Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

HELP$_4$ERS - Uma Metodologia para Auxiliar a Escrita de Documentos de Especificação de Requisitos de Sistemas

Hélcio de Abreu Soares

Teresina-PI, Agosto de 2016

Hélcio de Abreu Soares

HELP$_4$ERS - Uma Metodologia para Auxiliar a Escrita de Documentos de Especificação de Requisitos de Sistemas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Raimundo Santos Moura

Teresina-PI

Agosto de 2016

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca Setorial do CCN

S676h Soares,Hélcio de Abreu.
HELP$_4$ERS – Uma metodologia para auxiliar a escrita de documentos de especificação de requisitos de sistemas / Hélcio de Abreu Soares. – Teresina, 2016.
97f.: il.

Dissertação (Mestrado) – Universidade Federal do Piauí, Centro de Ciências da Natureza, Pós-Graduação em Ciência da Computação, 2016.
Orientador: Prof. Dr. Raimundo Santos Moura.

1. Sistemas de Computação. 2. Engenharia de Requisitos. 3. Linguagem Natural Controlada. I. Título.

CDD 004.2

Help4ERS: uma metodologia para conduzir a escrita de documentos de Especificação de Requisitos de Software

HÉLCIO DE ABREU SOARES

Dissertação apresentada ao Programa de Pós–Graduação em Ciência da Computação do Centro de Ciências da Natureza da Universidade Federal do Piauí, como parte integrante dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Aprovado por:

Prof. Raimundo Santos Moura

(Presidente da Banca)

Prof. Luiz Affonso Henderson Guedes de Oliveira

(Examinador Externo)

Prof. Erick Baptista Passos

(Examinador Interno)

Prof. Pedro de Alcântara dos Santos Neto

(Examinador Interno)

Teresina, 26 de agosto de 2016

Resumo

A Engenharia de Requisitos (ER) é o processo de definir, documentar e manter requisitos de *softwares*, tem como objetivo apoiar a criação e a manutenção do documento de Especificação de Requisitos de *Software* (ERS). Esse documento é a base para as demais atividades de desenvolvimento e sua qualidade é fundamental para o sucesso do projeto. Ele serve de contrato entre os participantes do desenvolvimento do *software* e deve ser produzido de forma que todos os envolvidos no desenvolvimento do sistema possam entendê-lo. Para isso descrições em linguagem natural e modelos de domínio são frequentemente utilizados. No entanto, o uso de linguagem natural pode dar origem a especificações de requisitos incompletas, inconsistentes e ambíguas, gerando interpretações equivocadas sobre os objetivos do sistema, além de impedir suporte computacional em especificações de requisitos. Nesse contexto, esta Dissertação propõe uma metodologia para auxiliar a escrita de documentos de requisitos completos, consistentes e não ambíguos, denominada HELP$_4$ERS. Para isso, define-se padrões de sentenças que ajudam a expressar requisitos de sistemas sem os problemas do uso de linguagem natural. A partir desses padrões, são criadas duas Linguagens Naturais Controladas (LNC) e dois Padrões Linguísticos que apoiam a verificação de inconsistências e ajudam na identificação automática de elementos de projeto (casos de uso, atores, classes, atributos, métodos, interfaces de usuário e elementos dessas interfaces). HELP$_4$ERS engloba atividades inerentes às fases de Engenharia de Requisitos, tais como análise de domínio da aplicação, definição semântica do domínio, escrita dos requisitos e verificação de inconsistência, além da fase de geração modelos de projeto (diagrama de classe, diagrama de casos de uso e protótipo de interface). A metodologia é apoiada pelo protótipo ERS-EDITOR que implementa os Padrões Linguísticos e as LNC's e dá suporte automatizado às suas atividades. Em um estudo experimental realizado, os atributos de qualidades obtidos com o uso de HELP$_4$ERS foram superiores àqueles alcançados sem o uso da metodologia. Porém, o mesmo estudo apresentou um esforço de tempo 30% maior com o uso da metodologia.

Palavras-chaves: Engenharia de Requisitos, Qualidade de Requisitos, Processamento de Linguagem Natural, Linguagem Natural Controlada.

Abstract

Requirements Engineering (RE) is the process of defining, documenting and maintaining software requirements, it aims to support the creation and maintenance of the Software Requirements Specification document (SRS). That document supports all other development activities and its quality is fundamental to project success. It serves as a contract between the of the software development and should be produced so that everyone involved can understand it. To achieve this, natural language descriptions and domain models are often used. However, the use of natural language can lead to incomplete, inconsistent, and ambiguous requirements specifications. This can create misconceptions about the system's objectives, and prevents computer support in requirements specifications. In this context, this Thesis proposes a methodology to assist the writing of complete, consistent, and unambiguous requirements document, called HELP4ERS. To do this, we define sentence patterns that help to express system requirements without the problems of using natural language. From these patterns are created two Controlled Natural Languages (CNL) and two Language Patterns that support inconsistency check and help in the automatic identification of project elements (use cases, actors, classes, attributes, methods, user interfaces and elements of these interfaces). HELP$_4$ERS includes activities related to the phases of requirements engineering, such as application domain analysis, semantic domain definition, written requirements, inconsistency check, and the phase of project templates generation (class diagram, use case diagram and interface prototype). The methodology is supported by the ERS-EDITOR prototype that implements the Language Patterns and the LNC's and provides automated support to its activities. In an experimental study performed, the quality attributes obtained using HELP4ERS were superior to those achieved without the use of the methodology. However, the same study showed a time effort 30% higher using the methodology.

Keywords: Requirement engineering, Quality Requirements, Natural Language Processing.

Lista de ilustrações

Figura 1 – Desafios das empresas. Adaptado de Jamasoftware (2011) 2
Figura 2 – Desafios das empresas. Adaptado de Arruda et al. (2014) 2
Figura 3 – Usuários de um ERS (SOMMERVILLE et al., 2008) 9
Figura 4 – Processo de engenharia de requisitos (SOMMERVILLE et al., 2008) . . 10
Figura 5 – Tipos de transações em um caso de uso 13
Figura 6 – Exemplo de Caso de uso com uma transação por sentença 13
Figura 7 – Gráfico segundo a lei de Zipf 16
Figura 8 – Gráfico de Zipf com os cortes propostos por Luhn 17
Figura 9 – Exemplo de uma gramática 18
Figura 10 – Padrão de requisitos para a seção *Requisitos de armazenamento* 28
Figura 11 – Padrão de requisitos para a seção *Descrições de casos de uso* 29
Figura 12 – A gramática *S-Lucas* 33
Figura 13 – Descrição do caso de uso "Validar certidão" escrita em *S-Lucas* 34
Figura 14 – Arvore sintática da sentença de ação 34
Figura 15 – Visão geral da HELP$_4$ERS 35
Figura 16 – Curva de ZIpf e Cortes de Luhn adaptados 39
Figura 17 – Definição de sinônimos do domínio 41
Figura 18 – Definição da semântica do domínio 41
Figura 19 – Itens da seção selecionada: *Atores e sistemas externos* 42
Figura 20 – Classificação de termos como atributos 44
Figura 21 – Gramática geradora do **Padrão 1** 44
Figura 22 – *Intellisense* na seção *Funções do produto* 45
Figura 23 – *S-Lucas*: Sentença de ação 46
Figura 24 – *Intellisense* na seção *Descrições de casos de uso* 46
Figura 25 – *Intellisense* para verbos do domínio da aplicação 47
Figura 26 – Aviso de sentença fora do padrão 48
Figura 27 – Erro de sintaxe 48
Figura 28 – Erro de semântica 49
Figura 29 – *Descrições de casos de uso* - Erro de sintaxe 50
Figura 30 – *Descrições de casos de uso* - Erro de semântica 50
Figura 31 – Verificação de termos classificados e escritos 51
Figura 32 – Exemplo de avisos 52
Figura 33 – Recurso explorar - Seção *Funções do produto* 53
Figura 34 – Recurso explorar - Seção *Descrição de casos de uso* 54
Figura 35 – Ator "sistema de autenticação" 67
Figura 36 – Pontos fracos da proposta 73

Figura 37 – Pontos fortes da proposta 74
Figura 38 – Avaliação da metodologia 74
Figura 39 – Avaliação do ERS-EDITOR 75
Figura 40 – Avaliação dos recursos 76

Lista de tabelas

Tabela 1 – Seções do documento de requisitos 26
Tabela 2 – Elementos de projeto/Seções do requisitos 27
Tabela 3 – Padrões linguísticos para identificar elementos de projeto 27
Tabela 4 – Verbos mais frequentes 32
Tabela 5 – Padrões linguísticos para identificar elementos de projeto 43
Tabela 6 – Resultados 55
Tabela 7 – Atividades do experimento 63
Tabela 8 – Mnemônicos para elementos de projetos 65
Tabela 9 – Elementos de projetos corretos 66
Tabela 10 – Elementos corretos - SEM 66
Tabela 11 – Elementos corretos - COM 66
Tabela 12 – Termos classificados e escritos SEM 68
Tabela 13 – Termos classificados e escritos COM 68
Tabela 14 – Termos escritos e classificados SEM 68
Tabela 15 – Termos escritos e classificados COM 68
Tabela 16 – Consistência SEM 69
Tabela 17 – Consistência COM 69
Tabela 18 – Não ambíguo - COM 70
Tabela 19 – LNC - FP 71
Tabela 20 – LNC - UC 71
Tabela 21 – Sentenças com erros de sintaxe 72
Tabela 22 – Questionário de avaliação do aprendizado da metodologia 74
Tabela 23 – Questões para avaliação do ERS-EDITOR 75
Tabela 24 – Recursos disponíveis 76

Lista de abreviaturas

CMMI *Capability Maturity Model Integration*

ER Engenharia de Requisitos

ERS Especificação de Requisitos de Softwares

ES Engenharia de Software

EBNF *Extended Backus Naur Form*

FSER Ferramenta de Suporte à Elicitação de Requisitos

GLC Gramática Livre de Contexto

GQM *Goal / Question / Metric*

IEEE *Institute of Electrical and Electronics Engineers*

LAL Léxico Ampliado da Linguagem

LLC Linguagem Livre de Contexto

LNC Linguagem Natural Controlada

PABRE *Pattern-based Requirements Elicitation*

PLN Processamento de Linguagem Natural

RSLingo *Requirements Specification Language*

RUP *Rational Unified Process*

TCE-PI Tribunal de Contas do Estado do Piauí

Sumário

1 INTRODUÇÃO 1
1.1 Contexto e Motivação 1
Contexto e Motivação 1
1.2 Objetivos 3
Objetivos 3
1.3 Contribuições 4
1.4 Estrutura da Dissertação 5
Estrutura do Trabalho 5

2 REFERENCIAL TEÓRICO 7
2.1 Requisitos 7
2.2 Engenharia de Requisitos 9
2.3 Casos de uso 11
2.4 Processamento de Linguagem Natural 13
2.4.1 Etiquetador (Part-Of-Speech Tagger) 14
2.4.2 *Stemming* 15
2.4.3 *Stopwords* 16
2.4.4 Gerador de Lexer e Parser - ANTLR 17
2.5 Linguagem Natural Controlada - LNC 19
2.6 Considerações finais 19

3 TRABALHOS RELACIONADOS 21
3.1 Miriam 21
3.2 PABRE 21
3.3 RSLingo 22
3.4 FSER 23
3.5 Considerações finais 23

4 A METODOLOGIA HELP$_4$ERS 25
4.1 Estrutura do documento de requisitos 25
4.2 Padrões linguísticos 27
4.3 Padrões de requisitos 28
4.4 Linguagens Naturais Controladas 29
4.4.1 *Funções do produto* 29
4.4.2 *Descrições de casos de uso* 31
4.5 Etapas da metodologia 34

4.6 O Suporte Automatizado . . . 38
4.6.1 Análise do domínio . . . 38
4.6.2 Definição da semântica do domínio . . . 40
4.6.3 Escrita de requisitos . . . 42
4.6.4 Verificação de inconsistências . . . 46
4.6.4.1 Erro sintático e semântico . . . 48
4.6.4.2 Verificação de termos classificados e escritos . . . 50
4.6.4.3 Avisos . . . 51
4.6.4.4 *Recurso explorar* . . . 52
4.7 Resultados preliminares . . . 53
4.8 Considerações finais . . . 56

5 EXPERIMENTOS E RESULTADOS . . . 57
5.1 Estudo Experimental . . . 57
5.1.1 Definição dos objetivos . . . 57
5.1.1.1 Resumo do objetivo . . . 58
5.1.2 Definição das hipóteses . . . 58
5.1.3 Questões da experimentação . . . 60
5.1.3.1 Seleção de variáveis . . . 61
5.1.4 Seleção dos participantes . . . 61
5.1.5 Instrumentação . . . 62
5.1.6 Projeto do estudo . . . 63
5.1.7 Operação . . . 64
5.1.8 Obtenção e interpretação dos resultados . . . 65
5.1.8.1 Completude . . . 65
5.1.8.1.1 Completude externa . . . 65
5.1.8.1.2 Completude interna . . . 66
5.1.8.2 Consistência . . . 68
5.1.8.3 Não ambíguo . . . 69
5.1.8.4 Linguagem Natural Controlada . . . 70
5.1.8.5 Esforço . . . 72
5.1.8.5.1 Pontos fortes e pontos fracos . . . 73
5.1.8.5.2 A metodologia . . . 73
5.1.8.5.3 O ERS-EDITOR . . . 75
5.1.8.5.4 Os recursos . . . 75
5.2 Validade do experimento . . . 77
5.2.1 Validade interna . . . 77
5.2.2 Validade externa . . . 77
5.2.3 Validade construção . . . 78
5.3 Considerações finais . . . 78

6 CONCLUSÃO 81
6.1 Contribuições 81
6.2 Limitações e dificuldades do trabalho 82
6.3 Trabalhos futuros 83

REFERÊNCIAS 85

APÊNDICES 91

APÊNDICE A – BASE DE VERBOS SINÔNIMOS 93

APÊNDICE B – *S-LUCAS*: ERROS SINTÁTICOS 95

APÊNDICE C – *S-LUCAS*: ERROS SEMÂNTICOS 97

1 Introdução

Neste capítulo serão descritos o contexto, as motivações da pesquisa, os objetivos da Dissertação e sua organização.

1.1 Contexto e Motivação

A Engenharia de Software (ES) é uma área da Ciência da Computação cujo foco é a construção de sistemas de alta qualidade, com custos adequados, dentro do cronograma estabelecido e com metodologia formalizada para o ciclo de vida de desenvolvimento. A base para a construção de um software é construída pela Engenharia de Requisitos (ER). Ela dá suporte à fase inicial do ciclo de vida do software. De acordo com Kotonya e Sommerville (1998), a Engenharia de Requisitos engloba todas as atividades envolvidas na descoberta, na documentação e na manutenção de um conjunto de requisitos para um sistema computacional. A Engenharia de Requisitos é uma importante disciplina da Engenharia de Software, é nela que os requisitos devem ser concretizados com a elaboração do documento de Especificação de Requisitos de Software. Esse documento orienta a construção do sistema e é determinante para a qualidade do produto e para a satisfação dos clientes (ESTECA et al., 2012).

Os trabalhos descritos em Jamasoftware (2011) e Arruda et al. (2014) descrevem pesquisas realizadas em empresas dos EUA e do Porto Digital de Recife, respectivamente, cujo objetivo é identificar tendências que contribuem para o sucesso na gestão de requisitos. As pesquisas abordaram aspectos importantes dentro da área de Engenharia de Software. Destaca-se, com relevância a este trabalho, a questão **"Quais os maiores desafios da empresa em relação à Engenharia de Requisitos?"**.

As Figuras 1 e 2 mostram o resultado para a questão. A conclusão dos trabalhos reforça a ideia de que a documentação e entendimento dos requisitos são fundamentais para o sucesso dos projetos de desenvolvimento. Segundo Koscianski e Soares (2007), a qualidade de um software depende em Software parte da especificação adequada dos requisitos. Documentos de requisitos incompletos, inconsistentes e com omissões resultam na construção de um software de baixa qualidade.

Ferreira e Silva (2012) defendem o uso de texto em linguagem natural para escrever especificações de requisitos por ser de uso comum entre as partes envolvidas no desenvolvimento do sistema, por contornar problemas de familiaridade e de proficiência em relação às notações mais formais, como diagramas e pseudocódigos, além de ser expressiva o suficiente para descrever o problema do domínio.

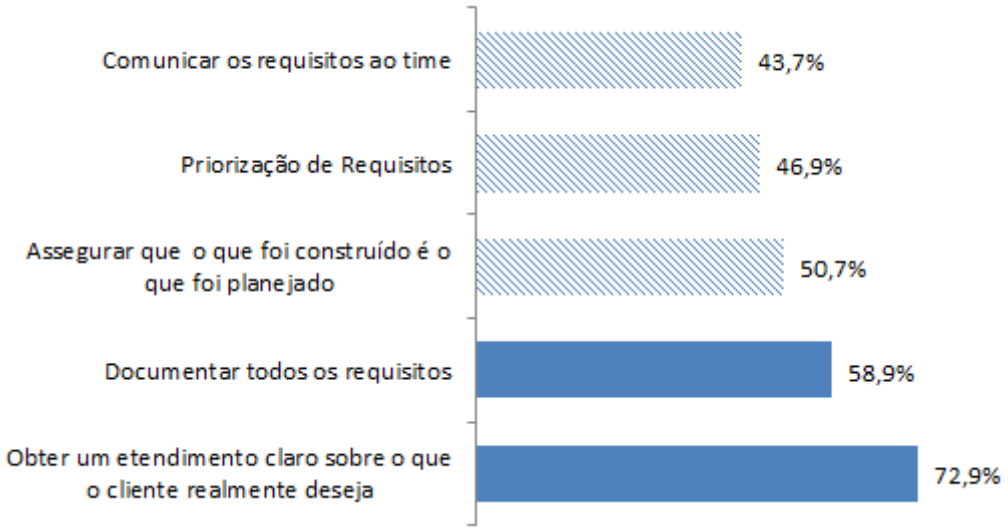

Figura 1 – Desafios das empresas. Adaptado de Jamasoftware (2011)

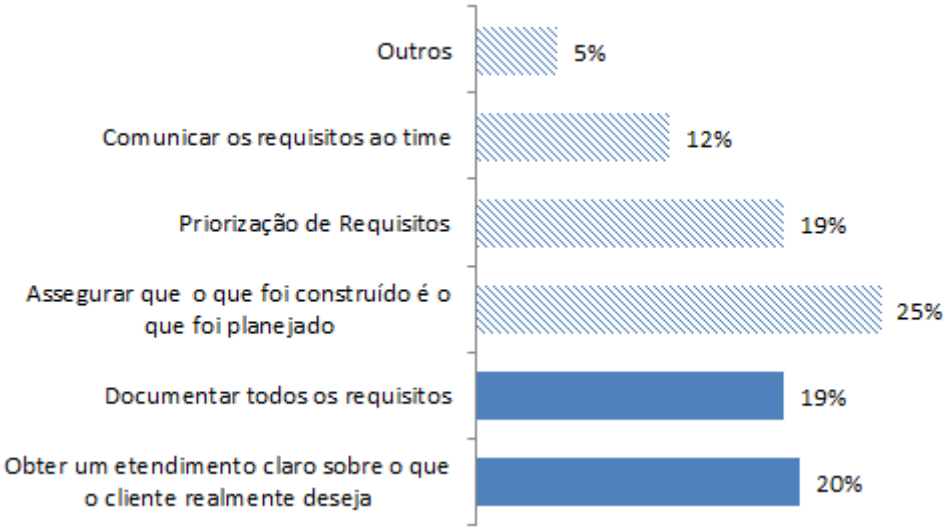

Figura 2 – Desafios das empresas. Adaptado de Arruda et al. (2014)

Apesar de possuir essas características, o uso da linguagem natural apresenta alguns problemas, tais como: ambiguidade, incompletude e inconsistências, que são encontrados em documentos de especificações de requisitos (IEEE, 1998), (YOUNG, 2004), (GAUSE; WEINBERG, 1989) e (KAINDL; SVETINOVIC, 2010). Esses problemas geram uma distância entre a especificação e a modelagem do sistema, uma vez que permitem múltiplas interpretações e dão origem a modelos que não contemplam as características que o software deve possuir (SOMMERVILLE et al., 2008).

Uma forma de reduzir os problemas do uso de linguagem natural é a definição de modelos e padrões que ajudam os engenheiros de requisitos e usuários a elicitar, expressar e registrar informações relevantes no domínio do sistema. Toro et al. (1999) desenvolveram modelos de requisitos e identificaram dois tipos de padrões: **padrões de requisitos** (*R-pattens*), que são modelos genéricos de requisitos encontrados frequentemente durante

o processo de ER e que podem ser reutilizados com algumas adaptações; e **padrões linguísticos** (*L-pattens*) que são agrupamentos de palavras, considerando a classificação gramatical (morfológica), frequentemente usadas para descrever **elementos de projeto** [1] e que podem ser parametrizadas. O uso desses padrões ajuda na obtenção de documentos de ERS de melhor qualidade tanto em conteúdo quanto em sintaxe (PALOMARES; QUER; FRANCH, 2011). A utilização desses padrões associado às técnicas de Processamento de Linguagem Natural (PLN) possibilita a extração de elementos de projeto a partir das descrições de requisitos. Anchiêta, Sousa e Moura (2013) e Juarez-Ramirez, Huertas e Inzunza (2014) apresentam o uso de PLN para extrair elementos de projeto a partir de descrições de casos de uso.

Outra forma de contornar os problemas da linguagem natural é o uso de Linguagem Natural Controlada (LNC) para guiar e restringir a criação de requisitos e descrições de casos de uso. Uma Linguagem Natural Controlada é um subconjunto de alguma linguagem natural, por exemplo o português e o inglês, que usa uma gramática restrita e um vocabulário predefinido de acordo com o domínio. O Principal objetivo do uso da LNC é evitar complexidade e ambiguidade em textos técnicos (SCHWITTER, 2010).

Dessa forma, assim como Ferreira e Silva (2012), defende-se um foco maior na especificação de requisitos textuais e que ferramentas devem ser capazes de extrair automaticamente os elementos de projeto pertinentes ao domínio do sistema a ser construído. Essas informações devem ser *(i)* escritas obedecendo padrões linguísticos e obedecendo as regras de uma LNC e; *(ii)* identificadas pela semântica das palavras, conforme sua posição na sentença e sua classificação no documento de requisitos.

1.2 Objetivos

Considerando a escrita de documentos de requisitos como um passo fundamental para o desenvolvimento de sistema, esta Dissertação tem como objetivo conceber uma metodologia apoiada por técnicas de Processamento de Linguagem Natural para conduzir a escrita de documentos de requisitos completos, consistentes e não ambíguos, que permite suporte automatizado nas descrições dos requisitos para posterior extração de elementos de projeto e geração de modelos com o uso de Linguagem Natural Controlada. A metodologia é composta por cinco etapas, que são semi-automatizadas pelo protótipo ERS-EDITOR desenvolvido com o propósito de dar suporte à metodologia. O trabalho pode ser detalhado nos seguintes objetivos específicos:

- Definir uma metodologia para auxiliar a escrita de documentos de requisitos completos, consistentes e não ambíguos;

[1] **elementos de projetos**: casos de uso, atores, classes, atributos, métodos, interfaces de usuário e elementos de interfaces.

- Propor uma estrutura de documento de requisitos que contenha informações sobre elementos de projeto;
- Definir um conjunto de Linguagem Natural Controlada para a escrita dos requisitos;
- Realizar a verificação de consistência do documento de requisitos.
- Permitir a extração automática de elementos e modelos de projeto.
- Desenvolver um protótipo para apoiar a execução das atividades do método proposto; e
- Realizar um experimento em um ambiente acadêmico para avaliar o método e o protótipo.

Durante o levantamento bibliográfico foram encontrados alguns trabalhos semelhantes, porém, nenhum deles apresenta todas as características desta Dissertação. Os trabalhos semelhantes são detalhados no Capítulo 2.

A metodologia foi testada com a realização de um estudo experimental em ambiente acadêmico. Utilizou-se alunos do curso de Ciências da Computação e do curso de Análise e Desenvolvimento de Sistemas. Neste experimento, documentos de requisitos foram gerados com e sem a utilização de HELP$_4$ERS. Como exemplo foi utilizado o Sistema de Acompanhamento dos Gastos dos Recursos da Sociedade (SAGRES), que foi desenvolvido pelo Tribunal de Contas do Piauí (TCE-PI). Os documentos gerados no experimento foram avaliados em relação aos atributos de qualidade consistência, completude e não ambiguidade. Os resultados são animadores, pois os valores dos atributos dos documentos gerados com a metodologia são melhores do que os gerados sem seu uso. Porém, o uso da metodologia requer um custo adicional no tempo para a geração dos documentos de requisitos. O estudo experimental é detalhado no Capítulo 5.

1.3 Contribuições

Este trabalho foi avaliado pela comunidade acadêmica através da publicação do artigo *A methodology to guide writing Software Requirements Specification document* (SOARES; MOURA, 2015). Além do artigo, destaca-se como contribuição desta Dissertação a métodologia HELP$_4$ERS que define etapas para a escrita de documentos consistentes, não ambíguos e completos. Destaca-se também, o protótipo ERS-EDITOR que dá suporte à metodologia e automatiza suas atividades. As contribuições serão detalhadas no Capítulo 6.

1.4 Estrutura da Dissertação

Além deste capítulo introdutório, o restante desta Dissertação está organizado da seguinte forma:

O Capítulo 2 apresenta o Referencial Teórico com os principais tópicos das áreas de Engenharia de Requisitos, Processamento de Linguagem Natural e Linguagem Natural Controlada necessários para o entendimento geral desta Dissertação. O Capítulo 3 discute os principais trabalhos relacionados, encontrados na literatura, que contribuíram para a formulação da Dissertação ora apresentada.

O Capítulo 4 apresenta a abordagem HELP$_4$ERS, uma metodologia para auxiliar a escrita de Documentos de Requisitos consistentes, não ambíguos e completos, com suporte automatizado à verificação de inconsistências e identificação de elementos de projeto, além de apresentar o protótipo ERS-EDITOR, que dá suporte à metodologia. Este protótipo foi construído baseado em técnicas de Processamento de Linguagem Natural. Ainda neste capítulo, apresentamos resultados de uma avaliação parcial da metodologia.

No Capítulo 5 é apresentado um estudo experimental que avalia HELP$_4$ERS em ambiente controlado, levando se em consideração tanto a qualidade do resultado quanto o custo para sua obtenção. Utilizou-se o processo de experimentação e a abordagem Goal/Question/Metric - GQM (BASILI; CALDIERA; ROMBACH, 1994), que obedece às exigências da Engenharia de Software Experimental (WOHLIN et al., 2000).

Finalmente, no Capítulo 6 são apresentadas as contribuições desta Dissertação, bem como as dificuldades encontradas no seu desenvolvimento e os trabalhos necessários para a evolução da pesquisa.

2 Referencial Teórico

De acordo com o IEEE *Standard Glossary of Software Engineering Terminology* (IEEE, 1990), o processo de desenvolvimento de *software* consiste na tradução das necessidades do usuário em um produto de *software*. O processo envolve traduzir essas necessidades em requisitos de *software*, transformar os requisitos de *software* em projeto, implementar o projeto em código, testar o código e, algumas vezes, instalar e checar o *software* para o uso operacional. Destaca-se que essas atividades podem ser sobrepostas ou realizadas iterativamente.

Na literatura sobre Engenharia de Software, existem diversos modelos para guiar o processo de desenvolvimento. Dentre eles, destacam-se: o modelo em cascata (do inglês: *Waterfall model*) (ROYCE, 1970), usado para desenvolvimento de *software* de forma sequencial, onde o processo é visto como uma sequência de fases; o modelo espiral, do inglês: *Spiral model* (BOEHM, 1988), é um processo de desenvolvimento iterativo e combina elementos dos estágios de projeto e prototipagem de *software* e os métodos ágeis que surgiram no final dos anos 90. Exemplos desses últimos são: Desenvolvimento de Software Adaptativo, Cristal, Desenvolvimento de Sistemas Dinâmicos, Programação Extrema (XP) e Desenvolvimento Dirigido a Características. Boehm (1988) apresenta uma visão geral das práticas de ES usadas desde os anos 50 (década a década) identificando os aspectos históricos envolvidos em cada tendência. A Engenharia de Requisitos de Software é o ramo da Engenharia de Software que envolve as atividades relacionadas com a atividade de traduzir as necessidades do usuário em requisitos de *software*.

Neste capítulo são apresentados os principais conceitos referentes as áreas de Engenharia de Requisitos e de Processamento de Linguagem Natural, além de ferramentas e recursos importantes para o entendimento geral desta Dissertação.

2.1 Requisitos

Requisitos são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Podem ser definidos como algo que o sistema deverá realizar para atender às necessidades dos clientes e usuários. Em alguns casos, requisito é simplesmente a descrição de alto nível de um serviço que o sistema deve fornecer ou uma restrição do sistema, em outros, pode ser uma definição formal e detalhada de uma função do sistema. Para solucionar problemas resultantes da falta de uma definição clara desses níveis de abstração, Sommerville et al. (2008) classifica os requisitos em *requisitos de usuário* e *requisitos de sistema*, que são definidos como segue:

- **Requisitos de usuário**: são requisitos de mais alto nível de abstração, geralmente são escritos em linguagem natural e deve ser inteligível para o usuário final. São acompanhados de diagramas indicando quais serviços o sistema deverá prover e suas restrições;

- **Requisitos de sistema**: São descrições detalhadas do que o sistema deve fazer, suas funções operacionais, serviços e restrições. Deve definir exatamente o que vai ser implementado e pode servir de contrato entre o comprador do sistema e os desenvolvedores.

Como dito anteriormente, requisitos são descrições dos serviços fornecidos pelo sistema (algo que deve ser produzido) e as suas restrições operacionais (caracterização de algum serviço). Sob esse ponto de vista, Sommerville et al. (2008) e Pressman (2009) classificam-nos da seguintes formas:

- **Requisitos funcionais**: São declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações, é proveniente do domínio da aplicação;

- **Requisitos não funcionais**: São restrições sobre os serviços e funções oferecidos pelo sistema e são aplicados frequentemente ao sistema como um todo e não apenas a serviços individuais.

Uma das formas de registrar requisitos é a utilização do documento de Especificação de Requisitos de Softwares (ERS). Esse documento é utilizado como a declaração oficial do que os desenvolvedores do sistema devem implementar. Em seu conteúdo deve-se incluir os requisitos de usuário e uma descrição detalhada dos requisitos de sistemas, além dos requisitos funcionais e não funcionais. Por esse motivo o documento de requisitos possui um conjunto diversificado de usuários, desde quem paga pelo sistema até aqueles que o desenvolvem. A Figura 3, adaptada de (SOMMERVILLE et al., 2008), mostra os possíveis usuários de um documento de ERS, comumente chamados de *stakeholders* [1].

O padrão IEEE Std. 830/1998 (IEEE, 1998) é uma recomendação prática para escrever documentos de ERS. Ele descreve o conteúdo e os quesitos de qualidades que devem estar presentes, além de conter uma grande quantidade de recomendações de como redigir requisitos. Quanto a estrutura do documento, o padrão propõe partes essenciais, tais como:

1. **Introdução:** propósito, escopo, definições e siglas, referências e visão do documento;

[1] Stakeholder é qualquer pessoa ou organização que tenha interesse ou que seja afetado pelo sistema.

