A methodology to guide writing Software Requirements Specification document

Hécio A. Soares
Departamento de Informática
Instituto Federal do Piauí
Email: helcio.soares@ifpi.edu.br

Raimundo S. Moura
Departamento de Computação
Universidade Federal do Piauí
Email: rsm@ufpi.edu.br

***Abstract*—The Requirements Engineering (RE) is the process of defining, documenting and mantaining requirements and it aims to support the creation and the maintenance of the Software Requirements Specification document (SRS). This document should be produced such way that all the participants can understand it. It is basis for all other activities of development and its quality is fundamental for the project success. This paper presents a methodology for guide writing SRS documents consistent, unambiguous and automated support in its content. The methodology is supported by ERS-EDITOR tool that uses Natural Language Processing techniques to automate the steps of the methodology. A preliminary assessment highlights promising results for the proposed approach.**

***Keywords*—Requirement engineering, Quality Requirements, Natural Language Processing.**

I. Introdução

De acordo com o padrão IEEE Std. 830/1998 [1], que dispõe sobre Especificação de Requisitos de Softwares - ERS, o processo de desenvolvimento de *software* deve iniciar com um acordo entre desenvolvedor e cliente sobre o que o *software* deve fazer. A Engenharia de Requisitos (ER) é o processo de definir, documentar e manter requisitos que tem como objetivo apoiar a criação e manutenção do documento de ERS [2]. Esse documento norteará a construção do sistema, que deve atender às reais necessidades de seus usuários, além disso, é determinante para a qualidade do produto e para a satisfação dos clientes [3]. A padronização de documentos de ERS facilita a comunicação entre os envolvidos no desenvolvimento do *software*, tais como, clientes, usuários, analistas de requisitos e programadores, comumente chamados de *stakeholders*[1]. Essa padronização ainda facilita o entendimento comum e melhora a consistência e inteligibilidade do documento de ERS [3].

Ferreira e Silva [4] defendem o uso de texto em linguagem natural para escrever especificações de requisitos por ser de uso comum entre os *stakeholders*, 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. Apesar de possuir essas características, o uso da linguagem natural apresenta problemas como ambiguidade, incompletude e inconsistências. Esses problemas geram uma distância entre a especificação e a modelagem do sistema, uma vez que permitem múltiplas interpretações que dão origem a modelos que não estão em conformidade com o documento de ERS [2].

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 e Bernárdez [5] desenvolveram modelos de requisitos e identificaram dois tipos de padrões: **padrões linguísticos** (*L-pattens*) que são frases em linguagem natural frequentemente usadas para descrever requisitos de sistema e que podem ser parametrizadas; e **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. O uso desses padrões ajuda na obtenção de documentos de ERS de melhor qualidade tanto em conteúdo quanto em sintaxe [6]. A utilização desses padrões associada às técnicas de Processamento de Linguagem Natural - PLN possibilita a extração de artefatos de projetos a partir das descrições de requisitos. Anchieta et al. [7] e Juarez-Ramirez et al. [8] apresentam o uso de PLN para extrair artefatos de modelagem como casos de uso, atores, classes, operações, atributos e interfaces de usuários, a partir de descrições de casos de uso que contém padrões linguísticos predefinidos.

Outra forma de contornar os problemas da linguagem natural é o uso de Linguagem Natural Controlada - LNC para restringir a criação de requisitos e casos de uso. Uma Linguagem Natural Controlada é um subconjunto de alguma linguagem natural, (e.g., português) que usa uma gramática restrita e um vocabulário predefinido de acordo com o domínio, com o objetivo principal de evitar complexidade e ambiguidade em textos técnicos [9].

Dessa forma, assim como Ferreira e Silva [4], defendemos um foco maior na especificação de requisitos textuais e que ferramentas devem ser capazes de extrair automaticamente as informações pertinentes ao domínio do sistema a ser construído. Essas informações devem ser *(i)* escritas obedecendo padrões linguísticos ou obedecendo às regras de uma LNC e; *(ii)* identificadas através da semântica das palavras e frases que são representadas textualmente e organizadas através do documento de ERS.

Nesse contexto e considerando a escrita de documentos de ERS como um passo fundamental para o desenvolvimento de sistemas, este trabalho propõe uma metodologia com os objetivos de: *i)* conduzir a escrita de documentos de ERS completos, consistentes e não ambíguos; e *ii)* permitir suporte

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

978-1-4673-9143-6/15/$31.00

automatizado nas descrições dos requisitos para posterior extração de artefatos de modelagem.

O trabalho está fundamentado em: *(i)* O padrão IEEE Std. 830/1998 [1] para a definição das seções presentes no documento de ERS com o objetivo de auxiliar a semântica do domínio; *(ii)* técnicas de mineração de textos para construir uma base de conhecimento sobre o domínio do sistema [10]; *(iii)* padrões de requisitos para definição de modelos de requisitos; e (iv) padrões linguísticos para auxiliar a escrita do documento e definir uma LNC, com o objetivo de identificar artefatos de modelagem [8] e definir/verificar da semântica do domínio.

O protótipo de uma ferramenta chamada ERS-EDITOR está sendo implementado par dar apoio automatizado à metodologia e validar suas etapas. Esse protótipo será disponibilizado para avaliação da comunidade acadêmica após sua completa implementação. Alguns testes foram aplicados em parte da metodologia e os resultados são animadores pois refletem os objetivos do trabalho.

O restante do artigo está organizado da seguinte forma. A Seção II apresenta a metodologia proposta para a escrita de documentos de ERS. A Seção III apresenta o suporte automatizado oferecido pelo protótipo da ferramenta ERS-EDITOR. Na Seção IV são descritos alguns resultados preliminares e discussões sobre os testes aplicados. A Seção V descreve os principais trabalhos relacionados e, finalmente, a Seção VI expõe as conclusões e trabalhos futuros.

