as ações ou estados dessas classes.

Os resultados obtidos das métricas de avaliação foram analisados por meio de uma análise estatística, a qual é definida por Silvestre (2007) como "[...] um conjunto de métodos adequados para recolher, explorar, descrever e interpretar conjuntos de dados numéricos". Os dados coletados de precisão e revocação foram organizados e descritos pela análise estatística descritiva. Para Silvestre (2007) esse tipo de análise "[...] está focado na medida das características dos elementos de toda a população".

## 3.2 Prova de conceito

Uma prova de conceito foi conduzida com uma população composta por especialistas em projetos de desenvolvimento de software. A amostragem foi de 10 participantes selecionados por conveniência. Buscou-se uma amostra que fosse representativa no contexto do tema da pesquisa. O critério aplicado para a inclusão dos participantes na pesquisa foi a experiência em trabalhar com requisitos em linguagem natural. Nessa modalidade de pesquisa é realizado um "[...] estudo profundo e exaustivo de um ou poucos objetos, de maneira que permita seu amplo e detalhado conhecimento, tarefa praticamente impossível mediante outros delineamentos já considerados" (GIL, 2002).

Para a coleta dos dados foi necessária a aprovação da pesquisa no Comitê de Ética em Pesquisa (CEP)[1](Apêndice A). Os participantes convidados avaliaram a aplicação do processo proposto em comparação com requisitos em linguagem natural. Como parte do processo de execução da prova de conceito, a pesquisadora desta tese apresentou o documento Termo de Consentimento Livre e Esclarecido (TCLE) para os participantes (Apêndice E). Definiu-se um questionário (Apêndice D) com um roteiro para ser aplicado de forma individual. Esse questionário foi publicado online na ferramenta Formulários Google. Foram desenvolvidas 13 questões para a avaliação, tanto abertas quanto fechadas. Para as questões fechadas, foi

[1]Número do CAAE: 20089919.7.0000.5404

utilizado a escala de *Likert* de cinco pontos. A duração média para o preenchimento do questionário foi de 40 minutos.

Os dados obtidos dos questionários foram avaliados por meio da técnica de análise de conteúdo de forma temática. Para Tozoni-Reis (2009), a principal finalidade desse tipo de análise é "[...] desvendar os sentidos aparentes ou ocultos de um texto, um documento, um discurso ou qualquer outro tipo de comunicação". O processo de aplicação pode ser sintetizado em seis etapas, cujos preceitos básicos são (BRAUN; CLARKE, 2006; SILVA; FOSSÁ, 2015):

1. Tratamento dos dados: Os materiais são organizados e preparados. Se os dados estiverem em áudio é preciso transcrevê-los. Uma leitura geral desses dados é realizada e recomenda-se fazer anotações com as ideias iniciais.

2. Geração de códigos: Formulação de códigos (rótulos) que possam abranger dados com características semelhantes, identificadas na leitura geral. Este é o primeiro nível de agrupamento dos dados. Para elaborar a codificação, os dados são fragmentados em unidades de registro, isto é, em palavras, frases ou parágrafos. Essa execução pode ser de forma manual ou apoiada por um software. Para este estudo, foi usada a versão gratuita da ferramenta QDA Miner[2], a qual possibilita organizar os códigos de forma intuitiva e em estrutura de árvore. Essa ferramenta também fornece alguns recursos como busca por códigos, frequência de código, entre outros.

3. Agrupamento temático: Nesta etapa é envolvido um nível de análise mais amplo, pois os códigos e os extratos dos dados são classificados em temas. São identificados padrões (significados) na codificação. O tema deve abranger os códigos que sejam relevantes e estejam associados a ele. Pode-se criar temas mais gerais para abranger subtemas. Algumas estratégias podem ser aplicadas para apoiar o agrupamento, como o uso de mapas mentais para a representação visual dessa classificação. Neste estudo foi utilizada a ferramenta diagrams.net[3] para construir um modelo de mapeamento temático.

4. Revisão de temas: Após o agrupamento por temas, uma revisão é conduzida para verificar se os temas são adequados e estão relacionados com os dados agrupados. É

[2]https://provalisresearch.com/products/qualitative-data-analysis-software/freeware/
[3]https://www.diagrams.net/

feito um refinamento nos temas. Como parte dessa etapa, a hierarquia formada na etapa anterior é analisada.

5. Definindo temas: Com base na revisão, os temas e subtemas que se adéquam ao contexto analisado podem ser definidos. Esse processo é recursivo, ou seja, durante a análise os agrupamentos e temas criados podem ser modificados. Os temas devem ser nomeados de maneira clara e objetiva.

6. Elaboração do relatório: Nesta parte, as inferências e as interpretações produzidas na análise são aplicadas no desenvolvimento do relatório. Como apoio são utilizados exemplos de fragmentos dos dados coletados. As análises devem estar de acordo com a questão de pesquisa do estudo.

## 3.3 Ameaças à validade

Para garantir a confiabilidade dos resultados, foram analisados e mitigados os possíveis fatores de ameaças para validar esta pesquisa. Utilizou-se os esquemas de classificação apresentados por Wohlin et al. (2012), ou seja, a validade de conclusão e construção, e também a validade interna e externa.

A validade da conclusão está associada à relação do tratamento ao resultado do experimento. A principal ameaça identificada foi semelhante ao estudo de Misra (2016). O uso de técnicas de PLN é dependente das características estatísticas do modelo treinado para a classificação. Desse modo, o valor de precisão pode variar. Para minimizar esse risco, foi selecionada a versão de um modelo estável e já treinado com uma grande quantidade de dados disponibilizado pela biblioteca Spacy.

A validade da construção está voltada para a relação da teoria e observação. Nesse sentido, avaliou-se quais métricas seriam adequadas para a avaliação de desempenho do processo automático. As métricas usadas foram a precisão, revocação e f-*measure*, consideradas confiáveis para esse tipo de avaliação (HRIPCSAK, 2005). Para a prova de conceito, buscando minimizar ameaças do tipo social com os participantes, como por exemplo o constrangimento de serem observados durante os testes, todas as atividades foram realizadas de forma online.

Ameaças à validade interna se concentram nos fatores que interferem nas relações causais dos elementos do estudo. A principal preocupação foi a seleção dos participantes e o conjunto de requisitos. Os problemas identificados e soluções aplicadas foram:

- Prova de conceito: Para obter uma amostra representativa para o contexto analisado, optou-se pelo critério de seleção dos participantes por conveniência, para incluir no estudo especialistas em projetos de desenvolvimento de software. Outro risco identificado foi a influência das questões do tipo fechada no questionário para a coleta de dados. Para reduzir essa restrição, essas questões foram completadas com questões do tipo aberta. Com relação à execução das tarefas dos participantes, existe a ameaça causada pelo fato de esta ser conduzida de modo remoto. Embora o ambiente virtual contribua com a flexibilidade de participação em termos de localização e tempo, pode-se enfrentar dificuldades com o entendimento do processo proposto e as etapas do estudo. Visando mitigar esse problema, foi fornecido um vídeo com explicações aos participantes, além de um documento com orientações.

- Experimentos: Para abranger requisitos distintos, os documentos utilizados foram extraídos de três domínios variados e em formatos diferentes (estória de usuário e casos de uso). O conjunto de dados dos experimentos foram anotados manualmente, pelo fato de não haver dados de referência para a avaliação. Visando reduzir alguma análise incorreta, como na identificação de sinônimos, foram realizadas várias revisões dessa anotação. Além disso, foi realizado apenas por um pesquisador para evitar viés de diferentes interpretações.

A validade externa está relacionada com a capacidade de generalização dos resultados obtidos da pesquisa para outros estudos. A prova de conceito utiliza a abordagem do estudo de caso, cuja intenção é prover uma generalização analítica para os resultados que são significativos a casos com características semelhantes (RUNESON; HÖST, 2009). Por essa razão, o estudo pode ser generalizado para equipes de software que lidam com requisitos expressos no formato de linguagem natural.

## 3.4 Considerações finais

Este capítulo dedicou-se a descrever a metodologia que foi estabelecida para atingir os objetivos desta pesquisa. Foram detalhados: o tipo e os métodos de pesquisa, as etapas dos

experimentos e prova de conceito, a população selecionada, as ferramentas e métodos para a coleta e análise dos dados e as questões relacionadas com a validade da pesquisa. Os próximos capítulos detalham a proposta deste estudo e suas avaliações descritas neste capítulo.

# Capítulo 4

# Representação estruturada e validação de requisitos de software

Este capítulo descreve a solução para a mitigação de problemas de interpretabilidade com a automatização para a representação estruturada semanticamente e validação de requisitos de software. Essa solução é composta pela: definição de um processo com os conceitos principais e um conjunto de atividades; descrição de um *framework* com as tecnologias que podem ser utilizadas para a automatização desse processo; desenvolvimento de um protótipo de uma ferramenta com a implementação dessas tecnologias. Além disso, mostra-se também uma exemplificação do uso dessa ferramenta com requisitos de um Sistema de Monitoramento Remoto de Pacientes.

## 4.1 Definição do processo para mitigar problemas de interpretabilidade

Este estudo parte da ideia de que automatizar a representação estruturada semanticamente e a validação de requisitos de software pode melhorar a qualidade de especificações escritas em linguagem natural e torná-las mais precisas. Por conseguinte, pode auxiliar na análise de requisitos com a mitigação de problemas de interpretabilidade como a falta de comunicação e compreensão na especificação do software. À vista disso, este trabalho propõe um processo linear, o qual é composto por três etapas sucessivas: estruturação, validação e representação dos requisitos. O fluxo do processo se constitui na execução das especificações de requisitos

em linguagem natural (entrada) nessa sequência de tarefas e obtém como resultado uma base de conhecimento (saída).

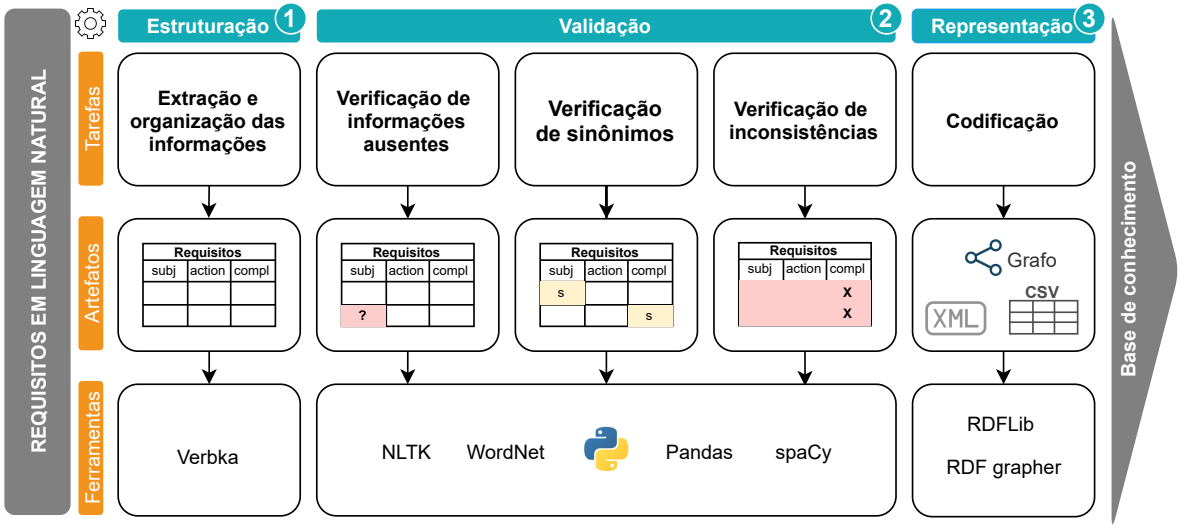
Este processo se diferencia por: tratar o texto para reduzir erros gramaticais; estruturar os requisitos com a aplicação da semântica verbal; proporcionar e facilitar sua aplicação em diferentes domínios; proporcionar a identificação de causa-efeito nas relações dos requisitos; criar automaticamente uma base de conhecimento e permitir a manipulação dos dados extraídos dos requisitos; dispensar o uso de glossário de termos; ser livre de formatação; e oferecer o suporte de uma ferramenta.

Para atender a esses propósitos, a primeira etapa visa à extração das informações. Essa etapa é composta por um conjunto de tarefas para organizar as informações dos requisitos e tratar o texto por meio da análise focada na semântica verbal, a fim de possibilitar uma modelagem conceitual com a identificação de sujeito, verbo e conceitos orientados a software. A aplicação de semântica verbal pode revelar os fluxos de afetação existentes entre os conceitos, permitindo a identificação dos conceitos causa e dos conceitos efeito extraídos dos requisitos (VASQUES et al., 2016). Essa transformação pode fornecer a redução de ruídos no texto.

A segunda etapa avalia a qualidade das informações em nível sintático e semântico. Um conjunto de regras de validação é aplicado nas informações estruturadas. A organização das informações permite a verificação de falhas associadas aos aspectos de qualidade dos requisitos, como incompletude, ambiguidade e inconsistência. Preceder a identificação e correção desses problemas durante a análise dos requisitos pode diminuir o retrabalho em atividades do desenvolvimento.

A última etapa representa os requisitos em um modelo, para torná-los processáveis por computadores. O formato do modelo deve fornecer a facilidade e flexibilidade para a integração com outras ferramentas. O processo explora diferentes formatos para expressar e compartilhar os requisitos. A Figura 4.1 mostra esse fluxo e ilustra como o processo pode ser implantado para sua automatização. As próximas seções apresentam e detalham os recursos computacionais implementados.

Figura 4.1: Visão geral do processo para a representação estruturada e validação de requisitos.




Fonte: Elaborado pela autora (2021).

## 4.2 *Framework* AutRQ - automatização para representar, estruturar e validar requisitos

Esta seção apresenta o AutRQ (Automatização de Requisitos), um *framework* desenvolvido no contexto deste trabalho, o qual implementa os conceitos propostos no processo para automatizar a representação estruturada e validação de requisitos. AutRQ emprega tecnologias de PLN e Web Semântica.

Uma das tarefas de PLN é a extração de informações, *Information Extraction* (IE) - no inglês, a qual busca a identificação de relações estruturadas. *Open* IE é um paradigma que lida com a extração de informações independente do domínio e gera como resultado Proposições (P) no formato de tripla, composta pela relação entre dois conceitos (*conceito_1, relação, conceito_2*) ou *n*-árias, isto é, considerando mais que dois conceitos no relacionamento (*conceito_1, ..., relação, ..., conceito_n*) (XAVIER; LIMA; SOUZA, 2015). Os recursos de *Open* IE podem fornecer a estruturação automática dos requisitos.

Para a implementação do AutRQ, utilizou-se um processo automático baseado no modelo *Open* IE, denominado Verbka (VASQUES et al., 2016; VASQUES, 2016). O processo não requer um entendimento antecipado do domínio (VASQUES et al., 2016; VASQUES, 2016). Inicialmente, esse processo foi testado por meio da aplicação manual e observou-se a necessidade da sua automatização para facilitar a aquisição do conhecimento. A autora desta

pesquisa colabora com outros pesquisadores, incluindo a responsável pela criação do processo (VASQUES et al., 2016), no desenvolvimento da automatização do Verbka.

Em comparação com outras abordagens de *Open* IE (XAVIER; LIMA; SOUZA, 2015), o Verbka se destaca por ser baseado em semântica verbal e contribuir com a correção de alguns problemas gramaticais, como a conversão de sentenças passivas e fragmentação de conjunções. Esse processo obteve resultados promissores em uma pesquisa anterior (SANTOS, 2017), na qual o Verbka apoiou uma estratégia para mitigar riscos em estórias de usuário de projetos que aplicam a metodologia ágil *Extreme Programming* (XP) (BECK, 2000) no desenvolvimento distribuído. Devido à eficiência apresentada em sua aplicação e suas contribuições para o tratamento do texto, aplicou-se o Verbka para a extração e organização das informações dos requisitos.

Verbka é composto por regras linguísticas e gramaticais para segmentar o texto em componentes linguísticos e reestruturar de forma ordenada, os conceitos e suas relações. As proposições geradas pelo processo são expressas por *n*-árias, que equivalem a uma *n*-tupla, onde *n* consiste no conjunto de *n* conceitos ordenados, neste *framework*, no formato <*sujeito, verbo, ..., complemento_n*>. Esse formato pode abranger informações que normalmente são perdidas na representação de triplas (MAUSAM et al., 2012). Para sua exemplificação, um requisito foi extraído do estudo de Goknil, Kurtev e Berg (2014) no idioma inglês. O Quadro 4.1 mostra o requisito selecionado. Salienta-se que a tradução em português foi inserida nos exemplos apresentados nesta seção para facilitar o entendimento do *framework*.

Quadro 4.1: Aplicação do processo Verbka: seleção do texto.

| | |
|---|---|
| *Requisito Original* | The system shall store patient temperature measured by the sensor in the central storage and it shall warn the doctor when the temperature threshold is violated.<br><br>(O sistema deve armazenar a temperatura do paciente medida pelo sensor no armazenamento central e deve avisar o médico quando o limite de temperatura for violado.) |

Fonte: Elaborado pela autora (2021).

Após a seleção do requisito (Quadro 4.1), o texto é submetido para o processamento do Verbka, o qual faz um tratamento no texto de maneira automática, com a transcrição dos verbos para sua forma base e em voz ativa, fragmentação de orações coordenadas e exclusão de termos acessórios. As proposições obtidas do texto são estruturadas de acordo com sua

função semântica e são apresentadas em uma tabela (Quadro 4.2). Essa tabela mostra dois grupos principais, o *agent* que executa a ação e o *patient* que é afetado. Como relatado por Vasques et al. (2016), os complementos são construídos com a fragmentação em objetos. No contexto desta pesquisa, os atores/sujeitos (*agent*) extraídos dos requisitos são relacionados a suas ações (*verb*) e atributos do software (*complements*). Para Ferreira e Silva (2013), esses complementos normalmente descrevem a estrutura/arquitetura, funções, estados e regras do negócio.

Quadro 4.2: Aplicação do processo Verbka: geração da tabela semântica.

| P | Agent | Patient | | |
|---|---|---|---|---|
| | | Verb | Compl. 1 | Compl. 2 |
| 1 | The sensor<br>(O sensor) | measure<br>(mede) | patient temperature<br>(a temperatura do paciente) | - |
| 2 | The system<br>(O sistema) | shall store<br>(deve armazenar) | patient temperature<br>(a temperatura do paciente) | in the central storage<br>(no armazenamento central) |
| 3 | The system<br>(O sistema) | shall warn<br>(deve avisar) | the doctor<br>(o médico) | when the temperature threshold is violated<br>(quando o limite de temperatura for violado) |

Fonte: Elaborado pela autora (2021).

O AutRQ apresenta algumas regras de qualidade criadas manualmente para validar os requisitos (tendo como entrada a tabela semântica). O foco é detectar problemas de incompletude, ambiguidade e inconsistência nas informações estruturadas. Elas são detalhadas a seguir.

***Detecção de informações ausentes.*** Visando identificar informações implícitas que podem comprometer o requisito, deve-se verificar se há sentenças sem atores e/ou sem complementos das ações. Essas sentenças com conceitos incompletos são destacadas, para que o engenheiro de software possa descobrir e completar essas informações. Por exemplo, em "Criar relatórios" não está contido o ator que cria esses relatórios. O Algoritmo 1 apresenta o pseudocódigo desenvolvido para essa verificação.

***Detecção de sinônimos.*** Uma das soluções para reduzir ambiguidades é a substituição de sinônimos (PITTKE; LEOPOLD; MENDLING, 2015), que podem ser identificados por um banco de dados lexical. Desse modo, para cada conceito, identificam-se sinônimos e sua presença nos

**Algoritmo 1:** Verificação de conceitos ausentes

```
Resultado: Identificação de sujeitos e complementos vazios
Entrada: Matrix de informações estruturadas
Saída: Anotações dos campos ausentes
Função VerificarCamposVazios(matrix):
    para cada linha da matriz faça
        S ⟵ verifique se há um conceito de sujeito em linha
        se S == Falso então
            linha[sujeito] = substitua o valor nulo para Quem?
        fim
        C ⟵ verifique se o número de complementos é maior que 0
        se C == Falso então
            linha[complemento] = substitua o valor nulo para O quê?
        fim
    fim
    retorna matrix
Fim Função
```

requisitos, como detalhado no Algoritmo 2. Se existirem, listam-se esses sinônimos e solicita-se ao engenheiro de software a seleção do conceito mais significativo para aquele contexto.

**Algoritmo 2:** Verificação de conceitos sinônimos

```
Resultado: Identificação de conceitos sinônimos
Entrada: Matrix de informações estruturadas
Saída: Lista de conceitos sinônimos
Função VerificarSinonimo(matrix):
    listaSIN ⟵ lista de sinônimos
    para cada linha da matriz faça
        para cada termo da linha faça
            se termo não existir em listaSIN então
                DL ⟵ Obtenha os sinônimos do termo no banco de dados lexical
                S ⟵ Verifique se os sinônimos em DL existem na matriz
                se S == Verdadeiro então
                    listaSIN += [sinônimos do termo]
                fim
            fim
        fim
    fim
    retorna listaSIN
Fim Função
```

***Detecção de inconsistências.*** Há três regras para alertar o engenheiro de software de possíveis conflitos nos requisitos, baseadas no estudo de Escalona et al. (2013):

1. Inconsistência entre atores: verificação se existem sentenças com ações e complementos iguais, mas com atores diferentes. Exemplo: “Médico residente acessa o sistema” e “Médico efetivo acessa o sistema”.

2. Inconsistência entre complementos de ações: extração de sentenças que apresentem o mesmo ator e a ação, mas com divergências nos complementos. Exemplo: “Sistema registra o nome do usuário” e “Sistema registra o nome de usuário com e-mail”.

3. Inconsistência na ação: Verificação de sentenças com os mesmos atores e complementos, mas com contradição nas ações/estados. Exemplo: “Usuário se cadastra por e-mail” e “Usuário não se cadastra por e-mail quando ele entra via rede social”.

O Algoritmo 3 fornece a lógica desenvolvida com a aplicação dessas regras. Após sua execução, as informações estruturadas e validadas podem ser representadas.

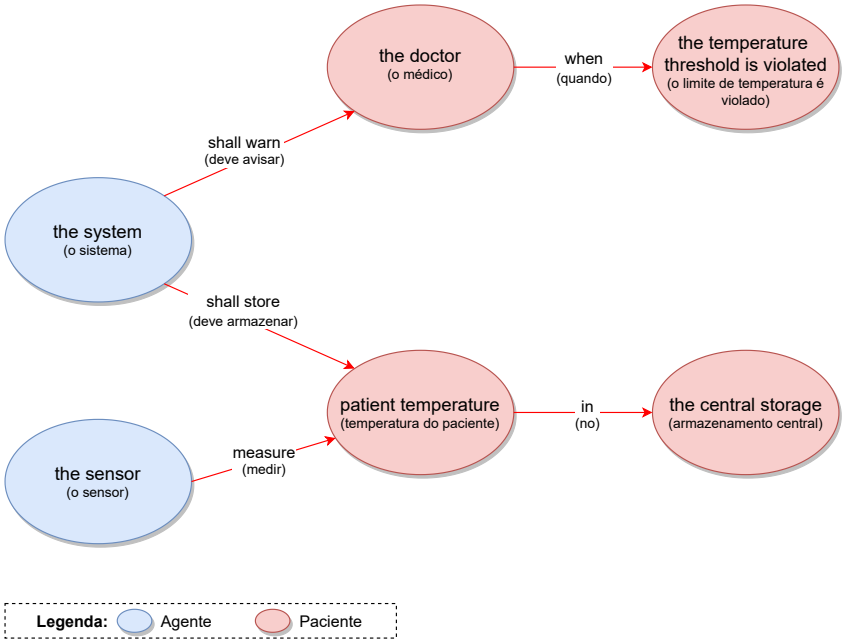
**Algoritmo 3:** Verificação de conceitos inconsistentes

```
Resultado: Identificação de inconsistências nos conceitos de sujeito, verbo e
           complemento
Entrada: Matrix de informações estruturadas
Saída: Lista de índices com conceitos inconsistentes
Função VerificarInconsistencia(matrix):
    listaINC ⟵ lista de índices
    S ⟵ verifique se há índices com o mesmo verbo e complemento, mas com
    sujeitos diferentes
    se S == Verdadeiro então
        listaINC[sujeitos] = índices com conceitos de sujeito inconsistentes
    fim
    V ⟵ verifique se há índices com o mesmo sujeito e complemento, mas com
    contradição no verbo
    se V == Verdadeiro então
        listaINC[verbos] = índices com contradição no conceito do verbo
    fim
    C ⟵ verifique se há índices com o mesmo sujeito e verbo, mas com
    os complementos superiores a 1 diferentes
    se C == Verdadeiro então
        listaINC[complementos] = índices com conceitos de complementos
          inconsistentes
    fim
    retorna listaINC
Fim Função
```

O Verbka representa o conhecimento obtido em mapas conceituais (VASQUES et al., 2016). Os conceitos agentes são relacionados por meio de setas (que indicam os verbos) aos

conceitos pacientes (complementos verbais). Para ilustrar esses mapas, a Figura 4.2 mostra o mapa gerado com base na tabela semântica.

Figura 4.2: Aplicação do processo Verbka: geração do mapa conceitual.




Fonte: Elaborado pela autora (2021).

Uma possível codificação para esse modelo conceitual é representar esse conhecimento com tecnologias de Web Semântica. A Web Semântica pode ser entendida como uma visão de uma rede composta por recursos de informações que são compreensíveis por máquinas (HAYES, 2004). O resultado esperado da aplicação da Web Semântica é a representação em modelos *Resource Description Framework* (RDF). Esse *framework* objetiva descrever recursos no contexto da Web, no entanto, pode ser utilizado para diversas áreas de aplicação e diferentes tipos de objetos como físicos, conceitos abstratos, entre outros (LASSILA; SWICK, 1998; MCBRIDE, 2004). Os recursos e suas relações são caracterizados formalmente por meio de *Uniform Resource Identifiers* (URIs). A principal vantagem dessa aplicação é que URIs referenciam os recursos de forma não ambígua (HAYES, 2004).

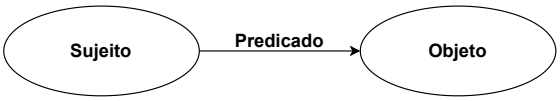
O modelo RDF é composto por três tipos de elementos, sendo (LASSILA; SWICK, 1998):

- *Resources* (Recursos): são todos os objetos que são descritos pelas expressões RDF;
- *Properties* (Propriedades): é a utilização de uma relação, aspecto específico, alguma característica para descrever um recurso;

- *Statements* (Declarações): consiste na relação de um recurso específico, uma propriedade e o valor dessa propriedade. A declaração RDF também pode ser encontrada como instrução ou afirmação.

A declaração RDF é formada por meio de triplas: sujeito (recurso), predicado (propriedade) e objeto (valor da propriedade). O objeto da declaração pode ser um recurso (definido por URI) ou um valor literal (podendo ser uma sequência de caracteres) (LASSILA; SWICK, 1998). A Figura 4.3 ilustra um exemplo de uma tripla representada em grafo RDF.

Figura 4.3: Exemplo de um grafo RDF (tripla).




Fonte: Adaptado de Consortium (2014).