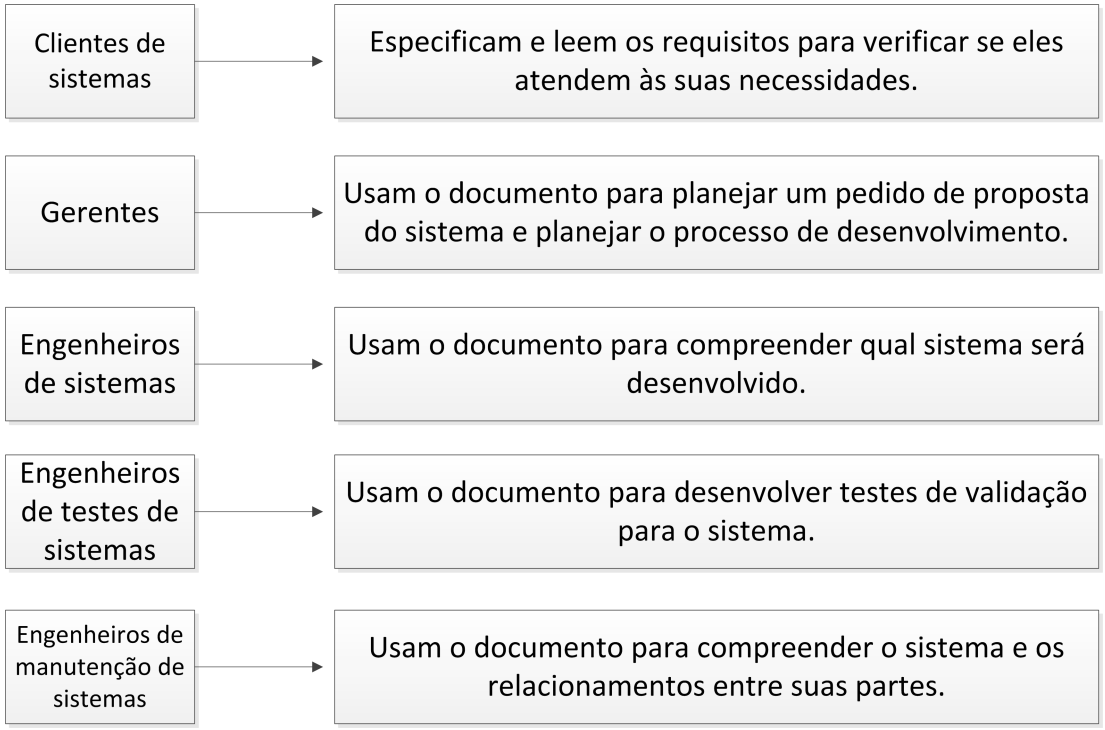

Figura 3 – Usuários de um ERS (SOMMERVILLE et al., 2008)

2. **Descrição Geral:** perspectiva e funções do produto, características dos usuários, restrições e suposições e dependências;

3. **Requisitos Específicos:** requisitos de interface externas (interfaces de usuário), requisitos funcionais e requisitos não funcionais.

Por ser muito geral, o padrão não é utilizado em sua forma completa. Porém, ele serve de *framework* que pode ser adaptado e configurado para definir um modelo dirigido às necessidades de uma organização.

2.2 Engenharia de Requisitos

O processo de descobrir, analisar e documentar requisitos de *software* é chamado de Engenharia de Requisitos (ER). Ele está relacionado à definição do que o sistema deve fazer, suas restrições e ao seu desenvolvimento. Esse processo estabelece uma base sólida para o projeto e a construção, sem ele, o *software* resultante tem uma alta probabilidade de não satisfazer às necessidades dos clientes. Sommerville et al. (2008) afirma que a ER é um estágio altamente crítico do processo de *software*, pois os erros nesse estágio conduzem inevitavelmente a problemas posteriores no projeto e na implementação do sistema.

Ainda segundo Sommerville et al. (2008), o processo de ER inclui quatro subprocessos de alto nível que ajudam a avaliar se o sistema é útil para a empresa, à obtenção de requisitos, à conversão desses requisitos em alguma forma padrão e à verificação se os

requisitos definem realmente aquilo que o cliente deseja. O objetivo principal da Engenharia de Requisitos é criar e manter um documento de Especificação de Requisitos de Sistemas. A Figura 4, adaptada de (SOMMERVILLE et al., 2008), mostra o processo de Engenharia de Requisitos e os artefatos gerados por cada atividade:

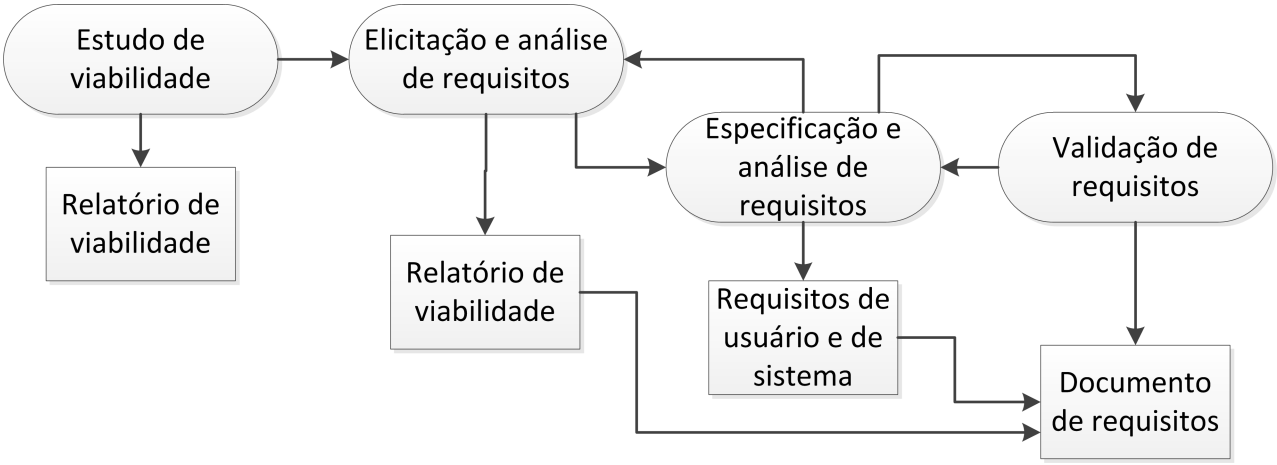

Figura 4 – Processo de engenharia de requisitos (SOMMERVILLE et al., 2008)

- **Estudo de viabilidade**: nesta fase é feita uma avaliação para analisar se as necessidades da empresa contratante podem ser atendidas através do desenvolvimento de um *software*, considerando se o custo é adequado do ponto de vista comercial e se o *software* pode ser desenvolvido dentro das restrições orçamentárias. Deve ser relativamente rápida e seu resultado deve fornecer informações de tomada de decisão para dar prosseguimento às próximas fases.

- **Elicitação e análise de requisitos**: é a derivação de requisitos de sistemas através da observação de sistemas existentes, discussões com usuários, análise de tarefas, entre outros. Durante esta fase, protótipos ou modelos de sistemas podem ser desenvolvidos para ajudar na compreensão do sistema.

- **Especificação de requisitos** Atividade de traduzir as informações coletadas durante a elicitação e análise em um documento que define um conjunto de requisitos. Os tipos de requisitos definidos na Seção 2.1 devem ser incluídos neste documento.

- **Validação de requisitos**: nesta fase os requisitos são validados em relação ao que realmente a empresa contratante precisa, à consistência e abrangência do que foi documentado. O objetivo desta fase é descobrir erros existentes no documento de requisitos e proceder com as correções necessárias.

Adicionalmente, o padrão IEEE Std. 830/1998 (IEEE, 1998) recomenda que o documento de ERS contenha a descrição de requisitos funcionais. Os requisitos tendem a ser uma mistura de requisitos de usuários e requisitos de sistema descritos de forma sequencial

no documento de requisito e em acordo com os *stakeholders* (ROSENBERG; STEPHENS, 2007). Esse procedimento é o passo inicial para se obter um conjunto completo, não ambíguo e consistente do comportamento do sistema, porém, não é o suficiente para se iniciar a fase de projeto. Uma maneira de detalhar os requisitos de usuários é escreve-los como casos de uso (COCKBURN, 2000) como discutidos na próxima Seção.

2.3 Casos de uso

Um caso de uso é uma forma estruturada de documentar requisitos funcionais para que, a partir deles, elementos de projetos e o comportamento do sistema possam ser identificados. Descrições de casos de uso ajudam a responder perguntas fundamentais, tais como: Quem são os usuários do sistema? O que esses usuários esperam do sistema? Como os usuários interagem com o sistema? Como o sistema manipula os conceitos de domínio?

Um caso de uso descreve o comportamento de um sistema sob a visão funcional à medida em que ele responde a uma requisição de um usuário chamado de ator principal. O ator inicia uma interação com o sistema para atingir algum objetivo. O sistema responde obedecendo às regras estabelecidas por todos os *stakeholders*. Cenários são diferentes sequencias de comportamentos que dependem das requisições feitas pelo ator principal e das condições que cercam essas requisições. Um caso de uso pode reunir diversos cenários.

De acordo com Cockburn (2000), um caso de uso fácil de ler é formado por sentenças escritas na voz ativa (sujeito ... verbo ... objeto) e por um passo de ação simples, na qual um ator alcança um objetivo ou que o sistema responde à uma solicitação do ator.

Casos de uso são elaborados para fornecer consideravelmente mais detalhes sobre como um usuário utiliza um sistema. Não existe um modelo padrão para a representação desses detalhes. Cockburn (2000) sugere que para obter um caso de uso com informações a respeito do cenário que será executado, as seguintes informações devem estar no modelo de caso de uso:

- **Nome**: Deve ser o objetivo do caso de uso, escrito com uma frase curta e de preferência iniciada com um verbo no infinitivo.
- **Contexto de uso**: Uma sentença maior do contexto de uso, se necessário.
- **Escopo**: Qual sistema está sendo considerado caixa-preta como desenvolvimento.
- **Nível**: Resumo, objetivo de usuário ou subfunção.
- **Ator primário**: Usuário que, no cenário, interage com o sistema.
- ***Stakeholders* interessados**: Lista contendo o nome e interesse dos *Stakeholders* interessados no cenário.

- **Pré-condição**: Situações que devem ser satisfeitas para que o caso de uso inicie.
- **Garantias mínimas**: Os interesses protegidos em qualquer saída.
- **Garantias de sucesso**: Os interesses satisfeitos em um final bem sucedido, também conhecido como Pós-condições.
- **Acionador**: Ação sob o sistema que inicia o caso de uso.
- **Cenário de sucesso principal**: Conjunto de sentenças que descrevem as ações do ator primário e resposta do sistema que atinjam um objetivo do ator.
- **Extensões**: Também chamado de Fluxo alternativo, é conjunto de sentenças que descrevem uma extensão no fluxo de sucesso. Cada conjunto referindo-se a um passo do fluxo de sucesso.
- **Variações tecnológicas e dados**: São as variações que causam bifurcações no cenário, também chamados de fluxo exceção.

Outros modelos são encontrados na literatura, tais como o modelo de duas colunas sugerido por Wirfs-Brock (1993) e o modelo da *Rational Unified Process* (RUP) sugerido por Kruchten (2003).

Cockburn (2000) também sugere um conjunto de diretrizes para a escrita de casos de uso, dentre elas, as mais relevantes para este trabalho são aquelas que dizem respeito à estrutura e ao tamanho de uma sentença, são elas:

1. **Usar gramática simples**: A maneira mais fácil e simples de deixar explícito quem executa a ação é escrever a frase na estrutura "sujeito ... verbo ... objeto ... frase preposicional", isso as tornam menos suscetível de serem mal interpretadas e permite que sejam identificados a ação, ator que executa a ação e objeto (um conceito importante no domínio da aplicação) que sofre a ação.
2. **Incluir um conjunto de ações razoável**: Baseado na recomendação de Jacobson (1992), uma sentença deve representar uma transação. Essas transações podem ser classificadas como uma solicitação do usuário, uma validação do sistema, uma alteração que o sistema realiza em algum conceito do domínio ou uma resposta do sistema ao usuário. A Figura 5 mostra os tipos de transações em casos de uso. Recomenda-se também que cada transição pode ser escrita separadamente em uma sentença ou combinando-se várias transações em uma única sentença. Neste trabalho escolheu-se utilizar uma transação por sentença, pois, segundo Cockburn (2000), cada uma delas torna-se uma unidade de teste para validação da consistência para os requisitos. A Figura 6 mostra um trecho de caso de uso e associa cada sentença com um tipo de transação.

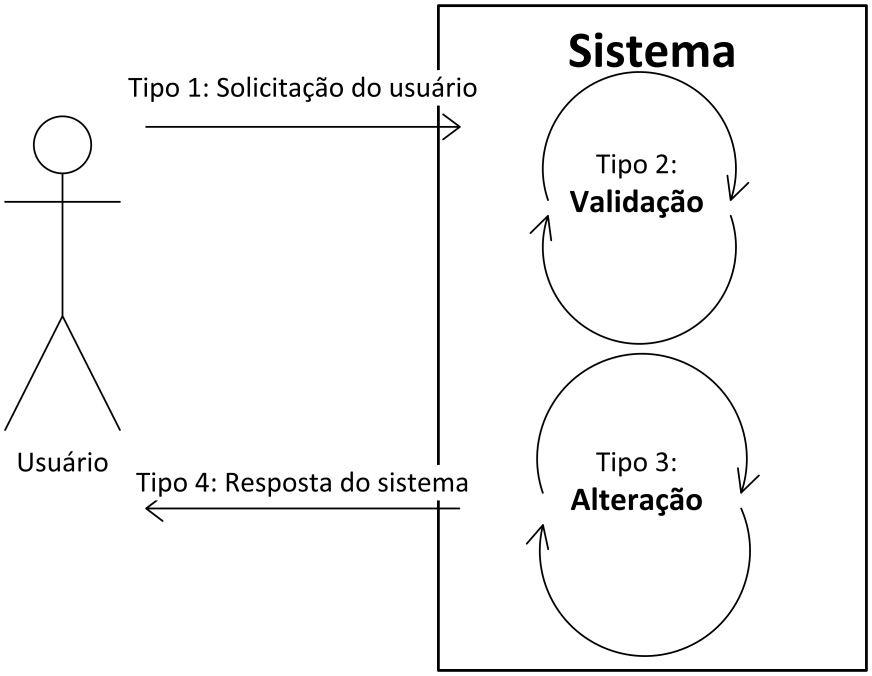

Figura 5 – Tipos de transações em um caso de uso

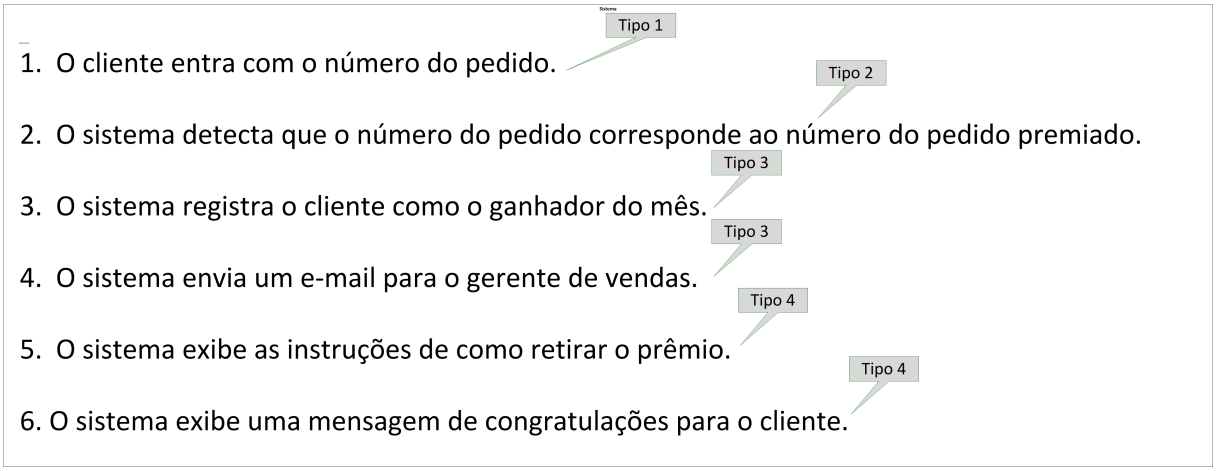

Figura 6 – Exemplo de Caso de uso com uma transação por sentença

2.4 Processamento de Linguagem Natural

Linguagem Natural é a forma de comunicação falada ou escrita (tais como Inglês, Português ou Espanhol) normalmente utilizada entre pessoas, (BIRD; KLEIN; LOPER, 2009b). O termo "Natural" é utilizado para distinguir a fala e escrita humana das linguagens mais formais, tais como notações matemáticas ou lógicas, ou linguagens de programação.

O conjunto de métodos computacionais formais desenvolvidos para tratar dessas Linguagens Naturais é denominado Processamento de Linguagem Natural (PLN). Segundo Jackson e Moulinier (2002) o termo Processamento de Linguagem Natural é normalmente utilizado para descrever a função dos componentes de *software* ou hardware em um sistema computacional que analisam ou sintetizam a linguagem escrita ou falada.

A área de PLN busca extrair informações estruturadas de textos livres. De modo geral, equivale a descobrir em um texto os seis principais elementos usados para organizar os pensamentos: quem, o que, quando, como, onde e por quê. Esses elementos são oriundos do jornalismo investigativo e são referenciados como *6W (who, what, when, how, where e why)*. Para isso, técnicas de PLN frequentemente utilizam conceitos linguísticos como classes gramaticais (Substantivos, Verbos e Adjetivos) e estruturas gramaticais (Sintagmas Nominais, Sintagmas Verbais e Sintagmas Preposicionais), além de tratar de grandes desafios como resolução de anáforas, ambiguidades e semântica.

As técnicas de PLN fazem uso de várias representações de conhecimento, como léxicos de palavras e seus significados, propriedades gramaticais, conjuntos de regras gramaticais e frequentemente outros conjuntos de recursos como ontologias de entidades e ações, ou ainda *Thesaurus* de sinônimos e abreviações (KAO; POTEET, 2010).

Nas subseções seguintes serão apresentadas técnicas e ferramentas pertencentes à área de PLN que foram utilizadas nesta Dissertação.

2.4.1 Etiquetador (Part-Of-Speech Tagger)

Em processamento de linguagem natural etiquetadores são sistemas que analisam um texto *tokenizado*[2] e inserem etiquetas morfológicas, gramaticais ou sintáticas a cada *token*. Um etiquetador morfossintático, analisa o texto e classifica cada palavra em uma categorial gramatical como: substantivo, verbo, adjetivo, advérbio, pronome, dentre outras. Para sinais de pontuação pode ser utilizado o próprio sinal, para palavras estrangeiras e fórmulas utiliza-se uma etiqueta única. Ou seja, etiquetar morfossintaticamente um texto de uma dada língua é atribuir um rótulo ou etiqueta (tag) de um conjunto de rótulos (tagset) a cada palavra da língua, símbolo de pontuação, palavra estrangeira, ou fórmula matemática de acordo com o contexto em que aparecem.

Um *Part-Of-Speech Tagger* (POS-Tagger) é um sistema responsável por etiquetar cada item lexical ao analisar um texto dado como entrada, sendo que o conjunto de rótulos são as classes gramaticais pertencentes a língua envolvida na etiquetagem referente a sua categoria morfossintática ou gramatical.

O processo de etiquetagem automática basicamente funciona em 3 etapas: pré-processamento do texto; classificação gramatical e desambiguador.

A etapa de pré-processamento tem o objetivo de preparar o texto de entrada no formato aceito pelo etiquetador para separação dos *tokens*. É importante lembrar que nem todos os etiquetadores possuem esta etapa por requerer que os textos já serão enviados para etiquetagem no formato aceito pela ferramenta.

[2] Do inglês *tokenize* é o processo de divisão do texto em unidades chamadas *tokens*, onde cada unidade é uma palavra, ou ainda um número ou uma marca de pontuação. Neste trabalho, utilizou-se o ANTLR para a tokenização de textos.

Neste trabalho, foi utilizado o etiquetador morfossintático Tree Tagger (SCHMID, 1994). O Tree Tagger é um etiquetador probabilístico para a língua inglesa que possui uma adaptação para o português. Utiliza árvores de decisão probabilísticas e a partir de trigramas (sequências de três palavras consecutivas encontradas em um *corpus*) cria relações entre as classes gramaticais. Para concluir qual a classe gramatical de determinada palavra é necessário responder afirmativa ou negativamente perguntas relativas às palavras que aparecem ao seu redor. À medida em que cada resposta afirmativa é dada, as informações na árvore são conectadas chegando-se a uma resposta: folha da árvore. O etiquetador também possui um léxico que foi criado a partir de uma parte do *corpus Penn Treebank*. Dois milhões de palavras desse *corpus* foram etiquetadas e serviram de treinamento, ou seja, a partir dos dados obtidos, criam-se regras probabilísticas que são utilizadas na tarefa de etiquetagem de quaisquer outros *corpora*.

O Tree Tagger também gera como saída o *lemma*, que é a forma canônica da palavra, ou seja, a formação da palavra. A forma canônica da palavra é muito importante pois reduz a forma flexionada das palavras, por exemplo as palavras: “executa” e “executou” irão gerar o mesmo *lemma* “executar”.

2.4.2 *Stemming*

Stemmers são analisadores morfológicos que associam variantes de um mesmo termo com sua forma raiz, chamada de radical (CHAVES, 2003). O processo de *stemming* visa a obtenção do radical de uma palavra, seja ela uma forma verbal flexionada, um substantivo, um adjetivo ou de outra classe gramatical. O processo de *stemming* é utilizado quando se deseja agrupar palavras com diferentes grafias, ou de diferentes categorias gramaticais, mas relacionados a um mesmo conceito, por exemplo, as palavras “inserção” e “inserir” possuem o mesmo radical “inser”, embora pertençam a classes gramaticais diferentes: “inserção” é um substantivo e “inserir” é um verbo no infinitivo.

A raiz resultante do processo de *stemming* não é necessariamente idêntica ao radical linguístico, mas servirá como uma denotação miníma não ambígua do termo. *Stemming* consiste em reduzir as palavras ao mesmo *stem*, por meio da retirada dos prefixos e sufixos, permanecendo apenas um radical (CHAVES, 2003).

A utilização de radicais e não dos próprios termos é uma forma de normalizar o processo de recuperação de informações, além de agrupar termos que apresentam similaridade morfológica e proximidade conceitual.

Um dos mais conhecidos algoritmos de *stemming* é o algoritmo de Porter, descrito em (PORTER, 1997), que foi desenvolvido originalmente para língua inglesa, mas existe uma adaptação para a língua portuguesa. No entanto, como as características da língua inglesa são distintas da língua portuguesa, preferiu-se utilizar o algoritmo proposto por

Orengo e Huyck (2001), desenvolvido exclusivamente para a língua portuguesa. Utilizou-se a ferramenta *PTStemmer* (OLIVEIRA, 2014) para a extração dos radicais de palavras.

2.4.3 *Stopwords*

Stopwords são termos de uma língua que não apresentam relevância semântica para a análise do problema. Para o português, alguns trabalhos consideram como *stopwords* palavras das classes gramaticais: artigo, pronome, preposição, numeral e conjunção. No entanto, existem listas genéricas de *stopwords* avaliadas para muitas línguas, porém elas não podem ser definitivas, pois a importância semântica de cada palavra depende do domínio do problema e do seu propósito.

De acordo com Miriam (2007) uma maneira de encontrar termos pouco representativos combina as leis de Zipf (ZIPF, 1949) e Luhn (LUHN, 1958). A teoria de Zipf afirma que o produto da frequência de um termo e sua classificação (*rank*) é aproximadamente constante. Obtém-se a frequência dos termos de um conjunto de documentos, gera-se a classificação desses termos em ordem descendente de frequência (*rank*) e gera-se um gráfico de classificação versus frequência. A Figura 7 apresenta a curva da lei de Zipf.

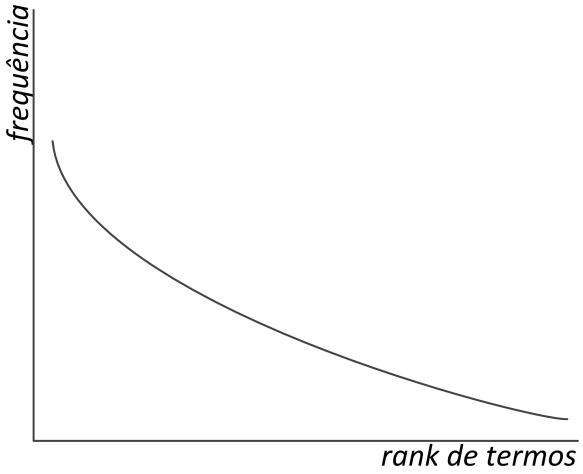

Figura 7 – Gráfico segundo a lei de Zipf

Esta teoria foi utilizada posteriormente por (LUHN, 1958) que ponderou os termos relevantes para a indexação de documentos que ficam concentrados numa região delimitada por dois pontos de corte (MIRIAM, 2007). Termos acima do limite superior e termos abaixo do limite inferior são considerados pouco discriminantes, por serem muito frequentes ou por serem muito raros. Uma das formas de implementar o conjunto de termos que pertencem ao limiar superior é agrupar artigos, pronomes, advérbios, preposições e conjunções num conjunto denominado de *stop list*. A Figura 8 mostra os limites superior e inferior para a curva de Zipf, segundo a abordagem de Luhn.

A região que engloba os termos com maior frequência inclui termos como artigos (a, o), preposições (de, da), conjunções (ou, e), que estão presentes em quase todos os

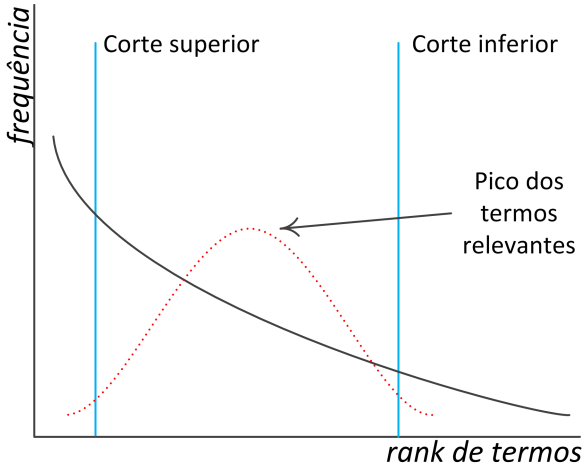

Figura 8 – Gráfico de Zipf com os cortes propostos por Luhn

documentos e não são significativos para seus conteúdos. A região central inclui termos como substantivos, adjetivos e verbos que podem ser utilizados para a representação de documentos. A terceira região inclui termos com baixa frequência de ocorrência, muitas vezes apenas uma ocorrência, e que são considerados "ruídos". A delimitação dessas regiões não é uma tarefa trivial e a decisão sobre os limites está associada a um certo grau de arbitrariedade (MIRIAM, 2007).

2.4.4 Gerador de Lexer e Parser - ANTLR

O ANTLR (***AN****other* ***T****ool for* ***L****anguage* ***R****ecognition*), descrito em (PARR, 2007), é um gerador de *parser* que automatiza a construção de reconhecedores para linguagens de domínios específicos. Ele gera uma gramática combinada que especifica tanto o *parser* quanto as regras léxicas da linguagem. O formato dos *tokens* da linguagem desejada é definido através de expressões regulares. As regras gramaticais do parser são descritas através de uma gramática livre de contexto, notação EBNF (***E****xtended* ***B****ackus* ***N****aur* ***F****orm*) adotada como padrão ISO/IEC 14977. Utilizou-se essa ferramenta para definição e reconhecimento dos padrões linguísticos e das Linguagens Naturais Controladas utilizadas nesta Dissertação.

Para exemplificar, a Figura 9 mostra uma gramática de nome "Expr" que gera uma sequência de atribuições de expressões matemáticas simples, a variáveis (Adaptado do exemplo da seção 5.5 de (PARR, 2007)). Os termos em minúsculo representam as regras de produção, os termos em maiúsculo são os terminais, o símbolo "?" significa que o termo pode ou não ocorrer, o símbolo "|" é semelhante ao "ou" lógico, o símbolo ":" separa uma regra de suas produções, o símbolo "+" indica que o termo/expressão deve ocorrer pelo menos uma vez e o símbolo "*" indica que o termo/expressão pode ocorrer nenhuma, uma ou várias vezes. A linha 1 define o nome da gramática.

As linhas de 3 a 5 definem as regras para os terminais. O terminal ID significa

```
1.      grammar Expr;
2.
3.      ID  :  ('a'..'z'|'A'..'Z')+;
4.      INT :  '0'..'9'+;
5.      NEWLINE:   '\r'? '\n';
6.      WS  :  (' '|'\t')+ { skip(); };
7.
8.      prog:  stat+
9.           ;
10.
11.     stat :  expr NEWLINE | ID '=' expr NEWLINE | NEWLINE
12.                ;
13.
14.     expr : multExpr ( '+' multExpr | '-' multExpr )*
15.                ;
16.
17.     multExpr :  atom ('*' atom )*
18.                ;
19.
20.     atom :  INT  |  ID  |  '(' expr ')'
21.                ;
```

Figura 9 – Exemplo de uma gramática

uma sequencia de letras. O terminal INT significa uma sequencia de números inteiros. O terminal NEWLINE significa um *Carriage Return* ou *Line Feed*. A linha 6 indica que analisador léxico (*lexer*) gerado pelo ANTLR deve ignorar espaços em branco e tabulações. A regra inicial da gramática é chamada “prog” e é definida na linha 8, essa regra gera um comando ou uma sequência de comandos a partir da regra “stat” (linha 11). Um “stat” pode ser uma “expr” seguida de uma mudança de linha (NEWLINE). Um “stat” também pode ser um identificador (ID) seguido de um sinal de igual (’=’) seguido de uma expressão e uma mudança de linha. Um “stat” ainda pode ser uma mudança de linha, que não fará nada. Já na linha 14 temos a definição da regra “expr”, que consiste da regra “multExpr” seguida de nenhuma, uma, ou várias repetições do que estiver entre parênteses. Entre parênteses temos o sinal de adição (’+’) seguido de uma “multExpr” ou (|) um sinal de subtração (’-’) seguido de uma “multExpr”. Uma “multExpr” (linha 14) significa uma expressão de multiplicação e consiste de uma regra “atom” seguida de nenhuma, uma, ou várias repetições do que estiver entre parênteses, neste caso o que temos entre parenteses é um sinal de multiplicação (*) seguido de uma regra “atom”. Finalmente, na linha 20 ocorre a definição da regra “atom”. Um “atom” pode ser um terminal INT ou um terminal ID ou uma “expr” entre parenteses. O *parser* gerado pelo ANTLR para essa gramática reconhecerá expressões como: x = 1; y = 2 e 3 * (x + y).

2.5 Linguagem Natural Controlada - LNC

Uma Linguagem Natural Controlada (LNC) é um subconjunto de linguagem natural, cujas gramática e dicionário foram restringidos de modo a reduzir ou eliminar a ambiguidade e complexidade. Segundo Schwitter (2007) uma LNC usa:

- **Um vocabulário de domínio específico (Léxico)**, a fim de evitar que se tenha dois termos diferentes referindo-se à mesma entidade no domínio da aplicação; e a ambiguidade léxica, isto é, o mesmo termo referindo-se a duas ou mais entidades no domínio da aplicação; e

- **Um conjunto restrito de regras gramaticais**, que podem ser gerais, exemplo "escreva frases curtas e simples", ou mais formais com regras gramaticais restringindo as estruturas sintáticas aceitáveis, no sentido de evitar a ambiguidade estrutural, ou seja, uma sentença sendo mapeada em duas ou mais estruturas sintáticas diferentes.