II. Metodologia Proposta

É importante mencionar que este trabalho faz parte de um projeto mais amplo que consiste em gerar artefatos de modelagem de *software* a partir de documentos de ERS. Anchiêta [11] apresenta a ferramenta *EasyGUI Prototyping* que contribui para a geração de protótipos de interfaces de usuários, diagramas de classes e diagramas de casos de uso a partir de descrições textuais de casos de uso utilizando técnicas de PLN. A metodologia proposta no nosso trabalho tem como objetivo auxiliar a escrita de documentos de ERS completos, consistentes e não ambíguos, permitindo, assim, o suporte automatizado nas descrições dos requisitos para posterior extração e geração de artefatos de modelagem. Destaca-se que essa metodologia é fundamentada nas recomendações do padrão IEEE Std. 830/1998 [1] e nos padrões de requisitos e padrões linguísticos de Toro e Bernárdez [5].

A. Estrutura do documento de ERS

A estrutura proposta para o documento de ERS auxilia a definição da semântica do domínio e o suporte automatizado às descrições textuais de suas seções. Esse suporte consiste em identificar, através de técnicas de PLN, artefatos de modelagem, tais como: atores, casos de usos, classes, atributos, métodos e interfaces de usuário. Para selecionar as seções do ERS e, ao mesmo tempo, relacioná-las aos artefatos, considerou-se o que o padrão IEEE Std. 830/1998 [1] e Toro e Bernárdez [5] recomendam que seja escrito em cada seção, dessa forma escolheu-se aquelas cujo conteúdo das seções refere-se a algum dos artefatos.

Assim, baseado em observações em documentos de ERS em conformidade com o padrão IEEE Std. 830/1998 [1] as seções *Funções do produto*, *Sistemas e usuários externos* e *Interfaces de usuário* foram selecionadas. De forma semelhante, a seção *Requisitos de armazenamento* foi selecionada a partir do trabalho de Toro e Bernárdez [5]. Além dessas seções, escolheu-se também duas outras consideradas importantes em um documento de ERS, são elas: *(i) Introdução* com as subseções *objetivo*, *escopo*, *referências* e *conceitos e siglas*; e *(ii) Descrições de casos de uso*. A Tabela I mostra todas as seções do documento de ERS proposto e um breve resumo sobre cada uma delas. Argumenta-se que este modelo de ERS contém as informações necessárias para suportar as atividades inerentes ao desenvolvimento de *software*. No entanto, o modelo precisa ser validado através da aplicação de alguns estudos de casos.

Tabela I: Seções do ERS

Seções do ERS	**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 referências, 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. Usuários 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 vai 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 artefatos de modelagem são associados a elementos das seções do documento de ERS, a saber: Atores são associados à seção *Usuários 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*. A Tabela II mostra os artefatos e as seções do documento de ERS relacionados.

Tabela II: Artefatos de modelagem/Seções do ERS

Artefatos de modelagem	**Seções**
Atores	Usuários 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

Para cada seção foi definido um **padrão de requisito** (i.e., um formulário para preenchimento das informações de cada seção) e, em alguns casos, uma LNC para permitir a identificação de padrões linguísticos e a definição e verificação semântica de termos extraídos dos documentos de referência. 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 1.5 que obedece ao processo Praxis [12]. Este padrão foi escolhido por ser uma representação tabular simples (com apenas os campos nome e descrição), de fácil entendimento e porque o processo Praxis segue as recomendações do padrão IEEE Std. 830/1998 [1]. Os padrões de requisitos das seções 4 e 6 são baseadas em Toro e Bernárdez [5], pois esses padrões são amplamente encontrados em especificações de requisitos, facilitam a comunicação entre os *stakeholders* e podem ser reutilizados com algumas adaptações além de ajudar na obtenção de documentos de ERS de melhor qualidade tanto em conteúdo quanto em sintaxe.

Adicionalmente, após observação dos **padrões linguísticos** definidos em [4]–[6], [11], [13] e em vários documentos de requisitos, verificou-se padrões que se repetem com muita frequência em descrições de requisitos funcionais. Dessa forma, definiu-se dois padrões para identificação de artefatos de modelagem e uma LNC para a seção *Funções do produto*. Os padrões linguísticos para identificação de artefatos de modelagem são mostrado na Tabela III, o símbolo '?' significa que a classe gramatical pode ou não ocorrer. O símbolo 'ǀ' 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 III: Padrões linguísticos para identificar artefatos de modelagem

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

A LNC definida para a seção *Funções do produto* aceita dois tipos de sentenças com propósitos bem definidos, são eles:

1) Quando o sistema deve disponibilizar uma funcionalidade para o usuário executar uma ação sobre um requisito de armazenamento.
   Padrão 1: "O sistema deve permitir" <*usuario*> <*acao*> <*requisito_de_armazenamento*>. Como exemplo, temos a sentença "O sistema deve permitir ao usuário chefe anexar uma certidão retificadora", dela pode-se extrair os seguintes elementos:
   - *usuario*: usuário chefe
   - *acao*: anexar
   - *requisito_de_armazenamento*: certidão retificadora
2) Quando o sistema deve executar uma ação sobre um requisito de armazenamento para um determinado usuário ou requisito de armazenamento, quando a uma ação sobre algum requisito de armazenamento acontecer.
   Padrão 2: "O sistema deve" <*acao*> <*requisito_de_armazenamento*> (<*usuario*>ǀ<*requisito_de_armazenamento*>) <*expressao_temporal*> <*usuario*> <*acao*> <*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:
   - *acao*: enviar
   - *conceito/requisito_de_armazenamento*: email
   - *usuario*: usuário solicitante
   - *expressao_temporal*: quando
   - *usuario*: usuário chefe
   - *acao*: finalizar
   - *requisito_de_armazenamento*: certidão

Inicialmente, buscou-se identificar padrões gerais. Porém, outros padrões devem ser definidos para reconhecer mais padrões que descrevem requisitos funcionais e atender especificidades de uma determinada organização ou domínio de aplicação. Portanto, novas observações devem ser realizadas em trabalhos futuros.