O AutRQ representa o conhecimento que é obtido pelo Verbka em um formato RDF/XML. O RDF é flexível e pode ser serializado (expresso textualmente) em diferentes formatos, como *Terse RDF Triple Language* (Turtle), *JavaScript Object Notation for Linked Data* (Json-LD), N-*Triples*, XML, entre outros. A XML foi selecionada pelo motivo de proporcionar a legibilidade humana e ter a capacidade de representar estruturas complexas (MILLER, 2005). A Figura 4.4 mostra o documento RDF/XML criado com as informações presentes no Quadro 4.2. O documento é composto pelas URIs utilizadas para a identificação de recursos e propriedades, como na linha 5 que declara uma URI de exemplo para conceitos do tipo sujeito.

Ressalta-se que o *framework* utiliza o recurso de contêiner do RDF para estender a tripla com os complementos extraídos da tabela semântica. Os contêineres armazenam e representam listas de recursos, como também os literais (LASSILA; SWICK, 1998). Neste caso a sequência dos complementos deve ser mantida, por isso, é utilizado o objeto contêiner do tipo sequência (por exemplo, linha 10 da Figura 4.4), o qual gera esses complementos de forma ordenada. Os objetos do contêiner estão relacionados com os complementos da tabela semântica, por exemplo, complemento 1 equivale ao primeiro objeto do contêiner e assim, sucessivamente.

Para facilitar a interpretação humana, nesta implementação, os valores de propriedades são no formato de texto. Para a descrição dos metadados dessas propriedades é adotado um