O estudo do processamento de linguagem em domínio limitado é importante por algumas razões: primeiro, a construção de uma aplicação de PLN de sucesso para uma sub-linguagem deve explorar suas restrições gramaticais e lexicais, permitindo "desvios" que podem ser perfeitamente aceitáveis por especialistas de domínio; uma segunda razão é que pode ser possível construir uma descrição linguística relativamente completa, mesmo com as limitações de vocabulário, sintaxe e semântica da sub-linguagem.

As LNCs têm características similares às de uma sub-linguagem, uma vez que são usadas em domínios específicos. Por exemplo, no domínio de um sistema de contabilidade, "receita" significa todos os recursos provenientes da venda de mercadorias ou de uma prestação de serviços; no âmbito de um sistema de saúde "receita" é uma ordem escrita, emitida por profissional habilitado com instruções detalhadas sobre medicamentos que devem ser dados ao paciente. Cada um desses domínios tem um vocabulário específico para suas necessidades. A principal diferença entre as LNCs e as sub-linguagens é que o léxico de uma linguagem controlada, a sintaxe e a semântica são construídos com objetivos particulares em mente, enquanto as restrições de uma sub-linguagem não são específicas (SCHWITTER, 2007).

2.6 Considerações finais

Neste capítulo foram apresentados alguns conceitos sobre Engenharia de Requisitos, Casos de uso, Processamento de Linguagem Natural e Linguagem Natural Controlada. Esses conceitos são de fundamental importância para o entendimento do trabalho, uma vez que servem como base para a definição e implementação da metodologia. O próximo capítulo

discute os principais trabalhos relacionados, encontrados na literatura, que contribuíram para a formulação da Dissertação ora apresentada.

3 Trabalhos Relacionados

Na literatura pesquisada, foram encontrados alguns trabalhos com objetivos semelhantes ao desta Dissertação. A seguir, discutiremos aqueles que consideramos mais relevantes e que mais contribuíram para a conclusão de nossa proposta.

3.1 Miriam

Miriam (2007) propõe uma estratégia que combina técnicas de Processamento de Linguagem Natural (PLN) e agentes de *software* para apoiar as atividades de verificação e validação de requisitos. Nessa estratégia são geradas visões textuais ou gráficas de grupos de requisitos relacionados. As visões apoiam a análise de completude, a identificação de duplicidades e identificação de dependências entre requisitos. São utilizadas técnicas de análise de conteúdo para apoiar a identificação de omissões em requisitos não funcionais, bem como uma estratégia para a construção ou atualização do Léxico Ampliado da Linguagem (LAL), originalmente proposto por Leite e Franco (1993). Agentes de *software* são usados para implementar os serviços que incorporam as estratégias da proposta e utilizam uma base de sufixo para a identificação de atores. Semelhante à essa proposta, a metodologia apresentada nesta Dissertação utiliza técnicas de mineração de textos para a criação de um léxico da aplicação e faz uso de um dicionário de sinônimos para redução da ambiguidade. Diferente da metodologia ora apresentada, Miriam (2007) não propõe e nem gera um documento de requisitos baseado em (IEEE, 1998), uma vez que sua proposta é analisar qualquer documento que contenha descrições de requisitos de sistemas. Além disso, não faz uso de padrões linguísticos, padrões de requisitos e nem de LNC para expressar ou reconhecer requisitos de sistemas.

3.2 PABRE

Palomares, Quer e Franch (2011) descrevem o método *Pattern-based Requirements Elicitation* (PABRE) que defende a reutilização de requisitos não-funcionais através da criação e evolução de um repositório de padrões de requisitos. O resultado é um catalogo de requisitos não funcionais para ser usado em processos de aquisição. O método PABRE consiste no acúmulo de experiência de requisitos com finalidades semelhantes e a partir deles deduzir um modelo refinado de requisitos bem formados e com espaços reservados para pontos de extensão. Além disso, cada modelo é enriquecido com metadados detalhados que está estruturado de acordo com uma organização baseada em modelos de formulários. O método PABRE também fornece duas ferramentas de suporte: *(i)* PABRE - Man

para auxiliar a gestão e evolução do catálogo padrões de requisitos; e *(ii)* PABRE - Proj para dar suporte à instanciação de padrões de requisitos em nível de projeto e aplicá-los a um contexto específico. Assim como PABRE, a metodologia hora apresentada é apoiada por uma ferramenta computacional, utiliza padrões linguísticos, baseados em frases frequentemente usadas para descrever requisito e padrões de requisitos baseados em formulários, porém, é específico para requisitos funcionais.

3.3 RSLingo

Ferreira e Silva Ferreira e Silva (2012) apresentam a *Requirements Specification Language* (RSLingo), uma abordagem para a melhoria da qualidade das especificações de requisitos que se baseia em duas linguagens e no mapeamento entre elas, a saber: RSL-PL, uma linguagem extensível para lidar com a extração de informações a partir de padrões linguísticos; e RSL-IL, uma linguagem formal com um conjunto fixo de construções para representar e transmitir conceitos específicos da Engenharia de Requisitos. Essa dissociação permite lidar com requisitos como itens de "caixa-branca"[1], possibilitando uma compreensão mais profunda em relação à semântica. Assim, o RSLingo permite a automação de tarefas de verificação que previnem problemas de qualidade de requisitos e estabelece uma base para integração de Engenharia de Requisitos com o paradigma *Model-Driven* (SCHMIDT, 2006) através de transformações de representações de requisitos em modelos de projeto. O RSLingo utiliza o alinhamento RSL-PL => RSL-IL, técnicas de PLN *chunk parsing* (BIRD; KLEIN; LOPER, 2009a) e o algoritmo da Distância de Levenshtein (NAVARRO, 2001) para a extração automática de informações (CUNNINGHAM, 2005) e semântica dos conceitos do domínio do sistema.

As semelhanças entre o RSLingo e a metodologia proposta neste trabalho são:

- **Uso de padrões linguísticos para capturar conceitos específicos do domínio do sistema**: O RSLingo espera que o especialista do domínio escreva o requisito nos padrões linguísticos definidos pelo arquiteto do sistema. A metodologia proposta nesta dissertação guia o analista de requisitos a escrever na estrutura da LNC definida para o requisito;

- **Criação do léxico do domínio**: No RSLingo o léxico é montado manualmente pelo analista de requisitos e o especialista do domínio.Na metodologia proposta existem duas maneiras de se construir o léxico: através da análise automática dos documentos de referência ou na etapa de verificação de consistência. Nas duas propostas o objetivo do léxico é de apoiar as tarefas de desambiguação e fornecer informações adicionais sobre o significado dos termos do domínio e suas relações léxicas;

[1] Itens caixa-branca são itens sobre os quais se conhecem sua estrutura interna e se possibilita que sejam escolhidas partes específicas para serem avaliadas.

- **Extração automática de informações**: A metodologia proposta nesta dissertação também utiliza técnicas de PLN através do uso de um *parser* para a LNC definida para os padrões linguísticos. Além disso, a estrutura do documento de ERS proposto reforça a semântica do domínio, o que permite uma compreensão mais profunda do que aquela sugerida na abordagem RSLingo.

3.4 FSER

Esteca et al. (2012) propõem a incorporação de um módulo de apoio à especificação de requisitos a uma ferramenta Web chamada Ferramenta de Suporte à Elicitação de Requisitos (FSER), com o intuito de contribuir para a criação de documentos de ERS completos, baseados em um padrão bem definido e amplamente utilizado na indústria de *software*, IEEE Std. 830/1998 (IEEE, 1998). O trabalho contribuiu para aprendizado de conceitos de especificação de requisitos por usuários inexperientes, dado o direcionamento para o preenchimento das informações que compõe o documento de requisito. Seu uso também contribui para apoiar as organizações em seus processos de melhoria da maturidade, fornecendo um recurso automatizado de apoio à realização das atividades cotidianas relacionadas às áreas de processo, gerência e desenvolvimento de requisitos definidas no modelo CMMI - *Capability Maturity Model Integration* (TEAM, 2000).

A única semelhança da metodologia apresentada nesta Dissertação com a FSER é o uso do padrão IEEE Std. 830/1998 (IEEE, 1998) como base para o documento de requisito proposto, porém, estruturas diferentes foram utilizadas nos dois trabalhos. Além disso, nossa proposta utiliza a estrutura do documento de requisitos para: definir a semântica do domínio; auxiliar as verificações de consistência; e auxiliar a identificação dos elementos de projeto, tudo isso com o apoio do protótipo ERS-EDITOR.

3.5 Considerações finais

Neste capítulo forma apresentados alguns trabalhos que ajudaram a elaboração da metodologia proposta. Os trabalhos foram analisados sob a perspectiva de definir as etapas da metodologia e encontrar e adaptar práticas que ajudassem a automatizar as atividades relacionadas às etapas. As principais contribuições dessa pesquisa bibliográfica foram: (*i*) uso do léxico da aplicação definido por Miriam (2007); (*ii*) uso de padrões linguísticos e padrões de requisitos proposto por Ferreira e Silva (2012), Palomares, Quer e Franch (2011), Toro et al. (1999); (*iii*) uso de etapas da metodologia sugerido por Ferreira e Silva (2012); (*iv*) uso de Linguagem Natural Controlada para expressar requisitos, conforme proposto por Palomares, Quer e Franch (2011), Ferreira e Silva (2012); e (*v*) o uso de ferramenta de apoio à Engenharia de Requisitos, semelhante ao proposto por Esteca et al. (2012).

No próximo capítulo a metodologia $HELP_4ERS$ será apresentada por meio de explicações das cinco etapas que a compõe. Além disso, mostra-se como essas etapas são automatizadas pelo protótipo ERS-EDITOR. Por fim, serão apresentados resultados preliminares de um experimento feito para testar parte da metodologia.

4 A Metodologia HELP$_4$ERS

Este capítulo tem como objetivo apresentar a metodologia HELP$_4$ERS para a escrita de documentos de requisitos. Suas etapas serão descritas como ações que o analista de requisitos e o analista de domínio devem executar. Para um melhor entendimento, inicialmente serão explicados como itens que dão suporte às ações de cada etapa foram construídos, são eles: *(i)* a Estrutura para o documento de requisitos gerado com a aplicação da metodologia; *(ii)* os Padrões de requisitos e os padrões linguísticos; e *(iii)* as Linguagens Naturais Controladas.

É importante mencionar que este trabalho faz parte de um estudo mais amplo que consiste em gerar elementos e modelos de projeto de *software* a partir de documentos de requisitos. Rafael Anchiêta, em (ANCHIETA, 2014), apresenta a ferramenta ***EasyGUI Prototyping*** que contribui para a geração de **modelos de projeto** [1] a partir de descrições textuais de casos de uso utilizando técnicas de PLN. A metodologia proposta neste trabalho tem como objetivo auxiliar a escrita de documentos de requisitos completos, consistentes e não ambíguos, permitindo, assim, o suporte automatizado nas descrições dos requisitos para extração de elementos de projeto e posterior geração dos modelos de projeto. Destaca-se que essa metodologia é fundamentada nas recomendações do padrão IEEE Std. 830/1998 (IEEE, 1998) e nos padrões de requisitos e padrões linguísticos de Toro et. al (TORO et al., 1999).

4.1 Estrutura do documento de requisitos

A estrutura do documento de requisitos foi definida para auxiliar a especificação da semântica do domínio e o suporte automatizado às descrições textuais de suas seções. Esse suporte consiste em identificar, com o uso de técnicas de PLN, elementos de projeto, tais como: atores, casos de usos, classes, atributos, métodos, interfaces de usuário e elementos de interface. Para selecionar as seções do requisitos e, ao mesmo tempo, relacioná-las aos elementos, considerou-se o que o padrão IEEE Std. 830/1998 (IEEE, 1998) e (TORO et al., 1999) recomendam que seja escrito em cada seção, dessa forma escolheu-se aquelas cujo conteúdo das seções refere-se a algum dos elementos.

Assim, baseado em observações a documentos de requisitos em conformidade com o padrão (IEEE, 1998) as seções *Funções do produto*, *Atores e sistemas externos* e *Interfaces de usuário* foram selecionadas. De forma semelhante, a seção *Requisitos de armazenamento* foi selecionada a partir do trabalho de Toro et al. (1999). Além dessas seções, escolheu-se

[1] **Modelos de projeto**: diagrama de classe, diagrama de casos de uso e protótipo de interface de usuários.

também duas outras seções consideradas importantes em um documento de requisitos, são elas: *(i) Introdução* com as subseções *objetivo*, *escopo*, *referências* e *conceitos e siglas*; e *(ii) Descrições de casos de uso*. A Tabela 1 mostra todas as seções do documento de requisitos proposto e um breve resumo sobre cada uma delas. Argumenta-se que esse modelo contém as informações necessárias para a realização das atividades inerentes ao projeto de *software*.

Tabela 1 – Seções do documento de requisitos

Seções do documento de requisitos	**Resumo**
1. Introdução	Oferece uma visão geral sobre o documento
1.1. Objetivo	Apresenta o propósito do documento
1.2. Escopo	Identifica o produto de software pelo nome, explica o que ele irá fazer, descreve sua aplicação
1.3. Referências	Fornece uma lista de documentos de referência, que servem de entrada para a construção do léxico do domínio
1.4. Conceitos e siglas	Fornece as definições de termos, acrônimos e abreviaturas necessários à interpretação do domínio
2. Atores e sistemas externos	Descreve as características dos usuários do sistema ou de outros sistemas com os quais poderá ser necessária uma comunicação
3. Interfaces de usuários	Contém uma descrição das entradas e saídas do sistema
4. Requisitos de armazenamento	Contém quais informações relevantes devem ser armazenadas pelo sistema
5. Funções do produto	Resumo das principais funções que o sistema irá desempenhar
6. Descrições de casos de uso	Deve conter uma sequência de ações de como sistema e usuários devem interagir para que os objetivos do produto sejam alcançados

Com o objetivo de auxiliar na definição semântica do domínio da aplicação, os elementos de projeto são associados a elementos das seções do documento de requisitos, a saber: Atores são associados à seção *Atores e sistemas externos*, classes e atributos são associados à seção *Requisitos de armazenamentos*, métodos e casos de uso são associados às seções *Funções do produto* e *Descrição de casos de uso*, além disso, as interfaces de usuários são associadas à seção *Interfaces de usuários*. A Tabela 2 mostra os elementos de projeto e as seções do documento de requisitos relacionados.

Para cada seção foi definido um **padrão de requisito** que ajuda o analista a expressar os requisitos do sistema. Também foram definidos dois **padrões linguísticos** para identificação de elementos de projetos. Além disso, definiu-se duas LNC's para permitir a identificação de padrões linguísticos e a verificação da consistência e da semântica de termos escritos nos documentos de requisitos.

Tabela 2 – Elementos de projeto/Seções do requisitos

Elementos de projeto	**Seções**
Atores	Atores e sistemas externos
Classes	Requisitos de armazenamento
Métodos	Funções do produto/Descrições de casos de uso
Atributos	Requisitos de armazenamento
Interfaces de usuário	Interfaces de usuários
Casos de uso	Funções do produto/Descrições de casos de uso
Elementos de interface	Descrições de casos de uso

4.2 Padrões linguísticos

Padrões linguísticos são agrupamentos de palavras, considerando a classificação gramatical (morfologica). Eles ajudam a identificar os elementos de projetos descritos nos documentos de referência e no documento de requisitos. Após a observação dos padrões linguísticos definidos em (TORO et al., 1999; FERREIRA; SILVA, 2012; PALOMARES; QUER; FRANCH, 2011; ANCHIETA, 2014) e em vários documentos de requisitos, verificou-se os padrões que se repetem com muita frequência em descrições de requisitos funcionais. Dessa forma, definiu-se dois padrões para identificação de elementos de projeto. De acordo com esses padrões linguísticos, define-se qual tratamento será dado, se ele é candidato a casos de uso, ator, classe, atributo, método, interface de usuário ou elemento de interface.

A Tabela 3 mostra os padrões linguísticos e os elementos de projeto correspondentes. O símbolo '?' significa que a classe gramatical pode ou não ocorrer, o símbolo '|' é equivalente ao 'ou' lógico. Por exemplo, na frase "O usuário externo pode consultar o status da sua solicitação" os termos "Usuário", "Usuário externo", "status", "status da solicitação" seriam identificados pelo padrão I e os termos "pode", "consultar", "consultar status da solicitação" seriam identificados pelo padrão II.

Tabela 3 – Padrões linguísticos para identificar elementos de projeto

Tipo	**Padrões linguísticos**	**Candidato a**
I	SUBSTANTIVO PREPOSIÇÃO? (SUBSTANTIVO | ADJETIVO)?	Atores, classes, atributos, interface de usuário e elementos de interface
II	VERBO SUBSTANTIVO? PREPOSIÇÃO? (ADJETIVO | SUBSTANTIVO)?	Casos de uso, métodos e interfaces de usuário

4.3 Padrões de requisitos

Padrões de requisitos são formulários para preenchimento das informações contidas nas seções do documento de requisitos, eles ajudam o analista a descrever os requisitos do sistema. O requisito é estruturado de uma forma fixa, o que promove o reuso e torna a verificação da ausência ou presença de uma informação facilmente tratada de forma automática. Os padrões de requisitos das seções 1.3, 1.4, 2, 3 e 5 são baseados no modelo das seções correspondentes da especificação de requisitos do sistema Merci[2] 1.5 que obedece ao processo Praxis (FILHO, 2001). O padrão foi escolhido por ser uma representação tabular simples (com apenas os campos nome e descrição), de fácil entendimento, ademais, o processo Praxis segue às recomendações do padrão IEEE Std. 830/1998 (IEEE, 1998). Os padrões de requisitos das seções 4 e 6 são baseados no trabalho de (TORO et al., 1999). Os autores afirmam que os padrões são amplamente encontrados em especificações de requisitos, facilitam a comunicação entre os *stakeholders* e podem ser reutilizados com algumas adaptações.

A Figura 10 ilustra o padrão de requisito para a seção 4 (*Requisitos de armazenamento*) adaptada para este trabalho. O objetivo dessa seção é registrar quais informações são relevantes e devem ser armazenadas pelo sistema. O que possibilita ao analista de requisitos a responder a questão "Quais informações relevantes no domínio do sistema devem ser armazenadas?", o padrão tem os seguintes campos:

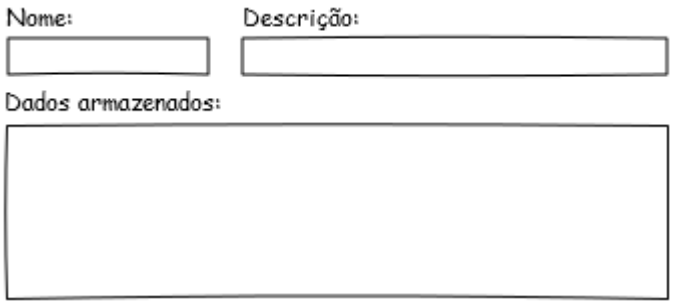

Figura 10 – Padrão de requisitos para a seção *Requisitos de armazenamento*

- **Nome**: Identificador de conceito relevante que deve ser armazenado pelo sistema;
- **Descrição**: Breve explicação sobre o conceito relevante; e
- **Dados armazenados**: Contém os dados específicos relacionados ao requisito de armazenamento. Os termos escritos neste campo que corresponderem com o padrão linguístico I da Tabela 3 poderão ser classificados como atributos, mais detalhes na Seção 4.6.3.

[2] O Merci é um exemplo de desenvolvimento de um software. Foi construído utilizando Java e possui sua modelagem descrita em UML, seguindo as convenções existentes no método PRAXIS.

A Figura 11 mostra o padrão de requisitos utilizado para a seção *Descrições de casos de uso*, baseado em (TORO et al., 1999) e nas recomendações de (COCKBURN, 2000) para quais informações devem está presente em um modelo de casos de uso, note que estas informações estão de acordo com o exposto na Seção 2.3.

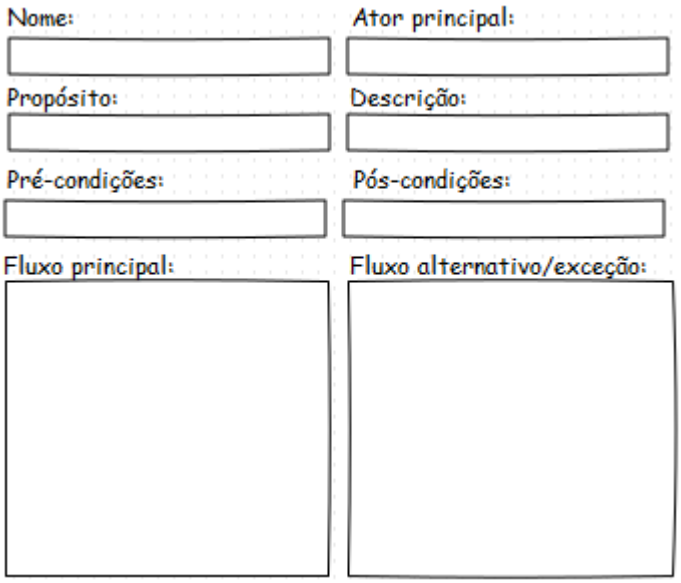

Figura 11 – Padrão de requisitos para a seção *Descrições de casos de uso*

4.4 Linguagens Naturais Controladas

Com o objetivo de minimizar a ambiguidade e complexidade nas descrições dos requisitos, definiu-se duas LNCs, uma para a seção *Funções do produto* e outra para a seção *Descrições de casos de uso*. A escolha dessas seções deve-se ao fato de que nelas, essencialmente, é descrito o comportamento do sistema.

4.4.1 *Funções do produto*

A LNC definida para a seção *Funções do produto*, denominada de S-LFS (*Simple Language to Features Specification*) aceita três tipos de sentenças com propósitos bem definidos, são eles:

1. O sistema disponibiliza que um usuário pratique uma ou várias ações sobre um requisito de armazenamento.

   Padrão 1: "O sistema deve permitir"<*ator*> ((<*ação*>)$^+$ <*requisito_de_armazenamento*> <complementos>?)$^+$. Como exemplo, temos a sentença "O sistema deve permitir ao usuário chefe anexar uma certidão retificadora". Dela, pode-se extrair os seguintes elementos:

   - *ator*: usuário chefe

- *ação*: anexar
- *requisito_de_armazenamento*: certidão retificadora

2. O ator deve executar alguma ação.

   Padrão 2: ("O"|"A") <*ator*> ("deve"|"pode") ((<*ação*>)$^+$ <*requisito_de_armazenamento*> <complementos>?)$^+$. Como exemplo, temos a sentença "O usuário externo pode baixar a certidão com a situação finalizada.", dela pode-se extrair os seguintes elementos:

   - *ator*: usuário externo
   - *ação*: baixar
   - *requisito_de_armazenamento*: certidão
   - *complementos*: situação finalizada

3. O sistema deve executar uma ação sobre um requisito de armazenamento, que pode ter um destinatário específico, quando outra ação for executada.

   Padrão 3: "O sistema deve" <*acao*> <*requisito_de_armazenamento*> <requisito_de_armazenamento_destino>? <*expressão_temporal*> <*ator*> <*ação*> <*requisito_de_armazenamento*>. Como exemplo, temos a sentença "O Sistema deve enviar um email ao usuário solicitante quando o usuário chefe finalizar sua certidão", dela pode-se extrair os seguintes elementos:

   - *ação*: enviar
   - *requisito_de_armazenamento*: email
   - *requisito_de_armazenamento_destino*: usuário solicitante
   - *expressão_temporal*: quando
   - *ator*: usuário chefe
   - *ação*: finalizar
   - *requisito_de_armazenamento*: certidão

O **fragmento** <*complementos*> que aparece nos padrões 1 e 2 é de escrita livre por parte do analista de requisito, porém, a metodologia propõe algumas inferências (descrito na Seção 4.6) para garantir a consistência no fragmento da sentença.

Inicialmente, buscou-se identificar padrões gerais. Porém, outros padrões devem ser definidos para atender especificidades de uma determinada organização ou domínio de aplicações. Portanto, novas observações devem ser realizadas em trabalhos futuros.

4.4.2 *Descrições de casos de uso*

Para garantir os atributos de qualidade perseguidos por este trabalho, definiu-se a *Simple Language to Use Case Specification* (*S-Lucas*), uma Linguagem Natural Controlada para especificação de casos de uso. *S-Lucas* foi construída com objetivos de automatizar os seguintes itens: *(i)* identificação dos elementos de projeto; *(ii)* verificação da semântica dos elementos de projetos; *(iii)* geração dos modelos de projeto; *(iv)* verificação da completude interna do documento de requisitos; e *(v)* a verificação de consistência interna do documento de requisitos. Para atingir a esses objetivos *S-Lucas* possui um Léxico com palavras reservadas, termos pré-definidos e uma gramática a fim de restringir as construções das sentenças para a especificação dos casos de uso.

A construção de *S-Lucas* foi baseada nas recomendações de boas práticas de escritas de casos de usos encontradas em (COCKBURN, 2000; ROSENBERG; STEPHENS, 2007) e da observação de 146 descrições de casos de usos, sendo 105 de sistemas desenvolvidos para o Tribunal de Contas do Estado do Piauí (TCE-PI), 31 casos de uso de um sistema desenvolvido para a Secretaria de Fazenda do Estado do Piauí (SEFAZ-PI) e 10 casos de uso da especificação do sistema Merci 1.5 (FILHO, 2001). Os casos de usos têm uma média de 32 sentenças.

As sentenças aceitas por *S-Lucas* estão de acordo com os tipos de transações explicados em (JACOBSON, 1992) e ilustrados na Figura 5 na Seção 2.3. Essas sentenças foram denominadas sentenças de ação, que são escritas na voz ativa conforme recomendado por (COCKBURN, 2000): *sujeito ... verbo ... complemento*. Esse formato facilita a identificação dos elementos de projeto estudados: *sujeito* –> ator; *verbo* –> método; e *complemento* –> requisito de armazenamento ou atributos ou interface ou elemento de interfaces. Além das sentenças de ação, *S-Lucas* aceitas uma sentença condicional com a utilização da palavra reservada “se” e uma sentença de repetição com a palavras reservada “para”.

Com o objetivo de formalizar as sentenças, efetuou-se uma análise nos 146 casos de usos disponíveis. A análise foi direcionada para responder as questões: *(i)* Quais são os verbos de ação mais frequentes nos casos de uso? *(ii)* Que tipo de objeto sofrem as ações desses verbos? *(iii)* Como são escritas as sentenças condicionais e de repetição? Os verbos que exprimem desejo não foram considerados para a resposta de nenhuma das questões, pois não estão de acordo com os objetivos das sentenças aceitas por *S-Lucas*. Após a seleção dos verbos mais frequentes verificou-se que a classificação dos complementos depende de uma análise de seu conteúdo, dessa forma, classificou-se os verbos em três categorias:

1. ***Sem objeto associado aos elementos de projeto***: Verbo cujo tipo do complemento não foi associado a algum tipo de elemento de projeto. Somente o verbo “validar” enquadrou-se nesta categoria;

2. ***Com objeto associado aos elementos de projeto***: Verbos que em todas as ocorrências nos casos de uso analisados estavam associados com o um tipo de elemento de projeto. Por exemplo, todas as ocorrências do verbo "abrir" estavam associadas à *interface de usuário*; e

3. ***Dependente do complemento***: Verbos que, para classificar o complemento como algum tipo de elemento de projeto, é necessária uma análise do conteúdo do complemento. Por exemplo, todas as ocorrências do verbo "inserir" seguido da expressão "os dados" referia-se a um requisito de armazenamento (*classe*), se não houve a presença da expressão, o verbo referenciava um dado armazenado (*atributo*).

Tabela 4 – Verbos mais frequentes

Ação	**Expressão reservada**	**Elemento esperado**
exibir	os dados os campos o campo	classe lista de atributos atributo
consultar	——	classe
inserir	os dados ——	classe atributo
executar	o comando o caso de uso	elemento de interface caso de uso
deletar	——	classe
alterar	——	atributo
imprimir	——	interface de usuário
gravar	——	classe
clicar	no botão na opção	elemento de interface elemento de interface
abrir	——	interface de usuário
selecionar	a opção o comando o botão o campo os dados	elemento de interface elemento de interface elemento de interface atributo classe
validar	——	——
escolher	a opção o comando o botão	elemento de interface elemento de interface elemento de interface

A Tabela 4 mostra os verbos selecionados, seus possíveis complementos e os elementos de projetos associados, esta associação é de fundamental importância para conduzir o analista de requisitos a escrever sentenças aceitas por *S-Lucas*, como será visto na seção 4.6.3. Definiu-se também, uma base de sinônimos para os verbos de ação. Os sinônimos foram escolhidos analisando-se os casos de usos selecionados, o trabalho de

Anchieta (2014) e a base de sinônimos TEP 2.0 [3]. A base de sinônimos encontra-se no Apêndice A.

```
1.  grammar S-Lucas;
2.
3.  sentenca : numero (sentenca_de_acao | sentenca_se | sentenca_enquanto);
4.  numero   : NUMERO (PONTO | NUMERO)*;
5.
6.  sentenca_de_acao      : (ATOR|SISTEMA) VERBO_DE_ACAO EXPR_RESERVADA elementos_de_projeto;
7.  elementos_de_projeto : elemento_de_projeto (elemento_de_projeto)*;
8.  elemento_de_projeto  : CASO_DE_USO | ATOR | REQUISITO_DE_ARMAZENAMENTO | ATRIBUTO |
9.                         INTERCAFE_DE_USUÁRIO | ELEMENTO_DE_INTERFACE;
10.
11. sentenca_se     : SE expr_logica ENTAO (sentenca_de_acao)+;
12. expr_logica     : expr ((E|OU) expr)*;
13. expr            : (sentenca_de_acao | expr_de_comp | expr_exist);
14. expr_de_comp    : elemento_de_projeto VERBO_DE_COMPARACAO comparacao elemento_de_projeto;
15. comparacao      : MAIOR | MENOR | MAIOR_OU_IGUAL | MENOR_OU_IGUAL | DIFERENTE;
16. expr_exist      : VERBO_EXISTENCIAL determinante elemento_de_projeto |
17.                       elemento_de_projeto determinante VERBO_EXISTENCIAL;
18.
19. determinante        : determinante_base pos_determinante;
20. determinante_base : (ARTIGO | PRONOME | NUMERAL);
21. pos_determinante  : (NUMERAL|PRONOME);
22. sentenca_enquanto :  ENQUANTO expr_logica FAÇA (sentenca_de_acao)+;
```

Figura 12 – A gramática *S-Lucas*

A Figura 12 mostra a gramática *S-Lucas* representada na notação **EBNF**. A regras foram obtidas pela observação das respostas às questões da análise dos casos de uso. Os terminais ATOR, REQUISITO_DE_ARMAZENAMENTO, ATRIBUTO, INTERCAFE_DE_USUÁRIO e ELEMENTO_DE_INTERFACE são reconhecidos pelo padrão I da Tabela 3. Os terminais CASO_DE_USO e INTERCAFE_DE_USUARIO são reconhecidos pelo padrão II da Tabela 3. O terminal VERBO_DE_COMPARAÇÃO é reconhecido pelos verbos de ligação "ser", "está", "permanecer" e "ficar", o termina VERBO_EXISTENCIAL, pelo verbo "existir". O terminal VERBO_DE_ACAO é reconhecido pelos verbos da Tabela 4 ou pelos verbos classificados como ação ou pelos verbos que fazem parte do domínio do sistema[4]. Os demais terminais são autoexplicativos.