B. Etapas da Metodologia

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

Como em Ferreira e Silva [4] 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, em colaboração com os analistas de domínio, descobrir os requisitos do sistema [14]. 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 1 mostra a visão geral da metodologia proposta. A seguir serão descritas suas etapas sem focar em como elas são implementadas.

A etapa 1 - Análise do domínio, consiste em gerar automaticamente o léxico da aplicação a partir dos documentos de referência. Essa etapa serve como base para as demais. Qualquer arquivo, cujo conteúdo diz respeito ao domínio do sistema e escrito em linguagem natural pode ser utilizados para criação do léxico. Por exemplo, atas de reuniões, manuais de sistemas legados e manuais técnicos.

Na mesma figura, a etapa 2 - Definição da semântica, consiste em o analista de requisitos classificar como artefatos de sistemas os termos do léxico da aplicação que foram gerados na etapa anterior. Essa tarefa pode ser realizada com o auxílio do analista de domínio, uma vez que este deve saber esclarecer o significado de cada termo do léxico. Ao classificar um termo do léxico, um registro é criado no formulário da seção correspondente ao artefato. Todos os formulários têm um campo nome, este campo é preenchido automaticamente com o termo selecionado. Nessa etapa, o analista de requisitos pode excluir os termos sem relevância para o domínio da aplicação.

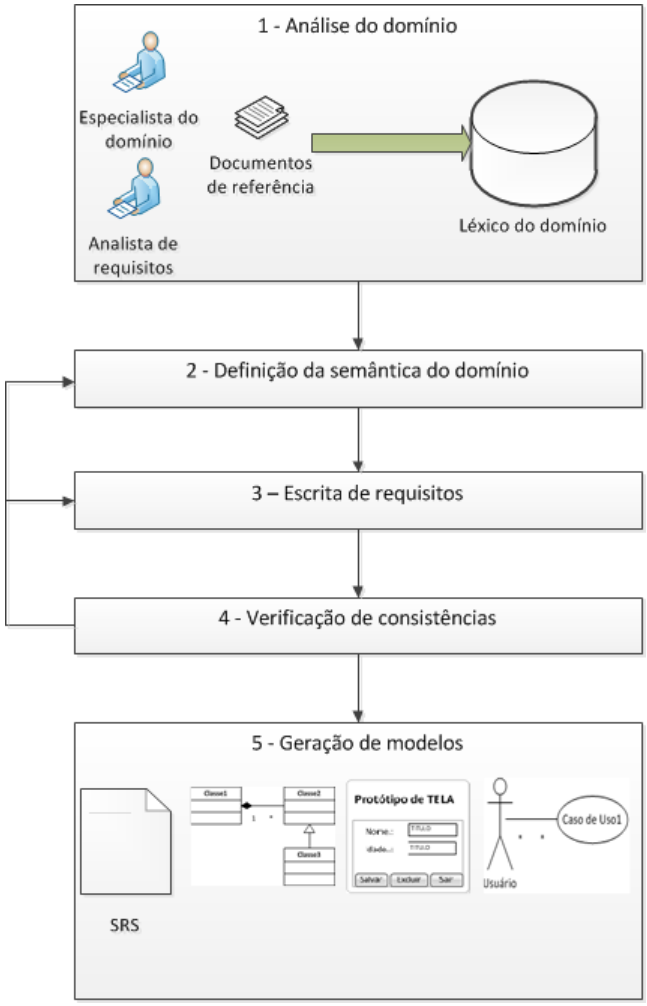

Figura 1: Visão geral da metodologia

Após a classificação dos termos do léxico o analista de requisitos pode passar para a etapa 3 - Escrita de requisitos. Essa etapa consiste no preenchimento dos formulários definidos para cada seção. Os itens de cada fomulário podem ser criados de duas maneiras, a primeira tem origem na etapa 2 - Definição semântica e o campo que identifica o item já está preenchido, neste caso o analista deve preencher os demais campos. A segunda maneira ocorre com a criação manual do item feita pelo analista, neste caso, ele preenche todos os campo do formulário. Para obter o suporte automatizado às descrições dos requisitos, o analista deve seguir a LNC definida para a seção.

O analista de requisitos tem a liberdade de escrever de maneira diversa à LNC, mas dessa forma não disponibilizará de todos os recursos oferecidos pela metodologia o pode levar a um documento de ERS 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 que descrevem os requisitos dos sistema. De forma geral, a verificação consiste na identificação de artefatos, se eles fazem parte do léxico, se estão classificados em alguma seção do documento e se estão escritos de acordo com a semântica do domínio. A metodologia propõe o uso de padrões linguísticos para construção de sentenças em conformidade com uma LNC definida para a seção, isso garante o suporte automatizado para descoberta dos artefatos de modelagem. A verificação das sentenças que estão em conformidade com a LNC pode resultar em dois tipos de mensagens: erros e avisos.

Na etapa 4 - Verificação de inconsistências, o analista de requisitos pode verificar a consistência das sentenças que descrevem os requisitos dos sistema. De forma geral, essa verificação consiste na identificação de artefatos, se eles fazem parte do léxico, se estão classificados em alguma seção do documento e se estão escritos de acordo com a semântica do domínio. A metodologia propõe o uso de padrões linguísticos para construção de sentenças em conformidade com uma LNC definida para a seção, isso garante o suporte automatizado para descoberta dos artefatos de modelagem. O analista de requisitos tem a liberdade de descrever o requisito de maneira diversa à LNC, mas dessa forma não disponibilizará de todos os recursos oferecidos pela metodologia. A verificação das sentenças que estão em conformidade com a LNC pode resultar em dois tipos de mensagens: erros e avisos.

Os erros impedem que os artefatos 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, descrito anteriormente, 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**: Os artefatos identificados na sentença, sintaticamente correta, não estão 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, pois o objeto do verbo "permitir" deve ser um usuário e o que aparece é o termo "certidão" que 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. 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 semanticamente classificado.
- **Sentença fora do padrão**: A sentença não está escrita de acordo com o a LNC definida para a seção. Assim, a metodologia alerta o analista de requisitos que ao escrever dessa forma o documento de ERS estará sujeito aos problemas de escrita em linguagem natural.