Figura 4.4: Requisito expresso no formato RDF/XML.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:vb="http://purl.org/linguistics/gold/Verbal"
5      xmlns:subj="http://example.com/subj/"
6      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
7
8      <rdf:Description about="http://example.com/subj/ The System (O Sistema)">
9          <vb:shall_store--deve_armazenar>
10             <rdf:Seq>
11                 <rdf:li> patient temperature (a temperatura do paciente)</rdf:li>
12                 <rdf:li> in the central storage (no armazenamento central)</rdf:li>
13                 <rdfs:label> subject affects complements (sujeito afeta complementos)</rdfs:label>
14             </rdf:Seq>
15         </vb:shall_store--deve_armazenar>
16         <vb:shall_warn--deve_alertar>
17             <rdf:Seq>
18                 <rdf:li> the doctor (o médico)</rdf:li>
19                 <rdf:li> when the temperature threshold is violated (quando o limite de temperatura for violado)</rdf:li>
20                 <rdfs:label> subject affects complements (sujeito afeta complementos)</rdfs:label>
21             </rdf:Seq>
22         </vb:shall_warn--deve_alertar>
23     </rdf:Description>
24     <rdf:Description about="http://example.com/subj/ The Sensor (O Sensor)">
25         <vb:measure--medir>patient temperature (a temperatura do paciente)</vb:measure--medir>
26         <rdfs:label> subject affects complements (sujeito afeta complementos)</rdfs:label>
27     </rdf:Description>
28 </rdf:RDF>
```

Fonte: Elaborado pela autora (2021).

vocabulário já definido de conceitos verbais (linha 4 da Figura 4.4). Outra contribuição é a etiquetagem dos recursos e propriedades com prefixos, ao invés de *Uniform Resource Locator* (URL), como exemplo o *Verb* (vb) que representa o verbo (predicado) e *Subject* (subj) que descreve o recurso.

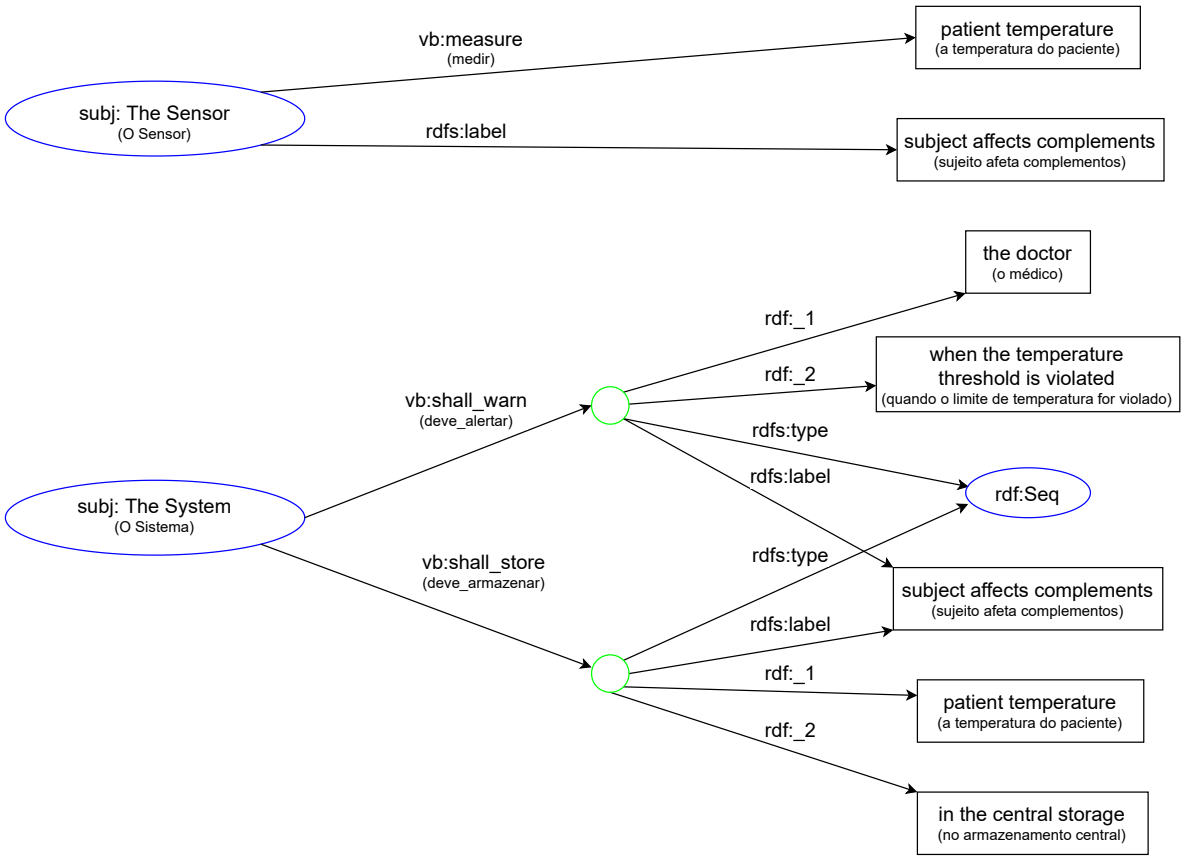
O documento RDF/XML pode ser empregado para gerar um grafo RDF (CONSORTIUM, 2014). A Figura 4.5 ilustra o exemplo do grafo gerado baseado na tabela semântica (Quadro 4.2) obtido de um serviço de visualização de grafos. O grafo é constituído por nós (sujeito, objetos) relacionados por arco dirigido (predicado) (MILLER, 2005). Os nós podem ser do tipo URIs, literais ou vazios (CONSORTIUM, 2014). Nesse exemplo, o formato de elipse simboliza os sujeitos e o tipo de contêiner (rdf:Seq) e os retângulos indicam os objetos. Os objetos do contêiner são relacionados por um nó vazio interligado com o predicado. Os relacionamentos entre as sequências do contêiner são descritos com uma numeração ordenada, como exemplo rdf:_1 e rdf:_2.

O processo Verbka proporciona a identificação do fluxo de afetação na relação dos conceitos, o qual é reconhecido pelo tipo de associação (verbo). Os tipos de verbos são (VASQUES et al., 2016; VASQUES, 2016):

- Ação-processo (sujeito afeta complementos): conceitos que executam uma ação que recai em outros conceitos, por exemplo, na sentença "O sistema cadastra médico", o conceito *médico* recebe uma ação do conceito *sistema*;
- Ação (sujeito afeta a si mesmo): conceitos que executam uma ação que recai sobre em eles próprios, como "Médico recebe o alerta no sistema", o próprio conceito *médico* recebe a ação;
- Ligação (conceitos estáticos): não existe um fluxo de afetação na relação, por exemplo, "Médico tem o atributo nome completo", o verbo de ligação *ter* é estático.

Para incorporar a identificação de causa-efeito, o *framework* utiliza a classe de propriedades *label* do RDF com a descrição do tipo de relação causal entre os conceitos, com seus respectivos valores: "*subject affects complements*" (sujeito afeta complementos), "*subject affects itself*" (sujeito afeta a si mesmo) e "*static concepts*" (conceitos estáticos).

Figura 4.5: Requisito expresso no formato de grafo RDF.




Fonte: Elaborado pela autora (2021).

A conversão das informações para uma linguagem que seja processável pelo computador possibilita a criação de uma base de conhecimento automática e a recuperação de informações. Os grafos gerados de informações estruturadas são armazenados em uma *triplestore* (ou RDF *store*), a qual consiste em uma base que lida com dados de semântica e sem esquemas (SELVARAJ; WEBER; LUTTEROTH, 2017).

## 4.3 Suporte de ferramenta

Um protótipo foi construído para ilustrar como seria uma ferramenta para dar suporte ao *framework* AutRQ. A ferramenta foi desenvolvida com as tecnologias descritas na seção 4.2 para processar requisitos de sistemas de informação. Para sua utilização, orienta-se seguir as regras de qualidade para a escrita de casos de uso (LILLY, 1999) e estórias de usuário (COHN, 2004b). Deve-se evitar o uso de frases irrelevantes e atores genéricos.

Para sua utilização, a ferramenta não requer um treinamento, no entanto, o texto submetido deve estar de acordo com algumas regras: as sentenças devem estar pontuadas corretamente; o texto deve ser em linguagem natural, a ferramenta não suporta requisitos no formato formal; e cada requisito deve estar relacionado a uma ação ou estado.

Para a submissão de requisitos extraídos de casos de uso, devem ser inseridas somente as descrições dos fluxos básicos e/ou secundários. Para isso, o ator deve estar explícito em cada etapa dos fluxos, ao invés de apresentar os atores apenas uma vez na documentação, por exemplo como a etapa: "*O **requerente** apresenta a reclamação com dados comprovativos [...]*". Para as estórias de usuário, o padrão suportado é o mesmo que apresentado na subseção 2.1.1, sendo: "Como <tipo de usuário>, eu quero <objetivo da estória> para que <razão>".

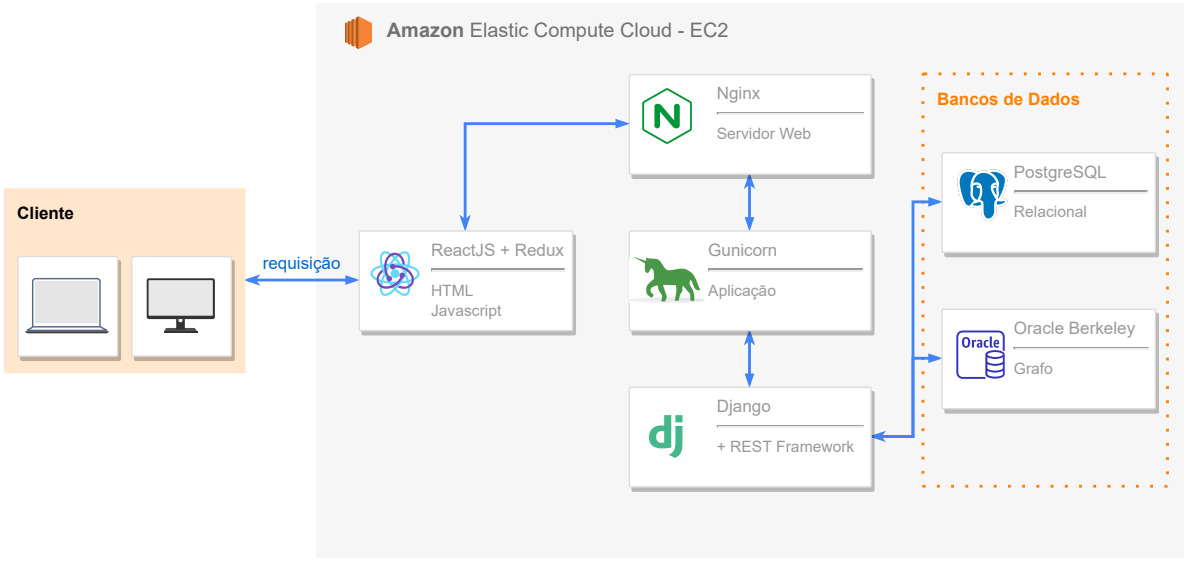
A ferramenta pode ser utilizada de forma online como um apoio para a especificação e gerenciamento dos requisitos. As seções seguintes detalham como ela pode ser implantada e as funcionalidades da ferramenta são apresentadas por meio de exemplos.

### 4.3.1 Desenvolvimento do protótipo da ferramenta

O protótipo, desenvolvido em python, utiliza uma estrutura cliente-servidor, com recursos da biblioteca React[1] e Redux[2] (bibliotecas em javascript) para *front-end* e do *framework* Django[3] para o *back-end*, com bancos de dados PostgreSQL[4] e, para grafos, Oracle Berkeley[5].

Para a construção do ambiente de produção, foram configurados os servidores Nginx[6] e Gunicorn[7]. Devido o alto consumo de processamento das ferramentas empregadas, a instância do Elastic Compute Cloud (EC2)[8] da Amazon Web Services (AWS), utilizada para alocar o protótipo, tem a capacidade computacional de 2 vCPUs (unidade de processamento virtual) e 4 GiB de memória. A Figura 4.6 representa a estrutura da implantação do protótipo.

Figura 4.6: Arquitetura da infraestrutura cliente-servidor do protótipo AutRQ.




Fonte: Elaborado pela autora (2021).

Há um conjunto de recursos utilizados para realizar os processamentos nos requisitos extraídos das partes interessadas e expressos em linguagem natural — neste protótipo, em inglês. Para o desenvolvimento das funções de estruturação, geração do relatório de validação de erros e a codificação do modelo de representação, aplicou-se: recursos para o

[1]https://reactjs.org/
[2]https://redux.js.org/
[3]https://www.djangoproject.com/
[4]https://www.postgresql.org/
[5]https://www.oracle.com/database/berkeley-db/
[6]https://www.nginx.com/
[7]https://gunicorn.org/
[8]https://aws.amazon.com/ec2/

PLN como a API de código fechado Verbka e as bibliotecas Industrial-Strength Natural Language Processing - spaCy (HONNIBAL; MONTANI, 2017), Natural Language Toolkit - NLTK (LOPER; BIRD, 2002) e seu banco de dados lexical WordNet incorporado; algumas funções da RDFLib (BOETTIGER, 2018) para operações de inclusão, remoção, armazenamento e recuperação de informações no formato RDF; e a biblioteca pandas[9] para a manipulação de tabelas.

A implantação dessas funções permite o mapeamento dos requisitos automaticamente para o formato RDF, mas ainda é necessária a intervenção do engenheiro de software para algumas decisões. A ferramenta cria e armazena uma base de conhecimento com informações estruturadas em uma *triplestore* local, extraídas dos requisitos. O protótipo oferece alguns recursos para o gerenciamento dessa base, como remoção e inserção de requisitos utilizando SPARQL, a linguagem de consulta para RDF. Além disso, é possível fazer consultas avançadas de informações por termos específicos contidos nos sujeitos, verbos ou objetos. Outras informações, como dados dos usuários e descrições da base, são armazenadas em um banco de dados relacional.

É possível apresentar as informações da base de conhecimento em formatos de tabelas ou grafos, utilizando serviços oferecidos pela iniciativa Linked Data Finland (LINKED DATA FINLAND, 2020), como o serviço RDF *grapher* utilizado para a construção dos grafos. Essas informações podem ser exportadas pelo protótipo no formato RDF em notação XML ou *Comma Separated Values* (CSV). O protótipo apresenta ainda a funcionalidade de gerenciamento para compartilhar a base de conhecimento, assim sendo, o engenheiro de software pode controlar o acesso da equipe do projeto.

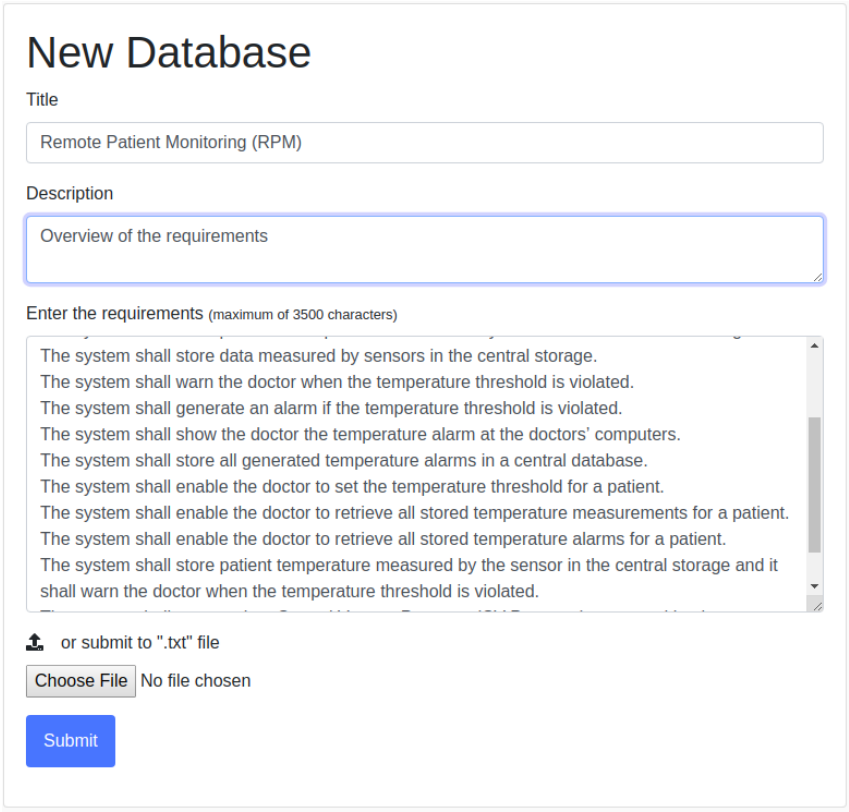
### 4.3.2 Exemplo de uso do protótipo

Esta seção apresenta um exemplo da aplicação do protótipo da ferramenta, ilustrando seu funcionamento e a interface Web. Os requisitos funcionais são relacionados a um Sistema de Monitoramento Remoto de Pacientes (*Remote Patient Monitoring*, RPM) (GOKNIL; KURTEV; BERG, 2014).

A primeira etapa é a inclusão dos requisitos na ferramenta. O engenheiro de software define o nome da base que será criada, uma descrição do sistema e os requisitos, que podem ser redigidos no campo específico ou submetidos por meio de um arquivo de texto (Figura 4.7).

[9] https://pandas.pydata.org

Figura 4.7: Exemplo de uso do AutRQ: criação da base de requisitos.



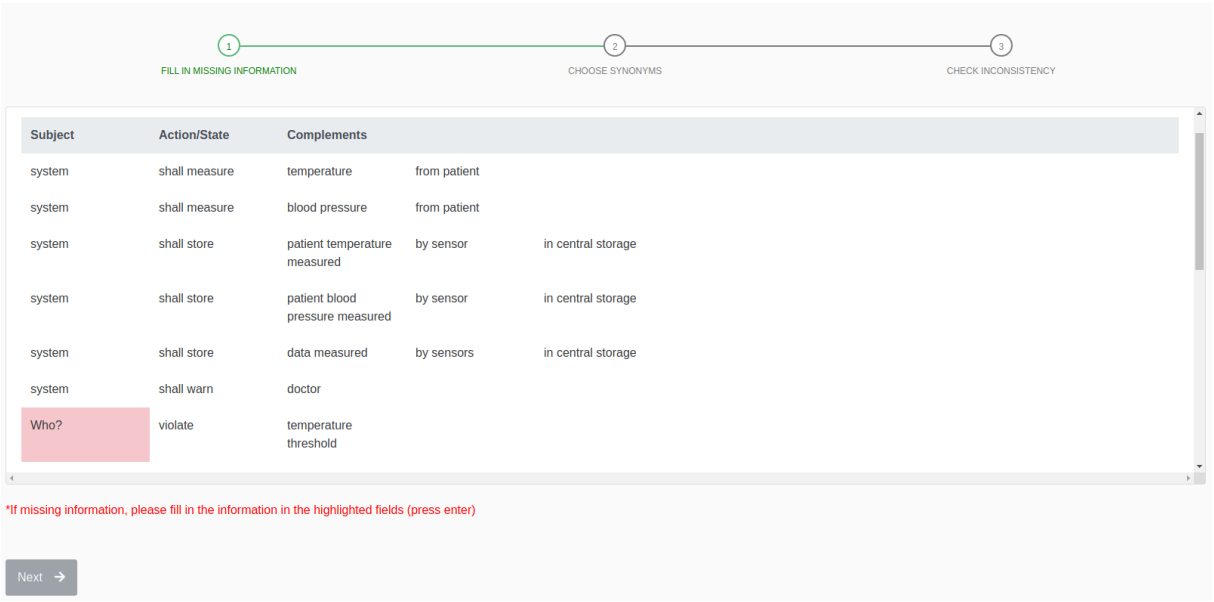

Fonte: *Print screen* do protótipo AutRQ.

Após a submissão dos requisitos, a ferramenta identifica eventuais lacunas. No exemplo (Figura 4.8), a ferramenta identifica a ausência do sujeito de uma das ações e destaca o campo com a cor vermelha e a pergunta *Who*? (Quem?), para que o engenheiro possa identificar e complementar a informação ausente. Para a falta de informação no complemento, o campo é destacado com a pergunta *What*? (O quê?).

A ferramenta verifica também se há conceitos sinônimos nos requisitos. A Figura 4.9 ilustra a detecção de dois termos semelhantes e a possibilidade de escolha, pelo engenheiro, do termo preferencial. A ferramenta substitui então as ocorrências dos conceitos sinônimos para o selecionado. Pittke, Leopold e Mendling (2015) enfatizam que é relevante o processo detectar e notificar a ambiguidade com sugestões para que os usuários possam escolher a opção mais apropriada.

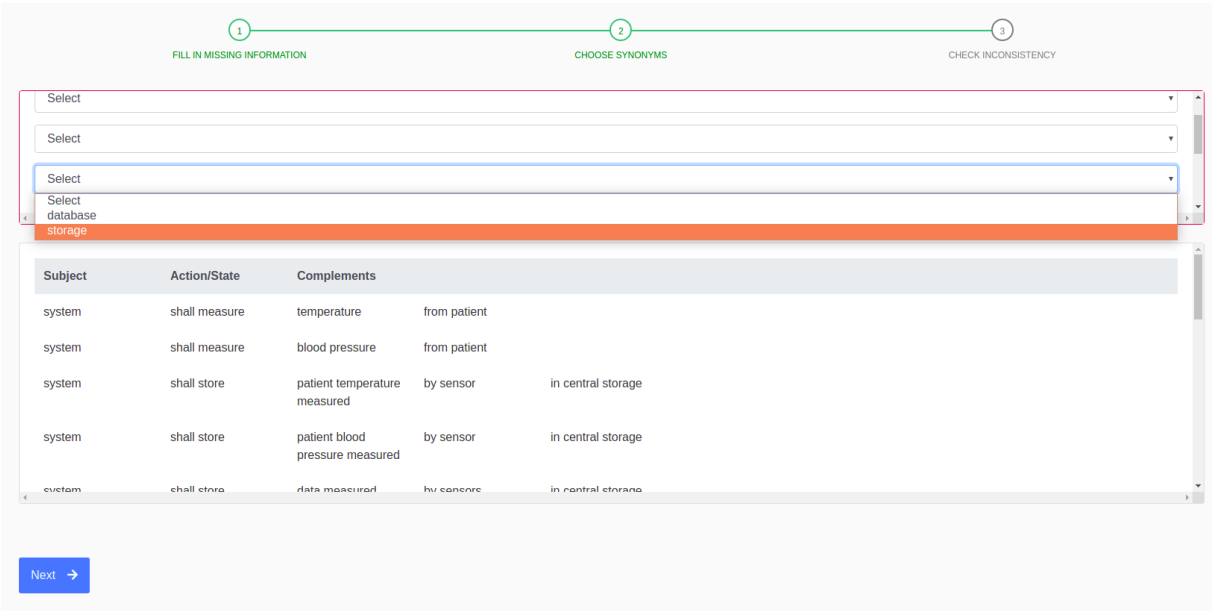
Na última etapa de verificação automática, a ferramenta retorna possíveis inconsistências nos requisitos, como informações duplicadas (como mostrado na Figura 4.10). Para resolver essas inconsistências, o engenheiro pode editar diretamente na tabela ou remover o requisito. Em casos de atores iguais, como os requisitos originais não apresentaram esse problema, foi

Figura 4.8: Exemplo de uso do AutRQ: verificação de ausência de informações.




Fonte: *Print screen* do protótipo AutRQ.

Figura 4.9: Exemplo de uso do AutRQ: verificação de sinônimos.

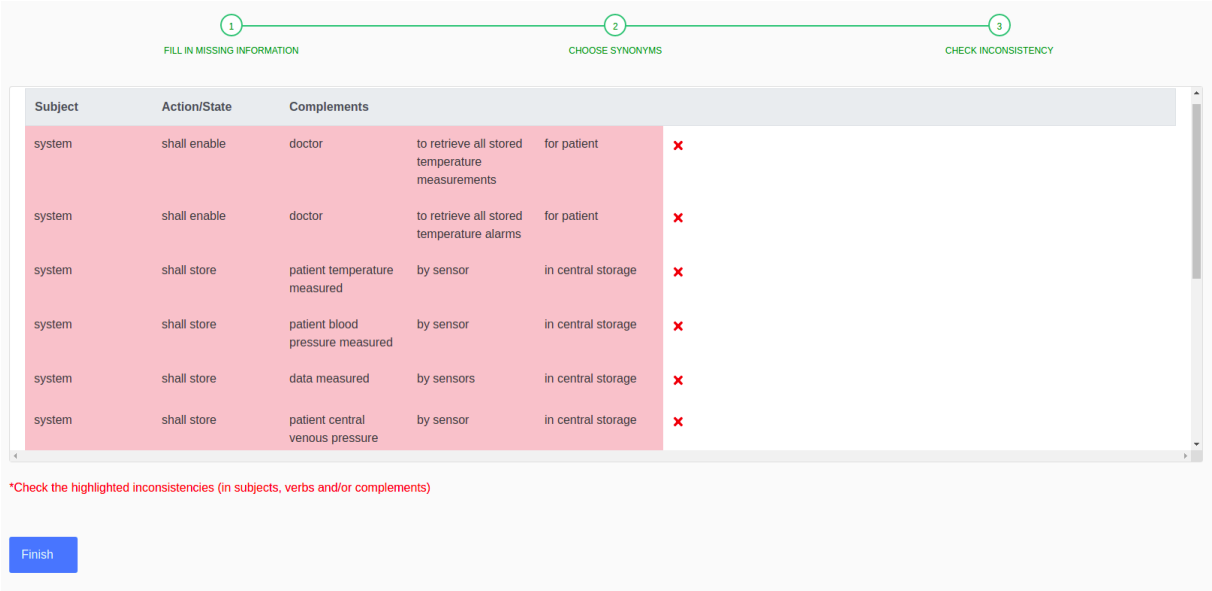



Fonte: *Print screen* do protótipo AutRQ.

feito um exemplo na Figura 4.11 para ilustrar como pode-se criar uma generalização de sujeitos iguais, com a definição de um conceito genérico.

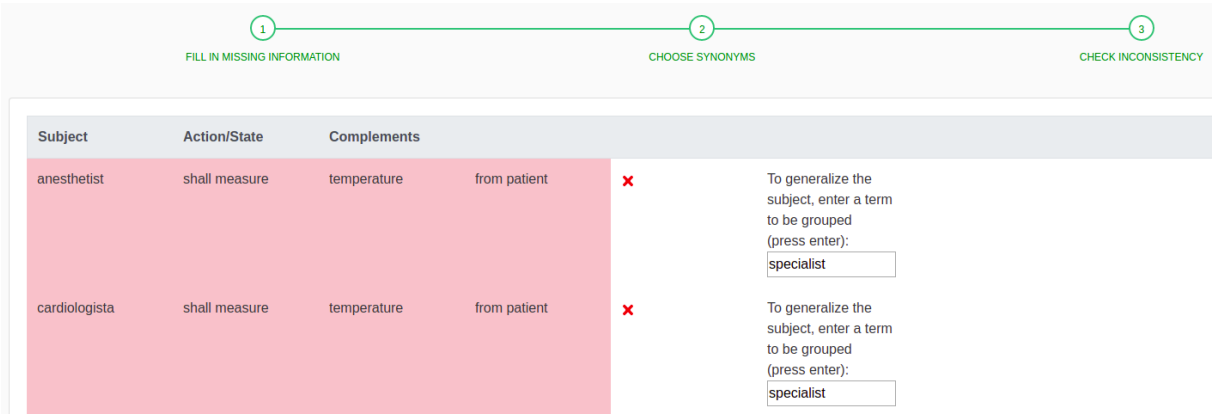
Após essas verificações, o protótipo cria a base de conhecimento extraído dos requisitos de forma automática, por exemplo, neste contexto cria-se uma base de conhecimento sobre o sistema de gestão hospitalar. Por meio da integração das informações e o acesso à base de

Figura 4.10: Exemplo de uso do AutRQ: verificação de inconsistências.




Fonte: *Print screen* do protótipo AutRQ.

Figura 4.11: Exemplo de uso do AutRQ: generalização dos sujeitos.



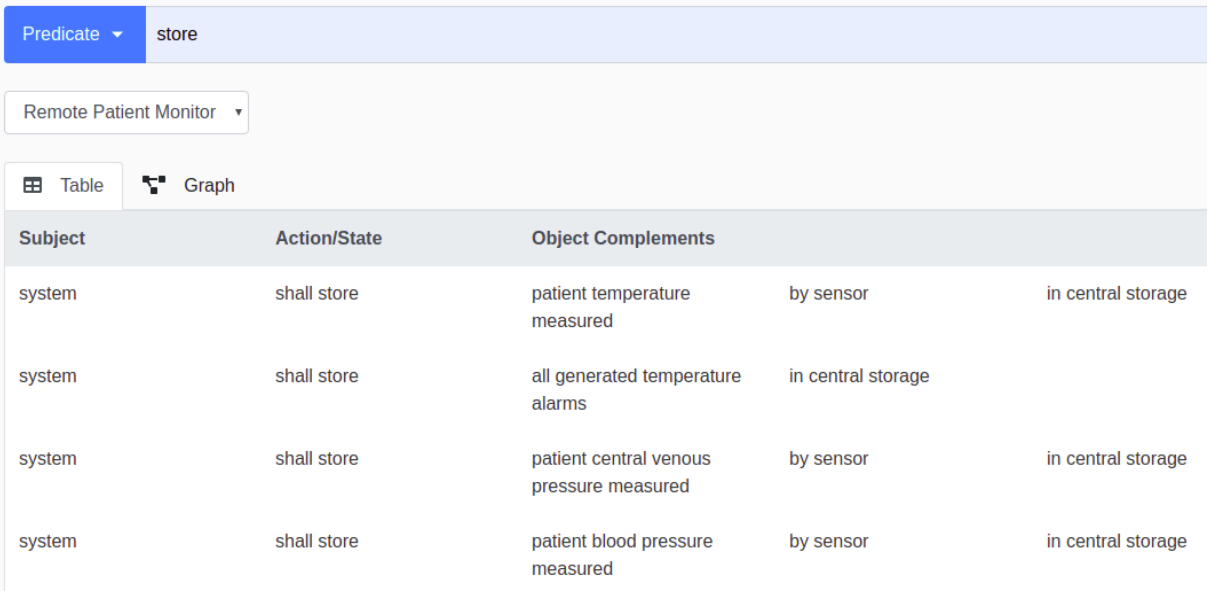

Fonte: *Print screen* do protótipo AutRQ.

conhecimento, o engenheiro pode realizar algumas tarefas como consultas específicas para a análise dos requisitos.

A Figura 4.12 apresenta um exemplo de consulta de requisitos que estavam relacionados com a ação (*predicate*) *store* (armazenar), tendo como objetivo investigar quais itens o sistema armazena. Nessa figura, o resultado da busca é apresentado em uma tabela.

O mesmo resultado pode ser apresentado em um grafo RDF (Figura 4.13). Existem algumas características específicas dos serviços utilizados para a construção do modelo, como as descrições de sequência dos objetos como *rdf:first* para o primeiro item e *rdf:rest*

Figura 4.12: Exemplo de uso do AutRQ: busca avançada.




Fonte: *Print screen* do protótipo AutRQ.

para os demais e os prefixos do verbo, marcado como *ns1*. Para facilitar o entendimento, foi acrescentado o termo *verb*.

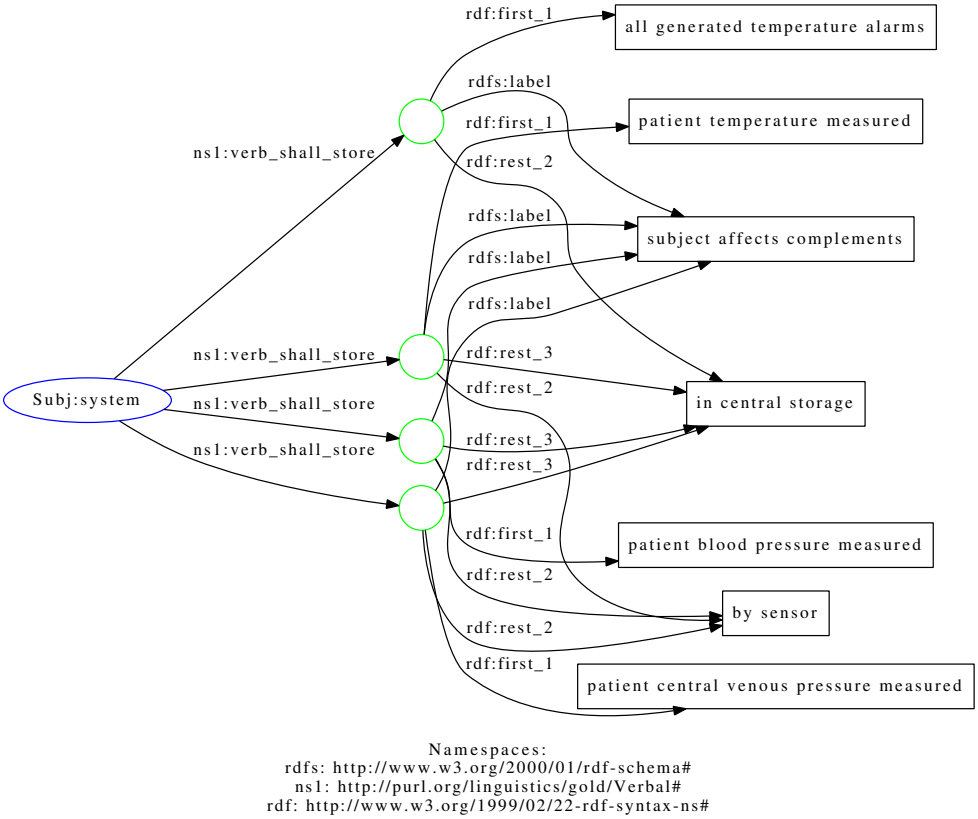
## 4.4 Considerações finais

Os conceitos citados neste capítulo possuem como princípio estabelecer um processo para automatizar uma representação estruturada e validação de requisitos de software. A proposta tem a premissa de que este processo pode melhorar a qualidade dos requisitos e, assim, ser capaz de reparar os problemas relacionados com a comunicação e a compreensão dos requisitos.

Esses conceitos foram implementados em um *framework*, denominado AutRQ, e ainda, implantado em um protótipo de uma ferramenta Web. Assim sendo, o capítulo também apresenta os recursos computacionais utilizados para a construção da ferramenta, como as tecnologias de Web Semântica, PLN, as bibliotecas de desenvolvimento Web e a infraestrutura utilizada.

Pode-se perceber, por meio de uma exemplificação com requisitos do domínio de monitoramento de pacientes, a capacidade do processo para: processar os requisitos automaticamente para a identificação de erros e a geração do modelo conceitual, criar a base de conhecimento com informações estruturadas extraídas desses requisitos, armazenar

Figura 4.13: Exemplo de uso do AutRQ: Resultado do grafo gerado na busca avançada.




Fonte: *Print screen* do protótipo AutRQ.

informações de forma centralizada, além da possibilidade de compartilhar essas informações. Os requisitos foram convertidos para um formato que permite seu processamento por máquinas, entretanto, preservou-se a expressividade na representação final. O próximo capítulo descreve a avaliação desta solução proposta.

# Capítulo 5

# Avaliação do processo proposto

Este capítulo apresenta a prova de conceito e os experimentos realizados, considerando a metodologia definida no Capítulo 3. Descrevem-se as etapas executadas, bem como a população do estudo e a análise dos resultados obtidos. Além disso, é conduzida uma discussão com as principais descobertas da pesquisa.

## 5.1 Prova de conceito com especialistas

Este estudo buscou observar as percepções de especialistas em projetos de desenvolvimento de software na aplicação do *framework* AutRQ em comparação com requisitos em linguagem natural em um cenário real. A execução e análise dos resultados da prova de conceito foram realizadas em um período de 3 meses.

Os participantes do estudo, os especialistas, atuam em projetos que empregam metodologias tradicionais e ágeis de desenvolvimento de software. Devido ao sigilo e privacidade, os participantes são mencionados pela sigla P associada a um número de identificação. Os participantes apresentam diferentes níveis profissionais, sendo: júnior (P2, P3 e P8), pleno (P6, P7 e P10) e sênior (P1, P4, P5 e P9). A maioria dos especialistas (7 de 10) tem experiência superior a cinco anos.

Os participantes executaram as atividades que compõem o estudo de forma remota e sem a interação entre eles. A pesquisadora desta tese atuou como observadora e mediadora durante este estudo. O estudo foi realizado em duas etapas.

Na primeira etapa, os participantes representaram, de forma livre, os requisitos de software em linguagem natural como empregam normalmente em seus projetos. Os

especialistas desenvolveram alguns requisitos no idioma inglês. Foram definidos 171 requisitos de diferentes domínios com a remoção daqueles duplicados.

Após a definição dos requisitos, na segunda etapa, esses especialistas acessaram o protótipo desenvolvido com a implementação das funcionalidades propostas no *framework* AutRQ. O objetivo foi disponibilizar os recursos da ferramenta na Web para os participantes avaliarem o *framework*. A pesquisadora utilizou alguns serviços da plataforma GoDaddy[1] para a publicação e hospedagem do protótipo. Posteriormente, os especialistas criaram bases com os requisitos desenvolvidos na primeira etapa. Para isso, a pesquisadora apresentou os recursos oferecidos pelo protótipo em um guia de orientações e ainda, mostrou por meio de um vídeo, um exemplo de como executar suas funcionalidades. Todos os participantes conseguiram acessar e utilizar corretamente os recursos fornecidos no protótipo. No entanto, um problema ocorreu, pois alguns participantes não adicionaram o ponto no final das sentenças e isso interferiu na primeira etapa da criação da base. Esta informação não estava clara. Mas, o problema foi corrigido com um aviso enviado aos participantes e uma atualização nas orientações. A pesquisadora esteve disponível para o esclarecimento de dúvidas. Os especialistas seguiram todos os passos sequenciais que são obrigatórios do protótipo, como a estruturação do texto, validações de qualidade e conversão automática para o modelo de representação. Os participantes criaram 28 bases de conhecimento.

A construção dessas bases possibilitou aos especialistas realizar algumas tarefas como a busca avançada com termos específicos e a visualização completa dos requisitos. Além disso, possibilitou também a exportação das informações estruturadas em diferentes formatos. A pesquisadora incentivou os participantes a explorarem esses recursos. Após a conclusão dessas tarefas, a pesquisadora disponibilizou o questionário, para que os participantes pudessem reportar suas experiências tanto quanto os resultados obtidos das intervenções nos requisitos com a utilização do *framework*.

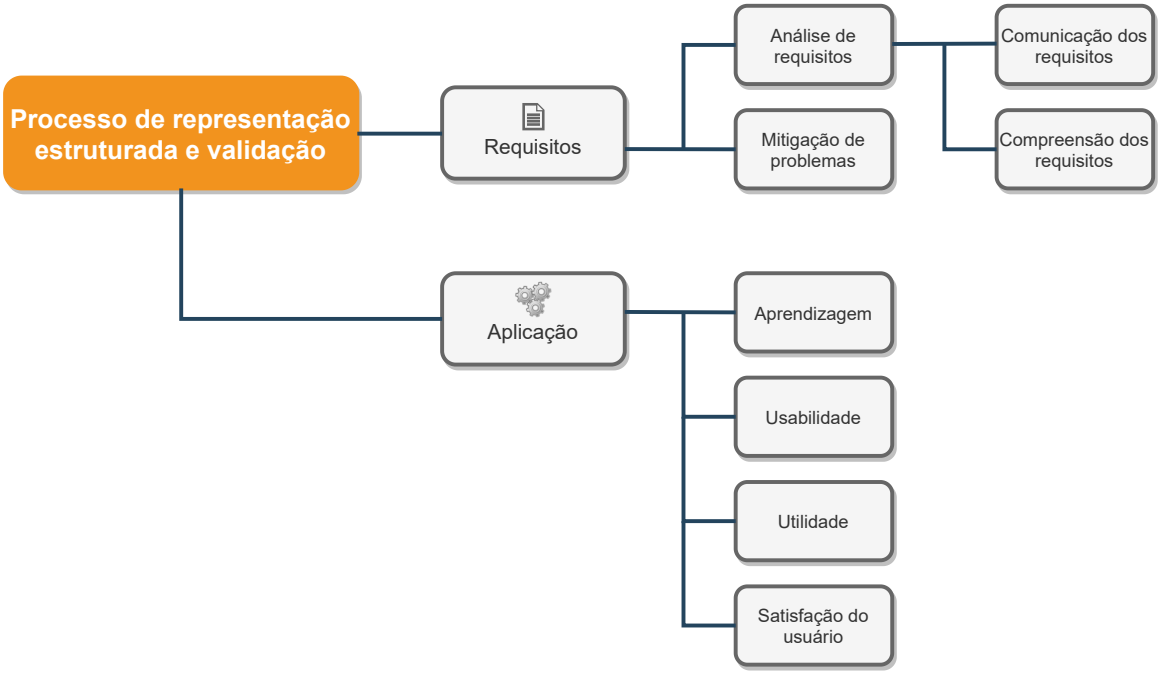
Os resultados obtidos foram classificados em duas categorias, como mostrado na Figura 5.1. A categoria "Requisitos" engloba a avaliação do *framework* a respeito da qualidade dos requisitos. Neste cenário, foi analisado como a proposta contribui para mitigar os problemas inerentes à linguagem natural e como auxilia a tarefa de análise de requisitos. A classificação "Aplicação" envolve a avaliação da execução do *framework* por meio do protótipo. Para a análise dessa aplicação, foram utilizados alguns critérios sugeridos como

[1]https://www.godaddy.com/

métricas por Lund (1998). Esses critérios são correlacionados entre si. Esta análise foi baseada na definição apresentada no estudo de Kamalrudin, Hosking e Grundy (2017):

- Facilidade de aprendizagem: quão fácil é a compreensão das funcionalidades fornecidas pela ferramenta;
- Facilidade de uso: quão fácil é a utilização. Este aspecto está relacionado com a interface de usuário e as funcionalidades da ferramenta;
- Utilidade: como a ferramenta pode ser útil para os especialistas e quais tarefas podem ser proveitosas em sua aplicação;
- Satisfação do usuário: quão satisfeito o especialista está com a capacidade da ferramenta em processar as tarefas.

Figura 5.1: Mapeamento temático dos resultados obtidos na prova de conceito.




Fonte: Elaborado pela autora (2021).

### 5.1.1 Percepções sobre os requisitos

Esta subseção apresenta os principais resultados (Quadro 5.1) obtidos na análise da avaliação feita pelos especialistas em referência às melhorias na qualidade dos requisitos. Além disso,

mostra como essas melhorias apoiaram a etapa de análise de requisitos de software.

Quadro 5.1: Principais resultados obtidos na análise da qualidade dos requisitos.

| | **Qualidade dos Requisitos** |
|---|---|
| Contribuições | Aprimoramento na escrita dos requisitos |
| | Validação durante a criação dos requisitos |
| | Estruturação e fragmentação de frases múltiplas |
| | Facilidade na transmissão de informações |
| | Descoberta de novos requisitos |
| | Facilidade de análise dos propósitos dos atores |
| | Identificação de relações tácitas |
| Limitações | Falhas na classificação de alguns conceitos no processamento |

Fonte: Elaborado pela autora (2021).

### Mitigação de problemas

Uma das principais descobertas no uso do *framework* para reduzir os problemas inerentes à linguagem natural foi a contribuição para escrita dos requisitos. De acordo com os especialistas, o uso do *framework* promoveu um direcionamento para a tarefa da escrita. De fato, proporciona recursos que auxiliam a escrita para a construção de um bom requisito, isto é, considera algumas características de qualidade definidas nas recomendações do IEEE (2018). O participante P4 destacou: "*A ferramenta me permitiu pensar melhor na escrita do requisito para achar uma versão mais simples e direta*" e completou "*os requisitos que eu criei foram na maioria atômicos, completos e testáveis*".

A melhoria na escrita citada pelos participantes foi em virtude da maneira como foram criados os requisitos pelo protótipo. Realizar os passos de validação juntamente com a adição dos requisitos estimulou aos especialistas a se atentarem ao desenvolvimento e às melhorias dos requisitos. Esse ponto foi apontado por alguns especialistas, um deles comentou (P4): "*ajudou a aperfeiçoar minha escrita de requerimentos*". Uma observação importante é que a estrutura fornecida para a criação dos requisitos e a fragmentação de frases múltiplas fez com que os próprios especialistas notassem algumas deficiências nos requisitos, como descrito pelo participante P6: "*os resultados ficaram mais claros quando colocamos os requisitos em frases simples, mantendo um requisito por frase*".

A estruturação dos requisitos permitiu que os especialistas verificassem se as informações estavam corretas. Para os participantes, ocasionou uma clareza nesses dados. Os especialistas

se adaptaram a essa estrutura, como citado pelo P1: "*o resultado em planilha ficou muito claro do que a ferramenta absorveu da linguagem natural*" e pelo P2 que confirmou que permitiu "*identificar lacunas nos requisitos do projeto*".

Para os especialistas, o protótipo facilitou a identificação de requisitos vagos. Este é um ponto importante na mitigação de problemas de linguagem natural, já que esse tipo de linguagem é subjetivo. Alguns participantes apontaram que identificaram esses requisitos na fase de verificação de inconsistência. Este fato chamou a atenção desses especialistas, como visto em um comentário do participante P5, o qual afirmou que poderia usar a ferramenta para reduzir requisitos dúbios.

Na maior parte dos casos, a etapa de validação de requisitos foi avaliada de forma positiva pelos participantes. Especialmente para identificação de informação faltante. O resultado corrobora que aplicar essa validação, torna os requisitos mais completos. Para o especialista P4 "*O passo de 'Fill in missing information', me permitiu repensar no meu próprio requisito*". O alerta mostrado na tela do protótipo, proporciona a fácil identificação dessas deficiências. Um dos participantes apontou (P7): "*a informação faltante fica mais clara*". Para o outro participante (P2): "*O software possibilitou preencher lacunas nos requisitos que às vezes de tão lógico que o mesmo seja, acaba não sendo exposto e possibilitando dupla interpretação para outros analistas*". Evitando esses vieses de interpretação como comentado pelo especialista, pode mitigar problemas futuros no projeto. Outro especialista (P10) aponta isso em sua observação: "*além de ser muito bom ter essas validações antes da criação, evitando, assim, possíveis erros por conta de requisitos*".

Quando os especialistas foram questionados por qual motivo eles adotariam o *framework*, a maioria das respostas foram relacionadas com a possibilidade de realizar a validação durante a criação dos requisitos. Este fato despertou o interesse deles.

#### Análise — comunicação e compreensão dos requisitos

Os resultados obtidos revelaram duas principais contribuições para a etapa de análise de requisitos: melhorias na comunicação e compreensão, o que vem ao encontro do objetivo deste estudo. Uma das razões desses avanços foi a mitigação precoce dos problemas presentes nos textos das especificações.