As validações sintáticas e semânticas são efetuadas utilizando-se consultas às seções do documento de requisitos. Por exemplo, para validar sintática e semanticamente um termo reconhecido como ATOR, deve-se consultar à seção *Atores e sist. externos*. A figura 13 mostra uma descrição de caso de uso exemplo escrito conforme a gramática de *S-Lucas*. A Figura 14 mostra a árvore sintática da sentença "3. o usuário preenche os campos Número da certidão, ano da certidão e a chave.". Mais detalhes da implementação da gramática e suas validações são descritas na Seção 4.6.

[3] http://nilc.icmc.usp.br/tep2/
[4] Na seção 4.5 será descrito como os verbos do domínio são selecionados

Fluxo principal
1. O usuário abre a tela Validar certidão.
2. O usuário preenche campos Número da certidão, ano da certidão e a chave.
3. O usuário clica na opção Validar.
4. O usuário digita os dados da Solicitação.
5. O sistema valida que existe uma certidão com o mesmo número.
6. O sistema exibe os campos Data da solicitação, Data da disponibilização, retificadora e o tipo de certidão.
7. O sistema exibe a mensagem "Certidão validada!"

Fluxo alternativo
6.1 O usuário clica no botão Baixar certidão.
6.2 O sistema faz o download da certidão.

Fluxo de exceção
5.1 Se a certidão existir e a chave de validação estiver errada então
5.2 O sistema exibe a mensagem "Código de validação incorreto"

5.3 Se a certidão não existir então
5.4 O sistema exibe a mensagem "Certidão não encontrada"

Figura 13 – Descrição do caso de uso "Validar certidão" escrita em *S-Lucas*

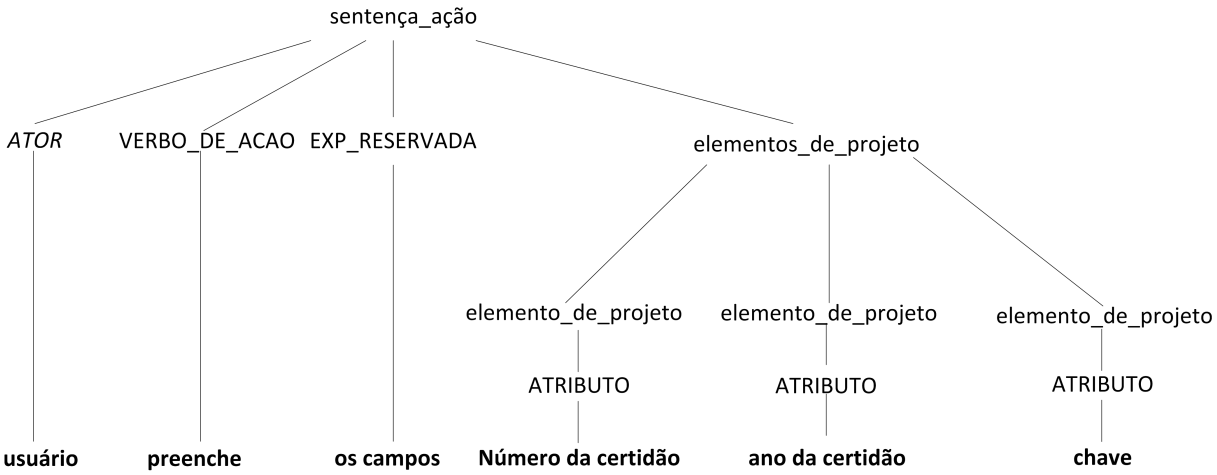

Figura 14 – Arvore sintática da sentença de ação

4.5 Etapas da metodologia

É importante destacar que ao seguir estas Etapas, os padrões linguísticos e a LNC's definidas, o analista de requisitos pode fazer uso de todos os recursos disponibilizados pela metodologia para a escrita de requisitos consistentes, o que permitirá a geração de um documento de requisitos com os atributos de qualidade desejados.

Como em Ferreira e Silva (2012) considera-se dois papeis importantes, o analista de requisitos e o analista de domínio que pode ser qualquer pessoa externa à equipe de desenvolvimento familiarizado com as regras de negócio e capaz de esclarecer dúvidas sobre o domínio do sistema. O analista de requisitos deve utilizar técnicas de ER para,

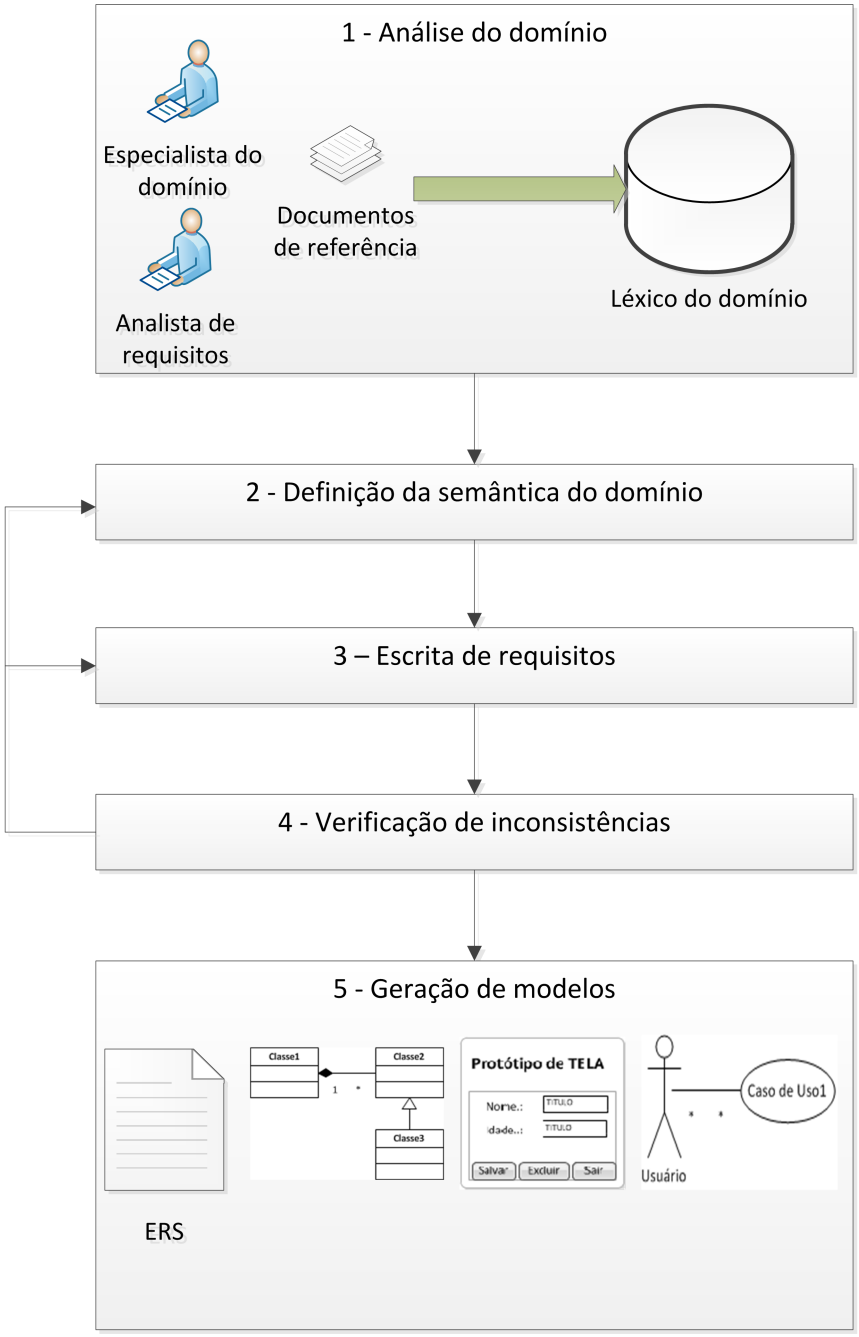

Figura 15 – Visão geral da HELP$_4$ERS

em colaboração com os analistas de domínio, descobrir os requisitos do sistema (YOUNG, 2004). Dessa forma, defende-se que tanto o analista de domínio quanto o analista de requisitos podem produzir documentos em linguagem natural sobre o domínio do sistema.

A Figura 15 mostra a visão geral da HELP$_4$ERS. A seguir serão descritas suas etapas sem focar em como elas são implementadas. As técnicas e ferramentas de PLN utilizadas para a implementação das etapas serão detalhadas no Seção 4.6.

A Etapa 1 - Análise do domínio, consiste em gerar automaticamente o léxico a partir dos documentos de referência incluídos na seção *Referências*. Esta Etapa serve como

base para as etapas posteriores. Arquivos cujo conteúdo diz respeito ao domínio do sistema e escrito em linguagem natural podem ser utilizados para criação do léxico, tais como atas de reuniões, manuais de sistemas legados e manuais técnicos. O analista de requisitos pode definir os sinônimos para mapear os termos do domínio semelhantes em uma única expressão. O dicionário de sinônimos ajuda a diminuir as inconsistências e ambiguidades em documentos de requisitos (IEEE, 1998).

A Etapa 2 - Definição da semântica do domínio, define o significado dos termos no domínio do sistema. Consiste em o analista de requisitos classificar os termos do léxico da aplicação, que foram gerados na Etapa anterior, como elementos de projeto. Essa tarefa pode ser realizada com o auxílio do analista de domínio, ele pode ajudar a esclarecer o significado de cada termo do léxico. Quando um termo é classificado, cria-se um registro no padrão de requisito da seção correspondente ao elemento. Por exemplo, ao classificar o termo "usuário externo" como ator, um registro na seção *Atores e sistemas externos* é inserido. Todos os padrões de requisitos têm um campo nome, que é preenchido automaticamente com o termo classificado. O analista de requisitos pode excluir os termos sem relevância para o domínio da aplicação. Outra atividade da etapa é a classificação dos padrões linguísticos do tipo I (Tabela 3), identificados na seção *Requisitos de armazenamento*, como Dados armazenados.

Após a classificação dos termos do léxico o analista de requisitos pode executar as atividades da Etapa 3 - Escrita de requisitos, que consiste no preenchimento dos padrões de requisitos definidos para cada seção. Os itens dos padrões de requisito podem ser criados na etapa 2, nesse caso o analista deve preencher os demais campos. Outra forma de criação dos itens é de maneira manual, quando o analista de requisito preenche todos os campo do padrão de requisito diretamente no formulário. Nesta Etapa, está disponível ao analista de requisitos o recurso de *intellisense*, que o ajuda a escrever o documento de requisitos de forma consistente, completa e não ambígua. Para fazer uso do recurso, o analista deve seguir as LNC's definidas para as seções *Funções do produto* e *Descrições de casos de uso*.

O analista de requisitos tem a liberdade de escrever de maneira diversa às LNC's, mas dessa forma não irá dispor de todos os recursos oferecidos pela metodologia o que pode levar a um documento de requisitos sem os atributos de qualidades desejados. Para detectar e corrigir essas situações, na etapa 4 - Verificação de inconsistências, o analista de requisitos pode verificar a consistência das sentenças cuja seção tenha uma LNC definida. De forma geral, a etapa consiste nos seguintes passos:

1. Identificar os padrões linguístico existentes;

2. Verificar se os padrões fazem parte do léxico;

3. Verificar se os padrões estão classificados como elementos de projetos; e

4. Verificar se os padrões estão escritos de acordo com essa classificação.

Para de garantir o suporte automatizado aos passos da verificação de consistência, a metodologia propõe o uso dos padrões linguísticos, mostrados na Tabela 3, para construção de sentenças em conformidade com as LNC's definidas. A verificação das sentenças que estão em conformidade com as LNC's pode resultar em dois tipos de mensagens: erros e avisos.

Os erros impedem que os elementos sejam identificados e detectam construções sintática e semanticamente incorretas. São classificados em dois tipos:

- **Erro de sintaxe**: O item da sentença não é o esperado pela LNC. Por exemplo, considerando o padrão 1 da LNC S-LFS, a sentença "O sistema deve permitir ao usuário chefe uma certidão retificadora" contém um erro sintático pois falta o verbo da sentença, ou seja, a ação praticada pelo usuário não foi informada.
- **Erro de semântica**: Neste caso, a sentença está sintaticamente correta, mas a classificação de um dos padrões linguísticos não está de acordo com a semântica do domínio. Utilizando novamente o padrão 1, um erro semântico ocorreria se a sentença fosse escrita da seguinte forma "O sistema deve permitir a uma certidão anexar uma certidão retificadora". Observe que a sentença está sintaticamente correta, porém, com um erro semântico. O objeto do verbo "permitir" deve ser um *ator* e o que aparece é o termo "certidão" que, no caso, é classificado como requisito de armazenamento.

Por outro lado, os avisos ajudam ao analista de requisitos a verificar se todos os elementos significantes, quer sejam casos de uso, atores, classes, operações, atributos e interfaces de usuários estão no léxico da aplicação e classificados, além de verificar se a sentença obedece à LNC. Os tipos de avisos são:

- **Termo não encontrado no léxico**: O elemento da sentença não faz parte do léxico da aplicação.
- **Termo não classificado**: O elemento da sentença faz parte do léxico da aplicação mas não foi classificado semanticamente.
- **Sentença fora do padrão**: A sentença não está escrita de acordo com o a regra da LNC definida para a seção. Assim, a metodologia alerta o analista de requisitos que ao escrever dessa forma o documento de requisitos estará sujeito aos problemas de escrita em linguagem natural.

Ao detectar um erro ou aviso, o analista de requisitos pode corrigi-lo retornando para as etapas 2 - Definição da semântica do domínio ou 3 - Escrita de requisitos. Esse

ciclo deve ser executado até que todos os erros sejam corrigidos e, opcionalmente, que nenhum aviso exista. A qualidade do resultado da etapa 5 depende da correção de todos os erros e avisos.

Por fim, a Etapa 5 – Geração de modelos, consiste em gerar de forma automática os modelos de projetos, ou seja, protótipos de interface de usuário, diagramas de classes e de casos de uso, a partir da seção *Descrições de casos de uso*. É importante destacar que essa etapa já fora descrita e validada em outros trabalhos do grupo de pesquisa em PLN da UFPI. Em especial, o trabalho (ANCHIETA, 2014) apresenta a ferramenta *EasyGUI Prototyping* que contribui para a geração de modelos de projeto a partir de descrições textuais de casos de uso utilizando técnicas de PLN.

4.6 O Suporte Automatizado

Para reduzir o esforço humano na criação e verificação de especificações de requisitos em linguagem natural, bem como auxiliar a descoberta dos requisitos e o aprendizado do domínio do sistema, além de tornar essas atividades menos propensas a erros, um conjunto de ferramentas e linguagens adequadas devem ser disponibilizadas para o analista de requisitos (FERREIRA; SILVA, 2012). Esta seção apresenta o suporte automatizado às etapas de 1 a 4 da metodologia e os detalhes da implementação de cada etapa. As ferramentas e técnicas de PLN utilizadas são aquelas apresentadas no Capítulo 2 e os exemplosnsão ilustrados utilizando o protótipo ERS-EDITOR, especialmente desenvolvido para dar suporte à HELP$_4$ERS.

4.6.1 Análise do domínio

Reportando-se ao domínio do sistema, o léxico é o conjunto de palavras e significados restrito ao domínio (REU-DEBOVE; MORAIS, 1984). Ele estabelece um vocabulário comum para termos-chaves do domínio (FERREIRA; SILVA, 2012). O objetivo da análise do domínio é a criação do léxico da aplicação, para isso são executados os seguintes passos para cada documento cadastrado na seção *Referências*:

1. **Etiquetagem**: Após a *tokenização* do texto, cada *token* recebe uma etiqueta que representa a classe gramatical da palavra.

2. **Calcular os termos relevantes**: Utiliza-se a abordagem *bag-of-words* (SALTON; MCGILL, 1983) para estruturação dos arquivos de referência e o corte inferior de (LUHN, 1958) aplicado à teoria de (ZIPF, 1949). Exclui-se as *stop words* (sinais de pontuação, artigos, preposições, numerais, pronomes, advérbios, conjunções, adjetivos e interjeições) e após o cálculo da frequência dos termos restantes (**VERBOS** e

SUBSTANTIVOS)[5], utiliza-se um parâmetro definido pelo analista de requisitos como o limite inferior. A Figura 16 mostra o corte de Luhn adaptado ao trabalho.

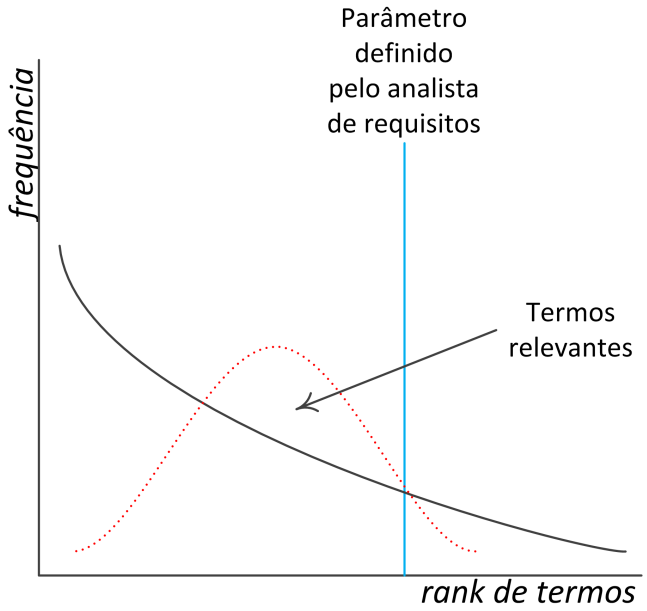

Figura 16 – Curva de ZIpf e Cortes de Luhn adaptados

3. **Construir o léxico da aplicação**: Neste passo o léxico é criado a partir dos termos relevantes e classificados como:

   - **Conceitos**: Consiste em uma lista com os SUBSTANTIVOS mais frequentes encontrados nos documentos de referência.
   - **Ações**: Consiste em uma lista com os VERBOS mais frequentes encontrados nos documentos de referência. Após a seleção, os verbos mais frequentes são reduzido à forma do infinitivo e adicionados à lista de Ações.
   - **Conceitos compostos** : São termos relevantes compostos por n-gramas[6]: bigramas (SUBSTANTIVO SUBSTANTIVO ou SUBSTANTIVO ADJETIVO) ou trigrama (SUBSTANTIVO PREPOSICAO SUBSTANTIVO) extraídos dos documentos de referência. Para cada elemento dessa lista verifica-se se um dos SUBSTANTIVOS faz parte da lista de conceitos. Se sim, o elemento é adicionado à lista de conceitos composto.
   - **Funcionalidades**: São ações sobre conceitos simples ou compostos. Consiste em uma lista composta por n-gramas extraídos dos documentos de referência pelo reconhecimento de um VERBO seguido de um conceito ou conceito composto. Para cada elemento dessa lista, reduz-se o VERBO à foma do infinitivo e

[5] Considera-se termos relevantes apenas VERBOS e SUBSTANTIVOS, pois o objetivo desse passo é reconhecer ações e conceitos mais significativos para o domínio da aplicação.

[6] n-gramas são expressões formadas por mais de uma palavra.

verifica-se se ele faz parte da lista de ações e se o(s) SUBSTANTIVO(S) faz (em) parte da lista de conceitos. Caso positivo o elemento permanece na lista, se não ele é excluído da lista de funcionalidades.

4. **Definição de sinônimos**: Neste passo o analista de requisitos define os termos sinônimos do domínio da aplicação. A Figura 17 mostra a tela que disponibiliza esse recurso, observa-se que o analista de requisitos tem as seguintes opções:

   - Adicionar sinônimo: Com esta opção o analista de requisitos adiciona um sinônimo não existente no léxico da aplicação ao termo selecionado. O sinônimo também é adicionado ao léxico da aplicação.
   - Agrupar sinônimos: Nesta opção, os temos selecionados são agrupados como sinônimos no domínio da aplicação, um dos termos selecionados é considerado como sinônimo principal e somente ele poderá ser utilizado como identificador nas seções do documento de requisitos. Na Figura 17 o sinônimo principal é "chave de validação".
   - Desagrupar sinônimos: Ao escolher está opção, o analista de requisitos exclui a associação como sinônimos entre termos do léxico da aplicação.

   É importante lembrar da regra que entre os termos agrupados, um é considerado como principal e apenas ele poderá ser utilizado como identificador nas seções do documento de requisitos. Por exemplo, na Figura 17, os termos "usuário chefe", "chefe" e "gestor" foram classificados como sinônimos e o termo "usuário chefe" foi considerado o principal, assim, apenas esse termo poderá ser utilizado no preenchimento do campo nome dos padrões de requisitos, porém, qualquer um dos termos poderá ser utilizado na escrita dos requisitos.

A Figura 18 mostra a tela do protótipo ERS-EDITOR com exemplos de termos do léxico da aplicação. As seguintes ferramentas são utilizadas para a definição do léxico: ANTLR para tokenização do texto, definição e reconhecimento dos padrões linguísticos; Tree Tagger para etiquetagem do texto; e PTStemmer (OLIVEIRA, 2014) para a obtenção do radical e o *lemma* do termo.

4.6.2 Definição da semântica do domínio

A definição da semântica do domínio consiste na classificação dos termos do léxico da aplicação em elementos de projeto. A implementação dessa fase é relativamente simples. Ao classificar um item do léxico como um elemento de projeto, um registro na seção correspondente é criado e o campo nome é preenchido com o termo selecionado. Vale lembrar que em cada padrão de requisito correspondente a um elemento de projeto, existe

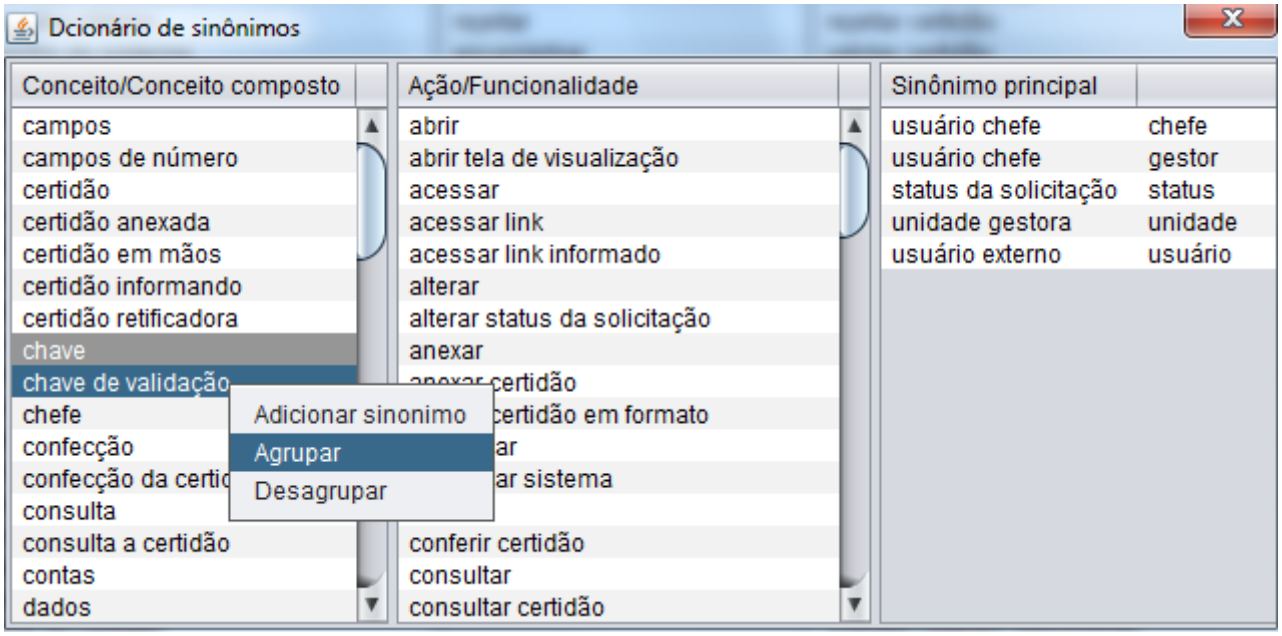

Figura 17 – Definição de sinônimos do domínio

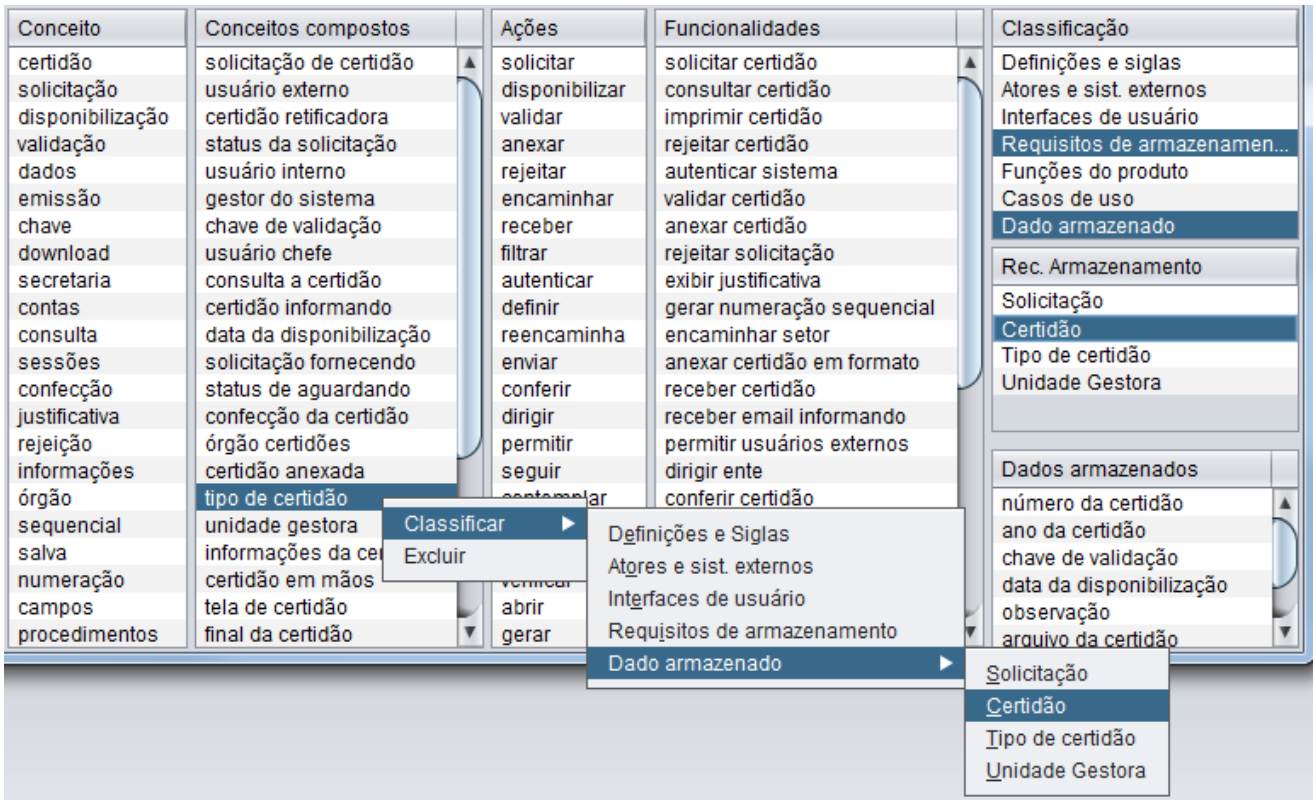

Figura 18 – Definição da semântica do domínio

um campo identificador que é formado pelo *lemma*[7] do campo nome, o *lemma* reduz a forma flexionada da palavra à sua forma canônica, assim, os termos "gestores dos sistemas" será reduzido "gestor do sistema". A Figura 18 mostra como esta etapa é suportada pelo

[7] *Lemma* é a forma canônica da palavra, ou seja, sua formação. A forma canônica da palavra é muito importante pois reduz a forma flexionada das palavras, por exemplo as palavras: executa, executou irá gerar o *lemma* executar

protótipo ERS-EDITOR. Na figura, o conceito composto “tipo de certidão” foi classificado como *Dado armazenado* do *Requisito de armazenamento* “Certidão”.

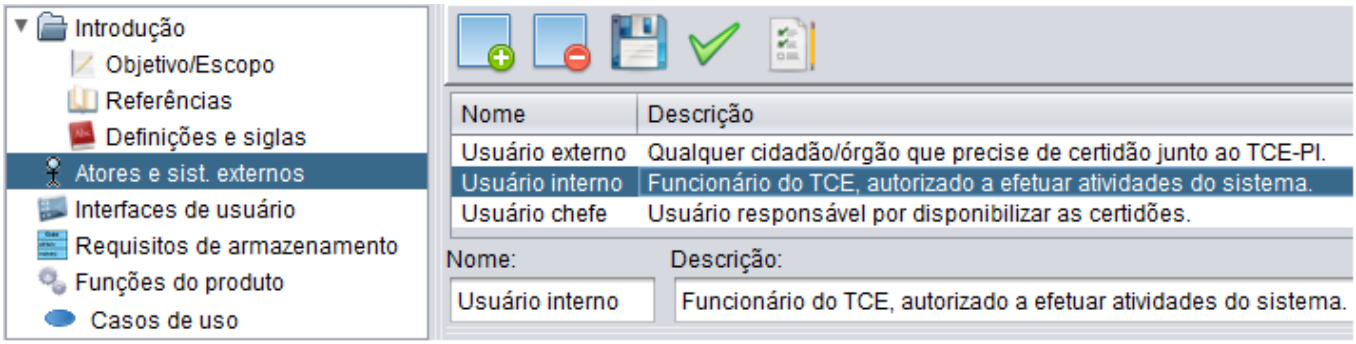

Figura 19 – Itens da seção selecionada: *Atores e sistemas externos*

A Figura 19 mostra como o registro da seção correspondente ao elemento de projeto é preenchido. No exemplo, um “usuário interno” foi descrito como um “Funcionário do TCE, autorizado a efetuar atividades no sistema”.

4.6.3 Escrita de requisitos

O objetivo dessa fase é conduzir o analista de requisitos a escrever sentenças não ambíguas, consistentes, semanticamente corretas e com suporte automatizado para extração de modelos e elementos de projeto. Para isso, faz-se o uso dos padrões linguísticos e de Linguagem Natural Controlada (LNC). Schwitter (2007) afirma que uma LNC é um subconjunto de linguagem natural, cuja gramática e dicionário são restringidos de modo a reduzir ou eliminar a ambiguidade e complexidade o que torna possível um processamento computacional em suas sentenças. Com esse objetivo, define-se uma Gramática Livre de Contexto para os padrões linguísticos, e para as LNC's das seções *Funções do produto* e *Descrições de casos de uso*. Ademais, faz-se uso dos padrões de linguísticos para reconhecimento e classificação de requisitos de armazenamento e do recurso de previsão de edição *intellisense*, que ajuda no processo de escrita e impõe as restrições das regras da LNC (SCHWITTER, 2010).

Os padrões de requisitos e as gramáticas das LNC's foram definidas com o uso da ferramenta ANTLR, que utiliza a notação EBNF (***E**xtended **B**ackus **N**aur **F**orm*) adotada como padrão ISO/IEC 14977(ISO, 1996). Detalharemos, a seguir, a utilização dos padrões de requisitos para classificação de dados armazenados na senão *Requisitos de armazenamento* e a implementação do recurso *intellisense* para as seções *Funções do produto* e *Descrição dos casos de uso*.