Ao detectar um erro ou aviso, o analista de requisitos tem a opção voltar para as etapas 2 ou 3 para corrigi-lo, esse ciclo deve ser executado até que todos os erros sejam corrigidos e, opcionalmente, que nenhum aviso seja exibido. 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 protótipos de interface de usuário, diagramas de classes e de casos de uso, a partir da seção *Descrições de casos de uso* do documento de ERS. É importante destacar que esta etapa já fora descrita e validada em outros trabalhos do grupo de pesquisa em PLN da UFPI. Em especial, o trabalho [11] apresenta a ferramenta *EasyGUI Prototyping* que contribui para a geração de protótipos de interfaces de usuários, diagramas de classes e diagramas de casos de uso a partir de descrições textuais de casos de uso utilizando técnicas de PLN.

III. O Suporte Automatizado

Para reduzir o esforço humano na criação e verificação de especificações de requisitos em linguagem natural bem como auxiliar o aprendizado do domínio do sistema e tornar essas atividades menos propensas a erros, um conjunto de ferramentas e linguagens adequadas devem ser disponibilizadas para o analista de requisitos [4]. Esta seção apresenta o suporte automatizado às etapas de 1 a 4 da metodologia e os detalhes da implementação de cada etapa. Ilustra-se exemplos utilizando o protótipo ERS-EDITOR. As ferramentas e técnicas de PLN utilizadas na implementação do protótipo são apresentados a seguir.

a) Etiquetagem: Do inglês *part-of-speech (POS) tagging*, consiste em uma análise do texto *tokenizado*[2] e inserção de etiquetas em cada *token*. Um *part-of-speech tagger* é um etiquetador morfossintático, que classifica cada palavra em uma classe gramatical. Utilizou-se o etiquetador *TreeTagger* [15], pois, segundo Anchieta et al. [7], obteve melhor resultado quando comparado ao MXPOST [16] e ao QTag [17].

b) Reconhecimento de padrões linguísticos: Para o reconhecimento dos padrões linguísticos mostrados na Tabela III, utilizou-se a técnica de PLN *chunk parsing* [18], que consiste em explorar um alinhamento entre as estruturas da sentença e as suas semânticas, baseando-se na posição relativa e *part-of-speech* de cada palavra, dessa forma é possível obter vantagens de modo a extrair o significado dos termos das sentenças [4]. Além de utilizar menos poder computacional, a utilização de *chunk parsing* é flexível e robusta ao extrair informação [18]. Para a implementação dos padrões linguísticos optamos pelo uso da ferramenta ANTLR (*Another Tool Language Recognition*) [19], por ser uma ferramenta para construção de analisadores léxico (*lexer*) e sintático (*parser*) de linguagens formais.

c) Radicalização: O processo de radicalização, do inglês Stemming, visa à obtenção do radical de uma palavra. É utilizado quando se deseja agrupar palavras com diferentes grafias, ou mesmos diferentes categorias gramaticais, mas relacionadas a um mesmo conceito. Utilizou-se o algoritmo proposto por Orengo e Huick [20] por ter sido desenvolvido exclusivamente para a língua portuguesa.

A. Análise do domínio

Reportando-se ao domínio do sistema, o léxico é o conjunto de palavras e significados restrito. Este tipo de dicionário estruturado é de suma importância, pois estabelece um vocabulário comum para termos-chaves do domínio [4]. O objetivo dessa etapa é a criação do léxico da aplicação, para isso são executados os seguintes passos para cada documento de referência:

1) **Etiquetagem**: Após a *tokenização* do texto, cada *token* recebe uma etiqueta que representa a classe gramatical da palavra.
2) **Calcular os termos relevantes**: Neste passo utiliza-se a abordagem *bag-of-words* [21] para estruturação dos arquivos de referência e o corte inferior de Luhn [22] aplicado à teoria de Zipf [23]. 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**)[3], calcula-se a frequência de cada um deles, ordena-os de forma decrescente e utiliza-se um parâmetro definido pelo analista de requisitos como o limite inferior, a Figura 2 mostra o corte de Luhn adaptado ao trabalho.

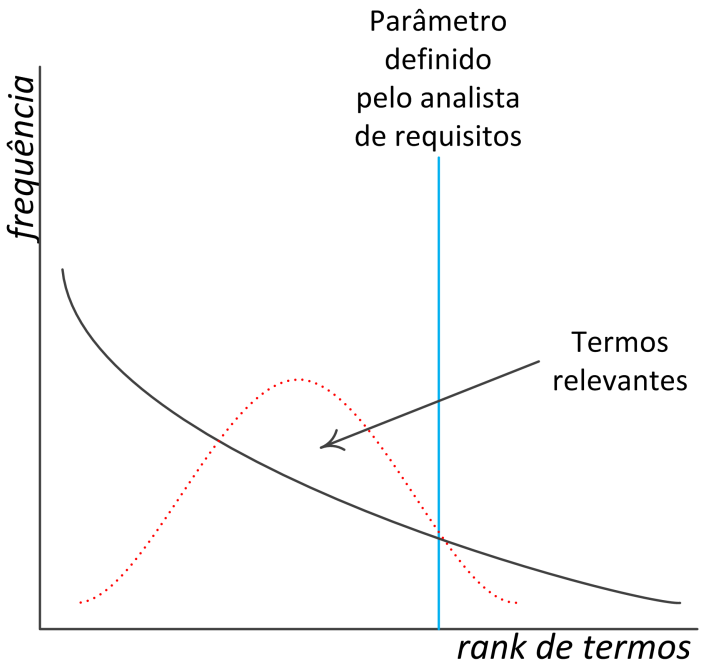

Figura 2: Curva de ZIpf e Cortes de Luhn adaptados

[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, utilizamos o ANTLR para a tokenização de textos.

[3]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.