O *framework* atuou como um mediador para reduzir as falhas na comunicação. Sua aplicação favoreceu a transmissão de informações. Oito participantes concordaram que a aplicação facilitou a comunicação dos requisitos. Para o participante P8: "*a comunicação entre eles foi elaborada de forma correta*". Os tipos de representações que o *framework* apresenta os requisitos (tabela e grafo), auxiliaram essa comunicação. Com relação a esses tipos de representações, obteve-se respostas similares ao comentário do participante P5 que se interessou por fornecer detalhes nos requisitos: "*uma visão mais ampla sobre o sujeito, ação ou estado e detalhes*".

Essa facilidade na comunicação ocasionou a descoberta de novos requisitos durante a fase de análise. Três especialistas identificaram a necessidade de gerar novos requisitos e incrementaram suas bases. Para o participante P3 teve-se a "*facilidade de gerar informações*". Outro especialista (P9) declarou que adotaria o *framework* para "*analisar os requisitos identificados e verificar se poderiam ter novos requisitos*". Com uma resposta semelhante, o participante P7 aponta que sua aplicação "*permite encontrar novos requisitos*". Conforme apontado pelos especialistas, os requisitos novos foram identificados em duas situações. Para um especialista (P5) a descoberta dos requisitos foi durante a validação do requisito, como declarado: "*nas sessões 'Fill in Missing Information e Choose Synonyms' o protótipo mostra os requisitos com ausência de informações e sinônimos, deixando mais claro a necessidade de formular mais requisitos novos*". Sob outra perspectiva, o especialista P7 argumentou que identificou "*comparando a tabela com os requisitos e seguindo o mapa*".

Os especialistas consideraram os requisitos compreensíveis. Nove participantes apontaram que o *framework* auxiliou no entendimento dos requisitos e suas relações. Essa facilidade para a análise dos requisitos foi mencionada pelo participante P8: "*O resultado apresentado pela ferramenta é relativamente simples, de fácil compreensão, não tive problemas para analisar*". Outras declarações foram semelhantes a essa, como registrado pelo participante P5 que abordou que o *framework* auxilia nesta fase, para ele: "*fica mais fácil compreender os requisitos*".

Os especialistas mencionaram que a composição da tabela e do grafo no formato de sujeito, verbo e complementos facilitou a compreensão. Um dos participantes (P5) comentou que: "*a visualização do grafo auxilia a equipe de desenvolvedores a enxergar e analisar os requisitos de forma mais simples*". Inclusive, proporcionou a identificação e o detalhamento dos atores desses requisitos. Para o participante P5, o *framework* "*mostra de forma muito simples todas as*

*referências de cada sujeito*". Outro participante (P4) complementou que pode "*ver rapidamente no grafo quais são os atores identificados*". Com isso, contribuiu com a análise dos propósitos de cada ator no software e, até mesmo, com a revelação de novos atores.

Com base nas declarações dos especialistas, uma das vantagens na fase de análise foi a identificação de relações tácitas e de causa e efeito. O participante P7 relata que ajudou "*a encontrar relações que podem ser omitidas nos requisitos em linguagem natural*". Outros comentários foram identificados acerca da rotulagem dos conceitos de causa e efeito, como mencionado por outro participante (P9): "*Eu achei interessante o fato do grafo mostrar, por meio de rótulos, as relações de causa e efeito*".

Explicitar esses tipos de relações também pode apoiar a análise das mudanças nos requisitos. A organização do *framework* pode apoiar os requisitos que são alterados ao longo do desenvolvimento do software. De acordo com o especialista P9: "*Essas informações podem ajudar na rastreabilidade de requisitos (sequenciamento) em um projeto de software*". Por meio dos relacionamentos entre os requisitos é possível construir uma matriz de rastreabilidade. Isto é, mapear esses relacionamentos para auxiliar no controle e investigação dos impactos das mudanças nas especificações.

### 5.1.2 Percepções da aplicação

Os especialistas do estudo avaliaram como foi a execução do *framework* por meio da utilização do protótipo. Investigaram suas experiências no processo de aprendizagem e uso. Ademais, os participantes forneceram comentários adicionais sobre as utilidades da ferramenta e mostraram sua satisfação. A Quadro 5.2 apresenta os principais resultados obtidos nessa análise, detalhados a seguir.

#### Aprendizagem

Para investigar como foi para os participantes o processo de aprendizagem de uso da ferramenta, optou-se em instruir os especialistas com informações básicas do uso para que eles pudessem explorar diferentes fluxos. Mesmo assim, os participantes apontaram que eles aprenderam a utilizar os recursos da ferramenta em um curto período de tempo. O participante P5 comentou: "*Tive um pouco de dificuldade no começo para entender o protótipo*", no entanto, em seguida declarou que "*logo fui me familiarizando e entendendo o sistema*". Um

Quadro 5.2: Principais resultados obtidos na análise da aplicação do *framework*.

| **Execução do *Framework*** | |
|---|---|
| Contribuições | Aprendizagem em curto período de tempo |
| | Clareza no uso dos recursos para diferentes níveis de experiência |
| | Apoio para a criação de modelagem |
| | Apoio para a análise de mudanças nos requisitos |
| | Facilidade na manipulação e organização de informações |
| | Eficiente processamento de busca e recuperação de informação |
| | Geração de agrupamento de informações e síntese do conhecimento |
| Limitações | Necessidade de melhorias na interface do usuário |
| | Falta de suporte para outros idiomas |

Fonte: Elaborado pela autora (2021).

dos participantes (P1) comentou: "*Acho que foi proposital deixar absolutamente aberto ao usuário escrever o que quiser com poucas instruções para preenchimento já pensando em UX e usabilidade também*", acrescentando que a aprendizagem foi "*Muito simples, mesmo com poucas instruções o uso foi rápido*". Outros participantes afirmaram a facilidade que tiveram em relação à aprendizagem do uso da ferramenta.

### Usabilidade

O protótipo mostrou clareza no uso dos recursos implantados. No geral os participantes avaliaram de forma positiva a facilidade na utilização do protótipo, quando questionados sobre quais facilidades tiveram em suas execuções. Para o participante P7 a interface da ferramenta "*é fácil de utilizar*".

Uma das facilidades declaradas pelos participantes foi a praticidade para a criação de requisitos. Um dos especialistas (P8) mencionou que "*os campos a serem fornecidos são intuitivos*". Outro especialista (P10) atesta que foi "*Bem objetivo para declaração dos requisitos, fácil visualização e alterações durante as validações*". Os resultados mostraram que a ferramenta foi acessível e atendeu a todos os níveis de experiências (júnior, pleno e sênior) dos especialistas. Por exemplo, o participante P2 (júnior) apontou que encontrou "*Facilidade de uso*". Essa afirmação também foi apresentada pelos participantes P10 (pleno) que a ferramenta fornece "*boa usabilidade*" e P5 (sênior) "*Particularmente, achei muito fácil a utilização do protótipo AutRQ*".

A ferramenta mostrou ser eficiente em outras etapas, como nas de validações e busca. Para um dos especialistas (P5), a ferramenta "*traz muitas funcionalidades simples, como, preencher as informações ausentes, escolher sinônimos, a pesquisa é bem detalhada e a visualização do grafo ajuda muito*". Os participantes não tiveram muitas dúvidas na execução dessas tarefas. Além disso, os fluxos para realizar as tarefas foi simples, o participante P4 comentou que "*Foi intuitivo, consegui criar o banco, adicionar e remover requerimentos*". Outro participante (P9) reconheceu que teve "*facilidades de uso relacionadas à navegação no framework*".

Apesar das avaliações positivas referentes à usabilidade, os participantes apontaram e sugeriram algumas melhorias na interface do usuário. Entre essas sugestões, foram mencionadas alterações como quantidade de caracteres permitidos na criação da base, alterações dos nomes de alguns campos e redirecionamentos de algumas páginas. Um exemplo é a declaração do participante P4 "*após de criar um requisito, eu era redirecionada para a tela de todos os bancos, eu preferia continuar no mesmo banco para continuar adicionando requisitos*".

Com base nas descrições extras dos especialistas, foram identificados alguns incrementos para serem implantados, visando melhorar a interface do usuário. Como apontado por um dos participantes (P7): "*Seria bastante útil que, quando pergunta a informação faltante ou quando pergunta pelos sinônimos, o usuário pudesse ler a sentença em linguagem natural para facilitar a identificação*". Implantar a visualização do texto original durante a validação pode contribuir para superar essa limitação. Observou-se que as mensagens e notificações de erros devem ser revisadas para uma melhor clareza. Ainda, serão consideradas as opções para refazer e desfazer ações durante a validação dos requisitos, visando reparar a dificuldade apontada pelo o participante P3: "*Na etapa de checagem cliquei em 'x', mas estava correto e não consegui desfazer a ação*".

Os especialistas apontaram a necessidade de adicionar um documento de ajuda para a ferramenta. Algumas declarações foram como a do participante P1: "*suponho que seriam disponibilizadas mais instruções para um direcionamento do analista de requisitos*" e P5 "*eu acho que faltam informações detalhadas*". No entanto, isso era esperado, visto que, como mencionado anteriormente, decidiu-se disponibilizar apenas as informações básicas para poder coletar as percepções dos especialistas em relação à usabilidade.

Observou-se que, em alguns casos, a representação da relação de causa-efeito no grafo pode ser irrelevante e confundir o entendimento. Um dos especialistas (P9) sugeriu "*Talvez*

*fosse interessante ter uma opção no protótipo em que o usuário pudesse ativar e desativar esse recurso (causa-efeito)*". Neste mesmo contexto, foi identificada a necessidade de descrever com mais detalhes a legenda do grafo, com as informações das etiquetas. Alguns participantes alegaram essa dificuldade quando questionados sobre as dificuldades que tiveram na utilização da ferramenta.

**Utilidade**

A ferramenta mostrou ser útil em diferentes contextos. Baseado nas declarações dos especialistas, sua utilização pode proporcionar aos integrantes da equipe de um projeto de desenvolvimento de software uma melhor gestão, produtividade e economia de tempo.

Alguns especialistas relataram o apoio no uso do protótipo nas etapas da engenharia de software, como declarado pelo participante P7: "*acho ele bastante útil para tarefas de engenharia de software*". Outro participante (P8) acrescenta que: "*A ferramenta é interessante e facilita o trabalho do desenvolvedor*". Essa foi uma das motivações apresentadas pelos especialistas para utilizarem a ferramenta em seus projetos. Para o participante P6, "*o levantamento de requisitos é uma fase importante do desenvolvimento de sistemas e que muitas vezes fica em segundo plano*". Nesse contexto, a aplicação da ferramenta pode priorizar as atividades dessa fase no ciclo de desenvolvimento do software. A ferramenta revelou-se capaz de apoiar a criação de modelagens como apontado pelo participante P6 "*O grafo pode ser de grande auxílio no quesito modelagem*".

Outra aplicação da ferramenta que despertou o interesse dos participantes foi a sua utilização para a manipulação da informação. Uma das vantagens identificadas é a possibilidade de indexação dos requisitos na ferramenta. Para o participante P6, proporcionou "*ter um controle maior da aplicação desenvolvida*". Dispor os requisitos estruturados em diferentes formatos e em uma base de conhecimento do domínio motivou os especialistas. Por exemplo, um desses especialistas (P6) afirmou que adotaria o protótipo pois "*Com o protótipo, conseguiríamos armazenar as informações originais das solicitações*". Outro participante (P4) declarou: "*Eu gostei porque posso ter registrado em um banco mesmo*". Identificou-se a relevância para exportar essas informações. Um dos especialistas (P5) comentou que utilizaria a ferramenta para "*a exportação dos dados em CSV e RDF, além de servir como um banco de dados de requisitos de software*".

A centralização dos requisitos ocasionou alguns ganhos que foram mencionados pelos participantes. Um exemplo foi a organização das informações. O especialista P3 declarou que "*O sistema organiza as informações de maneira simples e de fácil visualização, especialmente no modelo graphs"*. Essa organização pode auxiliar tarefas de agrupamento de informações com características similares, como observado na menção do especialista P7 "*Fica muito mais fácil agrupar os requisitos por atores*".

A ferramenta facilitou o acesso e a recuperação das informações extraídas dos requisitos. Sete especialistas responderam que obtiveram informações relevantes nas pesquisas realizadas. Dois participantes alegaram que não foram obtidas informações significativas e uma resposta foi neutra. Contudo, um desses participantes (P4) acrescentou que para cada banco criado por ele foram adicionados "*uns 10 requisitos, então não precisei muito usar o buscar. Acredito que com bancos maiores vai ter mais utilidade*". Os resultados obtidos das buscas foram precisos, como informado pelo participante P8: "*nos testes de consultas que fiz, todas retornaram informações úteis e corretas de acordo com os parâmetros da pesquisa*". Para a maioria dos participantes, o recurso de busca facilitou a sintetização do conhecimento extraído dos requisitos.

Os especialistas conseguiram obter detalhes durante as consultas como descrito pelo participante P5 "*o sistema traz de forma bem detalhada todas as consultas realizadas*". Esses resultados foram alcançados devido à busca avançada. Os participantes apontaram que tiveram facilidades nesse tipo específico de pesquisa, como exposto pelo participante P5 que afirmou "*a facilidade de pesquisar sobre cada item (sujeito, ação, estado e complementos)*" e P4 "*como tem a opção de filtrar os requisitos é uma solução para só focar na parte do grafo que eu preciso*". Foi observado que a busca recuperou as informações de forma rápida.

Receberam-se comentários mais específicos que apresentaram sugestões de valor. Entre eles, o participante P1 opinou que "*seria interessante criar subferramentas*". O especialista ainda descreveu que seriam extensões da ferramenta para fluxos alternativos nos requisitos, regras de validação de campos e registros de recursos com conteúdo de mensagens. Outro especialista (P6) recomendou implantar um versionamento das alterações realizadas nas bases criadas. Além dessas melhorias, um participante (P2) destacou a necessidade de abranger mais idiomas para o processamento dos requisitos.

**Satisfação do usuário**

Em geral, os participantes ficaram satisfeitos com o desempenho da ferramenta. Baseado nos resultados, a maioria dos requisitos foram processados de forma correta. Por exemplo, o participante P3 afirmou que "*Os requisitos criados foram compreendidos pelo sistema de maneira adequada*". Outras declarações foram próximas a essa, como a do participante P3: "*Não obtive problemas, o sistema identificou corretamente*" e a do participante P8: "*A ferramenta possui boa performance*", o qual completou que "*A ferramenta processa as informações de forma eficiente*".

Os conceitos extraídos do texto em linguagem natural foram mapeados nas classes certas em grande parte dos casos, como visto pelo participante P4: "*Os requisitos em linguagem natural com a estrutura <subject><verb><complement> foram interpretados corretamente pelo protótipo*". No entanto, alguns especialistas relataram dados que foram classificados de forma incorreta. Por exemplo, como descrito pelos especialistas P7: "*há problemas quando identifica adjetivos como 'correct' ou advérbios como 'when' como se fossem objetos*" e P4: "*usando a preposição 'as long as' os requisitos não eram bem interpretados*". Nesse problema, deve-se notar que a classificação dos dados extraídos dos requisitos é dependente de um modelo de dados treinado, o qual será atualizado em trabalhos futuros para melhorar esse tipo de classificação.

O protótipo abrangeu diferentes tipos de sintaxe de requisitos. Entretanto, no formato das estórias de usuário, em alguns casos, os sujeitos/atores escritos como "As a <user>" (Como <usuário>) causaram falhas no processamento como descrito pelo participante P4: "*obtive resultados que não foram os esperados*". No restante, alguns participantes avaliaram o protótipo como promissor e que atendeu as suas expectativas. O participante P7 mencionou que: "*a ferramenta cumpre o que promete*" e o participante P8 confirmou: "*Gostei muito do protótipo, acho que tem bastante potencial*". Outros comentários foram próximos a esses, como o do especialista P1 "*Eu fiquei impressionado. Foi um resultado mais preciso do que eu imaginaria que fosse*".

## 5.2 Experimentos de desempenho

Esta seção descreve as amostras e procedimentos usados para duas avaliações da capacidade de desempenho do *framework* AutRQ para a identificação de erros nos requisitos e geração

de modelos conceituais. O processo de execução dos experimentos e a análise dos resultados tiveram a duração de 4 meses.

Para explorar diferentes formatos de requisitos, utilizaram-se as notações de casos de uso e estórias de usuário. Como relatado pelos especialistas no estudo, a ferramenta apresentou restrições para o processamento de estórias de usuários. Por essa razão, os pesquisadores incrementaram melhorias para esse tipo de formato.

A amostra envolve documentos de requisitos disponíveis publicamente de três projetos: *Data Hub*, *Planning Poker* e ACME. Esses requisitos foram selecionados pelos motivos de: apresentarem diferentes níveis de complexidade; serem utilizados em estudos relacionados à modelagem conceitual e identificação de problemas; e estarem no formato de estória de usuário e caso de uso. *Data Hub* consiste em um repositório online para tarefas como coletar, organizar, compartilhar e pesquisar conjuntos de dados. *Planning Poker* é uma plataforma[2] que auxilia nas estimativas de estórias de usuários. Foram utilizadas as especificações aplicadas no experimento realizado pelos autores Dalpiaz e Sturm (2020a), os quais transcreveram as estórias de usuários extraídas de uma base de dados (DALPIAZ, 2018) para o formato de casos de uso (DALPIAZ; STURM, 2020b). ACME é um sistema genérico para o gerenciamento de recursos e tarefas de usuários. Essa documentação foi desenvolvida por um profissional da área. Reggio et al. (2018) utilizaram a documentação do ACME em sua pesquisa, a qual está relacionada com este trabalho. Foram utilizadas algumas das descrições dos casos de uso desse projeto.

Após a seleção desses documentos, os requisitos foram preparados. Como o projeto ACME apresenta uma quantidade maior de requisitos, foi selecionado um subconjunto de casos de usos das primeiras funções apresentadas no documento, pois são funções principais e não triviais. No experimento foram considerados os fluxos básicos e alternativos dos casos de uso. Para os casos de uso dos projetos *Data Hub* e *Planning Poker*, os atores foram adicionados nos fluxos básicos, como no formato dos casos de uso do ACME. A Tabela 5.1 apresenta algumas características desses documentos.

Para calcular a precisão da ferramenta foram utilizadas as métricas para os classificadores binários, apresentadas no Capítulo 3. Para isso, foi realizada a análise manual desses requisitos originais, conforme a descrição nas subseções 5.2.1 e 5.2.2

[2]https://www.planningpoker.com

Tabela 5.1: Métricas das especificações de software utilizadas nos experimentos.

| Itens | Data Hub | Planning Poker | ACME |
|---|---|---|---|
| Quantidade de estórias de usuário | 24 | 21 | - |
| Quantidade de diagramas de caso de uso | 4 | 3 | 8 |
| Quantidade de linhas em estórias de usuário | 38 | 40 | - |
| Quantidade de linhas em casos de uso | 39 | 36 | 66 |

Fonte: Adaptado de Dalpiaz e Sturm (2020a).

### 5.2.1 Relatório de erros

Essa avaliação visa descobrir a eficiência da primeira parte da ferramenta que é a identificação de erros nos requisitos. Os erros de ausência de informações, uso de sinônimos e inconsistências apontados automaticamente pela ferramenta AutRQ foram comparados com uma inspeção manual.

Para isso, a pesquisadora desta tese revisou os requisitos originais, dos projetos *Data Hub*, *Planning Poker* e ACME, manualmente e anotou as frases com esses erros relatados. Após essa verificação, os requisitos foram validados pela ferramenta. Os resultados obtidos (inspeção manual e automática) foram analisados por meio das métricas de precisão, revocação e f-*measure*. Nesse contexto, foi avaliado: se os erros reportados pela ferramenta existiam na revisão manual (verdadeiro positivo); se a ferramenta apontou problemas que não foram identificados na revisão manual (falso positivo); e se a ferramenta não identificou algum problema apresentado na revisão manual (falso negativo).

A Tabela 5.2 apresenta os resultados das métricas de avaliação para todos os erros validados nos requisitos. Pode-se observar que a ferramenta alcançou resultados satisfatórios, mesmo com requisitos em diferentes formatos e complexidade, com exceção do *Planning Poker*. O valor de precisão da avaliação dos casos de uso desse projeto foi abaixo do esperado e com isso permite investigar os possíveis fatores causadores desse resultado e incrementar melhorias na solução proposta. Como visto na Tabela 5.3, a qual detalha os resultados de cada etapa de avaliação, o valor obtido para a identificação de informações ausentes foi 40% de precisão. O mesmo problema ocorreu para o ACME que teve uma precisão de 73%. Isso se deve aos erros no PLN, como em alguns casos de conjunções que continham verbos presentes e foram classificadas incorretamente. Nesses casos o PLN não identificou o verbo da sentença.

Um problema durante a avaliação dos resultados foi na etapa de identificação de informações ausentes dos casos de uso do *Data Hub*, pois não foram identificados erros nem

na revisão manual e nem na validação automática. Por essa razão, os valores de precisão e revocação foram 0%, ou seja, não existia um valor de verdadeiro positivo.

Tabela 5.2: Resultados em porcentagem do desempenho do AutRQ para a identificação de erros.

| **Métricas** | ***Data Hub*** | | ***Planning Poker*** | | **ACME** |
|---|---|---|---|---|---|
| | **Estórias de Usuário** | **Casos de Uso** | **Estórias de Usuário** | **Casos de Uso** | **Casos de Uso** |
| **Precisão** | 84 | 82 | 76 | 66 | 89 |
| **Revocação** | 84 | 95 | 97 | 88 | 92 |
| **F-Measure** | 84 | 88 | 85 | 75 | 90 |

Fonte: Dados da Pesquisa. Elaborado pela autora (2021).

Tabela 5.3: Resultados em porcentagem detalhados de cada etapa do processo de identificação de erros.

| **Tipos** | **M** | ***Data Hub*** | | ***Planning Poker*** | | **ACME** | |
|---|---|---|---|---|---|---|---|
| | | **P** | **R** | **P** | **R** | **P** | **R** |
| **Estórias de usuários** | **AI** | 80 | 89 | 90 | 100 | - | - |
| | **S** | 83 | 80 | 70 | 95 | - | - |
| | **I** | 100 | 100 | 82 | 100 | - | - |
| **Casos de uso** | **AI** | 0 | 0 | 40 | 100 | 73 | 100 |
| | **S** | 100 | 92 | 72 | 81 | 89 | 76 |
| | **I** | 80 | 100 | 100 | 100 | 98 | 96 |

Fonte: Dados da Pesquisa. Elaborado pela autora (2021).
Notas: Verificações: (AI) Ausência de Informações, (S) Sinônimos e (I) Inconsistências.
Métricas (M): (P) Precisão e (R) Revocação.

Outras análises foram feitas baseadas na Tabela 5.3, como os valores entre 70-76% na etapa de verificação de sinônimos. A ferramenta AutRQ utiliza um banco de dados lexical genérico para a identificação de termos semelhantes, desse modo, esse resultado é derivado dos elementos presentes no corpus do banco (nesse caso as palavras). Alguns termos investigados na análise manual não foram identificados pela ferramenta e outros termos apontados pela ferramenta não eram sinônimos para o contexto do requisito, tornando necessário considerar o domínio do projeto.

Com relação à identificação de inconsistências, a ferramenta alcançou resultados positivos de precisão e revocação. Como o processo de revisão é sequencial, nessa etapa os requisitos estão mais completos e, com a redução do uso de sinônimos, a ferramenta consegue identificar as duplicações.

### 5.2.2 Modelo conceitual

A avaliação do modelo conceitual foi focada no desempenho da ferramenta AutRQ para representar os requisitos em atores, ações/estados e atributos do software. Neste estudo, considerou-se como um modelo conceitual a representação dos requisitos em um nível mais alto de abstração. Utilizou-se os mesmos requisitos dos projetos *Data Hub*, *Planning Poker* e ACME para a análise.