A seção *Requisitos de armazenamento* tem o objetivo de registrar as informações que devem ser persistidas pelo sistema em análise. O padrão de requisitos da seção possuí três campos: nome do requisito de armazenamento, descrição e dados armazenados. Conforme

visto na Tabela 2, ela está relacionada aos elementos de projetos *classes* e *atributos*. Na prática, o relacionamento é feito da seguinte forma: *classe* –> nome e *atributo* –> termos em conformidade com o padrão I da Tabela 3 classificados como *Dado armazenado* pelo analista de requisitos. A classificação de termos em atributos e classes ajuda na implementação do recurso de *intellisense* nas etapas de escrita de requisitos e geração de modelos de projetos.

O recurso é implementado da seguinte forma. O *parser* reconhece todos os padrões do tipo I do campo dados armazenados e os apresenta ao analista de requisitos, levando-se em consideração se o termo já foi classificado ou se o termo foi excluído. A Tabela 5 mostra como os termos são apresentados e as ações que podem ser executadas sobre eles.

Tabela 5 – Padrões linguísticos para identificar elementos de projeto

Classificação	**Situação**	**Ação**
Atributos	Termos classificados anteriormente como atributos e estão descritos no campo dados armazenados	Desclassificar
Não descritos	Termos classificados anteriormente como atributos e não estão descritos no campo dados armazenados	Excluir
Não classificados	Termos que não foram classificados com atributos	Classificar

A Figura 20 mostra um exemplo da utilização do recurso. Os termos atributos têm a notação de atributos, segundo a *Unified Modeling Language* - (UML) (BOOCH; RUMBAUGH; JACOBSON, 2005). Exemplos, “número da solicitação’ e “ano da solicitação”. Os termos atributos que também são requisitos de armazenamento tem a notação de classe. Por exemplo, o termo “solicitação”. O termo “tipo da solicitação” fora classificado como atributo, porém, posteriormente excluído do campo dados armazenados. Como termos não classificados temos “certidão” e “tipo de certidão” e, como mostrado, podem ser classificados como atributos.

O objetivo da seção *Funções do produto* é registrar um resumo das principais funções que o sistema deve desempenhar sem mencionar os detalhes que cada função necessita. O ERS-EDITOR oferece o recurso de *intellisense* para guiar o analista de requisitos a escrever sentenças em conformidade com a LNC definida para a seção. Para explicar sua implementação, será utilizado o trecho da gramática que reconhece o **Padrão 1**: “O sistema deve permitir” <*ator*> ((<*ação*>)$^+$ <*requisito_de_armazenamento*> <complementos>?)$^+$, descrito na Seção 1.4. A Figura 21 mostra o trecho da gramática que implementa esse padrão. A obtenção da predição do que será escrito é implementada com o auxílio de três tipos de associações entre as regras da gramática e os elementos das seções do documento de requisitos, a saber: *i*) a regra <*ator*> com a seção *Atores e sistemas externos*; *ii* a regra <*requisito_de_armazenamento*> com a seção *Requisitos de armazenamento*; e *iii*) a regra <*ação*> com a lista de ações do léxico da aplicação.

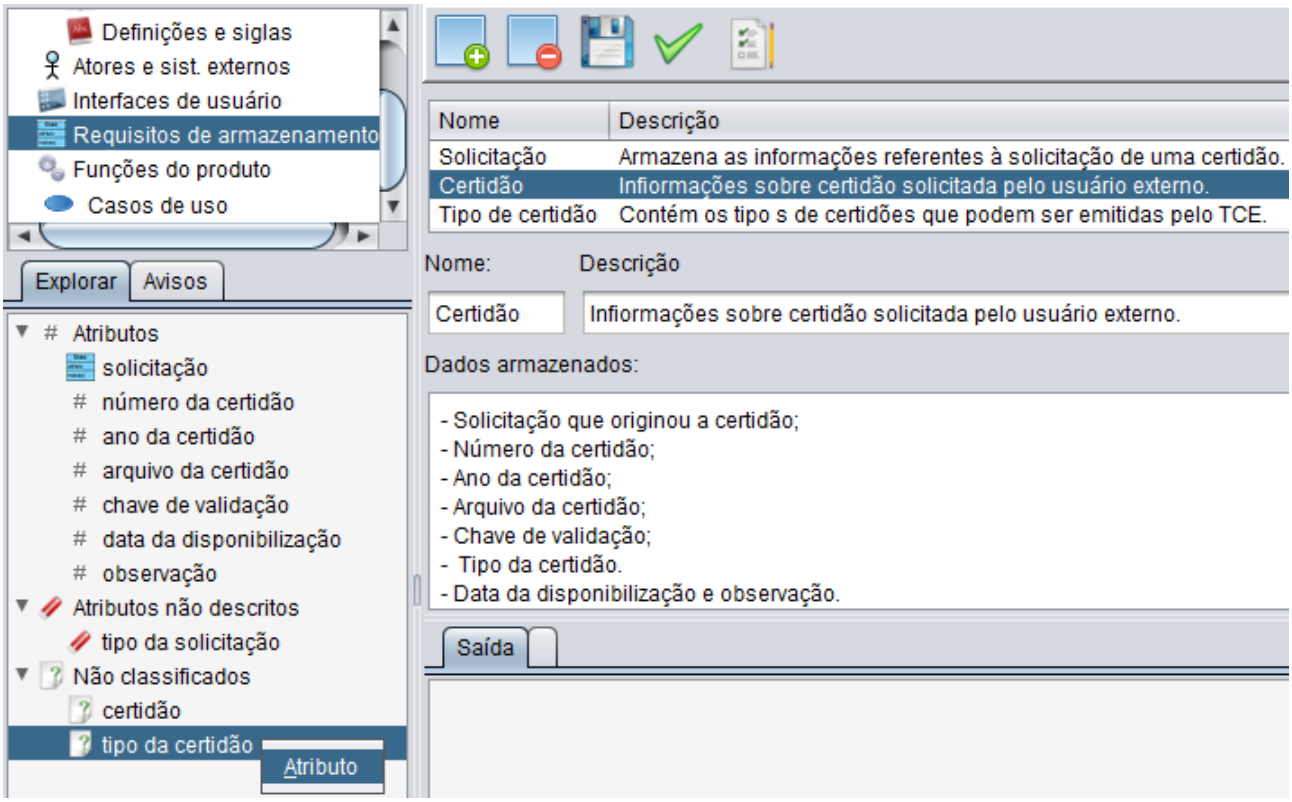

Figura 20 – Classificação de termos como atributos

```
sentenca                  : padrao_1 ator ((acao)+ requisito_de_armazenamento)+ complementos?
                          ;
padrao_1                  : ARTIGO SISTEMA VERBO_PADRAO PERMITIR
                          ;
usuario                   : locucao_substantiva
                          ;
acao                      : VERBO
                          ;
requisito_de_armazenamento : locucao_substantiva
                          ;
locucao_substantiva       : SUBSTANTIVO PREPOSICAO? (SUBSTANTIVO? | ADJETIVO?)
                          ;
complementos              : (palavras)*
                          ;
Palavras                  : ARTIGO | SUBSTANTIVO | PREPOSICAO | NUMERAL | ADJETIVO | ADVERBIO | PRONOME | CONJUNÇÃO
                          ;
```

Figura 21 – Gramática geradora do **Padrão 1**

O recurso é acionado quando as teclas “ctrl” + “espaço” são pressionadas sequencialmente. O próximo elemento é determinado com a execução das seguintes ações:

1. **Etiquetar sentença**: Essa etiquetagem ocorre em dois níveis:

   - **Nível gramatical**: Nesse nível são etiquetados os artigos, verbos, substantivos, preposições, adjetivos, numerais, advérbios, pronomes e conjunções com seus respectivos terminais ARTIGO, SUBSTANTIVO, PREPOSICAO, ADJETIVO, NUMERAL, ADVERBIO e PRONOME;
   - **Nível do léxico da LNC**: O substantivo “sistema” é etiquetado como o

terminal SISTEMA. O verbo “poder” em suas formas “pode”/“poderá” é etiquetado como o terminal VERBOPADRAO . O verbo “dever” em suas formas “deve”/“deverá”, são etiquetados como o terminal VERBOPADRAO. O verbo “permitir” é etiquetado como o terminal PERMITIR. Para ser reconhecida pela LNC a sentença deve iniciar com esse padrão .

2. **Analisar sentença**: Após a etiquetagem a sentença é analisada pelo *parser* da LNC que determina qual a próxima regra poderá ser executada.

3. **Consultar elementos**: Após a descoberta da próxima regra, uma consulta à seção ou à lista associada é feita e o resultado é exibido para que o analista de requisitos escolha qual elemento será adicionado à sentença.

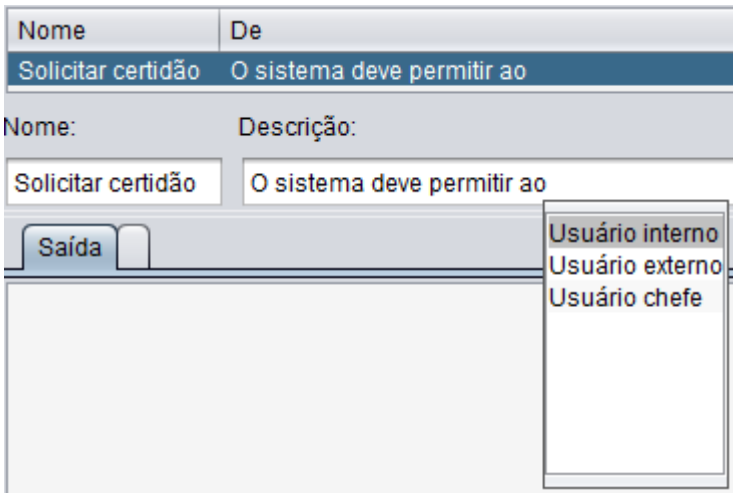

Figura 22 – *Intellisense* na seção *Funções do produto*

A Figura 22 ilustra um exemplo no uso do *intellisense* para a seção *Funções do produto*. Neste exemplo o analista de requisitos digita a sentença “O sistema deve permitir ao” e aciona o recurso, após a sentença ser tokenizada e etiquetada, o *parser* identifica a próxima regra que, neste caso, *<ator>*, então é feita uma consulta aos itens da seção *Atores e interfaces externas* e uma lista contendo os atores cadastrados é exibida. É importante observar que este recurso só permanecerá disponível se o analista de requisitos escrever sentenças de acordo com a LNC.

A Figura 23 mostra o trecho da gramática para a *sentença de ação* descrita na seção 4.4.2. Os passos da implementação do recursos para a seção *Descrições de casos de uso* são os mesmos explicados anteriormente, a única diferença é o uso da base de sinônimos para os verbos de ação.

A figura 24 mostra um exemplo do *intellisense* para uma sentença de ação em S-Lucas. Como exemplo, o analista de requisitos digita “O Usuário externo digita” e aciona o recurso. Após a sentença ser tokenizada e etiquetada o *parser* identifica que a próxima

```
sentenca_de_acao      : (ATOR|SISTEMA) VERBO_DE_ACAO EXPR_RESERVADA elementos_de_projeto;
elementos_de_projeto : elemento_de_projeto (elemento_de_projeto)*;
elemento_de_projeto  : CASO_DE_USO | ATOR | REQUISITO_DE_ARMAZENAMENTO | ATRIBUTO |
                       INTERCAFE_DE_USUÁRIO | ELEMENTO_DE_INTERFACE;
```

Figura 23 – *S-Lucas*: Sentença de ação

regra é EXPR_RESERVADA. Neste caso, o *parser* analisa o verbo digitado verificando se existe um sinônimo e analisando qual o tipo de verbo, segundo a classificação explicada na seção 4.4.2. Neste exemplo, o verbo é "digitar" que tem como sinônimo o verbo "inserir" que, segundo a Tabela 4, é do tipo 3 e tem como complementos o termo "os dados" ou os atributos cadastrados na seção *Requisitos de armazenamentos*. Assim, o *parser* carrega a expressão reservada "os dados" e os "dados armazenados" classificados como atributo.

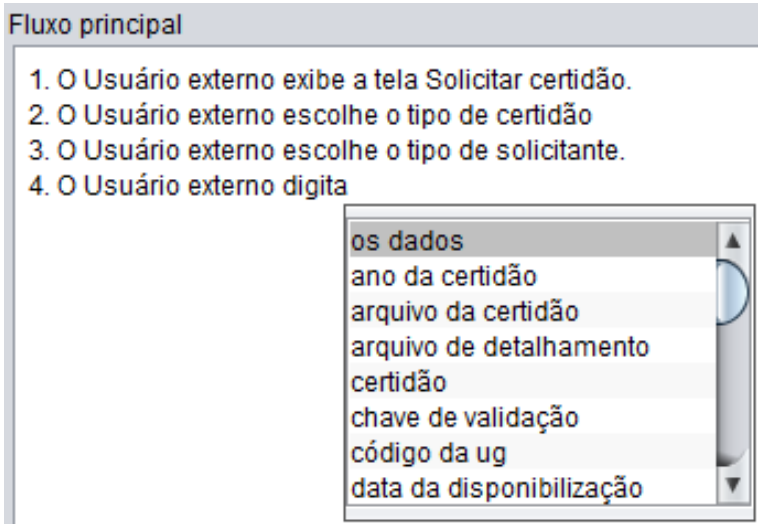

Figura 24 – *Intellisense* na seção *Descrições de casos de uso*

Outra possibilidade seria o analista de requisitos digitar uma sentença de ação com um verbo que não esteja na base dos verbos e sinônimos, esses verbos são chamados de verbos do domínio da aplicação. Neste caso o *parser* verifica se o verbo foi utilizado para definir algum nome de função do produto, nome de caso de uso ou se faz parte do léxico da aplicação na categoria de funcionalidades e apresenta as possibilidades de escrita. A Figura 25 ilustra um exemplo desta situação. O verbo "rejeitar" não faz parte da base de verbos, mas foi utilizado para definir a *função do produto* e o *caso de uso* "rejeitar certidão", além de existir a expressão "rejeitar solicitação" no léxico da aplicação. Assim, o *parser* seleciona os termos "certidão" e "solicitação" como possíveis complementos.

4.6.4 Verificação de inconsistências

O principal objetivo desta Etapa é verificar se as sentenças estão escritas de forma consistentes, semanticamente corretas e que seja possível extrair, de forma automática,

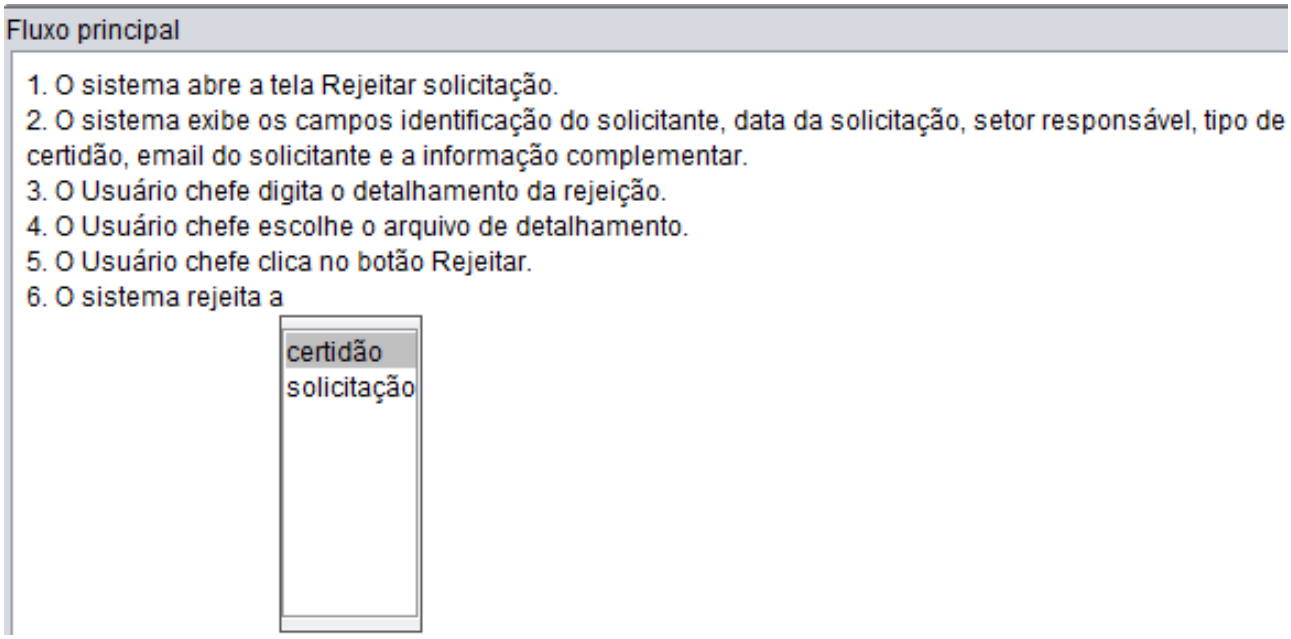

Figura 25 – *Intellisense* para verbos do domínio da aplicação

os modelos e elementos de projeto, além de verificar se todos os termos importantes do domínio foram classificados e referenciados no documento de requisitos (IEEE, 1998). Para alcançar esses objetivos as seguintes verificações são implementadas:

1. Verificar se as sentenças estão escritas, sintaticamente, em conformidade com a LNC's definida.

2. Verificar se elementos de projetos escritos na sentença estão semanticamente corretos de acordo com classificação da Etapa 2;

3. Verificar se todos os elementos reconhecidos pelos padrões linguísticos da Tabela 3 fazem parte do léxico da aplicação e estão classificados como elementos de projeto;

4. Verificar se todos os elementos de projetos das seções foram classificados e referenciados na seção *Descrições de casos de uso*.

O suporte à escrita das sentenças em conformidade com as LNC's tem objetivos diferentes nas duas seções onde elas são disponibilizadas. Na seção *Funções do produto*, o objetivo é fazer o analista de requisitos escrever de acordo com os padrões linguísticos definidos na Tabela 3. Por outro lado, na seção *Descrições de casos de uso*, o objetivo é fazer o analista de requisitos escrever os elementos de projetos conforme sua classificação semântica no documento de requisitos. A seguir será detalhada a implementação da verificação de inconsistências para as seções *Funções do produto* e *Descrições de casos de uso* e como as mensagens de erro e avisos são exibidas.

4.6.4.1 Erro sintático e semântico

Como dito anteriormente, o objetivo do suporte à escrita de requisitos na seção *Funções do produto* é conduzir o analista de requisitos a escrever os elementos de projetos de acordo com os padrões linguísticos da Tabela 3. Os erros sintáticos são descobertos analisado-se a classe gramatical dos termos e a estrutura da sentença. Para verificar a existência de erros, inicialmente ocorrem os processos de *tokenização* e etiquetagem da sentença conforme explicado nessa na Seção 4.6.3, em seguida, o *parser* verifica se o início da sentença está de acordo com algum dos padrões explicados na Seção 1.4, se não, um aviso de que a sentença está fora do padrão é gerada conforme ilustrado na Figura 26.

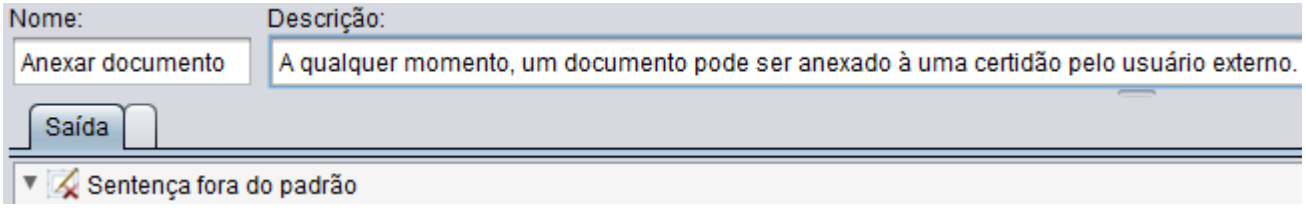

Figura 26 – Aviso de sentença fora do padrão

Após o passo anterior ser verificado, o *parser* analisa se a sentença está sintaticamente correta, ou seja, se os termos que a compõe estão de acordo com o esperado pela LNC, se não, uma mensagem de erro sintático é gerada. A Figura 27 mostra um exemplo para a sentença "O sistema deve permitir ao solicitar uma certidão", neste caso onde era esperada uma expressão do tipo SUBSTANTIVO PREPOSIÇÃO? (SUBSTANTIVO OU ADJETIVO)? foi encontrado o VERBO "solicitar".

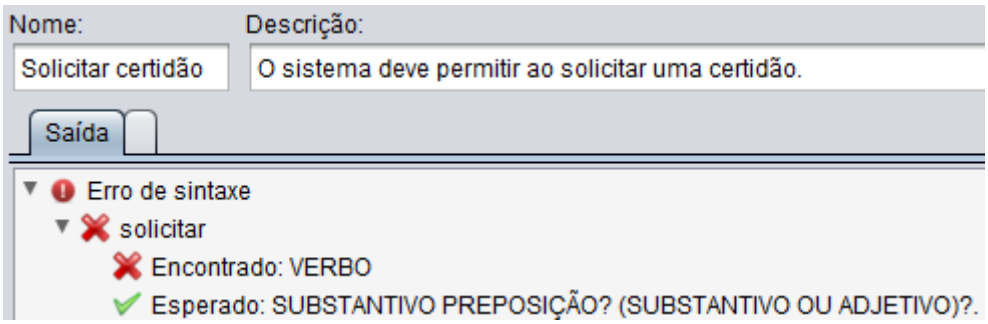

Figura 27 – Erro de sintaxe

Se a sentença estiver sintaticamente correta, o *parser* disponibiliza para o analisador semântico os padrões encontrados na sentença e suas respectivas classificações de acordo com a LNC, conforme os exemplos da Seção 1.4. Então, o analisador semântico verifica se a sentença está correta da seguinte forma, para cada padrão da sentença identificado pelo passo anterior verifica-se como ele foi classificado pela Etapa 2 - Definição da semântica do domínio, se essa classificação for diferente da classificação informada pelo *parser*, a mensagem de erro semântico é gerada. A Figura 28 mostra o exemplo para a sentença

"O sistema deve permitir a uma certidão anexar uma certidão retificadora", neste caso, o objeto do verbo "permitir" deve ser um *ator*, porém, o que aparece é o termo "certidão" que foi classificado como requisito de armazenamento.

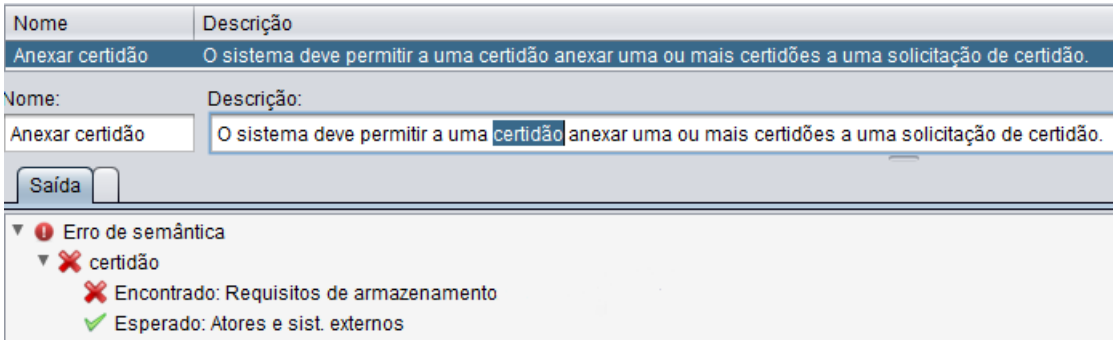

Figura 28 – Erro de semântica

Na seção *Descrições de casos de uso* o objetivo do suporte à escrita dos requisitos é induzir o usuário a escrever os elementos de projetos como foram classificados. O Suporte ajuda ao analista de requisitos escrever um documento com um alto grau de completude interna verificando quais termos escritos nos casos de uso não foram classificados nas seções do documento. Neste caso o *parser* espera que os elementos de projetos já obedeçam os padrões da Tabela 3. Por exemplo, se em uma sentença o *parser* encontrar um ator definido como "Usuário externo", ele não irá verificar se o termo está em conformidade com o padrão I da Tabela 3 e sim se existe um ator com esse nome na seção *Atores e sistemas externos*.

O foco da análise dos erros sintáticos é a estrutura das sentenças e não a classe gramatical do termos que a compõe. A fase de detecção dos erros sintáticos verifica se existem os principais elementos de cada tipo de sentença aceita pela linguagem *S-Lucas*. Por exemplo: ATOR, VERBO_DE_ACAO, EXR_RESERVADA e elemento_de_projeto para as sentenças de ação. A mesma ideia é utilizada para os demais tipos de sentenças. A Figura 29 mostra exemplos de erros sintáticos descritos para uma sentença de ação. Outros erros sintáticos são: *(i)* a falta de numeração das sentenças; e *(ii)* a exigência de existir apenas um verbo de ação na sentença, desta forma cada sentença torna-se uma unidade de teste para validação (COCKBURN, 2000). Todos os erros sintáticos para *S-Lucas* são apresentados no Apêndice B.

A detecção dos erros semânticos é semelhante à utilizada na seção *Funções do produto*. Os elementos de projetos são identificados e classificados conforme sua disposição na sentença sintaticamente correta. A partir daí, para cada termo classificado uma consulta à seção correspondente ao elemento de projeto é realizada. Caso o elemento não exista, uma mensagem de erro semântico deve ser apresentada. A fim de garantir que se possa gerar protótipos de interfaces de usuário corretas, o *parser* só aceita o uso de verbos que tenham como complemento elementos de interface (campos de edição, *labels* e botões) se uma

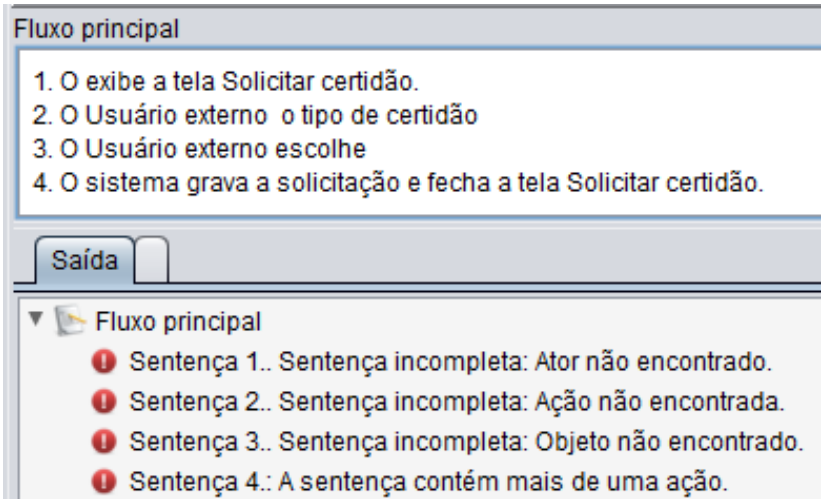

Figura 29 – *Descrições de casos de uso* - Erro de sintaxe

interface de usuário tiver sido acessada em linhas anteriores do caso de uso. Outros erros seguem as recomendações de (COCKBURN, 2000) que ajudam a garantir a clareza do caso de uso, são elas: *(i)* uso da sentença condicional "SE" apenas no fluxo alternativo/exceção; *(ii)* o uso da expressão "validar que" no lugar de "verificar se", desta forma evita-se o uso da sentença "SE" no fluxo principal e torna-a um ponto de extensão que deve ser detalhado no fluxo alternativo/exceção. A Figura 30 mostra um exemplo de alguns erros semânticos. Todos os erros semânticos para *S-Lucas* podem ser vistos no Apêndice C.

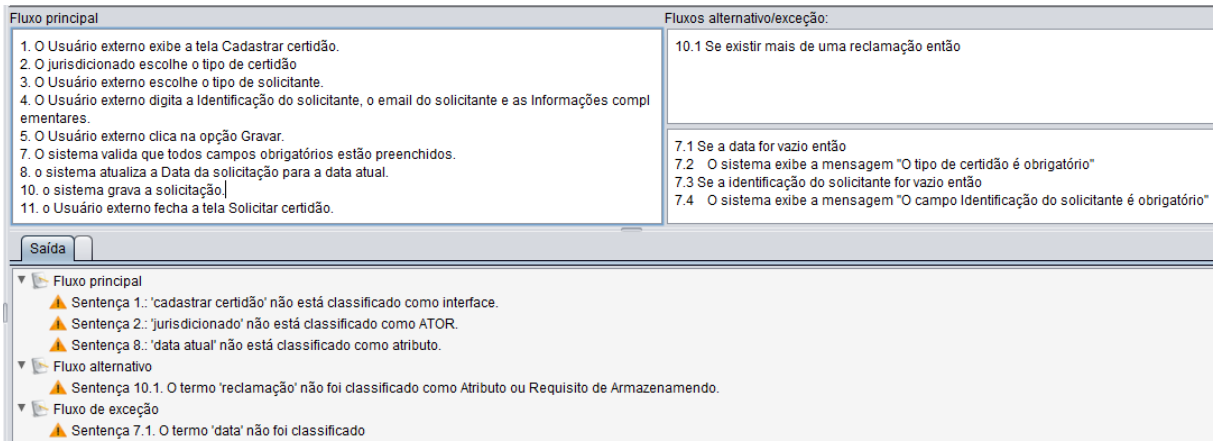

Figura 30 – *Descrições de casos de uso* - Erro de semântica

4.6.4.2 Verificação de termos classificados e escritos

A verificação de termos classificados e escritos que dá nome a esta subseção ajuda ao analista escrever documentos com alto grau de completude interna. Para isso, o protótipo verifica se todos os termos classificados nas seções do documentos são escritos com a mesma semântica nas descrições dos casos de uso. Exemplo, se o termo "chave de validação" foi classificados como *Dado armazenado*, ele dever ser escrito como tal em todos os casos de uso que aparecer.

Inicialmente, são selecionados todos os elementos de projetos das seções do documento, depois, para cada elemento selecionado é feita uma busca nas árvores explorar (Ver Figura 34) das descrições dos casos de uso para verificar se o elemento está escrito com a mesma semântica de sua classificação. Caso o elemento não esteja, ele é apresentado como elemento não descrito no documento. Os elementos não escritos são organizados conforme sua classificação nas seções do documento.

A Figura 31 mostra o resultado da verificação dos termos classificados e escritos nas descrições dos casos de uso. Neste exemplo, o termo "certidão negativa" foi classificado como uma definição do domínio mas não foi descrito em nenhum caso de uso, o mesmo acontece com os termos das seções *Atores e sistemas externos*, *Funções do produto* e *dados armazenados* que são mostrados. Por outro lado, todos os termos das seções *Interfaces de usuário*, *Requisitos de armazenamento* e *Casos de uso* foram descritos corretamente.