3) **Construção do 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 são reduzido à forma do infinitivo e adicionados à lista de Ações.
   - **Conceitos compostos** : São termos relevantes compostos por ngramas[4], 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. Caso positivo, o elemento é adicionado à lista de conceitos composto.
   - **Funcionalidades**: São ações sobre conceitos simples ou compostos. Consiste em uma lista composta por ngramas 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 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.

A Figura 3 mostra a tela do protótipo ERS-EDITOR com exemplos de termos do léxico da aplicação. As seguintes ferramentas são utilizadas: ANTLR para *tokenização* do texto, definição e reconhecimento dos padrões linguísticos; TreeTagger para etiquetagem do texto; e a ferramenta PTStemmer para a redução do termo ao seu radical.

B. Definição semântica

Como já descrito, a definição semântica consiste na classificação dos termos do léxico da aplicação em artefatos de sistemas. A implementação dessa fase é relativamente simples. Ao classificar um item do léxico como artefato, um registro na tabela correspondente é criado e o campo nome é preenchido com o termo selecionado. Vale lembrar que em cada tabela, correspondente a um artefato, existe um campo identificador que é formado pelo *lemma*[5] do campo nome. A Figura 3 mostra como esta etapa é suportada pelo protótipo ERS-EDITOR e a Figura 4 mostra como o registro da seção correspondente ao artefato é preenchido.

[4]ngramas são expressões formadas por mais de uma palavra

[5]*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

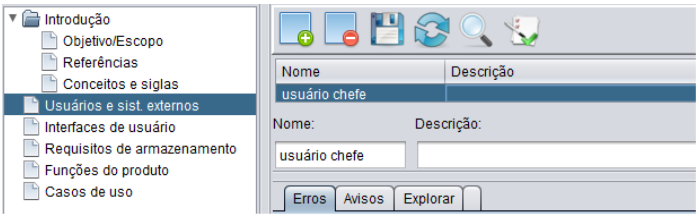

Figura 4: Itens da seção selecionada

C. Escrita de requisitos

O objetivo dessa fase é conduzir o analista de requisitos a escrever sentenças não ambíguos, consistentes, semanticamente corretas e com suporte automatizado para extração de modelos e artefatos de projeto. Para isso, faz-se o uso de LNC's. Schwitter [24] 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. Para atingir esse objetivo, define-se uma Gramática Livre de Contexto para a LNC da seção *Funções do produto* e faz-se uso do recurso de previsão de edição *intellisense*. Esse recurso ajuda o processo de escrita e impõe as restrições da LNC [9]. Detalharemos, a seguir, a implementação do *intellisense* para a seção *Funções do produto*.

Como exemplo, será utilizado o trecho da gramática que reconhece o **Padrão 1**: "O sistema deve permitir" <*usuário*> $(<$ *ação* $><$ *requisito_de_armazenamento* $>)^{+}$, descrito na Seção II-A. A Figura 5 mostra o trecho da gramática.

```
sentenca                    : padrao_1 usuario (acao requisito_de_armazenamento)+
                            ;
padrao_1                    : ARTIGO SISTEMA VERBO_PADRAO PERMITIR
                            ;
usuario                     : locucao_substantiva
                            ;
acao                        : VERBO
                            ;
requisito_de_armazenamento  : locucao_substantiva
                            ;
locucao_substantiva         : SUBSTANTIVO PREPOSICAO? (SUBSTANTIVO? | ADJETIVO?)
                            ;
```

Figura 5: Trecho da gramática que reconhece o padrão tipo 1

A gramática foi definida 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 [25]. 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 '+' indica que o termo/expressão deve ocorrer pelo menos uma vez e o símbolo '|' é equivalente ao "ou" lógico. Para obter a predição do que será escrito implementou-se dois tipos de associações entre as regras da gramática e os elementos do contexto, a saber: *i*) a regra <*usuario*> com a seção *Usuários e sistemas externas* e a regra <*requisito_de_armazenamento*> com a seção *Requisitos de armazenamento*; e *ii*) a regra <*ação*> com a lista de ações do léxico da aplicação.

O recurso é acionado quando as telcas "ctrl" + "espaço" são pressionadas de forma sequencial, a partir daí, o próximo