Os requisitos originais foram processados pela ferramenta. A pesquisadora desta tese preparou e submeteu esses requisitos para a representação em modelos conceituais. A ferramenta proporcionou a identificação de algumas deficiências relacionadas à qualidade da escrita dos requisitos nos casos de uso e estórias de usuário. O Quadro 5.3 apresenta alguns exemplos dos principais tipos de problemas observados. Em vários requisitos, os atores eram genéricos. Encontrou-se casos em que o ator foi descrito como sistema, plataforma, site, entre outros. Esses casos foram considerados como falhas, visto que de acordo com regras de qualidade de casos de uso, o sistema não deve ser representado com um ator (LILLY, 1999). Outra falha foi o uso de atores como atributos do software, como visto no quadro, os *estimators* são descritos como um complemento. Ademais, existiam frases irrelevantes para o contexto, como a segunda linha do quadro.

Quadro 5.3: Exemplos de problemas identificados nas especificações de software originais.

| **Subject** | **Action/state** | **Complement** |
|---|---|---|
| system<br>(sistema) | add<br>(adiciona) | estimators to game<br>(estimadores para o jogo) |
| estimator<br>(estimador) | back<br>(retorna) | to any of above operation<br>(para qualquer uma das operações acima) |

Elaborado pela autora (2021).

Inicialmente, esses requisitos e os modelos conceituais originais extraídos dos estudos de Dalpiaz e Sturm (2020a) e Reggio et al. (2018) seriam utilizados no experimento. No entanto, os problemas na qualidade dos requisitos reportados poderiam interferir nos resultados do experimento, pois o modelo conceitual gerado a partir de requisitos com falhas está sujeito a erros. Por essa razão, a pesquisadora reescreveu esses requisitos, seguindo regras de qualidade para casos de uso (LILLY, 1999) e para estórias de usuário (COHN, 2004b). Esses requisitos são descritos no Apêndice F. As tabelas semânticas geradas pela ferramenta são apresentadas no Apêndice H. Alguns ajustes foram realizados baseados na primeira

execução, como a eliminação da parte opcional *so that (some reason)* das estórias de usuário, já que existia a duplicação de informações nessa parte. Nesse mesmo sentido, nos casos de uso foram consideradas apenas as ações do fluxo dos atores. Após essa etapa, a pesquisadora produziu manualmente os modelos conceituais dos projetos a partir das especificações corrigidas (Apêndice G), os quais foram revisados pelo orientador desta tese. Cada um desses modelos foi utilizado como um modelo padrão-ouro para a comparação com os modelos produzidos pelo AutRQ.

Os modelos conceituais gerados pelo AutRQ dos requisitos corrigidos foram avaliados pelas métricas de precisão, revocação e f-*measure*. A pesquisadora comparou os dois modelos (padrão-ouro e o gerado pela ferramenta) e registrou os resultados. Foram analisadas as classes de atores, atributos e relacionamentos. Nesse contexto, o falso positivo consiste na identificação do AutRQ de uma classe ou relacionamento inexistente, enquanto um falso negativo seria a falta de identificação do AutRQ de uma classe ou relacionamento existente.

Os resultados obtidos nesta avaliação foram satisfatórios. Como visto na Tabela 5.4, grande parte dos resultados da precisão estavam entre 88% - 100%. Os resultados da revocação também foram positivos, apenas em um caso o valor foi inferior (75%). A principal razão desse valor está relacionada com o modelo conceitual criado manualmente que agrupa as estórias de usuário e os casos de uso. Os documentos usados foram revisados pelos autores Dalpiaz e Sturm (2020a) para alinhar seus conteúdos e o nível de granularidade para que um modelo pudesse abranger esses diferentes documentos. Desse modo, em algumas situações existiam relacionamentos específicos de cada documento e na avaliação isso foi considerado, como por exemplo, o requisito "*Moderator adds accounts to the game*" do projeto *Planning Poker* está contido no documento de casos de uso e não nas estórias de usuário, porém, quando as estórias foram processadas, o cálculo de revocação foi afetado, pois existia esse relacionamento no modelo.

O problema principal identificado que afetou alguns casos de precisão foi o fato de que alguns documentos incluíram requisitos não funcionais com a descrição de atributos focados em parâmetros como desempenho e tempo. Nesse cenário, a classificação de atributos falhou, como por exemplo o requisito "*As an estimator, I want to finish the round with a two-minute countdown timer*". É possível observar que a precisão das classes e relacionamentos de casos de uso foram melhores em relação às estórias de usuário e isso confirma o problema para

processar esses tipos de atributos, pois como as estórias de usuário apresentam mais detalhes nos requisitos, existem mais ocorrências desses parâmetros de requisitos não funcionais.

O resultado da média harmônica da precisão e revocação (f-*measure*) é incentivador, com destaque para o resultado da identificação das classes de 99% e o limite inferior de 92%. O resultado maior para a identificação de classes em comparação com os relacionamentos já era esperado, pois como apontado pelos autores Lucassen, Robeer et al. (2017), a identificação de um relacionamento depende além da detecção correta da relação, as classes (origem) e seu atributo (destino).

Tabela 5.4: Resultados em porcentagem da avaliação do desempenho do AutRQ para a geração de modelos conceituais.

| **Tipo** | **M** | ***Planning Poker*** | | ***Data Hub*** | | **ACME** |
|---|---|---|---|---|---|---|
| | | **Estória de usuário** | **Casos de Uso** | **Estória de usuário** | **Casos de Uso** | **Casos de Uso** |
| **Classes** | **P** | 93 | 98 | 92 | 98 | 98 |
| | **R** | 97 | 100 | 93 | 100 | 100 |
| | **F** | 95 | 99 | 92 | 99 | 99 |
| **Relações** | **P** | 94 | 100 | 88 | 100 | 96 |
| | **R** | 75 | 91 | 85 | 96 | 100 |
| | **F** | 83 | 95 | 87 | 98 | 98 |

Fonte: Dados da Pesquisa. Elaborado pela autora (2021).
(M) Métricas: (P) Precisão, (R) Revocação e (F) F-*Measure*.

## 5.3 Discussão a respeito dos resultados

O propósito da prova de conceito e os experimentos conduzidos nesta pesquisa foram validar na prática como o processo contribui com a mitigação de erros e qual o seu desempenho considerando diferentes documentos de requisitos. Baseado na avaliação dos especialistas de projetos de software, o processo mostrou ser eficaz para apoiar as tarefas de engenharia de software. Nesse sentido, o processo pode ser aplicado como uma tecnologia da engenharia de software assistida por computador, a qual tem como objetivo semi-automatizar tarefas como a obtenção e análise dos requisitos (DIAMANTOPOULOS et al., 2017). Uma das principais contribuições declaradas foi a melhoria na qualidade da escrita dos requisitos, devido à revisão de erros durante a produção dos requisitos. O processo contribuiu ainda com a clareza na comunicação e a compreensão das especificações e indica a aceitação da hipótese

H2. Os especialistas extraíram novos requisitos e identificaram relações tácitas. Em virtude desses resultados, existem evidências que a hipótese H0 foi rejeitada, pois o processo implicou na mitigação de problemas de interpretabilidade.

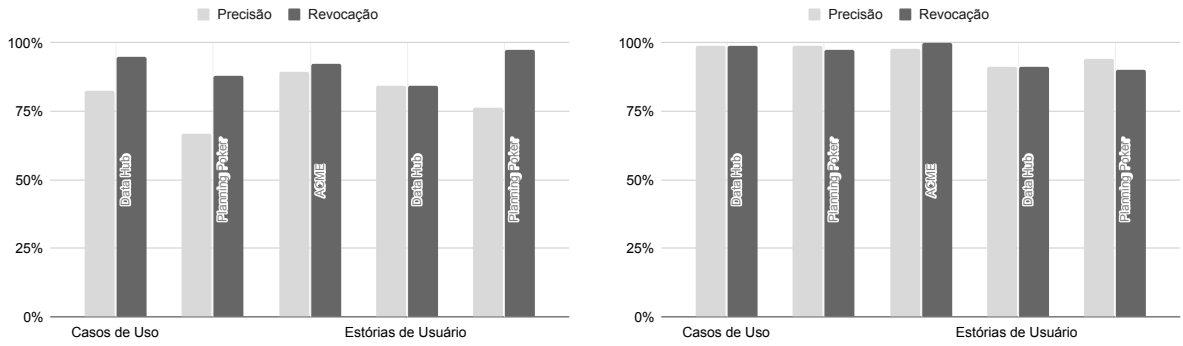
Em termos de aplicação, os conceitos propostos no processo foram avaliados por meio do protótipo. A capacidade de manipulação de informações que o protótipo oferece pode facilitar a organização dos requisitos, mesmo enfrentando mudanças no projeto. Alguns desafios foram associados à interface da ferramenta, no entanto, o protótipo pode ser facilmente aprimorado com novas atualizações.

Os resultados de desempenho foram satisfatórios em requisitos de diferentes contextos. A precisão e revocação do protótipo na identificação de problemas reais em especificação de software e a geração de modelos conceituais foram positivos. Em termos na detecção de problemas, além da identificação de informações ausentes, uso de conceitos sinônimos e inconsistências, o protótipo revelou algumas falhas na criação de casos de uso e das estórias de usuário. Essa contribuição permite que os engenheiros de software possam se atentar a esses problemas e reduzir consequências de problemas no desenvolvimento. Como relatado por Dalpiaz, Schalk et al. (2019), a análise de falhas como a ambiguidade, exige um alto investimento e uma incerteza de seu retorno. Desse modo, uma solução viável pode ser a implementação de ferramentas que detectem falhas durante a criação dos requisitos.

O processo permitiu uma melhoria tanto na identificação de problemas como na modelagem das informações. Foi possível observar que existia a duplicação de informações na parte opcional <razão> das estórias de usuário. Nesse sentido, as estórias de usuário escritas em um modelo simples sem descrições extensas de soluções evita a redundância nos requisitos e resultados melhores podem ser obtidos na análise de PLN (LUCASSEN; ROBEER et al., 2017). Essa informação se corrobora nos experimentos, como visto na Figura 5.2, a qual sintetiza os resultados dos dois experimentos, pois os valores para a geração dos modelos conceituais para estórias de usuário foram inferiores aos casos de uso.

A precisão e a revocação foram melhores para a segunda etapa do processo (geração do modelo conceitual). Essa melhoria na precisão indica evidências para a aceitação da H1, visto que na segunda etapa os requisitos foram estruturados, validados e representados de forma automática. Devido a estruturação das informações extraídas dos requisitos e sua validação, o protótipo classificou corretamente nos modelos os atores, relacionamentos e atributos. Observou-se apenas uma limitação para requisitos do tipo não-funcional, por exemplo, dados

Figura 5.2: Comparação da média da avaliação dos experimentos realizados.




(a) Identificação de erros

(b) Geração do modelo conceitual

Elaborado pela autora (2021)

temporais foram classificados como atributos de uma determinada entidade. Em comparação com os modelos originais do projetos *Data Hub* e *Planning Poker*, apresentados no estudo de Dalpiaz e Sturm (2020a), o modelo gerado pelo protótipo foi mais completo, pois a ferramenta extraiu um número maior de classes e relacionamentos, como observado na Tabela 5.5. Ademais, reduziu-se redundância, por exemplo, como encontrado em duas relações no modelo original: "*Publisher registers to the platform/site*" e "*Publisher opens an account*".

Tabela 5.5: Comparação dos modelos originais com os modelos gerados pelo AutRQ.

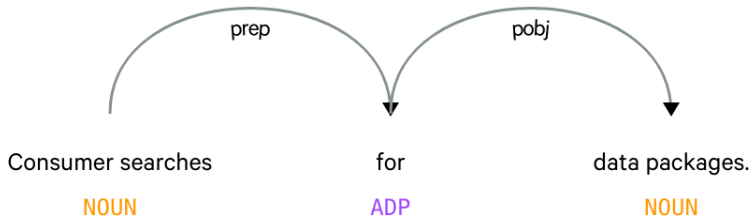
| **Tipo** | **Modelo Conceitual AutRQ** | | **Modelo Conceitual Original** | |
|---|---|---|---|---|
| | ***Data Hub*** | ***Planning Poker*** | ***Data Hub*** | ***Planning Poker*** |
| Classes | 12 | 9 | 10 | 6 |
| Relações | 23 | 23 | 13 | 8 |

Fonte: Dados da Pesquisa. Elaborado pela autora (2021).

Embora os valores de desempenho obtidos na identificação de erros sejam menos precisos, ainda são superiores a ferramentas similares como no estudo de Dalpiaz, Schalk et al. (2019). Como na primeira etapa do experimento, a avaliação do relatório de erros, os requisitos utilizados foram os originais, existiam erros na escrita que interferiram no PLN. Um exemplo de erro no PLN foi a classificação de POS-*tagging* - unidade linguística, a qual atribuiu verbo como substantivo, como exemplo a sentença ilustrada na Figura 5.3 mostra esse problema, na qual o verbo *search* (pesquisar, em português) foi atribuído como um substantivo (*noun*) e, desse modo, afetou a análise sintático/semântica, pois o núcleo

identificado da sentença foi a preposição "*for*". Esse tipo de problema, introduz erros na aplicação de regras do *framework*. Os resultados dessa etapa são afetados pelo modelo utilizado no algoritmo de classificação do serviço externo implementado no protótipo. Esse tipo de problema pode ser controlado com a utilização de outros modelos de dados personalizados.

Figura 5.3: Exemplo de dependências gramaticais da biblioteca spaCy.




Fonte: *Print screen* da classificação do spaCy pelo Explosion. Disponível em: https://explosion.ai/demos/displacy.

## 5.4 Considerações finais

Este capítulo visou mostrar como o processo de representação estruturada e validação de requisitos foi avaliado na prática. Descreveu-se os passos realizados em duas etapas de avaliação, constituídas pela prova de conceito com especialistas em projetos de software e pelos experimentos em requisitos de documentos públicos de três projetos de diferentes domínios. Foram detalhados o cenário deste estudo e dos experimentos, como também os critérios usados na avaliação.

O estudo visou explorar as percepções dos especialistas em relação à aplicação do processo em requisitos em linguagem natural. Abordou-se em um questionário os tópicos associados à mitigação de problemas e análise dos requisitos. Do ponto de vista da automatização, os experimentos investigaram a capacidade de desempenho para a detecção de erros e geração de modelo conceitual, por meio de métricas de precisão e revocação.

Os dados obtidos foram analisados por técnicas de análise de conteúdo e de estatística descritiva e foram relatados em uma ampla descrição. De modo geral, a utilização do processo em um cenário real revelaram resultados promissores, os quais foram discutidos neste capítulo.

# Capítulo 6

# Conclusão

A subjetividade relativa da língua natural, a qual é normalmente utilizada em especificações de software, é imprecisa e pode ocasionar problemas associados com a compreensão e comunicação das funcionalidades do software e, desse modo, pode comprometer sua qualidade. A correção manual dessas deficiências está sujeita a falhas. Diante desse fato, algumas soluções automáticas foram propostas na literatura. Apesar disso, a maioria das abordagens apresentadas concentra-se na avaliação da especificação de software e a falta de padronização dos requisitos pode afetar o seu processamento.

Alguns estudos representaram os requisitos de forma estruturada e validaram sua qualidade. Embora sejam poucos estudos, a validação dos requisitos foi eficaz. Todavia, a implantação dessas abordagens propostas é complexa devido ao seu nível de formalização e as regras de restrições impostas. Essas limitações podem ser reparadas com o uso de análise semântica.

Neste contexto, este estudo apresenta um processo automático que visa uma representação estruturada semanticamente e validação de requisitos de software. Os conceitos estabelecidos neste processo foram implantados em um *framework* e em um protótipo de uma ferramenta Web. Utilizaram-se tecnologias de Web Semântica e PLN.

A avaliação deste processo foi realizada por meio de uma prova de conceito com dez especialistas em projetos de software e dois experimentos em requisitos extraídos de bases públicas, para a investigação da geração do modelo conceitual e o relatório de erros. Foi possível constatar que o processo automático pode ser aplicado em diferentes domínios e mostrou ser eficaz para controlar problemas de compreensão e comunicação dos requisitos.

A mitigação desses problemas forneceu a descoberta de novos requisitos. Além disso, observaram-se melhorias na escrita dos requisitos.

O desempenho do processo automático foi positivo, apresentando uma média de 95% de precisão para a geração do modelo conceitual e 79% para a identificação de erros. Os modelos de requisitos gerados pela ferramenta foram mais completos e precisos. A ferramenta proporcionou a identificação de erros e requisitos vagos.

A ferramenta mostrou facilidade de uso, pois lida com uma formatação livre do texto, não requer nenhum vocabulário ou glossário de termos e não exige conhecimentos técnicos para sua aplicação. Também apoiou a etapa de análise de requisitos com a gestão das informações, como também a melhoria de tempo e esforços manuais para a detecção e correção de problemas no estágio inicial do projeto. A reparação de problemas no início do projeto pode reduzir significativamente o retrabalho das tarefas, testes, custos e tempo, evitando desafios maiores ou falhas no projeto. Foram obtidos ganhos com a automatização, como a criação de forma automática de bases de conhecimento que proporcionou a integração e a recuperação de informações estruturadas extraídas dos requisitos. A disponibilidade das informações em um armazenamento central permitiu o rápido acesso a essas informações e pode ser um apoio para a reutilização de software.

A principal contribuição deste estudo consistiu na redução de problemas na interpretação de requisitos em linguagem natural durante a fase de criação de requisitos, tornando-os mais precisos. Perante as dificuldades e incertezas enfrentadas pelas equipes de desenvolvimento de software com requisitos de software expressos em linguagem natural, as contribuições evidenciadas da avaliação do processo de representação estruturada e validação dos requisitos encoraja sua adoção em projetos de diferentes âmbitos.

## 6.1 Limitações e trabalhos futuros

Algumas limitações foram identificadas neste estudo e serão alvo de trabalhos futuros. O desempenho desse processo automático para lidar com grande quantidade de requisitos deve ser investigado. Ainda, a abrangência para requisitos não funcionais também deve ser explorada, como requisitos que envolvem atributos de tempo e desempenho, visto que esta pesquisa lida com requisitos do tipo funcional.

Em relação ao protótipo, resultados melhores poderão ser obtidos com a utilização de redes específicas para o contexto de cada domínio. Outra limitação é que a ferramenta só abrange as inconsistências pré-definidas nas regras deste estudo. A incorporação de novas regras poderá ser suportada com a utilização de outras ferramentas como Protégé (MUSEN, 2015). Como o protótipo proporciona a exportação da base de conhecimento no formato RDF, podem-se criar regras específicas para um domínio como no estudo de Bayoudhi, Sassi e Jaziri (2018). Por meio dessa exportação é possível incorporar facilmente os resultados da aplicação em outras ferramentas.

Um estudo sobre a interface do usuário deve ser realizado para melhorar a usabilidade do protótipo. Em novas versões serão revisados os fluxos das funcionalidades para prover uma melhor experiência de uso para os usuários. Além disso, será implementado um versionamento dos requisitos, visando manter o histórico de erros, bem como a documentação original.

Salienta-se que o processo automático será aprimorado para processar outros idiomas, como o português. Para suportar a internalização nos modelos, deve-se contar com a atualização das tecnologias de processamento de linguagem natural e uma nova estrutura para a modelagem do conhecimento.

O protótipo desenvolvido em uma versão virtualizada será aprimorado para uma versão de ferramenta que possa ser instalada localmente em computadores. Nesse ambiente, as empresas poderão manter uma base de conhecimento única, evitando duplicações e o retrabalho na definição de requisitos de seus projetos.

Os requisitos processados pela ferramenta poderão ser submetidos em um ambiente colaborativo, como os servidores RDF para tornar o conhecimento distribuído e permitir a consulta desses requisitos de qualquer dispositivo. Nesse contexto, como parte dos trabalhos futuros, será investigada a reutilização dos requisitos disponibilizados nesse modelo de representação aberto. A possibilidade de rastrear esses requisitos pode evitar omissões presentes em novos sistemas. Além disso, será possível referenciar esses requisitos e contribuir para a redução de casos em que os requisitos possam ser utilizados de forma parcial ou integral sem o reconhecimento da fonte.

# Referências bibliográficas

ADOLPH, S.; BRAMBLE, P.; COCKBURN, A.; POLS, A. **Patterns for Effective Use Cases**. [S.l.]: Addison-Wesley, 2003. ISBN 9780201721843.

ALVES, L. M.; SOUZA, G.; RIBEIRO, P.; MACHADO, R. J. Longevity of risks in software development projects: a comparative analysis with an academic environment. **Procedia Computer Science**, v. 181, p. 827–834, 2021. CENTERIS 2020 - International Conference on ENTERprise Information Systems / ProjMAN 2020 - International Conference on Project MANagement / HCist 2020 - International Conference on Health and Social Care Information Systems and Technologies 2020, CENTERIS/ProjMAN/HCist 2020. ISSN 1877-0509. DOI: 10.1016/j.procs.2021.01.236. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921002799>.

ANTINYAN, V.; STARON, M. Rendex: A method for automated reviews of textual requirements. **Journal of Systems and Software**, v. 131, p. 63–77, 2017. DOI: 10.1016/j.jss.2017.05.079.

APAZA, R. D. G.; BARRIOS, J. E. M.; BECERRA, D. A. I.; QUISPE, J. A. H. ERS-TOOL: Hybrid Model for Software Requirements Elicitation in Spanish Language. In: PROCEEDINGS of the International Conference on Geoinformatics and Data Analysis. New York, NY, USA: Association for Computing Machinery, 2018. (ICGDA '18), p. 27–30. ISBN 9781450364454. DOI: 10.1145/3220228.3220255.

ASHFAQ, F.; BAJWA, I. S. Natural language ambiguity resolution by intelligent semantic annotation of software requirements. **Automated Software Engineering**, v. 28, n. 2, p. 13, nov. 2021. ISSN 0928-8910. DOI: 10.1007/s10515-021-00291-0. Disponível em: <https://link.springer.com/10.1007/s10515-021-00291-0>.

ATOUM, I. A Scalable Operational Framework for Requirements Validation Using Semantic and Functional Models. In: PROCEEDINGS of the 2nd International Conference on Software Engineering and Information Management. New York, NY, USA: Association for Computing Machinery, 2019. (ICSIM 2019), p. 1–6. ISBN 9781450366427. DOI: 10.1145/3305160.3305166.

BAJWA, I. S.; BORDBAR, B.; LEE, M. OCL Usability: A Major Challenge in Adopting UML. In: PROCEEDINGS of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. New York, NY, USA: ACM, 2014. (RAISE 2014), p. 32–37. DOI: 10.1145/2593801.2593807.

BAKAR, N. H.; KASIRUN, Z. M.; SALLEH, N. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review.

**Journal of Systems and Software**, Elsevier Ltd., v. 106, p. 132–149, ago. 2015. DOI: 10.1016/j.jss.2015.05.006.

BARCELOS, L. V.; PENTEADO, R. D. Elaboration of software requirements documents by means of patterns instantiation. **Journal of Software Engineering Research and Development**, v. 5, n. 1, p. 3, dez. 2017. ISSN 2195-1721. DOI: 10.1186/s40411-017-0038-9. Disponível em: <http://jserd.springeropen.com/articles/10.1186/s40411-017-0038-9>.

BAYOUDHI, L.; SASSI, N.; JAZIRI, W. How to Repair Inconsistency in OWL 2 DL Ontology Versions? **Data Knowledge Engineering**, v. 116, p. 138–158, 2018. ISSN 0169-023X. DOI: 10.1016/j.datak.2018.05.010.

BECK, K. **Extreme programming explained: embrace change**. [S.l.]: addison-wesley professional, 2000.

BHATIA, M. P. S.; KUMAR, A.; BENIWAL, R. Ontology based framework for detecting ambiguities in software requirements specification. In: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). [S.l.: s.n.], 2016. p. 3572–3575.

BJØRNER, D. The vienna development method (VDM). In: MATHEMATICAL Studies of Information Processing. [S.l.]: Springer, 1979. p. 326–359.

BOETTIGER, C. **rdflib: A high level wrapper around the redland package for common rdf applications**. [S.l.], 2018. DOI: 10.5281/zenodo.1098478.

BRAUN, V.; CLARKE, V. Using thematic analysis in psychology. **Qualitative Research in Psychology**, v. 3, n. 2, p. 77–101, jan. 2006. ISSN 1478-0887. DOI: 10.1191/1478088706qp063oa. Disponível em: <http://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa>.

CAMPOS, J. P.; BRAGA, J. L.; RESENDE, A. M. P.; SILVA, C. H. O. Identification of aspect candidates by inspecting use cases descriptions. **ACM SIGSOFT Software Engineering Notes**, v. 35, n. 4, p. 1, jul. 2010. DOI: 10.1145/1811226.1811231.

CASTRO, M. M. H.; BEZERRA, J. M.; HIRATA, C. M. A CNL for requirements as the basis to automate tasks of critical system development. In: 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC). [S.l.]: IEEE, set. 2015. p. 8c2–1–8c2–9. DOI: 10.1109/DASC.2015.7311474.

CHERNAK, Y. Requirements Reuse: The State of the Practice. In: 2012 IEEE International Conference on Software Science, Technology and Engineering. [S.l.: s.n.], 2012. p. 46–53. DOI: 10.1109/SWSTE.2012.12.

CHIERCHIA, G. **Semântica**. Edição: Lígia Negri e Rodolfo Ilari. Tradução: Luiz Arthur Pagani. Campinas e Londrina: Editora da Unicamp e Editora da UEL, 2003.

COCKBURN, A. **Writing Effective Use Cases**. [S.l.]: Addison-Wesley, 2001.

COHN, M. **Advantages of the "As a user, I want" user story template**. [S.l.: s.n.], 2008. Disponível em: <http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>. Acesso em: 4 nov. 2016.

COHN, M. **Example User Stories**. [S.l.: s.n.], 2004. Disponível em: <https://www.mountaingoatsoftware.com/uploads/documents/example-user-stories.pdf>.