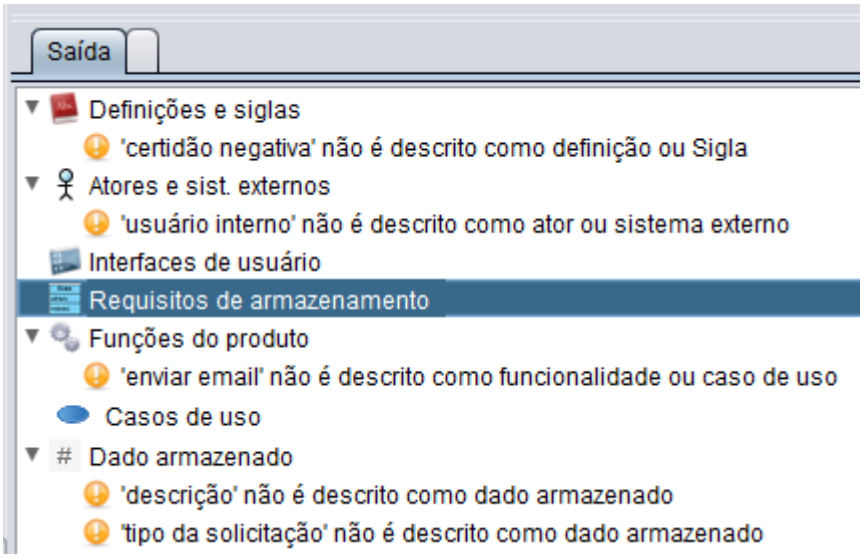

Figura 31 – Verificação de termos classificados e escritos

4.6.4.3 Avisos

Como explicado na Seção 4.5 os avisos ajudam ao analista de requisitos a verificar se todos os elementos significantes estão no léxico da aplicação e classificados, além de verificar se a sentença obedece à LNC. Os avisos são gerados mesmo que a sentença não esteja sintaticamente correta. A implementação e apresentação dos avisos são semelhantes para as seções *Funções do produto* e *Descrições de casos de uso* e ocorrem da seguinte forma. Após os processos de *tokenização* e etiquetagem da sentença o *parser* identifica todos os termos em conformidade com padrões da Tabela 3, para cada termo, é realizada uma consulta nas seções do documento de requisitos. Os termos não encontrados nessa consulta são procurados no léxico da aplicação, se o termo não for encontrado em nenhuma das consultas, ele é classificado como "Termo não encontrado no léxico". Caso contrário, se o termo for encontrado apenas no léxico ele é classificado como "Termo não classificado".

Os sinônimos do domínio encontrados na sentença também serão apresentados na tela de avisos. A Figura 32 mostra o resultado da geração de avisos para a sentença "A secretária deve enviar um email para o usuário externo quando a certidão estiver finalizada". Neste exemplo, o termo "secretária" faz parte do léxico na aplicação mas não foi classificado na etapa 2 da metodologia. Os termos "finalizada" e "email" não foram classificados nas seções do documento e nem fazem parte do léxico da aplicação. Os termos "usuário externo" e "usuário" são sinônimos e por isso também são apresentados.

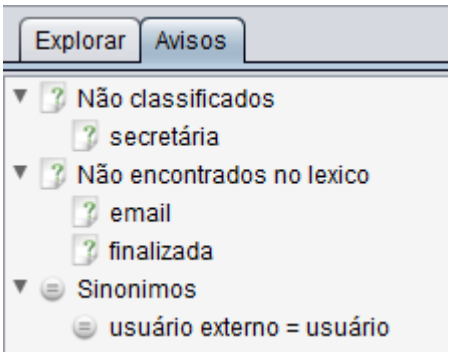

Figura 32 – Exemplo de avisos

4.6.4.4 *Recurso explorar*

Outro recurso oferecido pelo protótipo é função *explorar*, que consiste na visualização da classificação de cada elemento de projeto encontrado na sentença. Sua implementação traduz-se na identificação dos termos em conformidade com algum dos padrões da Tabela 3 e apresentá-los de acordo com sua classificação nas seções do documento de requisitos. Após os processos de *tokenização* e etiquetagem da sentença o *parser* identifica todos os termos em conformidade com padrões da Tabela 3. Para cada termo, é feita uma consulta nas seções do documento de requisitos. A partir do resultado da consulta é montada uma estrutura conforme a classificação dos termos. A Figura 33 mostra um exemplo para seção *Funções do produto* e utiliza a sentença "O sistema deve permitir ao usuário externo solicitar uma certidão negativa". No exemplo, o termo "usuário externo" foi classificado como elemento da seção *Atores e sistemas externos*, o termo "certidão" como elemento da seção *Requisitos de armazenamento* e o termo "solicitar certidão" como elemento da seção *Funções do produto*. Os termos "Sistema", "deverá" e "permitir" não foram apresentados pois fazem parte do léxico da LNC. O recurso está disponível para todas as sentenças escritas pelo analista de requisitos.

Na seção *Descrições de casos de uso* o objetivo deste trabalho é identificar os elementos de projetos referentes ao domínio do sistema. O *parser* analisa cada sentença do caso de uso na ordem escrita e identifica e classifica os elementos de projetos. Antes de exibir os termos classificados o *parser* os agrupa da seguinte forma: *(i)* os elementos de interfaces nas interfaces de usuários utilizadas no caso de uso; e *(ii)* os dados armazenados

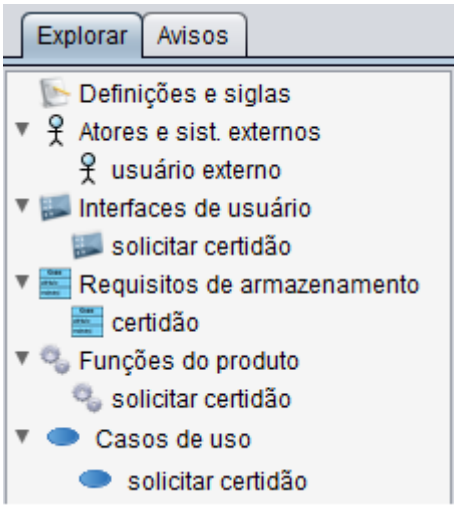

Figura 33 – Recurso explorar - Seção *Funções do produto*

nos seus requisitos de armazenamentos. As interfaces, elementos de interfaces, requisitos de armazenamento e dados armazenados são reconhecidos pela análise dos verbos e seus complementos. Por exemplo, na sentença "O usuário clica na opção Validar" o verbo "clicar", segundo a Tabela 4, espera uma expressão reservada e um elemento de interface. Assim, o termo "Validar" é classificado como um elementos de interface. Após o agrupamento, os elementos são exibidos. A Figura 34 mostra o recurso para o caso de uso "Validar certidão", mostrado na Figura 13.

Note que as figuras que representam os elementos de interfaces são escolhidos de acordo com o verbo utilizado. No exemplo, a figura apresentada ao lado do elemento "número da certidão" refere-se à sentença "2.", que tem o verbo "preencher", que dá ideia de inserção de um dado no sistema, portanto, utilizou-se uma figura que lembra um campo de edição, por outro lado, o elemento "data da certidão" da sentença "6." tem o verbo "exibir" seguido da expressão "campo" que remete à exibição de uma informação. Outro exemplo é o elemento "validar", sentença "3.", cujo verbo "clicar" seguido pela expressão "opção" remete ao uso de um botão.

4.7 Resultados preliminares

Como analise de viabilidade, efetuou-se uma avaliação preliminar da metodologia. Utilizou-se a metodologia para identificação dos elementos de projeto do Sistema de Emissão de Certidões - SEC[8], desenvolvido no Tribunal de Contas do Estado do Piaui - TCE-PI. Esse sistema permite que os usuários externos do TCE-PI solicitem ao órgão certidões negativas relacionadas às dívidas, às prestações de conta e às situações processuais. O papel de analista de requisitos foi desempenhado por um Auditor de Controle Externo mestre em Ciência da Computação, com mais de 10 anos de experiência em desenvolvimento de

[8] Para acessar: http://srvapp2.tce.pi.gov.br:8080/EmissaoDeCertidoes/

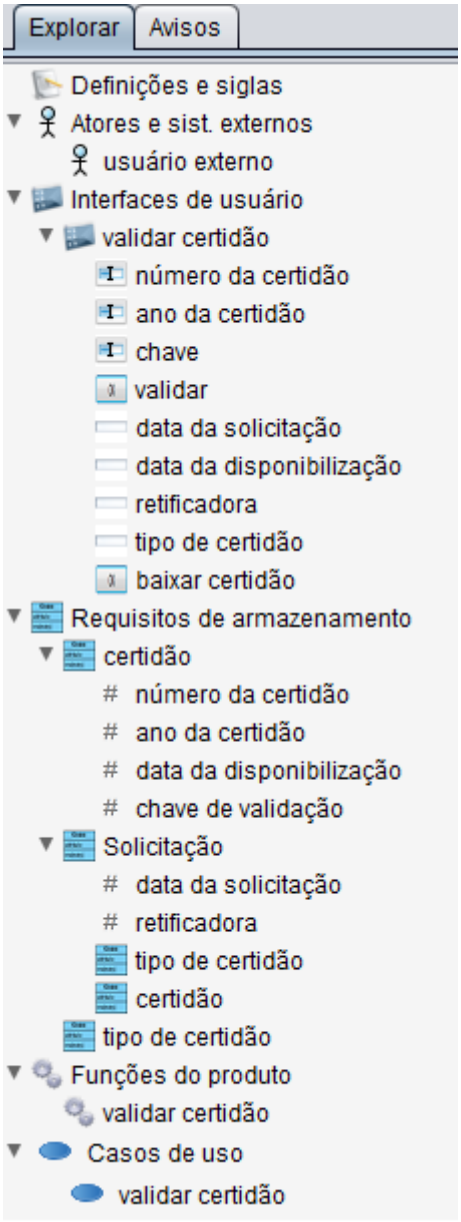

Figura 34 – Recurso explorar - Seção *Descrição de casos de uso*

sistemas e o de analista do domínio por um funcionário responsável pela confecção das certidões negativas, que até então eram elaboradas de forma manual. O resultado desta avaliação foi publicado no artigo "*A methodology to guide writing Software Requirements Specification document*", em (SOARES; MOURA, 2015).

Na execução da Etapa 1, utilizou-se apenas um documento de referência escrito pelo especialista do domínio. Esse documento contém 591 palavras e descreve algumas funcionalidades do sistema. O resultado dessa etapa aplicada ao arquivo de referência do Sistema de Emissão de Certidões é mostrado na Figura 18, apresentada anteriormente. O protótipo selecionou 88 termos relevantes.

Após isso, executou-se a Etapa 2 de duas maneiras distintas, a saber: **Método 1 (M1)**, de forma manual em que o analista de requisitos efetuou a leitura no documento de

referência e, com auxílio do especialista do domínio, identificou os elementos; e **Método 2 (M2)**, de forma automática com o auxílio do ERS-EDITOR, onde o analista de requisitos e o especialista do domínio classificaram os termos do léxico como elementos de projeto, conforme mostrado também na Figura 18. Orientou-se, em M2, que cada termo deveria ser classificado apenas como um elemento de projeto.

Dos 88 termos resultantes da Etapa 1, 29 foram identificados como falso-positivos com a utilização de M2. Durante a execução da Etapa 2, verificou-se a necessidade de desambiguação dos termos por meio de classificação em sinônimos para os casos onde o uso do radical não é suficiente, por exemplo, os termos "chefe", "gestor", "usuário chefe" e "gestor do sistema" são sinônimos no domínio da aplicação. Após a desambiguação restaram 58 termos, sendo que 23 deles continuaram como falso-positivos. A Tabela 6 mostra o resultado da classificação utilizando-se M1, M2, os elementos encontrados somente em M1 (**S1**), os elementos encontrados somente em M2 (**S2**), e a coluna total aceito que indica o número de elementos aceitos pelo analista de requisitos após a análise dos resultados. Por exemplo, a primeira linha da tabela indica que com o uso de M1 foram identificados 8 casos de uso; utilizando M2 foram identificados 10 casos de uso; dos 8 casos de uso identificados por M1, todos também foram identificados por M2, isto é, S1 igual a zero; dos 10 casos de uso identificados através de M2, 2 não foram identificados por M1, isto é, S2 igual a 2; por fim, após a avaliação dos métodos constatou-se que os 10 casos de uso identificados por M2 deveriam realmente ser implementados, ou seja, total aceito igual a 10.

Tabela 6 – Resultados

Elemento de projeto	**M1**	**M2**	**S1**	**S2**	**Total aceito**
Casos de uso	8	10	0	2	10
Atores	3	3	0	0	3
Classes	2	2	0	0	2
Métodos	11	12	1	2	12
Atributos	11	8	3	0	11
Interfaces de usuário	4	1	3	0	4

É importante destacar que o uso da metodologia ajudou o analista de requisitos a obter uma visão mais detalhada da semântica do domínio. Segundo sua análise, os dois casos de uso e métodos identificados apenas com M2 passaram despercebidos com a utilização do M1. Outro fato importante é que alguns elementos identificados apenas no M1 devem-se a problemas pontuais que serão objetos de estudo para melhoria da metodologia, são eles: *(i)* restrições no algoritmo de seleção de termos para o léxico, por exemplo, os atributos "cpf do usuário", "data de disponibilização" e "cnpj do usuário" não foram selecionado pois existe a restrição de que apenas palavras com mais de 4 caracteres

devem fazer parte do léxico, exceto para as funcionalidades; *(ii)* classificação errada do etiquetador, o único método não identificado por M2 foi o termo "emitir", pois não foi classificado como verbo pelo *Tree Tagger* e por isso não foi selecionado pelo algoritmo de construção do léxico da aplicação; e *(iii)*a recomendação de que o analista de requisitos deveria classificar cada termo como apenas um elemento de projeto em M2 foi o motivo da falta de identificação das interfaces de usuário "solicitar certidão", "consultar certidão" e "validar certidão", visto que estes termos foram classificados como casos de uso e interfaces de usuários quando da utilização do M1.

4.8 Considerações finais

Neste capítulo foi apresentada a HELP$_4$ERS. Inicialmente, mostrou-se a estrutura do documento de requisitos proposta e os motivos pelos quais as seções foram escolhidas. Também foram apresentadas as Linguagens Naturais controladas para que o analista de requisitos escreva de forma que o protótipo ERS-EDITOR reconheça os elementos de projetos contidos nas sentenças. Além disso, foram explicadas as cinco Etapas na metodologia, suas atividades e como o protótipo ERS-EDITOR foi implementado para dar suporte às atividades da metodologia. Por fim, foi apresentado um pequeno experimento que avaliou as Etapas 1 e 2 da metodologia, esse experimento mostrou que sua utilização ajuda o analista de requisitos a reconhecer elementos de projetos.

No próximo capítulo será apresentado um estudo experimental realizado com o intuito de avaliar a metodologia. Como destacado, a metodologia pode melhorar os atributos de qualidade de um documento de requisitos, porém, com um esforço maior quando comparado à metodologia manual.

5 Experimentos e resultados

A experimentação é o modo sistemático, disciplinado, computável e controlado para a avaliação da atividade humana. Por meio dela é possível verificar e explorar fatores críticos de novas teorias para que se possa formula-las e corrigi-las (BASILI; SHULL; LANUBILE, 1999). Na literatura são conhecidos vários tipos de estratégias para se realizar um estudo experimental. Wohlin et al. (2000) descreve as seguintes estratégias: *survey* ou pesquisa de opinião, estudo de caso e experimento. A aplicação de cada uma das estratégias depende do propósito da avaliação e das condições do estudo. As principais diferenças entre estas estratégias estão no nível de controle, no custo para realizá-las (tempo e esforços) e na facilidade para refazê-las.

Este capítulo tem como objetivo apresentar o estudo experimental para avaliar a metodologia, em um ambiente controlado, levando se em consideração tanto a qualidade do resultado quanto o custo para sua obtenção. Utilizou-se o processo de experimentação e a abordagem *Goal/Question/Metric* - GQM (BASILI; CALDIERA; ROMBACH, 1994), que obedece às exigências da Engenharia de Software Experimental (WOHLIN et al., 2000).

Os resultados apresentados são promissores, pois mostram que a abordagem HELP$_4$ERS melhora os atributos de qualidade: consistência, completude e não ambiguidade, além de se mostrar como uma ferramenta de aprendizado nas atividades que compõe a metodologia. Porém, sua utilização elevou o esforço temporal em 30% em comparação ao método manual. As seções seguintes apresentam a descrição do experimento de acordo com o modelo GQM.

5.1 Estudo Experimental

Nesta secção é descrito um estudo experimental que visa avaliar o uso da abordagem HELP$_4$ERS para produção de documento de requisitos com os atributos de qualidade perseguidos por esta Dissertação.

5.1.1 Definição dos objetivos

Como descritos anteriormente, a linguagem natural é utilizada para descrever requisitos de sistemas por ser de uso comum entre os envolvidos no desenvolvimento e de ser expressiva o suficiente para descrever o problema do domínio. Apesar disso, a linguagem natural possui problemas de ambiguidade, incompletude e inconsistência, que geram uma distância entre a especificação e a modelagem do sistema, uma vez que permite múltiplas interpretações e dão origem a modelos que não contemplam as características

que o *software* deve possuir. Para reduzir os problemas do uso da linguagem natural, esta Dissertação propõe o uso de uma metodologia que utiliza técnicas de Processamento de Linguagem Natural e o uso de Linguagem Natural Controlada. Para dar suporte à metodologia, o protótipo ERS-EDITOR foi desenvolvido em ambiente STANDALONE. No entanto, uma versão para Web deve ser desenvolvida como trabalho futuro.

O principal objetivo deste experimento é analisar se a metodologia conduz o analista de requisitos a escrever documentos de ERS com melhores atributos de qualidade (completeza, consistência e não ambiguidade) considerando-se o esforço quando comparadas à forma manual. Utilizou-se como base um documento que descreve os requisitos de um sistema desenvolvido pela industria e já em produção. Além disso, avaliou-se a capacidade das Linguagens Natural Controlada em descrever as funcionalidades de sistemas.

O experimento foi conduzido dentro de um ambiente controlado (*in vitro*), no contexto da disciplina de Engenharia de Software II para alunos de graduação em Ciência da Computação da Universidade Federal do Piauí (UFPI) e dos alunos do ultimo período do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal do Piauí (IFPI). A subseção 5.1.8 descreve o foco do experimento na qualidade do documento e no esforço de sua geração.

Como explicado, selecionou-se como participantes deste estudo os alunos de disciplinas específicas dos cursos Ciência da Computação e Análise e Desenvolvimento de Sistemas, o que ocasionou a ausência de um importante ingrediente em um estudo totalmente controlado: falta de aleatoriedade. Portanto, este estudo é classificado como um quasi-experimento.

5.1.1.1 Resumo do objetivo

Analisar a qualidade do documento de ERS gerado com o uso da metodologia **com o objetivo de** avaliar, **com respeito aos** atributos de qualidade não ambiguidade, completude e consistência do documento de ERS gerado e esforço de geração do documento de requisitos, além da capacidade das LNC's propostas em descrever os requisitos do sistema **do ponto de vista** do analista de requisitos, **no contexto de** alunos do curso de Ciência da Computação da UFPI e Análise e Desenvolvimento de Sistemas do IFPI, que já cursaram a disciplina de Análise Orientada a Objetos e já cursaram, ou estejam cursando, a disciplina de Engenharia de Software II.

5.1.2 Definição das hipóteses

- **Hipótese nula $(\mathbf{H0})_{\mathbf{qualidade}}$**: Não há diferença nos atributos de qualidade do documento de ERS gerado com e sem o uso da metodologia proposta.

  $H0_{\text{qualidade}}$: $Qualidade(com\ metodologia) = Qualidade(manual)$.

- **Hipótese alternativa (H1)$_{\text{qualidade}}$:** Os atributos de qualidade dos documento de ERS gerado com o uso da metodologia são diferentes daqueles gerados sem o uso da metodologia.

  H1$_{\text{qualidade}}$: $Qualidade(com\ metodologia) \neq Qualidade(manual)$.

- **Hipótese nula (H0)$_{\text{LNC}}$**: Os analistas de requisitos conseguem descrever 90% dos requisitos do sistema utilizando as LNC's propostas.

  H0$_{\text{LNC}}$: $PadraoLNC(requisitos) = 90\%$.

- **Hipótese alternativa (H1)$_{\text{LNC}}$**: Os analistas de requisitos conseguem descrever um valor diferente de 90% dos requisitos do sistema utilizando as LNC's propostas. Espera-se que esse valor seja maior que 90%.

  H1$_{\text{NLC}}$: $PadraoLNC(requisitos) \neq 90\%$.

- **Hipótese nula (H0)$_{\text{esforço}}$**: Não há diferença no esforço empregado para gerar o documento de requisitos com e sem o uso da metodologia proposta.

  H0$_{\text{esforço}}$: $Esforço(com\ metodologia) = Esforço(manual)$.

- **Hipótese alternativa (H1)$_{\text{esforço}}$**: Exite diferença no esforço gerar o documento de requisitos com e sem o uso da metodologia proposta.

  H1$_{\text{esforço}}$: $Esforço(com\ metodologia) \neq Esforço(manual)$.

Para verificar as hipóteses, algumas questões devem ser respondidas. A formulação das questões devem levar em consideração alguns pontos importantes da pesquisa, tais como:

1. Atributos de qualidade dos documentos de requisitos gerados pelo experimento, devem responder o quanto o documento possui:

   a Completude externa: Se todos os elementos de projetos do documento base são encontrados nos documentos de requisitos gerados pelos participantes do experimento e escritos com a mesma semântica.

   b Completude interna: Se todos os elementos de projetos registrados nas seções do documento são citados, com a semântica correta, na seção *Descrição de casos de usos* e vice versa.

   c Consistente: Refere-se à consistência interna, ou seja, se o elemento de projeto é sempre escrito da mesma forma (ou com o uso de sinônimo) em todas as sentenças em conformidade das LNC's proposta.

   d Não ambiguidade: Se o elemento de projeto está descrito com a mesma semântica da seção que ele está registrado.

2. O quanto as LNC's conseguem descrever os requisitos do sistema.

3. Qual o esforço gasto para gerar os documentos de requisitos com e sem a utilização da metodologia. Caso haja diferença, investigar quais motivos por meio da aplicação de um questionário pós-tarefa.

5.1.3 Questões da experimentação

As questões do experimento ajudam na análise e compreensão de seu resultado. Assim formulou-se as seguintes questões e métricas a verificação das hipóteses do experimento. As questões são:

Q1: O documento gerado está completo externamente(1a)?

Métrica (M1): O percentual de elementos de projeto encontrados nos documentos gerados pelo experimento que estão no documento base escritos com a mesma semântica.

Q2: O documento gerado está completo internamente(1b)?

Métrica (M2): O percentual de elementos de projeto que estão classificados e escritos nas sentenças em conformidade com *S-Lucas* na seção *Descrição de casos de uso* e vice versa.

Q3: O documento gerado está consistente internamente(1c)?

Métrica (M3): O percentual de elementos de projeto que são sempre escritos da mesma forma, ou com o uso de sinônimos previamente informados, nas sentenças em conformidade com as LNC's.

Q4: O documento gerado está ambíguo(1d)?

Métrica (M4): O percentual de sentenças não ambíguas escritas em conformidade com as LNC's. Esta métrica foi aplicada apenas nos documentos gerados pelo grupo COM.

Q5: Quanto as Linguagens Naturais Controladas propostas conseguem descrever os requisitos do sistema(2)?

Métrica (M5): Percentual de sentenças em conformidade com as LNC's e que descrevem corretamente o requisito do sistema.

Q6: O esforço utilizado na geração dos documentos de requisitos é diferente com e sem o uso da metodologia ?(3)

Métrica (M6): Tempo total gasto na geração dos documentos de requisitos.

Desta forma, as hipóteses poderão ser validadas relacionando-as com as métricas estabelecidas, da seguinte forma: $H_{qualidade}$ será validada por Q1, Q2, Q3 e Q4; H_{LNC}, por Q5; e $H_{esforço}$, por Q6.

5.1.3.1 Seleção de variáveis

Para responder às questões de pesquisa e validar as hipóteses, as seguintes variáveis foram definidas:

- Variáveis independentes, aquelas que podem ser controladas e que afetam as os valores das variáveis dependentes: metodologia proposta; conhecimento dos especialistas do domínio sobre o sistema utilizado no experimento; e conhecimento dos participantes do experimento sobre a metodologia proposta.

- Variáveis dependentes são medidas para verificar as variações encontradas no estudo, representam o efeito causado pelas variáveis independentes. As variáveis dependentes em nosso estudo e as questões que elas respondem são apresentadas a seguir: completude externa (COMP-EX) –> Q1; completude interna (COMP-IN) –> Q2; consistência (CONS) –> Q3; quanto o documento é não ambíguo (NA) –> Q4; capacidade da LNC descrever requisitos (LNC-FP) –> Q5, para a S-LSF e (LNC-UC) –> Q5, para *S-Lucas*; e o tempo total gasto para construção dos documentos de requisitos (TMP) –> Q6.

5.1.4 Seleção dos participantes

Os trabalhos apresentados em (HOST; REGNELL; WOHLIN, 2000; RUNESON, 2003; SVAHNBERG; AURUM; WOHLIN, 2008) indicam que grupos de estudantes e profissionais obtêm resultados semelhantes quando uma nova metodologia de Engenharia de Software é avaliada. Por esse motivo e por dificuldade de encontrar profissionais dispostos a participar do estudo, o experimento foi realizado utilizando-se estudantes dos cursos de Ciência da Computação (CC) da Universidade Federal do Piauí que estavam cursando a disciplina de Engenharia de Software II, a disciplina é oferecida para os alunos do sexto período do curso. Participaram também alunos do curso de Análise e Desenvolvimento de Sistemas (ADS) do Instituto Federal do Piauí que estavam cursando a disciplina de Tópicos em Sistemas de Informação, que é oferecida para os alunos também no sexto período. Destaca-se que no sexto período de ambos os cursos, os alunos possuem bons conhecimentos acadêmicos em análise de requisitos, análise orientada a objetos e descrição de casos de uso.

A participação dos alunos ocorreu de forma voluntária, assim, nem todos os alunos matriculados cooperaram com o experimento. O estudo iniciou com 21 alunos, sendo 11 de CC e 10 de ADS, como veremos na próxima seção, nem todos os alunos concluíram o estudo.

Devido à falta de material humano disponível para conduzir o experimento, atuar como analista de requisitos e instrutor da metodologia HELP$_4$ERS, apenas o autor deste

estudo representou esses papeis. Para minimizar o efeito da expectativa do experimentador descrito por Lima et al. (2011), o autor tentou interagir com os participantes somente quando solicitado e sem expor suas crenças e vontades para que o experimento desse certo. Apesar disso, o autor defende que em próximos experimentos outros atores atuem como analista de domínio, instrutor da metodologia e condutor do experimento.

5.1.5 Instrumentação

Os instrumentos usados neste experimento incluem o **documento de requisitos do Sistema Sagres-Web**; o **documento de referência do sistema Sagres-Web**; o **material de treinamento**; um **template do documento de requisitos**; e o **questionário pós-tarefa** que avalia a metodologia e o protótipo.

O documento de requisitos do Sistema Sagres-Web, chamado de **documento base**, foi escrito por sua equipe de desenvolvimento e contém os requisitos implementados no sistema. Devido ao processo interativo de desenvolvimento do Sagres-Web o documento poderia sofrer frequentes alterações, por esse motivo, utilizou-se a Versão 11 de 12.11.2015 e apenas os elementos de projetos contidos na versão foram levados em consideração para a análise dos resultados. O sistema está sendo desenvolvido no Tribunal de Contas do Estado do Piauí - TCE-PI em parceria com a Universidade Federal do Piauí por uma equipe formada por: analista de requisitos (um), especialista do domínio (um) e desenvolvedores (três). O principal objetivo do sistema é gerenciar o recebimento de prestações de contas contábeis e de folha de pagamento de forma eletrônica, além de disponibilizar chaves de acesso aos usuários para ações específicas no sistema. O sistema está em produção desde janeiro de 2016.

O documento de referência do sistema Sagres-Web foi escrito pelo analista de domínio em parceria com o analista de requisitos. Ele contém as informações do que deve ser implementado, segundo a visão dos analistas de domínio e de requisitos.

O material de treinamento da metodologia proposta é composto de dois conjuntos de slides, um para a explicação da dinâmica do experimento e outro sobre a as Etapas da metodologia HELP$_4$ERS e exemplos requisitos funcionais escritos em S-LFS e casos de uso descritos em *S-Lucas*.

O template do documento de requisitos foi entregue a cada participante do grupo que não utilizou a metodologia. O template tem as mesmas informações do documento de requisitos gerados pelo ERS-EDITOR e indicações do que deve ser escrito em cada seção.

Finalmente, o questionário pós-tarefa que tem o objetivo de avaliar a metodologia e o protótipo ERS-EDITOR. O questionário é composto de 24 questões distribuídas da seguinte forma: 4 questões para aviar o protótipo ERS-EDITOR; 7 questões para a avaliação da metodologia; 11 questões para avaliar a importância dos recursos disponibilizados pelo

protótipo; e 2 questões abertas para que os participantes elencassem os pontos fortes e fracos da metodologia/protótipo. Os questionários serão discutidos na subseção 5.1.8.5.

5.1.6 Projeto do estudo

O estudo foi realizado aplicando-se o mesmo procedimento, porém, em momentos diferentes, para os alunos das duas Instituições (UFPI e IFPI). Portanto, a partir de agora trataremos os alunos sem levar em consideração o curso uma vez que o objetivo do experimento é avaliar a metodologia. Outro fator importante foi a disponibilidade de tempo, uma vez que utilizou-se a carga horária das disciplinas, os professores disponibilizaram até 10h, por grupo, para a realização do experimento.

Os alunos foram divididos, por ordem alfabética, em dois grupos. O Grupo 01 não utilizou a metodologia (**SEM**). O Grupo 02 utilizou a metodologia/protótipo, durante o experimento (**COM**). Ambos os grupos iniciaram com 10 participantes. Não houve inversão de tarefas pelos grupos devido ao tempo limitado para a realização do experimento. As atividades do experimento são apresentadas na Tabela 7.

Tabela 7 – Atividades do experimento

Grupo	**Atividade**
	1.1 Leitura do doc. de referência e identificação dos elementos de projeto 1.2 Escrita do doc. requisito de forma manual 1.3 Responder questionário pós-tarefa
	2.1 Treinamento sobre a metodologia/ERS-EDITOR (2h) 2.2 Leitura do doc. de referência e identificação dos elementos de projeto 2.3 Escrita do doc. de requisito com a metodologia/ERS-EDITOR 2.4 Responder questionário pós-tarefa

A primeira linha da tabela descreve as atividades do grupo que gerou o documento de requisitos SEM a metodologia/ERS-EDITOR, os participantes deveriam ler o documento de referência e identificar os elementos de projetos e escrever o documento utilizando o template entregue. Após o término da escrita do documento os participantes deveriam preencher o questionário pós-tarefa.

As atividades do grupo que utilizou a metodologia/ERS-EDITOR está descrito na segunda linha da tabela. Inicialmente, os participantes desse grupo receberam um treinamento de duas horas sobre a metodologia, o protótipo ERS-EDITOR e as LNS' propostas. O treinamento é importante no sentido de dar condições aos participantes de desempenhar as etapas da metodologia e utilizar o ERS-EDITOR. O experimento iniciou, efetivamente, para o grupo COM, no início da atividade 2.2, que é semelhante à atividade 1.1 do grupo SEM. Na atividade 2.3 os participantes deveriam escrever os documento de requisitos seguindo as etapas da metodologia e, ao final, executar a atividade 2.4, sobre o

questionário pós-tarefa. É importante lembrar que a qualquer momento os participantes poderiam tirar dúvidas com o especialista de domínio, que ficou disponível durante todo o experimento. Em virtude da natureza iterativa da metodologia, o tempo de realização de cada atividade não foi controlada. A execução de cada atividade será detalhada na próxima subseção.

5.1.7 Operação

Antes de iniciar o estudo foi feita uma apresentação de como o experimento seria conduzido e as atividades que seriam executadas, sem explicar quais eram exatamente as hipóteses envolvidas.