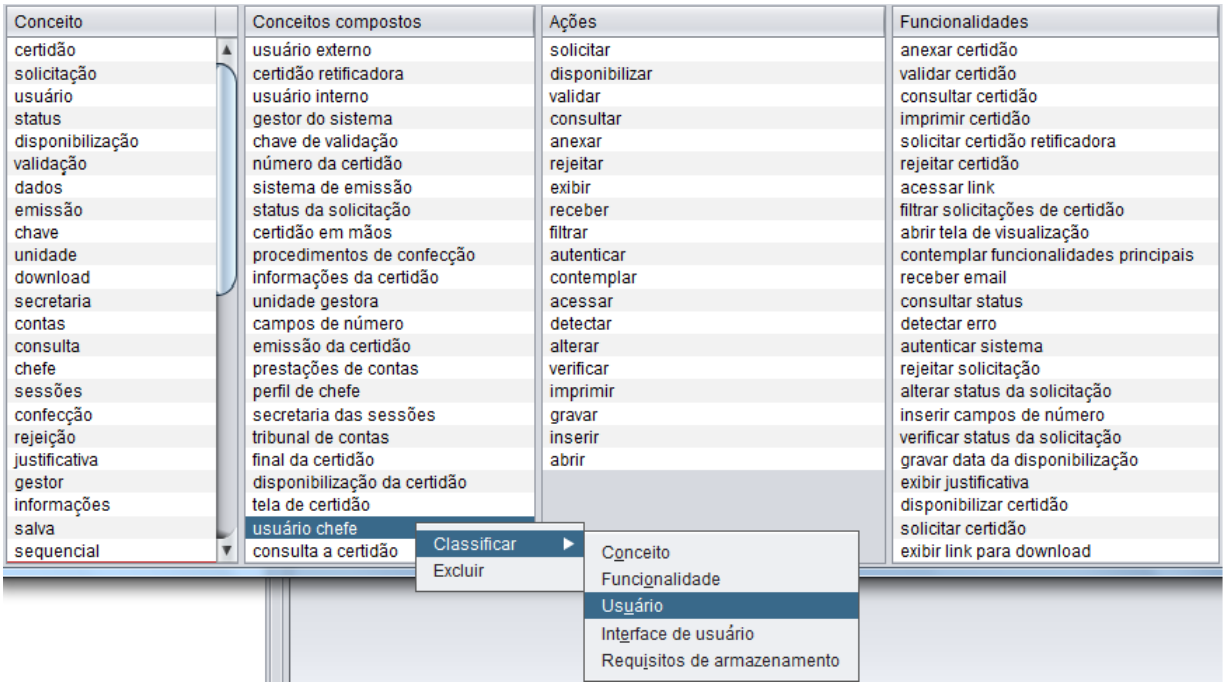

Figura 3: Léxico do domínio e classificação semântica

elemento é determinado através das seguintes ações:

1) **Etiquetar sentença**: Essa etiquetagem ocorre em dois níveis:
   - **Nível gramatical**: Nesse nível são etiquetados os verbos, substantivos, preposição e adjetivos respectivamente como os terminais VERBO, SUBSTANTIVO PREPOSICAO e ADJETIVO;
   - **Nível do léxico da LNC**: O artigo "O" é etiquetado como o terminal ARTIGO. 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. A sentença deve iniciar com esse padrão para ser reconhecida pela LNC.
2) **Analisar sentença**: Após a etiquetagem a sentença 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 a essa regra é feita e o resultado é exibido para que o analista de requisitos escolha qual elemento será adicionado à sentença.

A Figura 6 ilustra um exemplo no uso do *intellisense*, nele o analista de requisitos digita a sentença "O sistema deve permitir ao" e aciona o recurso, após a *tokenização* a sentença é etiquetada e a próxima regra identificada, que neste caso é a regra <*usuario*>, então é executada uma consulta aos itens da seção *Usuários e interfaces externas* e uma lista contendo os

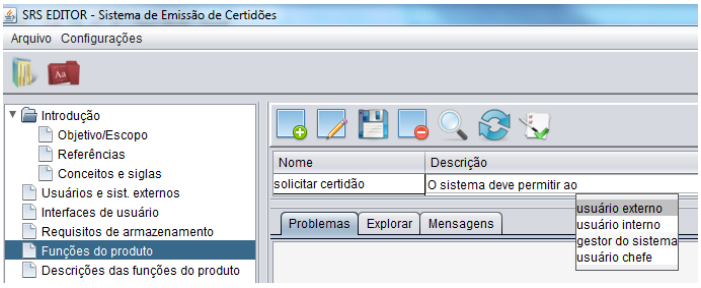

Figura 6: Uso do recurso *intellisense*

usuários cadastrados é exibida. É importante observar que este recurso só permanecerá disponível se o analista de requisitos escrever sentenças de acordo com a LNC definida para seção.

D. Verificação de inconsistências

O principal objetivo dessa fase é verificar se sentenças estão escritas de forma não ambíguas, consistentes, semanticamente corretas e com suporte automatizado para extração de modelos e artefatos de projeto, além de verificar a completude do documento de ERS, ou seja, se todos os termos importantes do domínio são definidos e referenciados no documento. Para alcançar esse objetivo as seguintes verificações são implementadas:

1) Verificar se as sentenças estão escritas sintaticamente conforme as LNC's definidas e semanticamente corretas de acordo com classificação da fase 2 - Definição semântica;
2) Verificar se todos os elementos reconhecidos pelos padrões linguísticos da Tabela III fazem parte do léxico e estão classificados;

3) Verificar se todos os elementos das seções são definidos e referenciados na seção *Descrições de casos de uso*.

A seguir será detalhada a implementação da verificação de inconsistências para a seção *Funções do produto* e como as mensagens de erro e avisos são exibidas.

1) Erro sintático e semântico: Inicialmente, ocorrem os processos de *tokenização* e etiquetagem da sentença conforme explicado na Seção III-C, depois o *parser* verifica se o inicio da sentença está de acordo com algum dos padrões explicados na Seção II-A, se não, um aviso de que a sentença a está fora do padrão do requisito é gerada conforme ilustrado na figura 7.

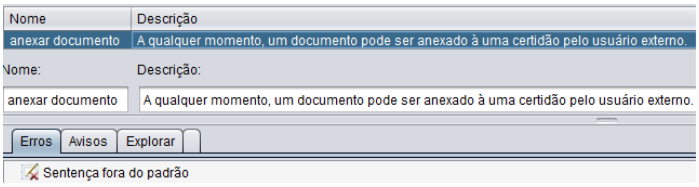

Figura 7: Aviso de sentença fora do padrão

Após o passo anterior validado, o *parser* verifica 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 8 mostra um exemplo para a sentença "O sistema deve permitir ao solicitar uma certidão", neste caso onde era esperado um SUBSTANTIVO foi encontrado o VERBO "solicitar".

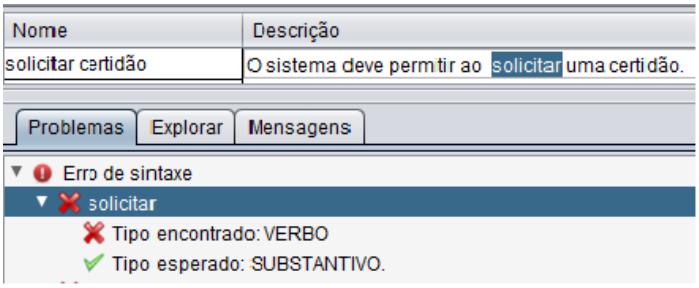

Figura 8: Erro de sintaxe

Caso o passo anterior seja validado 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 subseção II-A, 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, se essa classificação for diferente da classificação na sentença informada pelo *parser*, a mensagem de erro semântico é gerada. A Figura 9 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 usuário e o que aparece é o termo "certidão" que foi classificado como requisito de armazenamento.