COHN, M. **User Stories Applied: For Agile Software Development**. [S.l.]: Addison-Wesley, 2004. (Addison-Wesley signature series).

CONSORTIUM, W. W. W. RDF 1.1 concepts and abstract syntax. World Wide Web Consortium, 2014.

COUTO, R.; RIBEIRO, A. N.; CAMPOS, J. C. A Study on the Viability of Formalizing Use Cases. In: 2014 9th International Conference on the Quality of Information and Communications Technology. [S.l.]: IEEE, set. 2014. p. 130–133. DOI: 10.1109/QUATIC.2014.23.

DAL FORNO, G. M. B.; MULLER, F. M. Fatores Críticos em Projetos de Desenvolvimento de Software. **Revista Pretexto**, v. 18, n. 2, p. 100–115, set. 2017. DOI: 10.21714/pretexto.v18i2.5295.

DALPIAZ, F. **Requirements data sets (user stories)**. [S.l.: s.n.], 2018. DOI: 10.17632/7zbk8zsd8y.1.

DALPIAZ, F.; SCHALK, I. van der; BRINKKEMPER, S.; AYDEMIR, F. B.; LUCASSEN, G. Detecting terminological ambiguity in user stories: Tool and experimentation. **Information and Software Technology**, v. 110, p. 3–16, jun. 2019. ISSN 09505849. DOI: 10.1016/j.infsof.2018.12.007. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0950584918300715>.

DALPIAZ, F.; STURM, A. Conceptualizing Requirements Using User Stories and Use Cases: A Controlled Experiment. In: MADHAVJI, N.; PASQUALE, L.; FERRARI, A.; GNESI, S. (Ed.). **Requirements Engineering: Foundation for Software Quality**. Cham: Springer International Publishing, 2020. p. 221–238. ISBN 978-3-030-44429-7.

DALPIAZ, F.; STURM, A. **Experiment User Stories vs. Use Cases**. [S.l.]: Utrecht University, 2020. DOI: 10.23644/uu.c.4815591.v1. Disponível em: <https://uu.figshare.com/collections/Experiment%7B%5C_%7DUser%7B%5C_%7DStories%7B%5C_%7Dvs%7B%5C_%7DUse%7B%5C_%7DCases/4815591>. Acesso em: 1 nov. 2020.

DAVIES, R. The Power of Stories. **XP 2001**, ago. 2001.

DIAMANTOPOULOS, T.; ROTH, M.; SYMEONIDIS, A.; KLEIN, E. Software requirements as an application domain for natural language processing. **Language Resources and Evaluation**, v. 51, n. 2, p. 495–524, 2017. DOI: 10.1007/s10579-017-9381-z.

DJILANI, Z.; KHOURI, S. Understanding User Requirements Iceberg: Semantic Based Approach. In: MODEL and Data Engineering. Cham: Springer International Publishing, 2015. p. 297–310.

DORIGAN, J. A.; BARROS, R. M. A Process Model for Standardization and Increase in the Requirements Quality. **IEEE Latin America Transactions**, v. 12, n. 8, p. 1502–1507, 2014. DOI: 10.1109/TLA.2014.7014520.

DUGGAL, M.; SAXENA, N.; GURVE, M. SRS Automator - An Attempt to Simplify Software Development Lifecycle. In: 2020 6th International Conference on Signal Processing and

Communication (ICSC). [S.l.: s.n.], 2020. p. 278–283. DOI: 10.1109/ICSC48311.2020.9182768.

ECKHARDT, J.; VOGELSANG, A.; FEMMER, H.; MAGER, P. Challenging Incompleteness of Performance Requirements by Sentence Patterns. In: 2016 IEEE 24th International Requirements Engineering Conference (RE). [S.l.: s.n.], 2016. p. 46–55. DOI: 10.1109/RE.2016.24.

ESCALONA, M. J.; URBIETA, M.; ROSSI, G.; GARCIA-GARCIA, J. A.; LUNA, E. R. Detecting Web requirements conflicts and inconsistencies under a model-based perspective. **Journal of Systems and Software**, v. 86, n. 12, p. 3024–3038, 2013. DOI: 10.1016/j.jss.2013.05.045.

FANTECHI, A.; GNESI, S.; SEMINI, L. Ambiguity Defects as Variation Points in Requirements. In: PROCEEDINGS of the Eleventh International Workshop on Variability Modelling of Software-Intensive Systems. New York, NY, USA: Association for Computing Machinery, 2017. (VAMOS '17), p. 13–19. ISBN 9781450348119. DOI: 10.1145/3023956.3023964.

FARFELEDER, S.; MOSER, T.; KRALL, A.; STÅLHANE, T.; OMORONYIA, I.; ZOJER, H. **Ontology-Driven Guidance for Requirements Elicitation**. Edição: Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer e Jeff Pan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 212–226. ISBN 978-3-642-21064-8.

FEMMER, H.; MÉNDEZ FERNÁNDEZ, D.; WAGNER, S.; EDER, S. Rapid quality assurance with Requirements Smells. **Journal of Systems and Software**, v. 123, p. 190–213, 2017. DOI: 10.1016/j.jss.2016.02.047.

FERNANDES, J. M.; MACHADO, R. J. **Requisitos em projetos de software e de sistemas de informação**. [S.l.]: NOVATEC, 2017.

FERRARI, A.; DELL'ORLETTA, F.; ESULI, A.; GERVASI, V.; GNESI, S. Natural Language Requirements Processing: A 4D Vision. **IEEE Software**, v. 34, n. 6, p. 28–35, 2017. DOI: 10.1109/MS.2017.4121207.

FERRARI, A.; SPAGNOLO, G. O.; FISCELLA, A.; PARENTE, G. QuOD: An NLP Tool to Improve the Quality of Business Process Descriptions. In: BEEK, M. H. ter; FANTECHI, A.; SEMINI, L. (Ed.). **From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday**. Cham: Springer International Publishing, 2019. p. 267–281. ISBN 978-3-030-30985-5. DOI: 10.1007/978-3-030-30985-5_17. Disponível em: <https://doi.org/10.1007/978-3-030-30985-5_17>.

FERREIRA, D. A.; SILVA, A. R. RSL-IL: An interlingua for formally documenting requirements. In: 2013 3rd International Workshop on Model-Driven Requirements Engineering (MoDRE). [S.l.: s.n.], 2013. p. 40–49. DOI: 10.1109/MoDRE.2013.6597262.

FERREIRA, D. A.; SILVA, A. R. RSLingo: An information extraction approach toward formal requirements specifications. In: 2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE). [S.l.]: IEEE, set. 2012. p. 39–48. DOI: 10.1109/MoDRE.2012.6360073.

FOCKEL, M.; HOLTMANN, J. ReqPat: Efficient documentation of high-quality requirements using controlled natural language. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE). [S.l.: s.n.], 2015. p. 280–281. DOI: 10.1109/RE.2015.7320438.

FRAGA, A.; LLORENS, J.; ALONSO, L.; FUENTES, J. M. Ontology-Assisted Systems Engineering Process with Focus in the Requirements Engineering Process. In: COMPLEX Systems Design & Management. Cham: Springer International Publishing, 2015. p. 149–161. DOI: 10.1007/978-3-319-11617-4_11. Disponível em: <http://link.springer.com/10.1007/978-3-319-11617-4%7B%5C_%7D11>.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML: the systems modeling language**. [S.l.]: Morgan Kaufmann, 2014.

GANESH, N.; THANGASAMY, S. Issues identified in the software process due to barriers found during eliciting requirements on agile software projects: Insights from India. **International Journal of Computer Applications**, v. 16, n. 5, p. 7–12, fev. 2011. DOI: 10.5120/2011-2713.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2002. v. 4.

GOKNIL, A.; KURTEV, I.; BERG, K. V. D. Generation and validation of traces between requirements and architecture based on formal trace semantics. **Journal of Systems and Software**, v. 88, p. 112–137, 2014. DOI: 10.1016/j.jss.2013.10.006.

GRAMAJO, M. G.; BALLEJOS, L.; ALE, M. **Recurrent Neural Networks to automate Quality assessment of Software Requirements**. [S.l.: s.n.], 2021. arXiv: 2105.04757 [cs.SE].

GUEDES, G. T. A. **UML 2 - Uma Abordagem Prática - 1ª Edição:** [s.l.]: NOVATEC, 2011.

GUO, W.; ZHANG, L.; LIAN, X. **Automatically detecting the conflicts between software requirements based on finer semantic analysis**. [S.l.: s.n.], 2021. arXiv: 2103.02255 [cs.SE].

HASTIE, S.; WOJEWODA, S. **Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch**. [S.l.: s.n.], 2015. Disponível em: <https://www.infoq.com/articles/standish-chaos-2015>. Acesso em: 24 dez. 2018.

HAYES, J. A graph model for RDF. **Darmstadt University of Technology/University of Chile**, Citeseer, 2004.

HAYRAPETIAN, A.; RAJE, R. Empirically Analyzing and Evaluating Security Features in Software Requirements. In: PROCEEDINGS of the 11th Innovations in Software Engineering Conference. New York, NY, USA: Association for Computing Machinery, 2018. (ISEC '18). ISBN 9781450363983. DOI: 10.1145/3172871.3172879.

HIRAMA, K. **Engenharia de software: Qualidade e produtividade com tecnologia**. Rio de Janeiro: Elsevier Editora Ltda., 2012.

HONNIBAL, M.; MONTANI, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear. [S.l.], 2017.

HOYOS, J. P. A.; RESTREPO-CALLE, F. Fast Prototyping of Web-Based Information Systems Using a Restricted Natural Language Specification. In: DAMIANI, E.; SPANOUDAKIS, G.; MACIASZEK, L. (Ed.). **Evaluation of Novel Approaches to Software Engineering**. Cham: Springer International Publishing, 2018. p. 183–207. DOI: 10.1007/978-3-319-94135-6_9.

HRIPCSAK, G. Agreement, the F-Measure, and Reliability in Information Retrieval. **Journal of the American Medical Informatics Association**, v. 12, n. 3, p. 296–298, jan. 2005. ISSN 1067-5027. DOI: 10.1197/jamia.M1733. Disponível em: <https://academic.oup.com/jamia/article-lookup/doi/10.1197/jamia.M1733>.

HUGHES, D. L.; DWIVEDI, Y. K.; SIMINTIRAS, A. C.; RANA, N. P. Success and Failure of IS/IT Projects: A State of the Art Analysis and Future Directions. In: PROJECT Failure and Its Contributing Factors. Cham: Springer International Publishing, 2016. p. 3–25. DOI: 10.1007/978-3-319-23000-9_2.

IEEE. IEEE Recommended Practice for Software Requirements Specifications. **IEEE Std 830-1998**, p. 1–40, 1998. DOI: 10.1109/IEEESTD.1998.88286.

IEEE. IEEE Standard Glossary of Software Engineering Terminology. **IEEE Std 610.12-1990**, p. 1–84, 1990. DOI: 10.1109/IEEESTD.1990.101064.

IEEE. Systems and software engineering - Vocabulary. **ISO/IEC/IEEE 24765:2010(E)**, p. 1–418, 2010.

IEEE. Systems and software engineering – Life cycle processes – Requirements engineering. **ISO/IEC/IEEE 29148:2018(E)**, p. 1–104, 2018.

JUE, W.; SONG, Y.; WU, X.; DAI, W. A Semi-Formal Requirement Modeling Pattern for Designing Industrial Cyber-Physical Systems. In: IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society. [S.l.]: IEEE, out. 2019. v. 1, p. 2883–2888. ISBN 978-1-7281-4878-6. DOI: 10.1109/IECON.2019.8926665. Disponível em: <https://ieeexplore.ieee.org/document/8926665/>.

KAMALRUDIN, M.; HOSKING, J.; GRUNDY, J. MaramaAIC: tool support for consistency management and validation of requirements. **Automated Software Engineering**, v. 24, n. 1, 2017. DOI: 10.1007/s10515-016-0192-z.

KHTIRA, A.; BENLARABI, A.; ASRI, B. Duplication Detection When Evolving Feature Models of Software Product Lines. **Information**, v. 6, n. 4, p. 592–612, out. 2015. DOI: 10.3390/info6040592.

LASSILA, O.; SWICK, R. R. Resource description framework (RDF) model and syntax specification. Citeseer, 1998.

LAUE, R.; KOOP, W.; GRUHN, V. Indicators for Open Issues in Business Process Models. In: DANEVA, M.; PASTOR, O. (Ed.). **Requirements Engineering: Foundation for Software Quality**. Cham: Springer International Publishing, 2016. p. 102–116.

LEBEAUPIN, B.; RAUZY, A.; ROUSSEL, J. A language proposition for system requirements. In: 2017 Annual IEEE International Systems Conference (SysCon). [S.l.: s.n.], 2017. p. 1–8. DOI: 10.1109/SYSCON.2017.7934808.

LI, W.; HAYES, J. H.; TRUSZCZYŃSKI, M. **Towards More Efficient Requirements Formalization: A Study**. Edição: Samuel A. Fricker e Kurt Schneider. Cham: Springer International Publishing, 2015. p. 181–197. ISBN 978-3-319-16101-3.

LILLY, S. Use case pitfalls: top 10 problems from real projects using use cases. In: PROCEEDINGS of Technology of Object-Oriented Languages and Systems - TOOLS 30 (Cat. No. PR00278). [S.l.: s.n.], 1999. p. 174–183. DOI: 10.1109/TOOLS.1999.787547.

LINKED DATA FINLAND. **Living laboratory data service for the semantic web**. [S.l.: s.n.], 2020. Acesso em: 5 mai. 2020.

LOEBNER, S. **Understanding Semantics**. [S.l.]: Taylor & Francis, 2014. (Understanding Language). ISBN 9781134647156.

LOPER, E.; BIRD, S. Nltk: The natural language toolkit. **arXiv preprint cs/0205028**, 2002.

LUCASSEN, G.; DALPIAZ, F.; WERF, J. M. E. M. van der; BRINKKEMPER, S. Improving agile requirements: the Quality User Story framework and tool. **Requirements Engineering**, v. 21, n. 3, p. 383–403, set. 2016. ISSN 0947-3602. DOI: 10.1007/s00766-016-0250-x.

LUCASSEN, G.; ROBEER, M.; DALPIAZ, F.; WERF, J. M. E. M. van der; BRINKKEMPER, S. Extracting conceptual models from user stories with Visual Narrator. **Requirements Engineering**, v. 22, n. 3, p. 339–358, set. 2017. ISSN 0947-3602. DOI: 10.1007/s00766-017-0270-1. Disponível em: <http://link.springer.com/10.1007/s00766-017-0270-1>.

LUND, A. **Use questionnaire resource page**. [S.l.: s.n.], 1998. Disponível em: <https://www.researchgate.net/profile/Arnold_Lund/publication/298069359_USE/data/56e5a94008ae65dd4cc0c6ab/USE.zip>. Acesso em: 31 ago. 2020.

MADALA, K.; DO, H.; ACEITUNA, D. A combinatorial approach for exposing off-nominal behaviors. In: PROCEEDINGS - International Conference on Software Engineering. [S.l.: s.n.], 2018. Part F1371, p. 910–920. DOI: 10.1145/3180155.3180204.

MAHMUD, N.; SECELEANU, C.; LJUNGKRANTZ, O. ReSA tool: Structured requirements specification and SAT-based consistency-checking. In: IEEE. 2016 Federated Conference on Computer Science and Information Systems (FedCSIS). [S.l.: s.n.], 2016. p. 1737–1746.

MARKO, N.; LEITNER, A.; HERBST, B.; WALLNER, A. Combining Xtext and OSLC for Integrated Model-Based Requirements Engineering. In: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications. [S.l.: s.n.], 2015. p. 143–150. DOI: 10.1109/SEAA.2015.11.

MATSUMOTO, Y.; SHIRAI, S.; OHNISHI, A. A Method for Verifying Non-Functional Requirements. **Procedia Computer Science**, v. 112, p. 157–166, 2017. ISSN 18770509. DOI: 10.1016/j.procs.2017.08.006. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877050917313455>.

MAUSAM; SCHMITZ, M.; SODERLAND, S.; BART, R.; ETZIONI, O. Open Language Learning for Information Extraction. In: PROCEEDINGS of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language

Learning. Jeju Island, Korea: Association for Computational Linguistics, jul. 2012. p. 523–534. Disponível em: <https://aclanthology.org/D12-1048>.

MAYO, N.; ASANO, M.; BARBIC, S. When is a research question not a research question? **Journal of Rehabilitation Medicine**, v. 45, n. 6, p. 513–518, jun. 2013. DOI: 10.2340/16501977-1150.

MCBRIDE, B. The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In: HANDBOOK on Ontologies. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 51–65. DOI: 10.1007/978-3-540-24750-0_3.

MEDEIROS, J.; VASCONCELOS, A.; SILVA, C.; GOULÃO, M. Requirements specification for developers in agile projects: Evaluation by two industrial case studies. **Information and Software Technology**, v. 117, p. 106194, jan. 2020. ISSN 09505849. DOI: 10.1016/j.infsof.2019.106194. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0950584919302010>.

MEZGHANI, M.; KANG, J.; SÈDES, F. Using k-Means for Redundancy and Inconsistency Detection: Application to Industrial Requirements. In: SILBERZTEIN, M.; ATIGUI, F.; KORNY-SHOVA, E.; MÉTAIS, E.; MEZIANE, F. (Ed.). **Natural Language Processing and Information Systems**. Cham: Springer International Publishing, 2018. p. 501–508. ISBN 978-3-319-91947-8.

MILLER, E. An Introduction to the Resource Description Framework. **Bulletin of the American Society for Information Science and Technology**, v. 25, n. 1, p. 15–19, jan. 2005. DOI: 10.1002/bult.105.

MISRA, J. Terminological inconsistency analysis of natural language requirements. **Information and Software Technology**, v. 74, p. 183–193, jun. 2016. ISSN 09505849. DOI: 10.1016/j.infsof.2015.11.006. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0950584915002001>.

MOHER, D.; LIBERATI, A.; TETZLAFF, J.; ALTMAN, D. G. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. **PLoS medicine**, v. 6, n. 7, e1000097, jul. 2009. DOI: 10.1371/journal.pmed.1000097.

MOKAMMEL, F.; COATANÉA, E.; COATANÉA, J.; NENCHEV, V.; BLANCO, E.; PIETOLA, M. Automatic requirements extraction, analysis, and graph representation using an approach derived from computational linguistics. **Systems Engineering**, John Wiley & Sons, Ltd, v. 21, n. 6, p. 555–575, nov. 2018. ISSN 1098-1241. DOI: 10.1002/sys.21461.

MORENO, V.; GÉNOVA, G.; PARRA, E.; FRAGA, A. Application of machine learning techniques to the flexible assessment and improvement of requirements quality. **Software Quality Journal**, abr. 2020. ISSN 0963-9314. DOI: 10.1007/s11219-020-09511-4. Disponível em: <http://link.springer.com/10.1007/s11219-020-09511-4>.

MU, K.; HONG, J.; JIN, Z.; LIU, W. From inconsistency handling to non-canonical requirements management: A logical perspective. **International Journal of Approximate Reasoning**, v. 54, n. 1, p. 109–131, 2013. DOI: 10.1016/j.ijar.2012.07.006.

MULDER, H. **The Chaos Report**. [S.l.: s.n.], jan. 1994.

MUNIZ, A.; IRIGOYEN, A.; CORRÊA, D.; TARGINO, R. **Jornada Ágil do Produto: unindo práticas e frameworks para capacitar Donos do Produto (Product Owners) e Gerentes de Produtos (Product Managers) com foco no fluxo de valor entregue ao cliente**. Rio de janeiro: Brasport, 2020.

MUSEN, M. A. The ProtéGé Project: A Look Back and a Look Forward. **AI Matters**, Association for Computing Machinery, New York, NY, USA, v. 1, n. 4, p. 4–12, jun. 2015. DOI: 10.1145/2757001.2757003.

NAUMCHEV, A.; MEYER, B. Seamless requirements. **Computer Languages, Systems & Structures**, v. 49, p. 119–132, 2017. DOI: 10.1016/j.cl.2017.04.001.

OLIVEIRA, M.; RIBEIRO, L.; COTA, É.; DUARTE, L. M.; NUNES, I.; REIS, F. Use Case Analysis Based on Formal Methods: An Empirical Study. In: RECENT Trends in Algebraic Development Techniques. Cham: Springer International Publishing, 2015. p. 110–130.

OSAMA, M.; ZAKI-ISMAIL, A.; ABDELRAZEK, M.; GRUNDY, J.; IBRAHIM, A. Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements. In: IEEE. 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). [S.l.: s.n.], 2020. p. 651–661. DOI: 10.1109/ICSME46990.2020.00067.

OSMAN, M. H.; ZAHARIN, M. F. Ambiguous Software Requirement Specification Detection: An Automated Approach. In: PROCEEDINGS of the 5th International Workshop on Requirements Engineering and Testing. New York, NY, USA: Association for Computing Machinery, 2018. (RET '18), p. 33–40. ISBN 9781450357494. DOI: 10.1145/3195538.3195545.

OZKAYA, M. Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? **Information and Software Technology**, v. 95, p. 15–33, 2018. ISSN 0950-5849. DOI: 10.1016/j.infsof.2017.10.008. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584917304834>.

PALDÊS, R. A.; CALAZANS, A. T. S.; MARIANO, A. M.; CASTRO, E. J. R. de; SILVA, B. d. S. da. A utilização da linguagem natural na especificação de requisitos: um estudo por meio das equações estruturais. In: XIX Ibero-American Conference on Software Engineering – Workshop on Requirements Engineering (WER). [S.l.: s.n.], 2016. p. 365–378.

PARRA, E.; DIMOU, C.; LLORENS, J.; MORENO, V.; FRAGA, A. A methodology for the classification of quality of requirements using machine learning techniques. **Information and Software Technology**, v. 67, p. 180–195, 2015. DOI: 10.1016/j.infsof.2015.07.006.

PHANTHANITHILERD, N.; PROMPOON, N. Requirements Characteristics Verification Method and Tool Based on Rules Constructed Form Software Component Relationships. In: 2015 Second International Conference on Trustworthy Systems and Their Applications. [S.l.]: IEEE, jul. 2015. p. 61–70. ISBN 978-1-4673-9581-6. DOI: 10.1109/TSA.2015.20. Disponível em: <http://ieeexplore.ieee.org/document/7335945/>.

PHANTHANITHILERD, N.; PROMPOON, N. Verifying Software Requirements Characteristics Based on Rules Defined from Software Component Relationships. **Lecture**

**Notes on Software Engineering**, v. 4, p. 27–33, jan. 2016. DOI: 10.7763/LNSE.2016.V4.219.

PI, X.; SHI, J.; HUANG, Y.; WEI, H. **Automated Mining and Checking of Formal Properties in Natural Language Requirements**. Edição: Christos Douligeris, Dimitris Karagiannis e Dimitris Apostolou. Cham: Springer International Publishing, 2019. p. 75–87. ISBN 978-3-030-29563-9.

PITTKE, F.; LEOPOLD, H.; MENDLING, J. Automatic detection and resolution of lexical ambiguity in process models. **IEEE Transactions on Software Engineering**, v. 41, n. 6, p. 526–544, 2015. DOI: 10.1109/TSE.2015.2396895.

POHL, K. **Requirements engineering: fundamentals, principles, and techniques**. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 3642125778.

PRESSMAN, R.; MAXIM, B. **Software Engineering: A Practitioner's Approach**. 9. ed. New York: McGraw-Hill Education, 2019.

REED, A. H.; ANGOLIA, M. Risk Management Usage and Impact on Information Systems Project Success. **International Journal of Information Technology Project Management (IJITPM)**, IGI Global, v. 9, n. 2, p. 1–19, 2018.

REGGIO, G.; LEOTTA, M.; RICCA, F.; CLERISSI, D. DUSM: A Method for Requirements Specification and Refinement Based on Disciplined Use Cases and Screen Mockups. **Journal of Computer Science and Technology**, v. 33, n. 5, p. 918–939, set. 2018. ISSN 1000-9000. DOI: 10.1007/s11390-018-1866-8. Disponível em: <http://link.springer.com/10.1007/s11390-018-1866-8>.

RINE, D. C.; FRAGA, A. Chunking Complexity Measurement for Requirements Quality Knowledge Representation. In: FRED, A.; DIETZ, J. L.; LIU, K.; FILIPE, J. (Ed.). **Knowledge Discovery, Knowledge Engineering and Knowledge Management**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. p. 245–259. ISBN 978-3-662-46549-3. DOI: 10.1007/978-3-662-46549-3_16.

RIVERO, J. M.; GRIGERA, J.; DISTANTE, D.; MONTERO, F.; ROSSI, G. DataMock: An Agile Approach for Building Data Models from User Interface Mockups. **Software & Systems Modeling**, v. 18, n. 1, p. 663–690, fev. 2019. ISSN 1619-1366. DOI: 10.1007/s10270-017-0586-9. Disponível em: <http://link.springer.com/10.1007/s10270-017-0586-9>.

ROJAS, A. B.; SLIESARIEVA, G. B. Automated Detection of Language Issues Affecting Accuracy, Ambiguity and Verifiability in Software Requirements Written in Natural Language. In: PROCEEDINGS of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas. Los Angeles, California: Association for Computational Linguistics, 2010. (YIWCALA '10), p. 100–108.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering**, v. 14, n. 2, p. 131–164, abr. 2009. ISSN 1382-3256. DOI: 10.1007/s10664-008-9102-8. Disponível em: <http://link.springer.com/10.1007/s10664-008-9102-8>.

SABRIYE, A. O. J.; ZAINON, W. M. N. W. A framework for detecting ambiguity in software requirement specification. In: 2017 8th International Conference on Information Technology (ICIT). [S.l.: s.n.], 2017. p. 209–213. DOI: 10.1109/ICITECH.2017.8080002.

SAGAR, V. B. R. V.; ABIRAMI, S. Conceptual modeling of natural language functional requirements. **Journal of Systems and Software**, Elsevier, v. 88, p. 25–41, 2014.

SANDHU, G.; SIKKA, S. State-of-art practices to detect inconsistencies and ambiguities from software requirements. In: INTERNATIONAL Conference on Computing, Communication Automation. [S.l.: s.n.], 2015. p. 812–817. DOI: 10.1109/CCAA.2015.7148485.

SANNE, U.; WITSCHEL, H. F.; FERRARI, A.; GNESI, S. Ensuring action: Identifying unclear actor specifications in textual business process descriptions. In: IC3K 2016 - Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. [S.l.: s.n.], 2016. v. 3, p. 140–147. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85006980039%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=5fbff60f0c0c2e891f246954a290c4bc>.

SANTOS, G. S. **Uma estratégia baseada em aquisição do conhecimento para o gerenciamento de riscos nos requisitos em um desenvolvimento XP distribuído**. 2017. Dissertação de Mestrado – Universidade Estadual de Campinas. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/322754>.

SARMIENTO, E.; LEITE, J. C. S. P.; ALMENTERO, E. Using correctness, consistency, and completeness patterns for automated scenarios verification. In: 2015 IEEE Fifth International Work-shop on Requirements Patterns (RePa). [S.l.: s.n.], 2015. p. 47–54. DOI: 10.1109/RePa.2015.7407737.

SELVARAJ, G.; WEBER, G.; LUTTEROTH, C. Efficient Program Analyses Using Deductive and Semantic Methodologies. In: 2017 IEEE 13th International Conference on e-Science (e-Science). [S.l.: s.n.], 2017. p. 440–441. DOI: 10.1109/eScience.2017.61.

SELWAY, M.; GROSSMANN, G.; MAYER, W.; STUMPTNER, M. Formalising Natural Language Specifications Using a Cognitive Linguistics/Configuration Based Approach. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference. [S.l.: s.n.], set. 2013. p. 59–68. DOI: 10.1109/EDOC.2013.16.

SELWAY, M.; MAYER, W.; STUMPTNER, M. Semantic Interpretation of Requirements through Cognitive Grammar and Configuration. In: LECTURE Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Cham: Springer International Publishing, 2014. v. 8862. (Lecture Notes in Computer Science). p. 496–510. DOI: 10.1007/978-3-319-13560-1_40.

SHARMA, R.; BISWAS, K. K. Generating Logical Representations for Natural Language Requirements Using Syntactic Dependencies and Norm Analysis Patterns. In: BIEMANN, C.; HANDSCHUH, S.; FREITAS, A.; MEZIANE, F.; MÉTAIS, E. (Ed.). **Natural Language Processing and Information Systems**. Cham: Springer International Publishing, 2015. p. 432–436. ISBN 978-3-319-19581-0.

SILVA, A. R. Quality of Requirements Specifications - A Framework for Automatic Validation of Requirements. In: PROCEEDINGS of the 16th International Conference on Enterprise