O experimento com o grupo SEM foi realizado em três encontros de 2h e foi iniciado com uma explicação sobre os requisitos de qualidade e a estrutura do template do documento de requisitos. O tempo para essa explanação foi de 1h e não foi considerado como experimento. A partir de então, os participantes executaram as atividades 1.1 e 1.2 de forma iterativa. Durante todo o experimento os participantes tiraram dúvidas, sobre conceitos e execução do sistema Sagres-web, com o especialista do domínio, que esclareceu todas as dúvidas sem identificar os elementos de projetos existentes no sistema em produção.

Dos dez participantes, dois desistiram do experimento, um deles faltou a um encontro e não retornou por achar que não iria conseguir terminar o experimento e o outro alegou falta de motivação para continuar. O participantes terminaram o experimento entre 5 e 6h de estudo e geraram oito documentos de requisitos. Considerou-se o tempo de 6h para os cálculos do experimento.

O experimento com o grupo COM a metodologia/ERS-EDITOR foi realizado em cinco encontros de 2h. Inicialmente foi dado um treinamento sobre: a metodologia e suas fases, a utilização do ERS-EDITOR e explanações sobre S-LSF e *S-Lucas* com ilustrações de exemplos. O treinamento durou 2h e seu tempo não foi considerado como experimento. A partir de então os participantes executaram, de forma iterativa, as atividades 2.2 e 2.3 até concluírem o documento de requisitos. Mais uma vez, o analista de domínio esteve à disposição para dirimir dúvidas. Nesta etapa, as dúvidas sobre o domínio do sistema foram mais frequentes. Ocorreram muitas indagações para esclarecimento do uso do protótipo. Para que os participantes respondessem ao questionário pós-tarefa de forma mais objetiva foi solicitado que eles registrassem os pontos fortes e as dificuldades do uso da metodologia/ERS-EDITOR.

Dos dez participantes deste grupo três desistiram do experimento, um alegou problemas de saúde e os outros dois alegaram que, por prioridade, iriam aproveitar o tempo das aulas para estudar para outras disciplinas. Os participantes concluíram o

trabalho entre 6 e 8 horas de estudo e geraram sete documentos de requisitos. Considerou-se o tempo de 8h como tempo do experimento.

Em resumo, o estudo durou 17h com 3h de treinamento, 1h para o grupo SEM e 2h para o grupo COM. No total, quinze documentos de requisitos foram gerados. O grupo SEM gerou oito documentos e o grupo COM, sete. Dos vinte participantes do experimento, cinco desistiram, sendo dois do grupo SEM e três do grupo COM. Os dados dos quinze documentos gerados foram utilizados para a análise estatísticas descrita a seguir.

5.1.8 Obtenção e interpretação dos resultados

A análise dos dados foi feita com o objetivo de responder às questões de pesquisas. Devido ao pequeno número de amostras obtidas não foram realizados os testes das hipóteses de forma estatística. Outras amostras devem ser obtidas para que se possa realizar os testes das hipótese. Apesar disso, considera-se importante esclarecer quais questões verificam as hipóteses, são elas: Qualidade(H0,H1) são verificadas por: Q1, Q2, Q3 e Q4; LNC(H0,H1), por Q5; e Esforço(H0,H1), por Q6. A seguir são apresentados os resultados obtidos por meio da avaliação da metodologia em função das variáveis dependentes.

Um resumo do resultado de cada variável dependente é mostrado de forma descritiva e analítica por meio de tabelas, utilizando-se da métrica correspondente a cada variável. As colunas e linhas com o título **M** representam as médias por elemento e por documento, respectivamente. A ultima célula de cada tabela mostra a média geral do resultado apresentado. Para melhor aproveitamento do espaço das tabelas, utilizou-se de mnemônicos para os elementos de projetos, a Tabela 8 mostra os elementos de projetos e seus mnemônicos.

Tabela 8 – Mnemônicos para elementos de projetos

ATOR	Ator
UI	Interfaces de usuário
CLAS	Classes
ATR	Atributos
FUNC	Funcionalidades
UC	Casos de uso

5.1.8.1 Completude

5.1.8.1.1 Completude externa

Como descrito anteriormente, a completude está relacionada aos elementos de projetos do documento base, chamados de **elementos corretos**. A primeira linha da

Tabela 9 mostra a quantidade de elementos de projetos contidos no documento base na data do experimento e a segunda, os elementos que foram selecionados pelo pelo ERS-EDITOR na etapa 1 da metodologia.

Tabela 9 – Elementos de projetos corretos

	ATOR	UI	CLAS	ATR	FUNC	UC
Doc. Base	2	7	6	27	5	12
Etapa 1	2	7	6	19	5	12
	100%	100%	100%	70%	100%	100%

Assim, a completude externa foi analisada sob a ótica das métricas: (M1) o percentual de elementos de projeto encontrados nos documentos gerados pelo experimento que estão no documento base escritos com a mesma semântica (COM-EX). As Tabelas 10 e 11 mostram os valores da variável COM-EX para os grupos SEM e COM, respectivamente.

Tabela 10 – Elementos corretos - SEM

	ATOR	UI	CLAS	ATR	FUNC	UC	M
01	50%	43%	33%	33%	20%	42%	26%
02	50%	71%	50%	63%	80%	67%	64%
03	50%	71%	50%	59%	60%	58%	59%
04	50%	43%	33%	30%	60%	42%	37%
05	50%	71%	33%	74%	60%	42%	61%
06	50%	43%	33%	22%	20%	25%	27%
07	50%	100%	50%	74%	80%	25%	64%
08	50%	29%	33%	59%	60%	67%	54%
M	50%	59%	40%	52%	55%	46%	50%

Tabela 11 – Elementos corretos - COM

	ATOR	UI	CLAS	ATR	FUNC	UC	M
10	100%	100%	83%	85%	60%	75%	83%
11	100%	100%	50%	67%	20%	58%	64%
12	100%	71%	50%	89%	80%	75%	80%
13	100%	86%	83%	89%	60%	50%	78%
14	100%	57%	83%	63%	40%	83%	68%
15	100%	71%	83%	78%	100%	75%	80%
16	100%	86%	100%	81%	60%	92%	85%
M	100%	82%	76%	79%	60%	73%	77%

Note que a maior média de acertos no grupo SEM foi de 64%(documentos 02 e 07). Essa média é igual à menor média de acerto no grupo COM (documento 11). A última linha das tabelas mostram as médias de elementos corretos por grupo. Veja que todas as médias do grupo COM são melhores que as médias do grupo SEM, inclusive a média geral mostrada na última coluna da última linha das tabelas. Isso deve-se ao fato de estar disponível aos participantes do grupo COM, todos os termos candidatos aos elementos de projetos selecionados na etapa 1 da metodologia. Observe que na coluna ATOR, todos os participantes do grupo SEM acertaram 50% dos atores do sistema. Isso ocorreu porque o ator que não foi identificado trata-se de um sistema e não de um humano, segundo o Professor de Engenharia de Software, no início da disciplina os alunos têm dificuldade de reconhecer atores que são sistemas externos. Esse ator não passou despercebido no grupo COM porque foi selecionado pelo ERS-EDITOR como candidato à elemento de projeto, como mostra a Figura 35.

5.1.8.1.2 Completude interna

O atributo de qualidade completude interna foi medido por meio da métrica M3: o percentual de elementos de projeto que estão nas seções do documento e que estão escritos

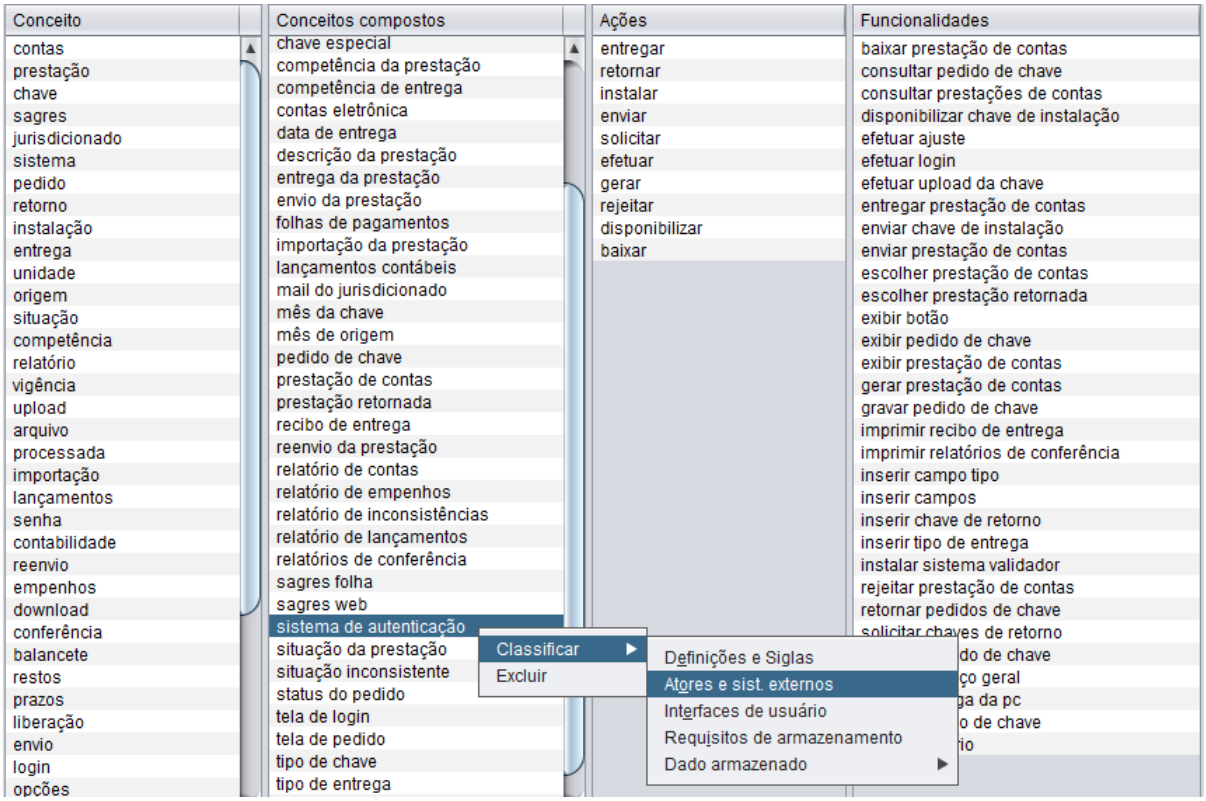

Figura 35 – Ator “sistema de autenticação”

nas sentenças em conformidade com as regras de *S-Lucas* na seção *Descrição de casos de uso* e vice versa. Essa medida corresponde ao valor da variável independente (COMP). O cálculo da variável consiste em :

1. Termos classificados e escritos: Verificar se todos os elementos de projeto classificados são descrito com a mesma semântica em algum caso de uso. Por exemplo, se o termos “sistema de autenticação” foi classificado como ator então ele deve ser descrito como ator em algum caso de uso;

2. Termos escritos e classificados: Verificar se todos os elementos descritos nos casos de uso estão definidos na seção correspondente à semântica utilizada na sentença do caso de uso. Por exemplo, na sentença “O jurisdicionado acessa a tela entregar prestação de contas” os termos “jurisdicionado” e “entregar prestação de contas” devem está cadastrados nas seções *Atores e sistemas externos* e *Interfaces de usuário*, respectivamente.

As Tabelas 12 e 13 mostram o percentual dos termos classificados e escritos para os grupos SEM e COM. A coluna C.S corresponde aos termos da seção *Conceitos e siglas*. Para calcular a consistência desses termos verificou-se sua presença em qualquer outro local do documento. Novamente todas as médias do grupo COM foram superiores às do grupo SEM.

Tabela 12 – Termos classificados e escritos SEM

	C.S	ATOR	UI	CLAS	ATR	M
01	100%	0%	20%	50%	64%	33%
02	20%	100%	57%	100%	78%	68%
03	67%	0%	71%	33%	67%	60%
04	0%	33%	80%	100%	89%	68%
05	57%	50%	33%	75%	79%	68%
06	67%	0%	0%	80%	4%	18%
07	25%	50%	0%	0%	17%	13%
08	60%	50%	40%	40%	82%	65%
M	53%	36%	41%	57%	58%	53%

Tabela 13 – Termos classificados e escritos COM

	C.S	ATOR	UI	CLAS	ATR	M
10	25%	100%	100%	50%	100%	64%
11	80%	100%	86%	100%	95%	93%
12	82%	100%	100%	80%	92%	90%
13	100%	67%	71%	100%	90%	88%
14	100%	100%	100%	100%	100%	100%
15	100%	100%	80%	100%	100%	97%
16	63%	100%	100%	100%	94%	91%
M	75%	93%	91%	91%	91%	89%

Da mesma forma, as Tabelas 14 e 15 mostram o percentual dos termos escritos e classificados. Observe que nas tabelas não existe a coluna C.S porque o foco da verificação são os elementos de projetos. As células cujo conteúdo é *** indicam que o participante não escreveu nenhum elemento de projeto do tipo específico na descrição do caso de uso.

Tabela 14 – Termos escritos e classificados SEM

	ATOR	UI	CLAS	ATR	M
01	0%	67%	60%	18%	29%
02	100%	75%	67%	46%	51%
03	***	75%	100%	48%	55%
04	100%	57%	25%	25%	32%
05	0%	67%	86%	75%	0%
06	***	***	80%	50%	38%
07	100%	***	***	100%	50%
08	50%	50%	100%	58%	51%
M	42%	63%	61%	47%	51%

Tabela 15 – Termos escritos e classificados COM

	ATOR	UI	CLAS	ATR	M
10	100%	100%	80%	96%	95%
11	100%	86%	86%	95%	92%
12	100%	100%	100%	100%	100%
13	100%	100%	100%	100%	100%
14	100%	83%	83%	78%	81%
15	100%	100%	80%	92%	91%
16	100%	100%	100%	94%	96%
M	100%	95%	90%	94%	94%

Como observado, as médias do grupo COM foram todas maiores que as do grupo SEM. Atribui-se esse fato à iteração das Etapas 2, 3 e 4 da metodologia e aos recursos oferecidos pelo ERS-EDITOR apresentados nas seções 4.6.4 e 4.6.4.2.

5.1.8.2 Consistência

O atributo de qualidade consistência foi medido pela métrica M4: o percentual de elementos de projeto que são sempre escritos da mesma forma ou com o uso de sinônimos previamente informados, nas sentenças em conformidade com as LNC's. Por exemplo se existir um ator com o nome de "jurisdicionado", todas as ocorrências desse ator nas sentenças das LNC's devem ser escritas como "jurisdicionado" ou o sinônimo que foi associado a ele na etapa 1 da metodologia. É fácil notar que o protótipo não faz, diretamente, essa verificação, pois, não há como ter certeza de que em uma sentença o ator dever ser o "jurisdicionado", qualquer termo classificado como ator deixaria a sentença

correta. Apesar disso, esse atributo foi o mais fácil de ser verificado e corrigido durante o experimento. As Tabelas 16 e 17 mostram os resultados da variável COMP para os grupos SEM e COM, respectivamente. As células cujo conteúdo é *** indicam que o participante não escreveu nenhum elemento de projeto, do tipo específico, na descrição do caso de uso.

Tabela 16 – Consistência SEM

	ATOR	UI	CLAS	ATR	M
01	0%	0%	0%	29%	15%
02	100%	50%	100%	76%	75%
03	***	0%	100%	47%	39%
04	0%	100%	50%	75%	73%
05	0%	50%	0%	79%	57%
06	***	***	50%	0%	33%
07	0%	***	***	75%	38%
08	0%	40%	50%	64%	63%
M	17%	44%	42%	64%	42%

Tabela 17 – Consistência COM

	ATOR	UI	CLAS	ATR	M
10	100%	100%	100%	100%	100%
11	100%	100%	100%	100%	100%
12	100%	100%	100%	100%	100%
13	100%	100%	100%	100%	100%
14	100%	100%	100%	100%	100%
15	100%	100%	100%	100%	100%
16	100%	100%	100%	100%	100%
M	100%	100%	100%	100%	100%

Os resultados mostram que o atributo consistência é:

1. O mais difícil de ser verificado sem a utilização da metodologia/ERS-EDITOR. Note que as médias são as menores de todos os atributos apresentados e que a ocorrência de *** é igual à do atributo completude interna; e

2. O mais fácil de ser verificado com a utilização da metodologia/ERS-EDITOR. As médias para todos os documentos foram de 100%. Dois fatos contribuíram para esse resultado, são eles : (*i*) o uso intenso do recurso *intellisense* minimizou a ocorrência deste tipo de inconsistência; e (*ii*) este tipo de inconsistência era o primeiro a ser notado pelos participantes quanto estes executavam os recursos das seções 4.6.4 e 4.6.4.2.

5.1.8.3 Não ambíguo

Conforme o (IEEE, 1998), um documento não ambíguo é aquele em que todos os conceitos expressos nele têm apenas uma interpretação. Neste trabalho buscou-se atingir esse objetivo utilizando três estratégias, são elas:

1. A estrutura do documento de requisitos, que garante a definição semântica de um termo importante no domínio do sistema;

2. Uso de um dicionário de sinônimos para reduzir a ambiguidade dos elementos de projetos;

3. As LNC's que só aceitam os termos sintática e semanticamente corretos. Mesmo que o termo signifique dois conceitos diferentes, ele só será aceito em uma sentença se ele estiver escrito com a semântica correta;

4. Uso da verificação de inconsistência, que apresenta ao usuário da metodologia todos so erros semânticos das sentenças.

Observe que para uma sentença estar em conformidade com a LNC ela deve obedecer o item 3. Por esse motivo, o atributo não ambíguo foi medido por meio da métrica M5 (NA), que mede o percentual de sentenças não ambíguas escritas em conformidade com S-LFS e *S-Lucas*.

A Tabela 18 mostra o resultado da avaliação dos 7 documentos do grupo COM.

Tabela 18 – Não ambíguo - COM

	Nº de sentenças	Ambíguas	Não ambíguas
10	78	0	100%
11	91	0	100%
12	53	0	100%
13	38	1	97%
14	74	3	96%
15	47	1	98%
16	125	0	100%
	506	5	99%

Considerou-se como ambíguas as sentenças da seção *Funções do produto* com erro semântico (ver Figura 28) e as sentenças da seção *Descrição de cados de uso* que continham os erros semânticos dos tipos de 1 a 5 (ver Apêndice C) cujo termo em destaque estava classificado como outro tipo de elemento de projeto. Por falta de uma definição objetiva de como definir que uma sentença é não ambígua para o grupo SEM, os documentos desse grupo não foram avaliados. No geral, 99% das sentenças escritas podem ser consideradas como não ambíguas.

5.1.8.4 Linguagem Natural Controlada

Um dos objetivos deste experimento é mensurar quanto S-LFS e *S-Lucas* são capazes de descrever de forma correta os requisitos de sistemas. Para isso, utilizou-se a métrica M6 - percentual de sentenças em conformidade com a LNC e que descrevem corretamente o requisito do sistema. A métrica deu origem às variáveis independentes LNC-FP, para S-LFS e LNC-UC, para *S-Lucas*.

Avaliou-se todas as sentenças das seções *Funções do produto* e *Descrição de casos de uso* dos documentos do grupo COM. Para ser considerada correta, uma sentença não

poderia ter erros sintáticos ou semânticos e deveria descrever corretamente, de forma individual, o requisito do sistema. Isso significa que nas avaliações das descrições dos casos de uso, o objetivo foi medir a capacidade de *S-Lucas* descrever corretamente a ação (ou condição) da sentença e não a interação do caso de uso.

As tabelas 19 e 20 mostram os resultados para os documentos do grupo COM. A coluna Sentenças mostra a quantidade de sentença no documento, a coluna Erradas, a quantidade de sentenças com erros sintáticos ou semânticos, a coluna Corretas, a quantidade de sentenças certas e a coluna %, os valores correspondentes às variáveis LNC-FP e LNC-UC.

Tabela 19 – LNC - FP

	Sentenças	Erradas	Corretas	%
10	5	0	5	100%
11	5	0	5	100%
12	8	0	8	100%
13	5	0	5	100%
14	3	0	3	100%
15	6	0	6	100%
16	5	0	5	100%
	37	0	37	100%

Tabela 20 – LNC - UC

	Sentenças	Erradas	Corretas	%
10	86	0	86	100%
11	45	4	41	91%
12	80	4	76	95%
13	33	0	33	100%
14	71	4	67	94%
15	41	0	41	100%
16	120	1	119	99%
	476	13	463	97%

A Tabela 19 mostra que todas as sentenças da seção *Funções do produto* estavam de acordo com os critérios estabelecidos, porém, é importante ressaltar que 35 delas foram escritas obedecendo o Padrão 1 de S-LFS e 2 delas, ao Padrão 2, fato explicado pelo uso do recurso *intellisense* que foi bastante intenso e neste caso induziu o usuário a escrever de acordo com um dos dois padrões. O resultado deixa claro que uma nova avaliação deve ser realizada com o objetivo de testar os Padrões 2 e 3.

A Tabela 20 mostra o resultado para a LNC *S-Lucas*. Observa-se que todos os documentos têm mais de 90% de sentenças corretas e, no geral, os participantes foram capazes de descrever corretamente 97% das sentenças. Atribui-se o resultado, novamente, ao uso do *intellisense* e à estrutura simples das sentenças de ação e condicional de *S-Lucas*. Porém, a sentença de repetição “Enquanto” não foi utilizada.

Para aprimorar *S-Lucas*, verificou-se os motivos das 13 sentenças conterem erros sintáticos. A tabela 21 mostra as sentenças com erros sintáticos, os tipos de erros (ver Apêndice B), a quantidade e o motivo de suas ocorrências. O erros mostram uma deficiência no Etiquetador Tree Tagger que não classifica algumas palavras como VERBO, tais como “consulta”, “pesquisa” e “anexa”, as duas primeiras são classificadas como SUBSTANTIVO e a ultima como ADJETIVO. Outro problema foi *S-Lucas* não reconhecer a sentença “O sistema inicia/encerra o caso de uso”.

Tabela 21 – Sentenças com erros de sintaxe

Sentença	Tipo	Quantidade	Motivo
O sistema consulta o ...	3.	5	O etiquetador não reconhece a palavra “consulta” como verbo.
O sistema encerra o caso de uso.	4.	6	“caso de uso” é expressão reservada da LNC.
O jurisdicionado anexa a ...	3.	2	O etiquetador não reconhece a palavra “anexa” como verbo.

Mesmo com os valores de LNC-FP e LNC-UC acima de 95%, o autor deste trabalho está consciente de que as LNC's devem ser melhoradas e reavaliadas, pois nem todas as situações do mundo real foram descritas nem as LNC's foram exploradas por completo.

5.1.8.5 Esforço

Neste experimento o esforço foi medido pelo tempo total gasto por cada grupo para gerar dos documentos. Como explicado anteriormente, por limitação de tempo disponível o estudo por grupo poderia ocorrer em até 10h, o tempo foi dividido em treinamento e experimento. O grupo SEM teve 1h de treinamento e 6h de experimento, apenas um participante entregou o documento em 5h e os demais em 6h. O grupo COM teve 2h de treinamento e utilizou as 8h restantes para a conclusão do experimento. O tempo de experimento do grupo COM foi 33% maior, o que já era esperado pelo fato do uso do protótipo e a falta de experiência dos participantes com a metodologia.

Sob a visão do pesquisador, após observação da operação do experimento, acredita-se que os seguintes fatores ajudaram no maior tempo gasto pelo grupo COM:

1. Dúvidas sobre o domínio do sistema: A Etapa 1 da metodologia gerou muitos questionamentos sobre os termos selecionados pelo protótipo;

2. Dúvida em relação ao uso do protótipo: Os participantes não estavam acostumados com a interface do protótipo, por isso ocorreram muitas dúvidas quanto a seu uso;

3. Dúvidas sobre *S-Lucas*: Inicialmente, algumas dúvidas sobre o uso de *S-Lucas* tomaram tempo dos participantes, mas rapidamente a LNC foi assimilada.

4. Erros no protótipo: alguns erros no protótipo fizeram os participantes perderem algum tempo em entender o que estava ocorrendo;

Para confrontar essa visão, aplicou-se um questionário onde os participantes deveriam avaliar quatro fatores importantes do experimento, são eles: (*i*) os pontos fortes e os pontos fracos da proposta; (*ii*) a metodologia; *(iii)* o ERS-EDITOR; e (*iv*) os recursos. A seguir, os resultados serão apresentados e discutidos.

5.1.8.5.1 Pontos fortes e pontos fracos

Os pontos fracos contribuem para o maior esforço na geração dos documentos de requisitos. A Figura 36 mostra os pontos fracos apontados pelos participantes do experimento. O principal problema apontado foi a usabilidade do protótipo (29%) que não se mostrou eficiente nas transições de telas para que outras informações fossem consultadas. Outros problemas foram: a falta de recursos de editor de texto na descrição dos casos de usos (21%) e a falta do recurso de *intellisense* na sentença "se" (21%) a interface de usuário (21%) e, como descrito anteriormente, alguns erros no programa (7%).

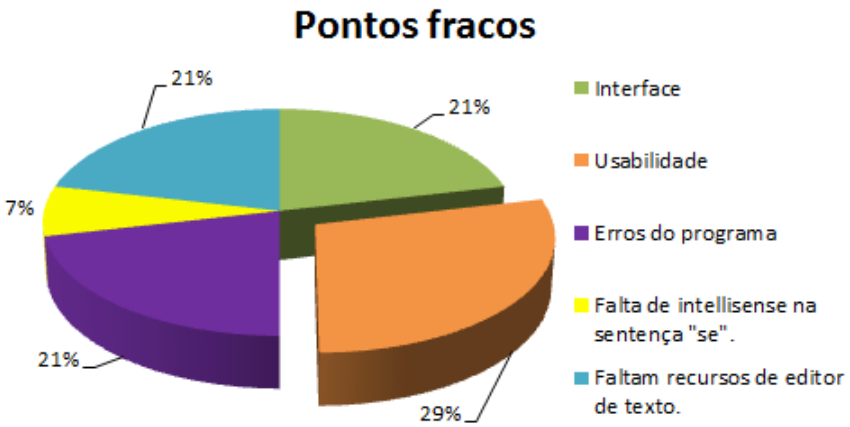

Figura 36 – Pontos fracos da proposta

Exceto o problema com a falta do recurso *intellisense*, todos os demais contribuíram para que o esforço na geração dos documentos de requisitos do grupo COM fosse maior que o do grupo SEM. Outro fato que comprova que necessita-se de mais esforço com o uso da metodologia é que o grupo SEM deixou de escrever 2% dos casos de uso encontrados pelos participante, enquanto o grupo COM deixou de descrever 10%.

A Figura 37 mostram os pontos fortes da metodologia. Segundo os participantes, a maior contribuição da metodologia é ajudar na descoberta e classificação dos requisitos do sistemas (40%), seguido da sua facilidade de aprendizado (20%) e geração de um documento de requisitos padronizado (20%). As mensagens de erros e avisos também foram citadas como pontos positivos, uma vez que os participantes não estão acostumados com este tipo de suporte. Por fim, o uso do recurso *intellisense* foi classificado como ponto positivo, os participantes utilizaram bastante o recurso e reclamaram sua ausência na sentença "se".

5.1.8.5.2 A metodologia

A metodologia foi avaliada, de forma qualitativa, por meio da aplicação de um questionário cujo objetivo era investigar a facilidade de uso e das LNC's. A Tabela 22

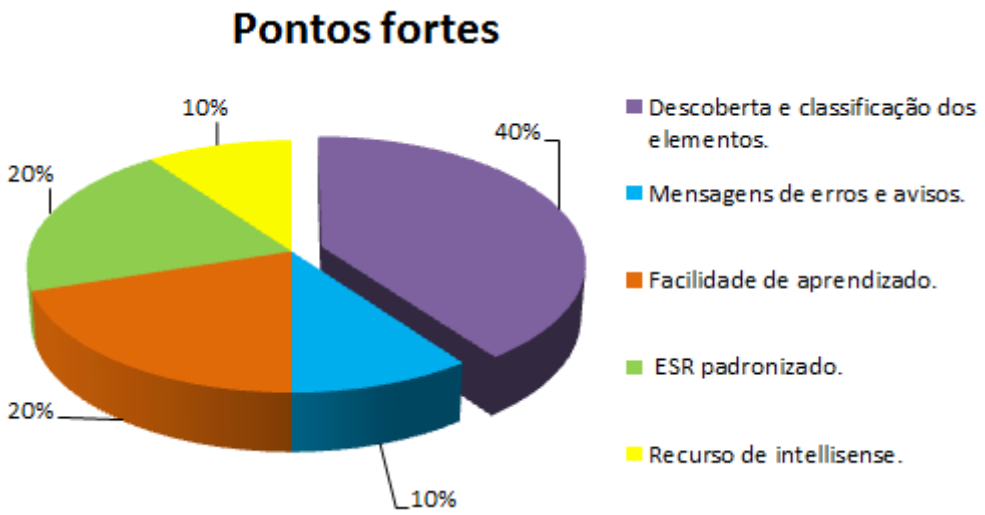

Figura 37 – Pontos fortes da proposta

mostra as questões do questionário. As respostas para as questões poderiam ser respondidas com as opções muto fácil, fácil, difícil e muito difícil.

Tabela 22 – Questionário de avaliação do aprendizado da metodologia

	Questões
Q1	Em relação ao aprendizado da metodologia, você o considera
Q2	Em relação ao aprendizado da LNC *S-LFS*, você o considera
Q3	Em relação ao aprendizado da LNC *S-Lucas*, você o considera

A Figura 38 mostra um gráfico com as respostas do questionário de avaliação da metodologia. As respostas corroboram com a observação de que, inicialmente, surgiram algumas dúvidas sobre as LNC's, mas seu uso foi assimilado pelos participantes do experimento. Além disso, está de acordo com a Figura 37 onde 20% dos pontos fortes apontam para a facilidade de aprendizado da metodologia.

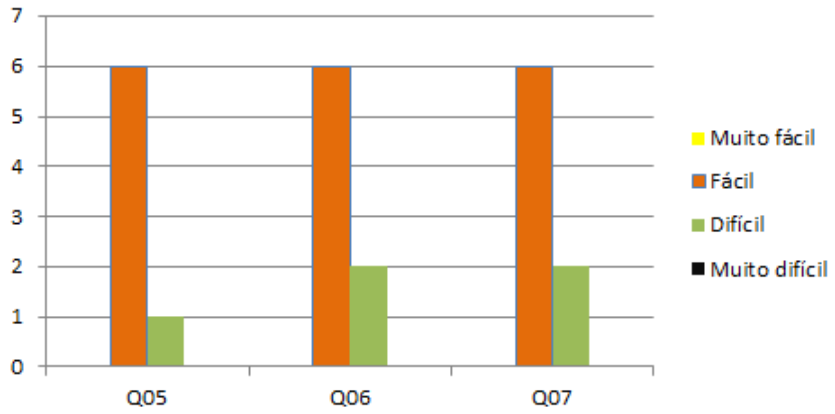

Figura 38 – Avaliação da metodologia

5.1.8.5.3 O ERS-EDITOR

Para uma avaliação qualitativa do ERS-EDITOR, elaborou-se um questionário baseado no trabalho de Esteca et al. (2012). A Tabela 23 mostra as perguntas do questionário. A questão (FA) refere-se ao grau de facilidade para aprender o protótipo, (QI) ao grau de satisfação em relação à interface do protótipo, (CD) ao auxílio que o protótipo oferece para a criação do documento de ERS e (AG) à qualificação do protótipo de um modo geral.