2) Avisos: Como explicado na Seção II-B os avisos são gerados de forma independente da verificação dos erros. Após

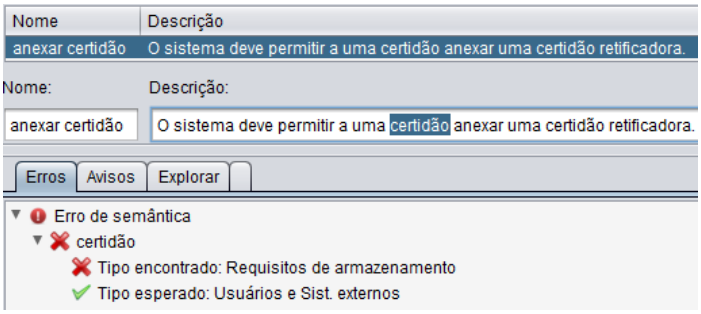

Figura 9: Erro de semântica

os processos de *tokenização* e etiquetagem da sentença o *parser* identifica todos os termos em conformidade com padrões da Tabela III, para cada termo, é feita uma consulta nas seções do documento de ERS. 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"; se o termo for encontrado apenas no léxico ele é classificado como "Termo não classificado". A Figura 10 mostra o resultado da geração de avisos para a sentença "A secretaria deve enviar um email para o usuário externo quando a certidão estiver finalizada". Neste exemplo os termos "secretaria" e "email" fazem parte do léxico na aplicação mas não foram classificados na etapa 2 da metodologia e os termos "finalizada" e "enviar email" não foram definidos nas seções do documento e nem fazem parte do léxico da aplicação.

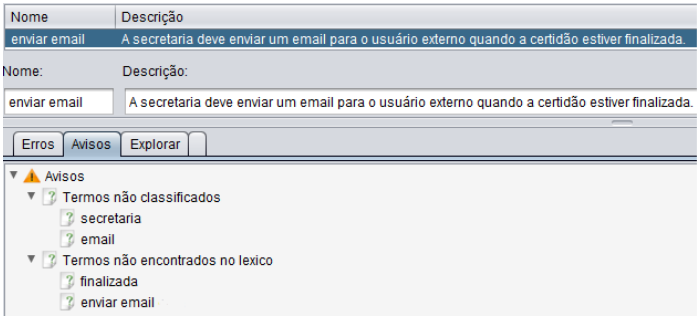

Figura 10: Avisos

Outro recurso oferecido pelo protótipo é a visualização da classificação dos artefatos encontrados na sentença. Esse recurso consiste em identificar os termos em conformidade com algum dos padrões da Tabela III e apresentá-los de acordo com sua classificação nas seções do documento de ERS. É implementado como descrito a seguir. Após os processos de *tokenização* e etiquetagem da sentença o *parser* identifica todos os termos em conformidade com padrões da Tabela III, para cada termo, é feita uma consulta nas seções do documento de ERS. A partir do resultado dessa consulta é montada uma estrutura conforme a classificação dos termos. A Figura 11 mostra o recurso para a sentença "O sistema deve permitir ao usuário externo solicitar uma certidão negativa". Neste documento de ERS, o termo "usuário externo" foi classificado como elemento da seção *Usuários 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. Não foram encontrados termos não classificados no documento de ERS e termos não encontrados no léxico da aplicação. O recurso está disponível para todas as sentenças escritas pelo analista de requisitos.

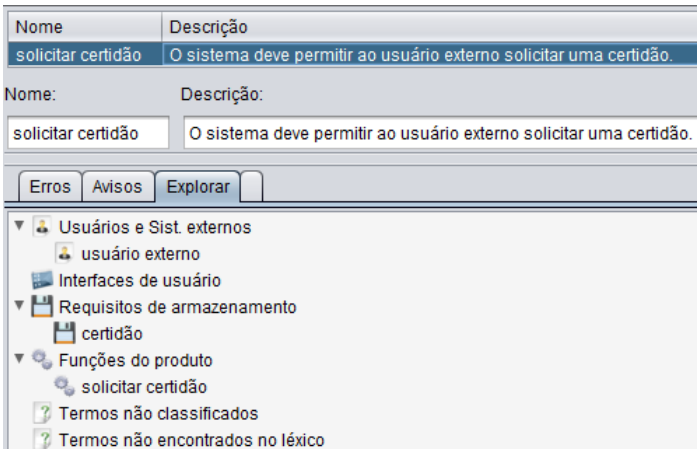

Figura 11: Classificação dos artefatos da sentença

IV. Resultados preliminares e Discussões

Como avaliação preliminar, a metodologia foi utilizada na identificação dos artefatos de modelagem do Sistema de Emissão de Certidões - SEC[6], desenvolvido no Tribunal de Contas do Estado do Piaui - TCE-PI. Este 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 sistemas e o de analista do domínio por um funcionário responsável pela confecção das certidões negativas, que até então eram elaboradas de forma manual.

Na execução da etapa 1 - Análise do domínio, 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 3. O protótipo selecionou 88 termos relevantes.

Após isso, executou-se a etapa 2 - Definição da semântica 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 artefatos conforme sua interpretação; 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 artefatos de modelagem, conforme mostrado na Figura 3. Nesse método, orientou-se que cada termo deveria ser classificado apenas como um artefato.

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

Dos 88 termos resultantes da etapa 1 - Análise do domínio, 29 foram identificados como falso-positivos com a utilização de M2. Durante a execução do método, 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 contexto da aplicação. Após a desambiguação restaram 58 termos, sendo que 23 deles continuaram como falso-positivos. A Tabela IV mostra o resultado da classificação utilizando-se M1, M2, os artefatos encontrados somente em M1 (**S1**), somente em M2 (**S2**), e a coluna total aceito que indica o número de artefatos aceitos pelo analista de requisitos após a análise dos resultados. Por exemplo, a primeira linha da tabela indica que através de M1 foram identificados 8 casos de uso; através de 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.