Information Systems. [S.l.]: SCITEPRESS - Science, 2014. p. 96–107. DOI: 10.5220/0004951900960107.

SILVA, A. R. SpecQua: Towards a Framework for Requirements Specifications with Increased Quality. In: INTERNATIONAL Conference on Enterprise Information Systems. [S.l.: s.n.], 2015. p. 265–281. DOI: 10.1007/978-3-319-22348-3_15.

SILVA, A. H.; FOSSÁ, M. I. T. Análise de conteúdo: exemplo de aplicação da técnica para análise de dados qualitativos. **Qualitas Revista Eletrônica**, v. 16, n. 1, 2015. ISSN 1677-4280. DOI: 10.18391/qualitas.v16i1.2113. Disponível em: <http://revista.uepb.edu.br/index.php/qualitas/article/view/2113>.

SILVESTRE, A. L. **Analise de Dados e Estatistica Descritiva**. São Paulo: Escolar Editora, 2007.

SINGH, M. Using Natural Language Processing and Graph Mining to Explore Inter- Related Requirements in Software Artefacts. **SIGSOFT Softw. Eng. Notes**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 1, p. 37, 2019. ISSN 0163-5948. DOI: 10.1145/3310013.3310018.

SKERSYS, T.; DANENAS, P.; BUTLERIS, R. Extracting SBVR business vocabularies and business rules from UML use case diagrams. **Journal of Systems and Software**, v. 141, p. 111–130, jul. 2018. ISSN 01641212. DOI: 10.1016/j.jss.2018.03.061. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S016412121830061X>.

SNEED, H. M.; VERHOEF, C. Natural language requirement specification for web service testing. In: 2013 15th IEEE International Symposium on Web Systems Evolution (WSE). [S.l.: s.n.], 2013. p. 5–14. DOI: 10.1109/WSE.2013.6642410.

SOARES, H. A.; MOURA, R. S. A methodology to guide writing Software Requirements Specification document. In: 2015 Latin American Computing Conference (CLEI). [S.l.: s.n.], 2015. p. 1–11. DOI: 10.1109/CLEI.2015.7360001.

SOMMERVILLE, I. **Software Engineering**. 10. ed. São Paulo: Pearson, 2016.

STACHTIARI, E.; MAVRIDOU, A.; KATSAROS, P.; BLIUDZE, S.; SIFAKIS, J. Early validation of system requirements and design through correctness-by-construction. **Journal of Systems and Software**, v. 145, p. 52–78, 2018. DOI: 10.1016/j.jss.2018.07.053.

THITISATHIENKUL, P.; PROMPOON, N. Quality Assessment Method for Software Requirements Specifications Based on Document Characteristics and Its Structure. In: 2015 Second International Conference on Trustworthy Systems and Their Applications. [S.l.: s.n.], 2015. p. 51–60. DOI: 10.1109/TSA.2015.19.

TIWARI, S.; AMETA, D.; BANERJEE, A. An Approach to Identify Use Case Scenarios from Textual Requirements Specification. In: PROCEEDINGS of the 12th Innovations on Software Engineering Conference (Formerly Known as India Software Engineering Conference). New York, NY, USA: Association for Computing Machinery, 2019. (ISEC'19). ISBN 9781450362153. DOI: 10.1145/3299771.3299774.

TOZONI-REIS, M. F. C. **Metodologia de Pesquisa**. 2. ed. Curitiba: IESDE Brasil S.A., 2009.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. **Introdução à Engenharia de Software Experimental**. Rio de Janeiro, 2002.

VALLON, R.; DA SILVA ESTÁCIO, B. J.; PRIKLADNICKI, R.; GRECHENIG, T. Systematic literature review on agile practices in global software development. **Information and Software Technology**, v. 96, p. 161–180, 2018. ISSN 0950-5849. DOI: 10.1016/j.infsof.2017.12.004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584917302975>.

VASQUES, D. G. **Verbka: processo para aquisição de conhecimento baseado em semântica verbal**. 2016. f. 167. Dissertação (Mestrado em Tecnologia) – Universidade Estadual de Campinas. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=000973205>.

VASQUES, D. G.; ZAMBON, A. C.; BAIOCO, G. B.; MARTINS, P. S. An Approach to Knowledge Acquisition Based on Verbal Semantics. In: HAWAII International Conference on System Sciences (HICSS). [S.l.]: IEEE, jan. 2016. p. 4144–4153. DOI: 10.1109/HICSS.2016.514.

VEIZAGA, A.; ALFEREZ, M.; TORRE, D.; SABETZADEH, M.; BRIAND, L. On systematically building a controlled natural language for functional requirements. **Empirical Software Engineering**, v. 26, n. 4, p. 79, jul. 2021. ISSN 1382-3256. DOI: 10.1007/s10664-021-09956-6. Disponível em: <https://link.springer.com/10.1007/s10664-021-09956-6>.

VÉRAS, P. C.; VILLANI, E.; AMBROSIO, A. M.; VIEIRA, M.; MADEIRA, H. A benchmarking process to assess software requirements documentation for space applications. **Journal of Systems and Software**, v. 100, p. 103–116, 2015. DOI: 10.1016/j.jss.2014.10.054.

WALTER, B.; HAMMES, J.; PIECHOTTA, M.; RUDOLPH, S. A Formalization Method to Process Structured Natural Language to Logic Expressions to Detect Redundant Specification and Test Statements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). [S.l.]: IEEE, set. 2017. p. 263–272. ISBN 978-1-5386-3191-1. DOI: 10.1109/RE.2017.38. Disponível em: <http://ieeexplore.ieee.org/document/8048912/>.

WAN, W.; CHEONG, H.; LI, W.; ZENG, Y.; IORIO, F. Automated transformation of design text ROM diagram into SysML models. **Advanced Engineering Informatics**, v. 30, n. 3, p. 585–603, 2016. DOI: 10.1016/j.aei.2016.07.003.

WANG, H. H.; DAMLJANOVIC, D.; SUN, J. An automated tool for semantic accessing to formal software models. **Science of Computer Programming**, v. 95, p. 93–111, 2014. DOI: 10.1016/j.scico.2014.02.027. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016764231400094X>.

WANG, Y.; JIANG, B.; WANG, T. Using workflow patterns to model and validate service requirements. In: IEEE. 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW). [S.l.: s.n.], 2016. p. 281–288.

WANG, Y.; ZHANG, J. Experiment on automatic functional requirements analysis with the EFRF's semantic cases. In: 2016 International Conference on Progress in Informatics and Computing (PIC). [S.l.]: IEEE, dez. 2016. p. 636–642. DOI: 10.1109/PIC.2016.7949577.

WHITE, S. A. Introduction to BPMN. **Ibm Cooperation**, v. 2, 2004.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in Software Engineering**. [S.l.]: Springer Berlin Heidelberg, 2012. (Computer Science).

WONG, S. Y.; MIT, E.; SIDI, J. Integration of use case formal template using mapping rules. In: 2016 Third International Conference on Information Retrieval and Knowledge Management (CAMP). [S.l.: s.n.], 2016. p. 25–31. DOI: 10.1109/INFRKM.2016.7806329.

XAVIER, C. C.; LIMA, V. L. S. de; SOUZA, M. Open information extraction based on lexical semantics. **Journal of the Brazilian Computer Society**, v. 21, n. 1, p. 4, dez. 2015. ISSN 0104-6500. DOI: 10.1186/s13173-015-0023-2. Disponível em: <https://journal-bcs.springeropen.com/articles/10.1186/s13173-015-0023-2>.

YAMADA, S.; OMORI, T.; OHNISHI, A. Verification method of reliability requirements. **Procedia Computer Science**, v. 159, p. 860–869, 2019. ISSN 18770509. DOI: 10.1016/j.procs.2019.09.245. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877050919314334>.

YAN, R.; CHENG, C. H.; CHAI, Y. Formal consistency checking over specifications in natural languages. In: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). [S.l.: s.n.], 2015. p. 1677–1682.

ZAINUDDIN, F. B.; LIU, S. PowerPoint Add-in Tool Support for Informal and Semi-Formal Specification Animation. In: 2015 Asia-Pacific Software Engineering Conference (APSEC). [S.l.]: IEEE, dez. 2015. p. 24–31. DOI: 10.1109/APSEC.2015.18.

ZAMAN, Q. uz; NADEEM, A.; SINDHU, M. A. Formalizing the use case model: A model-based approach. Edição: Alejandro F Villaverde. **PLOS ONE**, v. 15, n. 4, e0231534, abr. 2020. ISSN 1932-6203. DOI: 10.1371/journal.pone.0231534. Disponível em: <https://dx.plos.org/10.1371/journal.pone.0231534>.

# Apêndice A

# Protocolo da primeira pesquisa bibliográfica

Este apêndice apresenta o protocolo da primeira revisão de literatura. Nesse processo foram realizadas as seguintes etapas: definição do objetivo e questão de pesquisa, construção da estratégia de busca e aplicação de critérios de inclusão e exclusão de estudos.

**Objetivo e Questão de Pesquisa**

O objetivo principal da revisão consistiu em identificar abordagens atuais que contribuam com a estruturação, representação e validação de requisitos de software para reparar problemas ocasionados pela aplicação da linguagem natural. Para auxiliar a elaboração da questão de pesquisa foi utilizada a estratégia PICO. Essa metodologia contribui para o desenvolvimento da questão de pesquisa com base em quatro elementos: a população alvo da pesquisa, a intervenção a ser aplicada, uma intervenção alternativa para comparação e os resultados dessa aplicação (MAYO; ASANO; BARBIC, 2013). O resultado dessa aplicação é apresentado no Quadro A.1.

Quadro A.1: Aplicação da estratégia PICO.

| | |
|---|---|
| *P:* | Requisitos |
| I: | Abordagens para a estruturação, validação e representação de requisitos |
| C: | Requisitos expressos em linguagem natural |
| O: | Precisão e eficiência da abordagem |

Fonte: Elaborado pela autora (2021).

Com base na metodologia PICO, a questão definida para a busca foi: Qual a eficiência e precisão das abordagens para automatizar a estruturação, validação e representação de requisitos em comparação a requisitos em linguagem natural?

**Estratégia de Busca**

Os conceitos envolvidos na pesquisa bibliográfica foram extraídos da questão definida para a revisão (Quadro A.2). A definição desses conceitos foi baseada no contexto da engenharia de software.

Quadro A.2: Principais conceitos utilizados na revisão da literatura.

| | |
|---|---|
| Requisitos | descrevem as funcionalidades que o software deverá exercer. |
| Estruturação | organização/padronização das informações extraídas dos requisitos. |
| Validação | processo para a avaliação de qualidade das informações. |
| Representação | a forma como as informações são apresentadas e documentadas. |
| Linguagem natural | linguagem utilizada pelo ser humano para a comunicação. |
| Precisão | essa característica indica o acerto na definição dos requisitos. |

Fonte: Elaborado pela autora (2021).

Após a definição dos conceitos, para cada conceito foram explicitados os termos que os representam. Esses termos extraídos são mostrados no Quadro A.3.

Quadro A.3: Extração de termos para a busca de estudos.

| |
|---|
| *Software specification, software requirement, system requirement, requirement, domain* |
| *Linguistic analysis, structuring, structure, natural language processing, formal specification* |
| *Representation, modeling, document, database* |
| *Validation, verification* |
| *Linguagem natural* |
| *Quality, precision, efficiency* |

Fonte: Elaborado pela autora (2021).

Foram identificadas as bases de dados bibliográficos relevantes para a área de ciências da computação, como também em engenharia de software, com base no estudo de Bakar, Kasirun e Salleh (2015). As bases selecionadas são apresentadas no Quadro A.4.

Uma *string* de busca foi construída para ser aplicada nessas bases de dados. Foram utilizados operadores booleanos OR e AND para combinar os termos. A *string* de busca definida para a pesquisa foi: *((“software specification” OR “software requirement” OR “system*

Quadro A.4: Bases de dados utilizadas na primeira revisão da literatura.

| **Bases de Dados da Revisão de Literatura** | |
|---|---|
| ACM Digital Library | www.dl.acm.org |
| Elsevier | www.sciencedirect.com |
| IEEE Xplore Digital Library | www.ieeexplore.org |
| SCOPUS | www.scopus.com |

Fonte: Elaborado pela autora (2021).

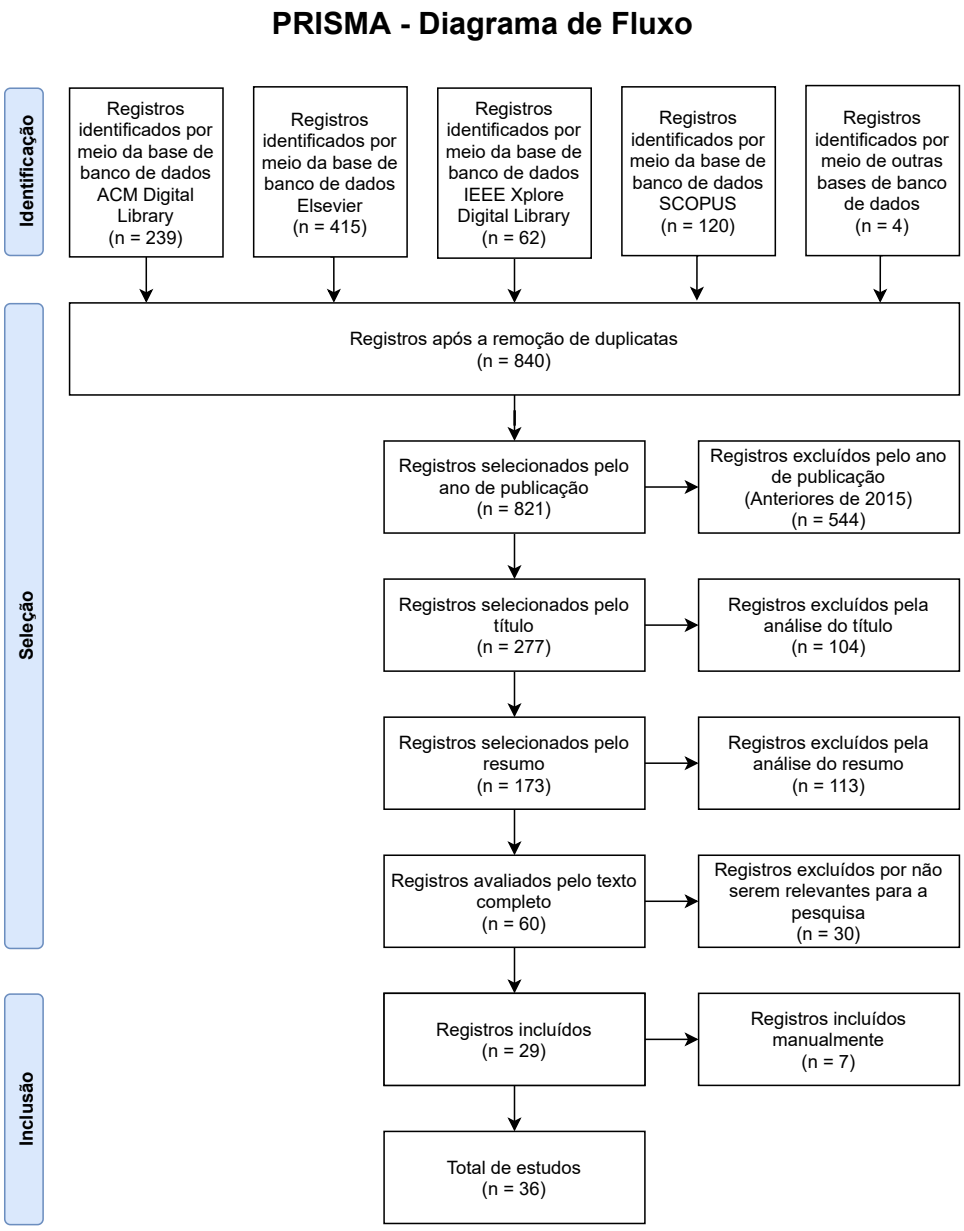
*requirement" OR requirement OR domain) AND ((representation OR modeling OR document OR database) AND ("linguistic analysis" OR structuring OR structure OR "natural language processing" OR "formal specification") AND (validation OR verification)) AND "natural language" AND (quality OR precision OR efficiency))*

Para as bases de dados Elsevier e SCOPUS a *string* foi adaptada para ter uma quantidade menor de termos, pelo motivo da limitação de caracteres nos campos de busca dessas bases. Definiu-se a *string*: *("software specification" OR "software requirement")AND ((representation OR modeling) AND ("natural language processing" OR structuring) AND validation) AND "natural language" AND (quality OR precision)*

**Critérios de Inclusão e Exclusão**

No processo de seleção dos estudos, o primeiro critério aplicado para a exclusão de artigos foi à duplicação de publicações. Em seguida, foram removidos artigos anteriores de 2014, por não serem considerados recentes. Foram consideradas as publicações que atendessem algum nível do objetivo da pesquisa. Para isso, foi realizada a filtragem dos artigos pela análise dos títulos, resumos e o texto completo. Ainda, foram adicionados alguns estudos relacionados e relevantes para a pesquisa, mesmo que não atendessem os critérios anteriores. Por fim, foram selecionados 39 estudos relevantes para a pesquisa. O fluxo de informações das fases da revisão é ilustrado no diagrama PRISMA (Figura A.1).

Figura A.1: Processo de seleção de artigos da primeira busca na literatura.




Fonte: Elaborado pela autora (2021).

# Apêndice B

# Protocolo da segunda pesquisa bibliográfica

Uma segunda busca foi realizada na literatura, visando abranger mais artigos relacionados com a estruturação, validação e representação de requisitos de software. Para isso, a *string* de busca (Quadro B.1) foi atualizada com base nas palavras-chave dos estudos identificados na primeira revisão.

Quadro B.1: Termos incrementados na *string* de busca.

| |
|---|
| *Software specification, software requirement, system requirement, requirement* |
| *Natural language* |
| *Structuring, structure, text analysis* |
| *Representation, modeling* |
| *Validation, verification* |

Fonte: Elaborado pela autora (2021).

Além das bases de dados utilizadas na primeira busca de estudos, foram adicionadas outras bases para abranger mais pesquisa. O Quadro B.2 apresenta essas bases.

A *string* de busca definida foi: *((“software specification” OR “software requirement” OR requirement) AND “natural language” AND ((representation OR modeling) AND (structuring OR structure OR “text analysis”) AND (validation OR verification))).* Essa *string* foi ajustada para cada base, por exemplo, para a base Elsevier foi: *(“software requirement” AND “natural language” AND ((representation OR modeling) AND (structuring OR structure OR “text analysis”) AND (validation OR verification)))*

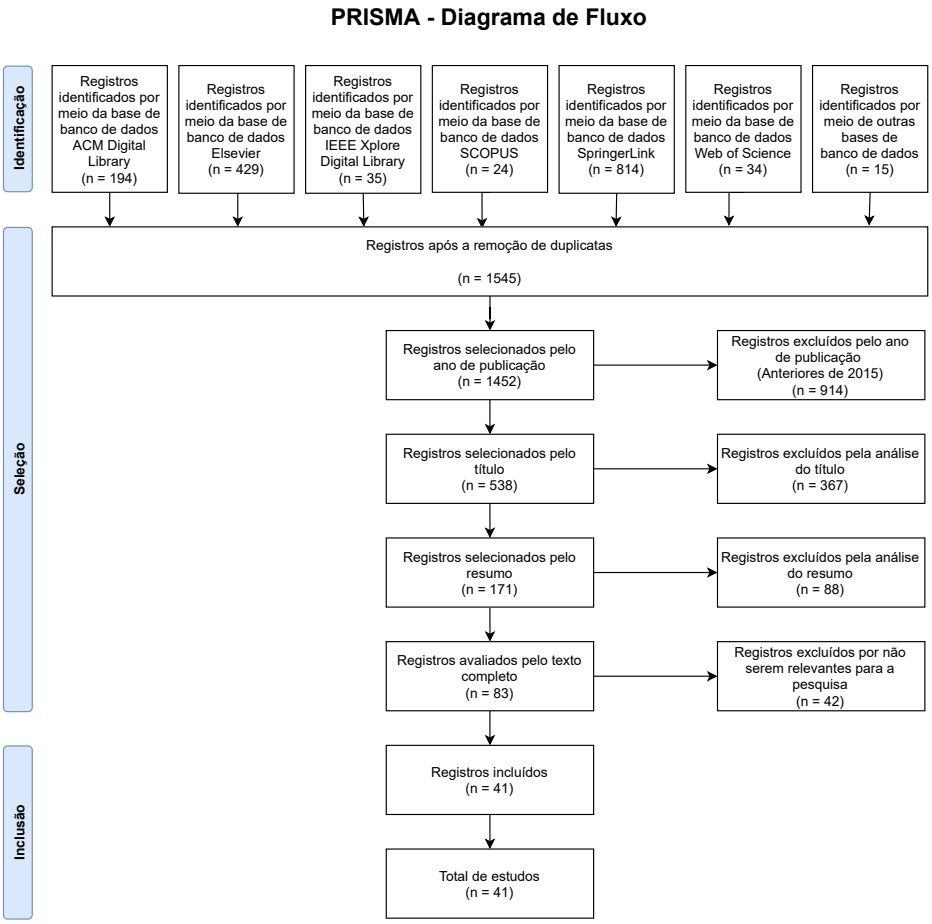
Quadro B.2: Bases de dados utilizadas na segunda revisão da literatura.

| **Bases de Busca da Revisão de Literatura** | |
|---|---|
| ACM Digital Library | www.dl.acm.org |
| Elsevier | www.sciencedirect.com |
| IEEE Xplore Digital Library | www.ieeexplore.org |
| SCOPUS | www.scopus.com |
| SpringerLink | www.link.springer.com |
| Web of Science | www.webofknowledge.com |

Fonte: Elaborado pela autora (2021).

Os critérios de inclusão e exclusão de estudos foram os mesmos considerados na primeira revisão, com exceção do ano, pois foram considerados estudos anteriores de 2015. O diagrama PRISMA desse processo é apresentado na Figura B.1.

Figura B.1: Processo de seleção de artigos da segunda busca na literatura.




Fonte: Elaborado pela autora (2021).

# Apêndice C

# Síntese dos estudos da revisão da literatura

Este apêndice apresenta um resumo dos estudos coletados das duas revisões da literatura. O Quadro C.1 apresenta os autores, ano, problema principal, abordagem proposta e o mapeamento da fase envolvida na abordagem (estruturação, representação e validação de requisitos) de cada artigo.

Quadro C.1: Síntese dos estudos da revisão da literatura.

| **Autor(es) e Ano** | **Problemas** | | | | | | | **Proposta** | **Tarefas** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | | E | R | V |
| Antinyan e Staron (2017) | | | | | | x | | Método de revisão e classificação | | | x |
| Apaza et al. (2018) | | | x | | | | x | Modelo híbrido | x | | |
| Ashfaq e Bajwa (2021) | | | x | | | | | Anotação semântica inteligente | x | x | |
| Atoum (2019) | x | x | | | | | | Modelos semânticos | | | x |
| Bajwa, Bordbar e Lee (2014) | | | x | | x | | | Framework para gerar descrições | x | | |
| Barcelos e Penteado (2017) | x | x | x | | | | | Conjunto de padrões | x | | |
| Bhatia, Kumar e Beniwal (2016) | x | | x | | | | | Framework de detecção de problema | | x | x |
| Castro, Bezerra e Hirata (2015) | | | | | | x | | Linguagem natural controlada | x | x | |
| Couto, Ribeiro e Campos (2014) | | | | | | | x | Formalização das declarações | x | x | |
| Dalpiaz, Schalk et al. (2019) | | x | x | | | x | | Abordagem baseada em ferramentas | | | x |
| Diamantopoulos et al. (2017) | x | x | x | | | | | Estruturação formal | x | x | |
| Djilani e Khouri (2015) | | | | | | | x | Framework baseado em semântica | x | x | |
| Dorigan e Barros (2014) | x | | | | | | x | Modelo de processo | x | | x |
| Eckhardt et al. (2016) | | x | | | | | | Framework para especificação | x | | |
| Escalona et al. (2013) | x | | | | | | | Abordagem dirigida por modelo | | | x |

Fonte: Elaborado pela autora (2021).

Notas: Problemas - (1) inconsistência, (2) incompletude, (3) ambiguidade, (4) redundância, (5) ilegibilidade, (6) limitação para o processamento automático e (7) falta de padronização.

Tarefas - (E) estruturação, (R) representação e (V) validação.

Continuação do Quadro

| Autor(es) e Ano | Problemas | | | | | | | Proposta | Tarefas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | E | R | V |
| Femmer et al. (2017) | | | | | | x | | Abordagem de análise de requisitos | | | x |
| Ferrari, Spagnolo et al. (2019) | | | x | | | | | Modelo de qualidade | | | x |
| Guo, Li Zhang e Lian (2021) | x | | | | | | | Análise semântica | x | | x |
| Gramajo, Ballejos e Ale (2021) | x | | x | x | | | | Redes neurais recorrentes | | | x |
| Hayrapetian e Raje (2018) | | x | x | | | | | Método de avaliação | | | x |
| Hoyos e Restrepo-Calle (2018) | | | | | | | x | Esquema de prototipagem | x | x | |
| Jue et al. (2019) | | x | x | | | | | Extensão de método de modelagem | x | x | |
| Kamalrudin, Hosking e Grundy (2017) | x | x | | | | | | Ferramenta automatizada | | | x |
| Khtira, Benlarabi e Asri (2015) | | | | x | | | | Abordagem de detecção de problema | | | x |
| Laue, Koop e Gruhn (2016) | | x | x | x | | | | Indicadores de problemas | | | x |
| Lebeaupin, Rauzy e Roussel (2017) | | | x | | | | | Linguagem Natural Controlada (lógica) | x | | |
| Li, Hayes e Truszczyński (2015) | x | x | x | | | x | | Linguagem formal intermediária | x | x | |
| Lucassen, Dalpiaz et al. (2016) | | | | | | x | | Ferramenta para identificar problemas | | | x |
| Lucassen, Robeer et al. (2017) | x | | | x | | | | Combinação de ferramentas | | x | x |
| Madala, Do e Aceituna (2018) | | | x | | | | | Modelagem com base em componentes | x | x | |
| Mahmud, Seceleanu e Ljungkrantz (2016) | x | | x | | | | | Conjunto de ferramentas | x | | x |
| Marko et al. (2015) | | | x | | x | | | Auxílio na escrita de requisitos | x | | |
| Matsumoto, Shirai e Ohnishi (2017) | x | x | x | x | | | | Método de verificação | x | | x |
| Medeiros et al. (2020) | | x | | | | | | Novo modelo de especificação | | x | |
| Mezghani, Kang e Sèdes (2018) | x | | | x | x | | | Abordagem de classificação | | | x |
| Misra (2016) | | | x | | | | | Abordagem genérica | | | x |
| Mokammel et al. (2018) | x | | | | | | | Abordagem de extração e análise | | x | x |
| Moreno et al. (2020) | | | | | | x | | Método de avaliação | | | x |
| Mu et al. (2013) | x | x | | x | | | | Framework baseado em lógica | | | x |
| Naumchev e Meyer (2017) | | | | | x | | | Combinação de linguagens | | x | |
| Oliveira et al. (2015) | | x | x | | | | | Transformação de gráficos | | x | |
| Osama et al. (2020) | | | x | | | x | | Análise de pontuação | | | x |
| Osman e Zaharin (2018) | | x | x | | | | | Abordagem de classificação | | | x |
| Parra et al. (2015) | | | | | | x | | Abordagem automática | | | x |
| Phanthanithilerd e Prompoon (2015) | | | x | | | | | Método de verificação | | | x |
| Phanthanithilerd e Prompoon (2016) | | | x | | | | | Regras para verificação | | | x |
| Pi et al. (2019) | | | x | | | | | Abordagem de verificação | x | | x |
| Pittke, Leopold e Mendling (2015) | | | x | | | | | Técnica de detecção de problema | | | x |
| Reggio et al. (2018) | x | x | x | | x | | | Método para descrição | x | x | x |
| Rine e Fraga (2015) | | | | | x | | | Métricas de complexidade | | | x |
| Rivero et al. (2019) | x | x | | | | | | Geração de modelos de dados | | x | x |
| Sabriye e Zainon (2017) | | | x | | | x | | Framework para detecção de problemas | | | x |
| Sagar e Abirami (2014) | | | | | x | | | Extração de conceitos e suas relações | | x | |

Continuação do Quadro

| Autor(es) e Ano | Problemas | | | | | | | Proposta | Tarefas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | E | R | V |
| Sanne et al. (2016) | | x | | | | | | Algoritmo baseado em regras | | | x |
| Sarmiento, Leite e Almentero (2015) | x | x | | | | x | | Catálogo de padrões | | | x |
| Selway et al. (2013) | | | x | | | x | | Geração de modelos formais | x | | |
| Selway, Mayer e Stumptner (2014) | x | x | x | | x | | | Interpretação semântica dos requisitos | x | | |
| Sharma e Biswas (2015) | x | x | | | | x | | Representações Lógicas | x | x | |
| Silva (2015) | x | x | x | | | | | Framework genérico | | | x |
| Singh (2019) | x | | x | x | | | | Requisitos inter-relacionados | | x | x |
| Skersys, Danenas e Butleris (2018) | x | | x | x | | | | Linguagem natural controlada | x | x | |
| Sneed e Verhoef (2013) | | | | | | x | | Linguagem natural estruturada | x | | |
| Soares e Moura (2015) | x | x | x | | | | | Auxílio na escrita de requisitos | x | x | x |
| Stachtiari et al. (2018) | | | x | | | | | Modelagem formal | x | x | x |
| Thitisathienkul e Prompoon (2015) | | | x | | x | | | Método de avaliação | | | x |
| Tiwari, Ameta e Banerjee (2019) | x | x | | x | | | | Transformação sistemática | x | | x |
| Veizaga et al. (2021) | | x | x | | | | x | Linguagem natural controlada | x | | |
| Véras et al. (2015) | | | | | | | x | Benchmark para a avaliação | | | x |
| Walter et al. (2017) | | | | x | | | | Processo de formalização | x | x | x |
| Wan et al. (2016) | | x | x | | | | | Extração de conhecimento | x | x | |
| Wang e Zhang (2016) | | | | | | x | | Extração de informações estruturadas | x | | |
| Wang, Jiang e Wang (2016) | | | | | | | x | Modelagem de requisitos de serviços | | x | x |
| Wong, Mit e Sidi (2016) | x | x | x | | | | | Modelo formal | x | x | |
| Yamada, Omori e Ohnishi (2019) | x | x | x | x | | | | Método de verificação | x | | x |
| Yan, Cheng e Chai (2015) | x | | | | | | | Framework de manutenção | | | x |
| Zainuddin e Liu (2015) | | | x | | x | | | Representação em animação | | x | |
| Zaman, Nadeem e Sindhu (2020) | x | | x | | | | | Abordagem baseada em modelo | x | x | |

# Apêndice D

# Questionário individual

O Quadro D.1 detalha as treze questões (abertas e fechadas) que compõem o questionário. Esse questionário foi aplicado individualmente aos participantes da prova de conceito realizada nesta pesquisa.

Quadro D.1: Questionário individual para os participantes da prova de conceito.

| Nº | Questões | | | |
|---|---|---|---|---|
| 1 | Como foi a compreensão dos requisitos gerados pelo protótipo comparado com os requisitos em linguagem natural? | Difícil | \| 1 \| 2 \| 3 \| 4 \| 5 \| | Fácil |
| 2 | Justifique sua resposta. | | | |
| 3 | Como foi a comunicação dos requisitos gerados pelo protótipo em comparação com os requisitos em linguagem natural? | Difícil | \| 1 \| 2 \| 3 \| 4 \| 5 \| | Fácil |
| 4 | Justifique sua resposta. | | | |
| 5 | Foram obtidas informações relevantes nas consultas realizadas no protótipo? | Não obtive | \| 1 \| 2 \| 3 \| 4 \| 5 \| | Obtive |
| 6 | Especifique sua resposta. | | | |
| 7 | Foram descobertos novos requisitos? | Não identifiquei | \| 1 \| 2 \| 3 \| 4 \| 5 \| | Identifiquei |
| 8 | Se identificados novos requisitos, como foram descobertos? | | | |
| 9 | Descreva suas facilidades na utilização do protótipo. | | | |
| 10 | Descreva suas dificuldades na utilização do protótipo. | | | |
| 11 | Qual a probabilidade de você adotar o protótipo em seu projeto? | Baixa | \| 1 \| 2 \| 3 \| 4 \| 5 \| | Alta |
| 12 | Por qual motivo você adotaria o protótipo? | | | |
| 13 | Observações ou comentários em relação ao protótipo. | | | |

Fonte: Elaborado pela autora (2021).

# Apêndice E

# Documento TCLE

**TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO**

**AUTRQ - Um Framework para Automatizar a Estruturação, Representação e Validação de Requisitos de Software em Linguagem Natural**

Glaucia Schnoeller dos Santos

**Número do CAAE:** 20089919.7.0000.5404

Você está sendo convidado a participar de uma pesquisa. Este documento, chamado Termo de Consentimento Livre e Esclarecido, visa assegurar seus direitos como participante da pesquisa e é elaborado em duas vias, assinadas e rubricadas pelo pesquisador e pelo participante/responsável legal, sendo que uma via deverá ficar com você e outra com o pesquisador.