Tabela 23 – Questões para avaliação do ERS-EDITOR

Questões
(FA) Facilidade de aprendizado
(QI) Qualidade da interface
(CD) Contribuição para a criação do documento
(AG) Avaliação geral

Os valores possíveis para as respostas podem ser: péssimo, ruim, regular, bom ou ótimo. a Figura 39 mostra as respostas dos participantes. O item (QI) obteve a pior avaliação e o item (CD) a melhor. Note que o resultado está de acordo com os gráficos mostrados nas Figuras 36 e 37.

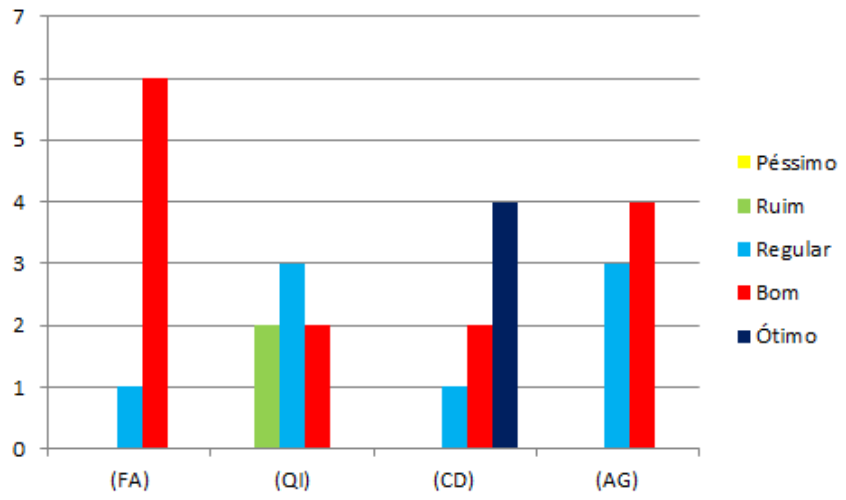

Figura 39 – Avaliação do ERS-EDITOR

5.1.8.5.4 Os recursos

Os recursos disponibilizados pela metodologia/ERS-EDITOR também foram objetos de avaliação, uma vez que sua proposta é ajudar na escrita dos documentos de requisitos. Para avaliar esses recursos, um conjunto de questões foi submetido ao participantes do grupo COM. A Tabela 24 mostra as questões, que deveriam ser respondidas com “dispensável”, “importante” e “essencial”.

Tabela 24 – Recursos disponíveis

	Recurso
R1	Sinônimos do domínio
R2	Léxico da aplicação
R3	Definição semântica
R4	Identificação dos dados armazenados
R5	LNC para Funções do produto
R6	*Intellisense*
R7	LNC para descrições de caso de uso
R8	Árvore explorar
R9	Árvore de avisos
R10	Árvore de erros
R11	Verificação de completude

A Figura 40 mostra o resultado da aplicação do questionário, observe que apenas duas opiniões consideram itens dispensáveis. R1 - Sinônimos do domínio (ver Figura 17), que obteve quatro opiniões como importante, duas como essencial e uma como dispensável. O outro item que obteve uma opinião como dispensável foi R9 - Árvores de avisos (ver Figura 32), além desta opinião, o item obteve quatro opiniões como essencial e duas como importante. Mesmo assim, esses recursos obtiveram suas maiores pontuações em importante e indispensável, o que caracteriza a sua aceitação entre a maioria dos participantes.

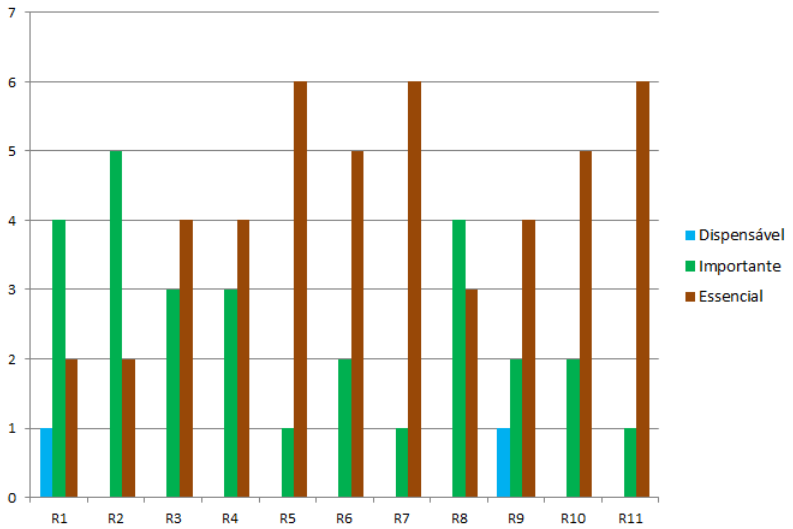

Figura 40 – Avaliação dos recursos

No geral, todos os recursos tiveram a maioria das opiniões entre importante e essencial. Um destaque deve ser dado aos recursos R5, R7 e R11 que obtiveram a opinião de que são essenciais por seis participantes.

5.2 Validade do experimento

Uma das premissas em relação aos resultados de um experimento é o quanto são válidos. Os resultados precisam ser válidos para que possam ser generalizados. Nesta seção, discutiremos a validade dos resultados do experimento, descrevendo algumas ameaças e a confiabilidade dos indicadores encontrados. A seguir são discutidos os quatro tipos de validade previstas em Wohlin et al. (2000).

5.2.1 Validade interna

Os problemas relacionados a este tipo de validade estão ligados aos participantes. Fatores como cansaço, desânimo, sentimento de obrigatoriedade durante a atividade são exemplos de fatores que influenciam na validade interna do experimento. Para evitar tais riscos, os participantes do experimento foram convidados a participar do estudo de forma voluntária, não recebendo nenhum tipo de pro labore ou outras vantagens. Uma ameaça para validade do experimento poderia ser a experiência dos participantes, neste caso, considerou-se que todos os participantes possuem o conhecimento mínimo para as tarefas de levantamento e especificação de requisito e os participantes do grupo COM receberam treinamento no uso de HELP$_4$ERS e do ESR-EDITOR. A seleção dos participantes dos grupos SEM e COM foi feita pela ordem na caderneta da disciplina, não houve o estudo com a inversão dos participantes devido ao tempo disponível ser limitado.

5.2.2 Validade externa

A validade externa define as condições limitantes para generalizar os resultados obtidos no experimento para a população representada pelos participantes. Para alcançar uma validade externa deve-se levar em consideração o perfil dos participantes, o lugar e o tempo de aplicação do experimento. Problemas podem acorrer quando os participantes não representam a população sob interesse, ou quando os instrumentos são inadequados à prática cotidiana, ou quando o experimento é executado em dia e tempo que venham afetar os resultados.

Como descrito anteriormente, os participantes do experimento podem ser considerados representativos para a população de estudantes da área de desenvolvimento de software. Claramente, usar estudantes de graduação para avaliar métodologia (como em experimento) não é o mesmo que usar profissionais com experiência na área. No entanto, como já citados, (HOST; REGNELL; WOHLIN, 2000; RUNESON, 2003; SVAHNBERG; AURUM; WOHLIN, 2008) argumentam que grupos de estudantes e profissionais obtêm resultados semelhantes quando uma nova metodologia de Engenharia de Software é avaliada. Em todo caso, estamos conscientes de que outros experimentos, com profissionais, devem ser construídos.

5.2.3 Validade construção

A validade de construção de um experimento diz respeito aos artefatos usados e os fatores humanos envolvidos no estudo. Ameaças decorrentes de comportamentos inadequados tanto por parte dos participantes quanto dos pesquisadores podem fragilizar o trabalho. Problemas como sub-representação da teoria que envolve o experimento, participantes envolvidos em mais de um estudo e experimentos projetados para dar resultados já esperados; tudo isso, são exemplos de fatores que influenciam a validade da construção.

Quanto à validade dos documentos de requisitos gerados durante o experimento, pode-se dizer que estes são artefatos já trabalhados em sala de aula, porém, sem a mesma padronização exigida no experimento. Isto nos leva a crer que os problemas apresentados possuem, no mínimo, considerável significância para o processo ensino-aprendizagem.

Devido à inexistência de pessoas experientes em $HELP_4ERS$ e do ESR-EDITOR, apenas o autor deste estudo esteve envolvido na condução das atividades do experimento. Estamos conscientes da possibilidade de sobrevalorizar de um ou outro método avaliado. É por isso que, para os próximos estudos, outros profissionais previamente treinados em em $HELP_4ERS$ e do ESR-EDITOR devem ser convidados para conduzir o experimento e atuar como especialista do domínio.

5.3 Considerações finais

Neste capítulo foi apresentado um estudo experimental realizado como intuito de avaliar da metodologia, focando em aspectos relacionados aos atributos de qualidade de um documento de requisitos: não ambiguidade, completude e consistência. O experimento mostrou uma tendencia de que a metodologia pode melhorar os atributos de qualidade, uma vez que apresentam valores acima de 50% maiores do que aqueles obtidos sem o uso da metodologia.

A capacidade de descrever requisitos de S-LFS e *S-Lucas* também foi objeto de avaliação. Sob este aspecto, o experimento mostrou que as elas foram capazes de descrever corretamente 97% das sentenças descritas no estudo. Todavia, não foi possível testar todas os tipos de sentenças da *S-LFS*, além disso faltam situações reais para serem testadas descritas em *S-Lucas*.

Avaliou-se o esforço da geração do documento de requisitos. Neste experimento o uso da metodologia despendeu de 30% a mais de tempo em comparação ao método manual de geração de documentos de requisitos. Por fim, um questionário foi aplicado ao final do experimento a fim de investigar os pontos fortes e os pontos fracos da metodologia.

Pode-se concluir que neste experimento, mesmo sem os testes das hipóteses, o uso

da metodologia mostrou-se promissor pois os resultados dos atributos de qualidades obtidos foram maiores em comparação à metodologia manual e a capacidade das LNC's descreverem os requisitos do sistema alcançou valor acima de 95%. Apesar disso, o experimento também revelou os pontos fracos da metodologia que devem ser corrigidos, principalmente em relação ao protótipo ERS-EDITOR, para que se possa alcançar melhores resultados em relação ao esforço de geração de documentos de requisitos.

6 Conclusão

Este capítulo apresenta as contribuições desta Dissertação, bem como as dificuldades encontradas no seu desenvolvimento e os trabalhos necessários para evoluir à pesquisa.

6.1 Contribuições

Neste trabalho foi apresentada a proposta de uma metodologia focada na melhoria dos atributos de qualidade dos documentos de requisitos: não-ambiguidade, consistência e completude. Para isso, definiu-se 5 Etapas compostas por atividades que levam o analista de requisitos a descrever requisitos em forma de sentenças suscetíveis a suporte automatizado para a verificação dos atributos de qualidades e identificação de elementos de projetos. O suporte automatizado utiliza técnicas de Processamento de Linguagem Natural e Linguagens Naturais Controladas. As principais contribuições deste trabalho são:

- **A metodologia**: Durante a fase de pesquisa bibliográfica, não foi encontrada nenhuma metodologia semelhante à HELP$_4$ERS. Alguns trabalhos semelhantes foram descritos no Capítulo 3. O trabalho apresentado em (FERREIRA; SILVA, 2012) é o que mais se parece com a HELP$_4$ERS, porém, além de não apresenta a Etapa de Definição semântica e não propõe uma estrutura de documento de requisito que apoie essa definição. Os demais trabalho são fontes de inspiração para técnicas que contribuem com o desenvolvimento da HELP$_4$ERS.

- **As LNC's**: As Linguagens Naturais Controladas S-LFS e *S-Lucas* foram desenvolvidas com os objetivos de descrever sentenças inteligíveis a todos os *stakeholders* e com suporte automatizado no sentido de identificar elementos de projetos de forma consistente e não ambígua. Outros trabalhos de definição de LNC para escritas de casos de uso em português são encontrados em (LIMAVERDE, 2006; HORIS, 2010).

- **O protótipo**: Apesar de conter algumas limitações, o protótipo ERS-EDITOR tem recursos ainda não explorados na Engenharia de Requisitos. Por exemplo o uso do *intellisense*, que foi muito utilizado pelos participantes do estudo experimental, contribui para a obtenção de alta consistência no documento de requisitos. Enfim, a ideia de desenvolver uma ferramenta com os mesmos recursos do ERS-EDITOR mostrou-se viável para o auxílio da escrita de documentos de requisitos.

- **Avaliação da metodologia**: Foi realizado um estudo experimental repetível com o objetivo de avaliar o uso da metodologia seguindo as recomendações da Engenharia

de Software Experimental (WOHLIN et al., 2000). Nesse estudo constatou-se que o uso da metodologia pode trazer aumento nos valores dos atributos de qualidades consistência, não-ambiguidade e completude por meio do uso dos recursos disponíveis no ERS-EDITOR. Porém constatou-se também que o uso da metodologia pode aumentar o esforço na geração dos documentos de requisitos.

- **A metodologia/ERS-EDITOR como ferramenta de aprendizagem**: Outro fato importante é que a metodologia proposta apresenta-se como uma ferramenta de aprendizado. Um questionário sobre a metodologia foi submetido para avaliação do Professor da Disciplina de Engenharia de Software da UFPI. Ao responder um questionário com as perguntas "Você utilizaria a metodologia/ERS-EDITOR para criar os documentos de requisitos?" Sua resposta foi que sim, porque com o uso da metodologia teria a garantia de que os padrões de criação dos casos de uso seriam seguidos e estariam sempre de acordo com o *template* definido na ferramenta, ou seja, os casos de uso sempre estariam formatados uniformes, as pré-condições, atores, passos dos fluxos principal, fluxos alternativos e fluxos de exceção também estariam uniformes, além disso auxilia na geração do documento de especificação de requisitos unificado em um padrão e sem o uso da ferramenta, cada projeto tende a seguir um *template* diferente e não padronizado. A outra pergunta do questionário foi "Você considera que a metodologia ajuda no aprendizado das atividades de Engenharia de Requisitos ?" A resposta foi que sim porque a ferramenta "doutrina" uma sequência lógica do detalhamento de cada caso de uso, além de auxiliar na geração e incremento do documento de requisitos. Além disso, como visto no Capítulo 5, os participantes do estudo experimental apontaram que a metodologia proposta é fácil de usar e útil para gerar o documento de requisitos.

6.2 Limitações e dificuldades do trabalho

Durante a realização deste trabalho foram encontradas algumas dificuldades que geraram limitações na obtenção dos objetivos. A seguir, serão apresentadas algumas dessas limitações.

- **Desenvolvimento do ERS-EDITOR**: A principal dificuldade no desenvolvimento do protótipo foi o número de erros de etiquetagem do *Tree Tagger*, o que gerava mensagens erradas. Para solucionar o problema, a etiquetagem correta da palavra deveria ser adicionada no módulo etiquetador. Porém não se sabe quantas palavras da língua portuguesa serão etiquetadas de forma errada. Outra limitação é o tempo de execução da análise das sentenças pelo parser das LNC. O tempo de execução é alto devido à complexidade das análises feitas pelo ANTLR, *Tree Tagger* e as consultas para a verificação semântica dos termos da sentença. Em alguns testes de

performance o tempo médio de execução para um caso de uso com 10 sentenças foi de 2,8s.

- **Interface do ERS-EDITOR**: Durante o experimento, a interface para *Desktop* mostrou-se limitada para a forma como as informações deveriam ser apresentadas para o usuário. A troca de telas para obter informações sobre elementos de projetos cadastrados fez os usuários gastarem tempo para terminar os documentos de requisitos. É necessário que se estude interfaces alternativas que disponibilizem a usabilidade necessária para as atividades da metodologia.

- **Estudo experimental**: Devido a limitações de ambientes ou regionais, os seguintes fatores não contribuíram para um estudo experimental mais robusto: (*i*) encontrar profissionais dispostos a realizar o experimento; (*ii*) o pequeno número de amostras não foi suficiente para que se realizasse um teste estatísticos das hipóteses; (*iii*) o tempo limitado para o experimento; e (*iv*) a falta de experiência em estudo experimental.

6.3 Trabalhos futuros

Durante este trabalho alguns questionamentos com vistas a melhoria da metodologia sugiram e serão tratados nos próximos passos da pesquisa, são eles:

- **Redução do número de falsos-positivo na Etapa 1**: Baseado na construção de um Léxico Ampliado da Linguagem - LAL, descrito em (LEITE; FRANCO, 1993), deseja-se associar os elementos de projeto aos sintagmas e após a seleção utilizando-se a lei de Zipf e os cortes de Luhn, selecionar apenas aqueles termos que exercerem a função dos sintagmas associados;

- **Integração com o trabalho de** (ANCHIETA, 2014): A integração com a ferramenta *EasyGui* permitirá a geração automática dos modelos de projetos a partir das descrições dos casos de uso.

- **Solução do problema do Etiquetador**: Estudo de técnicas e etiquetadores alternativos para minimização/eliminação dos erros de etiquetagem dos termos das sentenças.

- **Melhoria da interface/usabilidade do ERS-EDITOR**: Estudo de técnicas de Interação Humano-Computador (IHC) para melhoria do ERS-EDITOR e implementação de uma versão para WEB.

- **Estudo experimental**: Uma nova avaliação da metodologia com as seguintes características: (*i*) Experimento com profissionais; (*ii*) sem tempo definido para

conclusão, o que poderá permitir medir a corretude do documento de requisitos; (*iii*) medição do esforço de cada Etapa da metodologia, isso implicará numa avaliação do esforço detalhado para cada etapa; e (*iv*) a obtenção de mais amostras para as validações estatísticas das hipóteses.

Referências

ANCHIETA, R. T. *Uso de PLN para extrair elementos de interface de usuário a partir de descrições de requisitos de softwares*. Dissertação (Mestrado) — Universidade Federal do Piauí,UFPI, 2014. Citado 5 vezes nas páginas 25, 27, 33, 38 e 83.

ANCHIÊTA, R. T.; SOUSA, R. F. de; MOURA, R. S. Using NLP techniques for identifying GUI prototypes and UML diagrams from use cases. In: *The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, MA, USA, June 27-29, 2013*. [S.l.: s.n.], 2013. p. 48–53. Citado na página 3.

ARRUDA, D. et al. Engenharia de requisitos: Um survey realizado no porto digital, recife/brasil. In: *WER*. [S.l.: s.n.], 2014. Citado 3 vezes nas páginas 9, 1 e 2.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. In: *Encyclopedia of Software Engineering*. [S.l.]: Wiley, 1994. Citado 2 vezes nas páginas 5 e 57.

BASILI, V. R.; SHULL, F.; LANUBILE, F. Building knowledge through families of experiments. *IEEE Trans. Softw. Eng.*, Piscataway, NJ, USA, v. 25, n. 4, p. 456–473, jul. 1999. ISSN 0098-5589. Citado na página 57.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python*. [S.l.]: O'Reilly, 2009. (O'Reilly Series). ISBN 9780596516499. Citado na página 22.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009. ISBN 978-0-596-51649-9. Citado na página 13.

BOEHM, B. W. A spiral model of software development and enhancement. *IEEE Computer*, IEEE Press, v. 21, n. 5, p. 61–72, 1988. Citado na página 7.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *The unified modeling language user guide*. Upper Saddle River, NJ: [s.n.], 2005. Citado na página 43.

CHAVES, M. S. Um estudo e apreciação sobre algoritmos de stemming para a língua portuguesa. *IX Jornadas Iberoamericanas de Informática*, 2003. Citado na página 15.

COCKBURN, A. *Writing Effective Use Cases*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201702258. Citado 6 vezes nas páginas 11, 12, 29, 31, 49 e 50.

CUNNINGHAM, H. Information extraction, automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, Elsevier, 2005. Citado na página 22.

ESTECA, A. M. N. et al. Computational support for the process of software requirement specification. *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, Ieee, p. 1–10, out. 2012. Citado 3 vezes nas páginas 1, 23 e 75.

FERREIRA, D. de A.; SILVA, A. da. Rslingo: An information extraction approach toward formal requirements specifications. In: *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*. [S.l.: s.n.], 2012. Citado 8 vezes nas páginas 1, 3, 22, 23, 27, 34, 38 e 81.

FILHO, W. de P. P. *Engenharia de software: fundamentos, métodos e padrões*. [S.l.]: Livros Técnicos e Científicos, 2001. Citado 2 vezes nas páginas 28 e 31.

GAUSE, D. C.; WEINBERG, G. M. *Exploring Requirements: Quality Before Design*. New York, NY, USA: Dorset House Publishing Co., Inc., 1989. ISBN 0932633137. Citado na página 2.

HORIS, r. A. A. *ucsCNL – A Controlled Natural Language for Use Case Specifications*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, UFPE, 2010. Citado na página 81.

HOST, M.; REGNELL, B.; WOHLIN, C. Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering*, v. 5, n. 3, nov. 2000. Citado 2 vezes nas páginas 61 e 77.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. 1990. 1-84 p. Citado na página 7.

IEEE, E. *IEEE Recommended Practice for Software Requirements Specifications*. [S.l.], 1998. Citado 10 vezes nas páginas 2, 8, 10, 21, 23, 25, 28, 36, 47 e 69.

ISO (Ed.). *ISO/IEC 14977:1996 Information Technology - Syntactic Metalanguage - Extended BNF*. 1996. Citado na página 42.

JACKSON, P.; MOULINIER, I. *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization*. [S.l.]: John Benjamins Pub., 2002. (Natural language processing). ISBN 9789027249890. Citado na página 13.

JACOBSON, I. *Object-oriented Software Engineering*. New York, NY, USA: ACM, 1992. ISBN 0-201-54435-0. Citado 2 vezes nas páginas 12 e 31.

JAMASOFTWARE. State of requirements management survey. 2011. Citado 3 vezes nas páginas 9, 1 e 2.

JUAREZ-RAMIREZ, R.; HUERTAS, C.; INZUNZA, S. Automated Generation of User-Interface Prototypes Based on Controlled Natural Language Description. *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, Ieee, p. 246–251, jul. 2014. Citado na página 3.

KAINDL, H.; SVETINOVIC, D. On confusion between requirements and their representations. *Requirements Engineering*, jan. 2010. ISSN 0947-3602. Citado na página 2.

KAO, A.; POTEET, S. R. *Natural Language Processing and Text Mining*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1849965587, 9781849965583. Citado na página 14.

KOSCIANSKI; SOARES. *Qualidade de Software - 2ª Edição*. [S.l.]: Novatec, 2007. ISBN 978-85-7522-112-9. Citado na página 1.

KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering: Processes and Techniques*. [S.l.]: John Wiley, Sons, Inc., 1998. ISBN 0471972088. Citado na página 1.

KRUCHTEN, P. *The Rational Unified Process: An Introduction*. 3. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321197704. Citado na página 12.

LEITE, J.; FRANCO, A. A strategy for conceptual model acquisition. In: *Proceedings of IEEE International Symposium on Requirements Engineering*. [S.l.]: IEEE Computer Society Press, 1993. Citado 2 vezes nas páginas 21 e 83.

LIMA, V. et al. Investigação experimental e Práticas ágeis : ameaças à validade de experimentos envolvendo a prática ágil Programação em Par . n. 2004, 2011. Citado na página 62.

LIMAVERDE, G. F. *Geração de Especificação Formal de Sistemas a partir de Documento de Requisitos*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, UFPE, 2006. Citado na página 81.

LUHN, H. *The Automatic Creation of Literature Abstracts (auto-abstracts)*. [S.l.]: IBM Research Center, International Business Machines Corporation, 1958. Citado 2 vezes nas páginas 16 e 38.

MIRIAM, S. *Verificação e Validação em Requisitos : Processamento da Linguagem Natural e Agentes*. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, PUC, 2007. Citado 4 vezes nas páginas 16, 17, 21 e 23.

NAVARRO, G. A guided tour to approximate string matching. *ACM Computing Surveys*, v. 33, n. 1, p. 31–88, 2001. Citado na página 22.

OLIVEIRA, P. *PTStemmer – A Stemming toolkit for the portuguese language*. 2014. Disponível em: <https://code.google.com/archive/p/ptstemmer/>. Citado 2 vezes nas páginas 16 e 40.

ORENGO, V. M.; HUYCK, C. A Stemming Algorithm for Portuguese Language. In: *Proc. of Eigth Symposium on String Processing and Information Retrieval (SPIRE 2001) - Chile*. [S.l.: s.n.], 2001. p. 186–193. Citado na página 16.

PALOMARES, C.; QUER, C.; FRANCH, X. PABRE-Man: Management of a requirement patterns catalogue. *2011 IEEE 19th International Requirements Engineering Conference*, IEEE, p. 341–342, ago. 2011. Citado 4 vezes nas páginas 3, 21, 23 e 27.

PARR, T. *The Definitive ANTLR Reference: Building Domain-Specific Languages*. [S.l.]: Pragmatic Bookshelf, 2007. ISBN 0978739256. Citado na página 17.

PORTER, M. F. Readings in information retrieval. In: JONES, K. S.; WILLETT, P. (Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. cap. An algorithm for suffix stripping, p. 313–316. ISBN 1-55860-454-5. Citado na página 15.

PRESSMAN, R. *Engenharia de Software*. [S.l.]: McGraw Hill Brasil, 2009. Citado na página 8.

REU-DEBOVE, J.; MORAIS, C. D. Léxico e dicionário. *ALFA: Revista de ...*, p. 45–69, 1984. Citado na página 38.

ROSENBERG, D.; STEPHENS, M. *Use Case Driven Object Modeling with UMLTheory and Practice*. [S.l.: s.n.], 2007. 438 p. ISBN 1590597745. Citado 2 vezes nas páginas 11 e 31.

ROYCE, W. W. Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON, Los Angeles*, p. 1–9, August 1970. Citado na página 7.

RUNESON, P. Using students as experiment subjects–an analysis on graduate and freshmen student data. In: *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering.–Keele University, UK*. [S.l.: s.n.], 2003. p. 95–102. Citado 2 vezes nas páginas 61 e 77.

SALTON, G.; MCGILL, M. *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1983. Citado na página 38.

SCHMID, H. *Probabilistic Part-of-Speech Tagging Using Decision Trees*. 1994. Citado na página 15.

SCHMIDT, D. C. Guest editor's introduction: Model-driven engineering. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 39, n. 2, p. 25–31, fev. 2006. ISSN 0018-9162. Citado na página 22.

SCHWITTER, R. Natural languages. technical report. *Centre for Language Technology, Macquarie University*, 2007. Citado 2 vezes nas páginas 19 e 42.

SCHWITTER, R. Controlled natural languages for knowledge representation. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (COLING '10), p. 1113–1121. Citado 2 vezes nas páginas 3 e 42.

SOARES, H. A.; MOURA, R. S. A methodology to guide writing software requirements specification document. In: *Computing Conference (CLEI), 2015 Latin American*. [S.l.: s.n.], 2015. p. 1–11. Citado 2 vezes nas páginas 4 e 54.

SOMMERVILLE, I. et al. *Engenharia de software*. São Paulo: Pearson Prentice Hall, 2008. ISBN 9788588639287 8588639289. Citado 5 vezes nas páginas 9, 2, 7, 8 e 10.

SVAHNBERG, M.; AURUM, A.; WOHLIN, C. Using students as subjects - an empirical evaluation. In: ROMBACH, H. D.; ELBAUM, S. G.; MüNCH, J. (Ed.). *ESEM*. [S.l.]: ACM, 2008. p. 288–290. ISBN 978-1-59593-971-5. Citado 2 vezes nas páginas 61 e 77.

TEAM, C. P. D. *CMMI for Systems Engineering/Software Engineering, Version 1.02, Staged Representation (CMMI-SE/SW, V1.02, Staged)*. Pittsburgh, PA, 2000. Citado na página 23.

TORO, A. D. et al. A requirements elicitation approach based in templates and patterns. In: *Anais do WER99 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, Setembro 9-10, 1999*. [S.l.: s.n.], 1999. p. 17–29. Citado 6 vezes nas páginas 2, 23, 25, 27, 28 e 29.

WIRFS-BROCK, R. Designing scenarios: Making the case for a use case framework. *The Smalltalk Report*, v. 3, n. 3, Nov-Dec 1993. Citado na página 12.

WOHLIN, C. et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN 0-7923-8682-5. Citado 4 vezes nas páginas 5, 57, 77 e 82.

YOUNG, R. *The requirements engineering handbook*. [S.l.]: Boston: Artech House, 2004. I-XX, 1-254 p. Citado 2 vezes nas páginas 2 e 35.

ZIPF, G. *Human behavior and the principle of least effort: an introduction to human ecology*. [S.l.]: Addison-Wesley Press, 1949. Citado 2 vezes nas páginas 16 e 38.

f

Apêndices

APÊNDICE A – Base de verbos sinônimos

Verbo	**Sinônimo**
exibir	apresentar mostrar
consultar	localizar pesquisar
inserir	acrescentar cadastrar digitar entrar fornecer incluir informar introduzir preencher
executar	acionar chamar invocar
deletar	apagar excluir remover
alterar	atualizar corrigir editar modificar mudar
imprimir	emitir
gravar	armazenar salvar
clicar	pressionar
abrir	acessar
selecionar	——
validar	verificar
escolher	——

APÊNDICE B – *S-Lucas*: Erros sintáticos

	Mensagem	**Exemplo**
1.	A sentença *"sentença"* não está numerada	O Usuário externo abre a tela Solicitar certidão.
2.	Sentença incompleta: Ator não encontrado	1. O abre a tela Solicitar certidão.
3.	Sentença incompleta: Ação não encontrada	1. O Usuário externo a tela Solicitar certidão.
4.	Sentença incompleta: Objeto não encontrado	1. O Usuário externo abre.
5.	Sentença incompleta: ENTÃO não encontrado.	7.1 Se o tipo de certidão for vazio
6.	Cada expressão lógica deve possuir um verbo.	7.1 Se o tipo de certidão vazio então
7.	O verbo *"**verbo**"* necessita de dois operandos. (Sentença condicional com verbos de comparação)	7.1 Se o tipo de certidão **for** igual a então | 7.1 Se **for** igual ao tipo de certidão então
8.	A ação *"**verbo**"* necessita de um ator que a inicie. (Sentença condicional com verbos de ação)	7.1 Se **clicar** na opção Cancelar então
9.	A ação *"**verbo**"* necessita de um objeto. (Sentença condicional com verbos de ação)	7.1 Se o usuário externo **clicar** na opção então
10.	A sentença contém mais de uma ação	4. O sistema **grava** a solicitação e **emite** o recibo

APÊNDICE C – *S-Lucas*: Erros semânticos

	Mensagem	Exemplo
1.	O termo *"**nome do termo**"* não foi classificado como Ator.	2. O **Jurisdicionado** escolhe o tipo de certidão
2.	O termo *"**nome do termo**"* não foi classificado como *"elemento de projeto"*.	2. 1. O Usuário externo acessa a **tela de certidão**.
3.	O termo *"nome do termo"* não foi classificado como Atributo ou Requisito de Armazenamento.	7.1 Se existir mais de um **pedido** então
4.	O termo *"**nome do termo**"* não foi classificado como atributo.	8. o sistema atualiza a Data da solicitação para a **data atual**.
5.	O termo *"**nome do termo**"* não foi classificado	7.1 Se o tipo de certidão for igual a **retificadora** então
6.	Nenhuma interface de usuário foi acessada antes.	Qualquer sentença cujo verbo de ação espeta um objeto do tipo "elemento de interface" e nenhuma interface tenha sido acessada. Ex: 3. O Usuário externo clica no botão Gravar.
7.	É recomendado o uso da sentença 'se' apenas nos fluxos alternativo ou de exceção	7. Se o tipo de certidão for vazio então (No fluxo principal)
8.	Para maior clareza, trocar "verificar se" por "validar que"	1. O sistema **verifica se** todos campos obrigatórios estão preenchidos.