Por favor, leia com atenção e calma, aproveitando para esclarecer suas dúvidas. Se houver perguntas antes ou mesmo depois de assiná-lo, você poderá esclarecê-las com o pesquisador. Se preferir, pode levar este Termo para casa e consultar seus familiares ou outras pessoas antes de decidir participar. Não haverá nenhum tipo de penalização ou prejuízo se você não aceitar participar ou retirar sua autorização em qualquer momento.

**Justificativa e objetivos:** O objetivo principal desta pesquisa é apresentar um framework para automatizar a estruturação, representação e validação dos requisitos, visando torná-los mais precisos para mitigar problemas na compreensão e comunicação de especificações de software escritas em linguagem natural.

**Procedimentos:** Participando do estudo você está sendo convidado a: contribuir com a representação e avaliação de requisitos de software em uma prova de conceito. Este estudo terá a duração de uma semana e será realizado de forma online. O estudo será realizado em três etapas:

- Primeira etapa: representar requisitos em linguagem natural utilizando ferramentas convencionais de especificações de software;
- Segunda etapa: representar os requisitos utilizando o framework proposto nesta pesquisa. Nesta etapa será apresentado aos participantes um protótipo disponibilizado na web com a implementação do framework. Os participantes receberão as instruções com as funcionalidades desse protótipo;
- Terceira etapa: responder um questionário online sobre esses requisitos com a duração de 40 minutos (aproximadamente). Em casos de questões que você não deseja responder, a questão poderá estar em branco.

**Desconfortos e riscos:** Você não deve participar deste estudo se tiver algum impedimento para seguir as orientações apresentadas pela pesquisadora responsável.

A pesquisa não apresenta riscos previsíveis. O desenvolvimento e a coleta de dados serão realizados online, assim não irá trazer desconfortos para os voluntários, pois não irá interferir em suas atividades de rotina. A duração aproximada para a coleta dos dados (preenchimento do questionário) é de 40 minutos.

**Benefícios:** Esta pesquisa contribuirá com a redução de problemas de comunicação e compreensão de especificações de software, evitando desafios e falhas de projetos de diferentes domínios. O framework proposto irá proporcionar uma redução de tarefas manuais de equipes de desenvolvimento, como correções de erros inerentes ao uso da linguagem natural e consequentemente ocasionando ganhos em termos de tempo e custo.

**Acompanhamento e assistência:** Você tem o direito à assistência integral e gratuita devido a danos diretos e indiretos, imediatos e tardios, pelo tempo que for necessário. O estudo será acompanhado de forma presencial e remota pela pesquisadora responsável. Os participantes terão a garantia da assistência após o encerramento ou interrupção da pesquisa. Os resultados desta pesquisa estarão sempre disponíveis para os participantes.

Em caso de ausência ou preenchimento incorreto do questionário, o participante será descontinuado do estudo.

**Sigilo e privacidade:** Você tem a garantia de que sua identidade será mantida em sigilo e nenhuma informação será dada a outras pessoas que não façam parte da equipe de pesquisadores. Na divulgação dos resultados desse estudo, seu nome não será citado.

**Ressarcimento e Indenização:** Caso o participante tenha gastos para participar da pesquisa fora de sua rotina, as despesas serão ressarcidas integralmente. Você terá a garantia ao direito à indenização diante de eventuais danos decorrentes da pesquisa.

**Contato:** Em caso de dúvidas sobre a pesquisa, você poderá entrar em contato com a pesquisadora Glaucia Schnoeller dos Santos, por meio do endereço da Faculdade de Tecnologia da Unicamp: R. Paschoal Marmo, 1888 - Jd. Nova Itália - CEP 13484-332 - Limeira, SP - Departamento de Pós-Graduação.

Em caso de denúncias ou reclamações sobre sua participação e sobre questões éticas do estudo, você poderá entrar em contato com a secretaria do Comitê de Ética em Pesquisa (CEP) da UNICAMP das 08:00hs às 11:30hs e das 13:00hs as 17:30hs na Rua: Tessália Vieira de Camargo, 126; CEP 13083-887 Campinas – SP; telefone (19) 3521-8936 ou (19) 3521-7187; e-mail: cep@fcm.unicamp.br.

**O Comitê de Ética em Pesquisa (CEP).** O papel do CEP é avaliar e acompanhar os aspectos éticos de todas as pesquisas envolvendo seres humanos. A Comissão Nacional de Ética em Pesquisa (CONEP), tem por objetivo desenvolver a regulamentação sobre proteção dos seres humanos envolvidos nas pesquisas. Desempenha um papel coordenador da rede de Comitês de Ética em Pesquisa (CEPs) das instituições, além de assumir a função de órgão consultor na área de ética em pesquisas.

**Consentimento livre e esclarecido:** Declaro que sou maior de 18 anos e que participarei por livre vontade desta pesquisa. Após ter recebido esclarecimentos sobre a natureza da pesquisa, seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa acarretar, aceito participar e declaro estar recebendo uma via original deste documento assinada pelo pesquisador e por mim, tendo todas as folhas por nós rubricadas:

Nome do (a) participante da pesquisa:

---

(Assinatura do participante da pesquisa ou nome e assinatura do seu RESPONSÁVEL LEGAL)

Data: _____

**Responsabilidade do Pesquisador:** Asseguro ter cumprido as exigências da resolução 466/ 2012 CNS/MS e complementares na elaboração do protocolo e na obtenção deste Termo de Consentimento Livre e Esclarecido. Asseguro, também, ter explicado e fornecido uma via deste documento ao participante da pesquisa. Informo que o estudo foi aprovado pelo CEP perante o qual o projeto foi apresentado e pela CONEP, quando pertinente. Comprometo-me a utilizar o material e os dados obtidos nesta pesquisa exclusivamente para as finalidades previstas neste documento ou conforme o consentimento dado pelo participante da pesquisa.

---

(Assinatura do pesquisador)

Data: _____

# Apêndice F

# Experimento - requisitos

Este Apêndice apresenta os requisitos reescritos que foram utilizados no experimento para a geração automática dos modelos conceituais deste trabalho. Esses requisitos foram extraídos dos projetos de desenvolvimento *Data Hub*, *Planning Poker* (DALPIAZ, 2018; DALPIAZ; STURM, 2020a) e ACME (REGGIO et al., 2018). Os requisitos originais foram processados pela ferramenta AutRQ, a qual apontou alguns erros de qualidade. Esses erros foram corrigidos seguindo algumas diretrizes da literatura de boas práticas.

**Casos de Uso - *Data Hub***

| | |
|---|---|
| 1 | Site admin specifies the repository with title and owner. |
| 2 | Site admin defines name for pricing plans. |
| 3 | Site admin determines pricing plans scheme. |
| 4 | Site admin views usage metrics for user, API and download. |
| 5 | Site admin determines the billing scheme. |
| 6 | Site admin sends invitations to visit the repository. |
| 7 | Publisher creates an account. |
| 8 | Publisher views pricing plans. |
| 9 | Publisher chooses the pricing plan. |
| 10 | Publisher adds accounts to membership. |
| 11 | Publisher removes accounts from the membership. |
| 12 | Publisher views usage metrics. |
| 13 | Publisher searches for data packages. |
| 14 | Publisher chooses a data package. |
| 15 | Publisher imports data packages. |
| 16 | Publisher validates the data packages. |
| 17 | Publisher tags data package. |
| 18 | Publisher specifies data package permissions. |
| 19 | Publisher removes data package publication. |
| 20 | Publisher deletes data package. |
| 21 | Consumer searches data package. |
| 22 | Consumer views data packages with metadata. |
| 23 | Consumer downloads the data package. |

**Estórias de Usuário - *Data Hub***

| | |
|---|---|
| 1 | As a Publisher, I want to sign up for an account. |
| 2 | As an Admin, I want to send invitations to visit the repository. |
| 3 | As a Publisher, I want to import the data packages. |
| 4 | As a Publisher, I want to publish data packages. |
| 5 | As a Publisher, I want to remove data packages publications. |
| 6 | As a Publisher, I want to delete data packages. |
| 7 | As a Publisher, I want to validate publications data packages. |
| 8 | As a Publisher, I want to tag the data packages. |
| 9 | As a Publisher, I want to view examples of data packages. |
| 10 | As a Consumer, I want to search the data packages. |
| 11 | As a Consumer, I want to view online data packages. |
| 12 | As a Consumer, I want to download the data package in one file. |
| 13 | As a Consumer, I want to view the downloaded data packages. |
| 14 | As a Consumer, I want to check the quality and reliability of data packages. |
| 15 | As a Publisher, I want to view pricing plans. |
| 16 | As a Publisher, I want to choose a pricing plan. |
| 17 | As an Admin, I want to have a pricing plan. |
| 18 | As an Admin, I want to have a billing scheme. |
| 19 | As an Admin, I want to specify the repository with title and owner. |
| 20 | As an Admin, I want to view the usage metrics for users, API and downloads. |
| 21 | As a Publisher, I want to add accounts to membership. |
| 22 | As a Publisher, I want to remove accounts from the membership. |
| 23 | As a Consumer, I want to view the account data packages. |

**Casos de Uso - *Planning Poker***

| | |
|---|---|
| 1 | Moderator creates a game with a name and an optional description. |
| 2 | Moderator specifies the estimation policy for the game. |
| 3 | Moderator sends invitations to the game. |
| 4 | Moderator adds accounts to the game. |
| 5 | Moderator starts a round. |
| 6 | Moderator views items. |
| 7 | Moderator adds items in the round. |
| 8 | Moderator edits items in the round. |
| 9 | Moderator deletes items from the round. |
| 10 | Moderator chooses an item. |
| 11 | Moderator publishes the item for estimation. |
| 12 | Moderator views item estimates. |
| 13 | Moderator calculates the average estimation per item. |
| 14 | Moderator submits the final estimate. |
| 15 | Moderator finishes the round. |
| 16 | Moderator finishes the game. |
| 17 | Estimator views items. |
| 18 | Estimator submits estimates. |
| 19 | Estimator updates estimates. |
| 20 | Estimator changes the estimates. |
| 21 | Estimator views items estimates. |
| 22 | Estimator finishes the round with a two minute countdown. |

**Estórias de Usuário - *Planning Poker***

| | |
|---|---|
| 1 | As a moderator, I want to create a game with a name and an optional description. |
| 2 | As a moderator, I want to send invitations to the game. |
| 3 | As a moderator, I want to specify the estimation policy for the game. |
| 4 | As a moderator, I want to start a round with an item. |
| 5 | As a moderator, I want to view the items of the round. |
| 6 | As a moderator, I want to select an item for estimation. |
| 7 | As a moderator, I want to add an item to the round. |
| 8 | As a moderator, I want to edit an item in the round. |
| 9 | As a moderator, I want to delete an item from the round. |
| 10 | As a moderator, I want to show all estimates. |
| 11 | As an estimator, I want to change my estimate. |
| 12 | As a moderator, I want to accept the average of estimates. |
| 13 | As a moderator, I want to automate the acceptance of unanimous estimates. |
| 14 | As a moderator, I want to submit the final estimate. |
| 15 | As an estimator, I want to view the estimates item. |
| 16 | As an estimator, I want to identify the estimates of other game participants. |
| 17 | As an estimator, I want to view all estimates at the same time. |
| 18 | As an estimator, I want to finish the round with a two-minute countdown timer. |

**Casos de Uso - ACME**

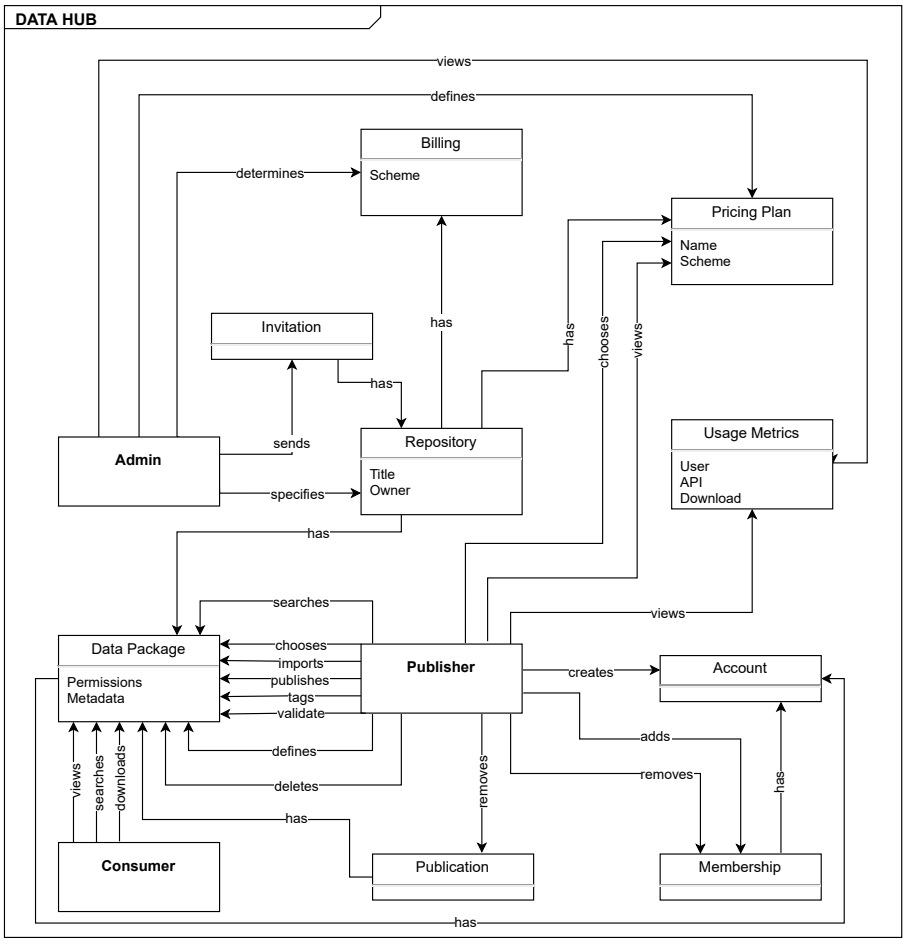
| | |
|---|---|
| 1 | External user accesses as unknown account. |
| 2 | External user submits the username and password. |
| 3 | External user logs in to his account. |
| 4 | Administrator submits the username and password. |
| 5 | Administrator logs in to his account. |
| 6 | Administrator creates accounts with name, group, username and password. |
| 7 | Administrator views the accounts with username and group. |
| 8 | Administrator edits accounts with name, group, username and password. |
| 9 | External user views the accounts with username and group. |
| 10 | External user edits his account with name, group and username. |
| 11 | External user submits the old and new password. |
| 12 | External user changes his password. |
| 13 | Administrator deletes the accounts. |
| 14 | Administrator views account jobs. |
| 15 | Administrator removes the jobs from accounts. |

# Apêndice G

# Experimento - modelos conceituais

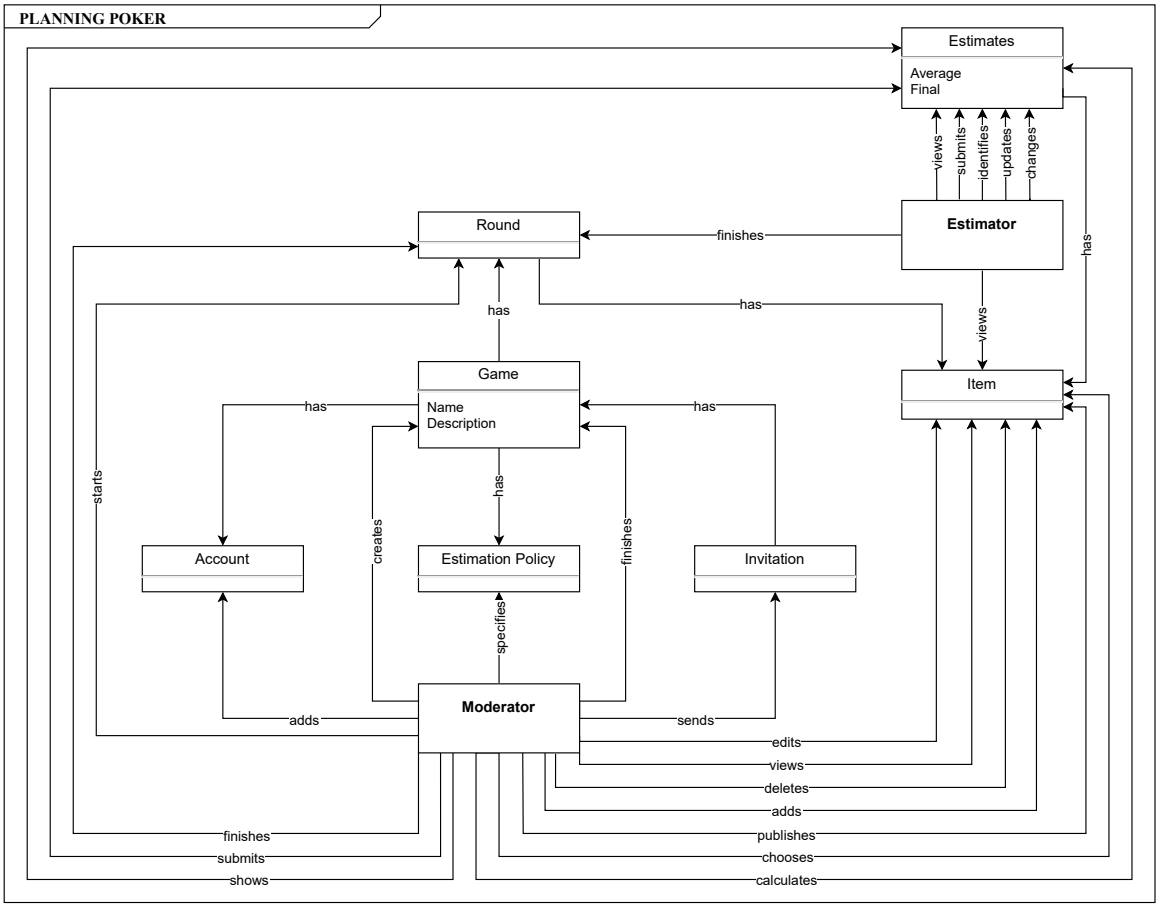
Os modelos conceituais utilizados como padrão-ouro dos projetos *Data Hub* (Figura G.1), *Planning Poker* (Figura G.2) e ACME (Figura G.3) são ilustrados neste Apêndice.

Figura G.1: Modelo conceitual padrão-ouro do projeto *Data Hub*.



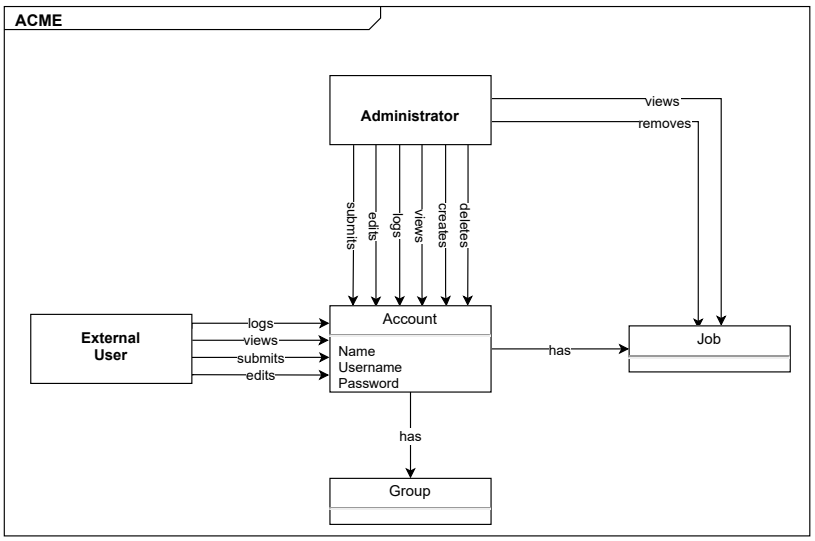

Fonte: Elaborado pela autora (2021).

Figura G.2: Modelo conceitual padrão-ouro do projeto *Planning Poker*.




Fonte: Elaborado pela autora (2021).

Figura G.3: Modelo conceitual padrão-ouro do projeto ACME.




Fonte: Elaborado pela autora (2021).

# Apêndice H

# Experimento - tabelas semânticas

As tabelas semânticas extraídas dos requisitos dos projetos *Data Hub* (Quadros H.1 e H.2), *Planning Poker* (Quadros H.3 e H.4) e ACME (Quadro H.5) são apresentadas neste Apêndice.

Quadro H.1: Tabela semântica do projeto *Data Hub* - Casos de Uso.

| **Subject** | **Action/state** | **Complement 1** | **Complement 2** |
|---|---|---|---|
| publisher | remove | accounts | from membership |
| consumer | search | data package | |
| publisher | tag | data package | |
| consumer | download | data package | |
| publisher | choose | data package | |
| site admin | define | repository | with title |
| site admin | view | usage metrics | for user |
| consumer | view | data packages | with metadata |
| site admin | define | billing scheme | |
| publisher | view | pricing plans | |
| publisher | add | accounts | to membership |
| publisher | delete | data package | |
| publisher | define | data package permissions | |
| publisher | create | account | |
| publisher | import | data packages | |
| site admin | define | repository | with owner |
| publisher | remove | data package publication | |
| site admin | view | usage metrics | for download |
| publisher | view | usage metrics | |
| site admin | view | usage metrics | for api |
| publisher | validate | data packages | |
| site admin | define | pricing plans scheme | |
| publisher | choose | pricing plan | |
| publisher | search | for data packages | |
| site admin | send | invitations | to visit repository |
| site admin | define | name | for pricing plans |

Fonte: Tabela gerada pelo AutRQ.

Quadro H.2: Tabela semântica do projeto *Data Hub* - Estórias de Usuário.

| **Subject** | **Action/state** | **Complement 1** | **Complement 2** |
|---|---|---|---|
| consumer | view | downloaded data packages | |
| admin | specify | repository | with title |
| consumer | check | reliability of data packages | |
| admin | view | usage metrics | for downloads |
| publisher | validate | publications data packages | |
| admin | specify | repository | with owner |
| publisher | remove | data packages publications | |
| admin | view | usage metrics | for api |
| admin | send | invitations | to visit repository |
| publisher | publish | data packages | |
| publisher | remove | accounts | from membership |
| consumer | view | online data packages | |
| consumer | download | data package | in one file |
| publisher | view | pricing plans | |
| admin | have | billing scheme | |
| publisher | view | examples of data packages | |
| publisher | add | accounts | to membership |
| publisher | sign up | for account | |
| admin | view | usage metrics | for users |
| publisher | tag | data packages | |
| admin | have | pricing plan | |
| publisher | delete | data packages | |
| consumer | check | quality of data packages | |
| publisher | import | data packages | |
| consumer | search | data packages | |
| consumer | view | account data packages | |
| publisher | choose | pricing plan | |

Fonte: Tabela gerada pelo AutRQ.

Quadro H.3: Tabela semântica do projeto *Planning Poker* - Casos de Uso.

| **Subject** | **Action/state** | **Complement 1** | **Complement 2** |
|---|---|---|---|
| estimator | change | estimates | |
| moderator | add | items | in round |
| moderator | edit | items | in round |
| estimator | finish | round | with two minute countdown |
| moderator | create | game | with name |
| estimator | submit | estimates | |
| moderator | submit | final estimate | |
| moderator | finish | game | |
| estimator | view | items estimates | |
| estimator | view | items | |
| moderator | publish | item | for estimation |
| estimator | update | estimates | |
| moderator | finish | round | |
| moderator | create | game | with optional description |
| moderator | start | round | |
| moderator | choose | item | |
| moderator | specify | estimation policy | for game |
| moderator | view | items | |
| moderator | send | invitations | to game |
| moderator | add | accounts | to game |
| moderator | calculate | average estimation | per item |
| moderator | delete | items | from round |

Fonte: Tabela gerada pelo AutRQ.

Quadro H.4: Tabela semântica do projeto *Planning Poker* - Estórias de Usuário.

| **Subject** | **Action/state** | **Complement 1** | **Complement 2** |
|---|---|---|---|
| moderator | create | game | with name |
| moderator | specify | estimation policy | for game |
| estimator | identify | estimates of other game participants | |
| moderator | automate | acceptance of unanimous estimates | |
| moderator | show | all estimates | |
| estimator | change | your estimate | |
| moderator | send | invitations | to game |
| moderator | delete | items | from round |
| estimator | finish | round | with two - minute countdown timer |
| moderator | create | game | with optional description |
| moderator | submit | final estimate | |
| moderator | calculate | average estimation | |
| estimator | view | estimate items | |
| estimator | view | all estimates | at same time |
| moderator | select | item | for estimation |
| moderator | add | items | in round |
| moderator | edit | items | in round |
| moderator | start | round | with item |
| moderator | view | items of round | |

Fonte: Tabela gerada pelo AutRQ.

Quadro H.5: Tabela semântica do projeto ACME - Casos de Uso.

| **Subject** | **Action/state** | **Complement 1** | **Complement 2** |
|---|---|---|---|
| administrator | view | accounts | with username |
| administrator | view | account jobs | |
| administrator | edit | accounts | with name |
| external user | edit | his account | with group |
| external user | view | accounts | with group |
| administrator | edit | accounts | with group |
| external user | access | as unknown account | |
| administrator | remove | jobs | from accounts |
| external user | edit | his account | with name |
| administrator | create | accounts | with password |
| external user | submit | new password | |
| administrator | create | accounts | with name |
| administrator | submit | password | |
| administrator | submit | username | |
| external user | edit | his account | with username |
| administrator | edit | accounts | with password |
| administrator | log in | to his account | |
| external user | submit | password | |
| external user | submit | old password | |
| external user | log in | to his account | |
| administrator | view | accounts | with group |
| external user | change | his password | |
| external user | view | accounts | with username |
| administrator | create | accounts | with group |
| external user | submit | username | |
| administrator | delete | accounts | |

Fonte: Tabela gerada pelo AutRQ.

# Anexo A

# Parecer Consubstanciado do CEP


**PARECER CONSUBSTANCIADO DO CEP**

**DADOS DO PROJETO DE PESQUISA**

**Título da Pesquisa:** AutRQ - Um Framework para Automatizar a Estruturação, Representação e Validação de Requisitos de Software em Linguagem Natural

**Pesquisador:** Glaucia Schnoeller dos Santos

**Área Temática:**

**Versão:** 2

**CAAE:** 20089919.7.0000.5404

**Instituição Proponente:** Faculdade de Tecnologia

**Patrocinador Principal:** Financiamento Próprio

**DADOS DO PARECER**

**Número do Parecer:** 3.676.042

UNICAMP - CAMPUS CAMPINAS

Continuação do Parecer: 3.676.042

**Conclusões ou Pendências e Lista de Inadequações:**
As pendências foram sanadas.

**Considerações Finais a critério do CEP:**
- O participante da pesquisa deve receber uma via do Termo de Consentimento Livre e Esclarecido, na íntegra, por ele assinado (quando aplicável).

- O participante da pesquisa tem a liberdade de recusar-se a participar ou de retirar seu consentimento em qualquer fase da pesquisa, sem penalização alguma e sem prejuízo ao seu cuidado (quando aplicável).

- O pesquisador deve desenvolver a pesquisa conforme delineada no protocolo aprovado. Se o pesquisador considerar a descontinuação do estudo, esta deve ser justificada e somente ser realizada após análise das razões da descontinuidade pelo CEP que o aprovou. O pesquisador deve aguardar o parecer do CEP quanto à descontinuação, exceto quando perceber risco ou dano não previsto ao participante ou quando constatar a superioridade de uma estratégia diagnóstica ou terapêutica oferecida a um dos grupos da pesquisa, isto é, somente em caso de necessidade de ação imediata com intuito de proteger os participantes.

- O CEP deve ser informado de todos os efeitos adversos ou fatos relevantes que alterem o curso normal do estudo. É papel do pesquisador assegurar medidas imediatas adequadas frente a evento adverso grave ocorrido (mesmo que tenha sido em outro centro) e enviar notificação ao CEP e à Agência Nacional de Vigilância Sanitária – ANVISA – junto com seu posicionamento.

- Eventuais modificações ou emendas ao protocolo devem ser apresentadas ao CEP de forma clara e sucinta, identificando a parte do protocolo a ser modificada e suas justificativas e aguardando a aprovação do CEP para continuidade da pesquisa. Em caso de projetos do Grupo I ou II apresentados anteriormente à ANVISA, o pesquisador ou patrocinador deve enviá-las também à mesma, junto com o parecer aprovatório do CEP, para serem juntadas ao protocolo inicial.

- Relatórios parciais e final devem ser apresentados ao CEP, inicialmente seis meses após a data deste parecer de aprovação e ao término do estudo.

-Lembramos que segundo a Resolução 466/2012 , item XI.2 letra e, “cabe ao pesquisador

**Endereço:** Rua Tessália Vieira de Camargo, 126
**Bairro:** Barão Geraldo **CEP:** 13.083-887
**UF:** SP **Município:** CAMPINAS
**Telefone:** (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br

UNICAMP - CAMPUS CAMPINAS

Continuação do Parecer: 3.676.042

apresentar dados solicitados pelo CEP ou pela CONEP a qualquer momento".

-O pesquisador deve manter os dados da pesquisa em arquivo, físico ou digital, sob sua guarda e responsabilidade, por um período de 5 anos após o término da pesquisa.

**Este parecer foi elaborado baseado nos documentos abaixo relacionados:**

| Tipo Documento | Arquivo | Postagem | Autor | Situação |
|---|---|---|---|---|
| Informações Básicas do Projeto | PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_1406226.pdf | 19/10/2019 23:13:43 | | Aceito |
| Outros | Carta_Resposta.pdf | 19/10/2019 23:12:48 | Glaucia Schnoeller dos Santos | Aceito |
| Projeto Detalhado / Brochura Investigador | Projeto_Pesquisa_V2.pdf | 19/10/2019 23:12:16 | Glaucia Schnoeller dos Santos | Aceito |
| TCLE / Termos de Assentimento / Justificativa de Ausência | TCLE_V2.pdf | 19/10/2019 23:11:03 | Glaucia Schnoeller dos Santos | Aceito |
| Outros | AtestadoMatricula.pdf | 21/08/2019 16:24:21 | Glaucia Schnoeller dos Santos | Aceito |
| Cronograma | Cronograma_Glaucia.pdf | 15/08/2019 22:47:59 | Glaucia Schnoeller dos Santos | Aceito |
| Outros | Questionario_Glaucia.pdf | 15/08/2019 22:43:58 | Glaucia Schnoeller dos Santos | Aceito |
| TCLE / Termos de Assentimento / Justificativa de Ausência | TCLE_Glaucia.pdf | 15/08/2019 22:41:49 | Glaucia Schnoeller dos Santos | Aceito |
| Projeto Detalhado / Brochura Investigador | Projeto_Pesquisa_Glaucia.pdf | 15/08/2019 22:41:29 | Glaucia Schnoeller dos Santos | Aceito |
| Folha de Rosto | Folha_Rosto_Glaucia.pdf | 15/08/2019 22:31:05 | Glaucia Schnoeller dos Santos | Aceito |

**Situação do Parecer:**
Aprovado

**Necessita Apreciação da CONEP:**
Não

**Endereço:** Rua Tessália Vieira de Camargo, 126
**Bairro:** Barão Geraldo **CEP:** 13.083-887
**UF:** SP **Município:** CAMPINAS
**Telefone:** (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br