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 desapercebidos com a utilização do M1. Outro fato importante é que alguns artefatos identificados apenas no M1 devem-se a problemas pontuais que serão objetos de estudo para melhoria da metodologia, são eles: 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; classificação errada do etiquetador, o único método não identificado por M2 foi o termo "emitir", pois não foi classificado como verbo pelo *TreeTagger* e por isso não foi selecionado pelo algoritmo de construção do léxico da aplicação; e a recomendação de que o analista de requisitos deveria classificar cada termo como apenas um artefato em M2 foi o motivo da falta de identificação das interfaces de usuário "solicitar certidão", "consultar certidão" e "validar certidão",l visto que estes termos foram classificados como casos de casos de uso e interfaces de usuários quando da utilização do M1.

Tabela IV: Resultados

Artefatos	**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	12	7	5	0	12
Interfaces de usuário	4	1	3	0	4

V. Trabalhos Relacionados

Na literatura pesquisada, foram encontrados alguns trabalhos com objetivos semelhantes ao apresentado nesta proposta. A seguir, discutiremos aqueles que consideramos mais relevantes.

Miriam [10] 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, essas 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 [26]. 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 neste artigo utiliza técnicas de mineração de textos para a criação de um léxico da aplicação e faz uso de uma lista de sufixos para reconhecimento de atores em descrições de casos de uso.

Palomares et al. [6] descreve 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. 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.

Ferreira e Silva [4] apresentam o *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 ER. Esta dissociação permite lidar com requisitos como itens de "caixa-branca", possibilitando uma compreensão mais profunda a nível semântico. Assim, o RSLingo permite a automação de tarefas de verificação que previnem problemas de qualidade de requisitos e estabelece uma base para integração de RE com o paradigma *Model-Driven* [27] 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* [18] e o algoritmo da Distância de Levenshtein [28] para a extração automática de informações [29] e semântica dos conceitos do domínio do sistema. As semelhanças entre o RSLingo e a metodologia proposta neste artigo são: *(i)* uso de padrões linguísticos para capturar conceitos específicos do domínio do sistema, porém, enquanto o RSLingo espera que o especialista do domínio escreva o requisito nos padrões linguísticos definidos pelo arquiteto do sistema, a metodologia proposta neste artigo guia o analista de requisitos a escrever na estrutura da LNC definida para o requisito; *(ii)* a 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; e *(iii)* extração automática de informações, a metodologia proposta também utiliza técnicas de PLN *chunk parsing* através do uso de um *parser* para a LNC definida para os padrões linguísticos, além disso, a estrutura do ERS proposta reforça a semântica do domínio, o que permite uma compreensão semântica mais profunda do que aquela sugerida na abordagem RSLingo.

Esteca et. al [3] propõe a incorporação de um módulo de apoio à especificação de requisitos a uma ferramenta Web de apoio à ER chamada Ferramenta de Suporte à Elicitação de Requisitos (FSER), com o intuito de contribuir para a criação de documentos ERS completos, baseados em um padrão bem definido e amplamente utilizado na indústria de *software*, IEEE Std. 830/1998 [1] com o objetivo de melhoria da qualidade dos produtos de *software* obtidos. 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 ERS. 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* [30]. A única semelhança da metodologia apresentada neste artigo com a FSER é o uso do padrão IEEE Std. 830/1998 [1] como base para os documentos de ERS propostos. Porém, com estruturas diferentes. Além disso, a metodologia apresentada neste artigo utiliza sua estrutura para: definir a semântica do domínio; auxiliar as verificações de consistência; e auxiliar a identificação dos artefatos de modelagens, tudo isso com o apoio da ferramenta ERS-EDITOR.

VI. Conclusões e Trabalhos Futuros

Neste trabalho, foi apresentada uma metodologia para a escrita de documentos de ERS completos, consistentes e não ambíguos; e com possibilidade de suporte automatizado em seu conteúdo escrito em linguagem natural. Para auxiliar a metodologia, a ferramenta ERS-EDITOR está sendo desenvolvida com o objetivo de automatizar suas etapas, isso reduz o esforço humano e torna as atividades de ER menos propensas a erros.

Avaliou-se as etapas 1 - Análise do domínio e 2 - Definição da semântica do domínio através do Sistema de Emissões de Certidões, em desenvolvimento no Tribunal de Contas do Estado do Piauí - TCE-PI. Os resultados preliminares indicam que a proposta é bastante promissora, entretanto, a fim de dar

maturidade à metodologia e avaliá-la de forma completa, tem-se as seguintes atividades como trabalhos futuros:

1) Desenvolvimento de um módulo para auxilio automatizado à desambiguação de termos do léxico;
2) Uso de sintagmas nominais para redução do número de falsos-positivo. Baseado na construção de um Léxico Ampliado da Linguagem - LAL, descrito por Leite e Franco [26], deseja-se associar os artefatos 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;
3) Definir a LNC para a seção 6 - Descrições de casos de uso utilizando-se como modelo o trabalho de Toro e Bernárdez [5];
4) Avaliar a metodologia proposta em um ambiente controlado.

REFERÊNCIAS

[1] "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998, pp. 1-140*, 1998.

[2] I. Sommerville, S. S. S. Melnikoff, R. Arakaki, E. d. A. Barbosa, and K. Hirama, *Engenharia de software*. São Paulo: Pearson Prentice Hall, 2008.

[3] A. M. N. Esteca, A. Simonato, R. C. G. de Souza, C. R. Valencio, R. E. Garcia, M. L. Tronco, and V. D. A. Borges, "Computational support for the process of software requirement specification," *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, pp. 1–10, Oct. 2012.

[4] D. de Almeida Ferreira and A. R. da Silva, "RSLingo: An information extraction approach toward formal requirements specifications," *2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pp. 39–48, Sep. 2012.

[5] A. D. Toro, B. B. Jiménez, A. R. Cortés, and M. T. Bonilla, "A requirements elicitation approach based in templates and patterns." in *WER*, 1999, pp. 17–29.

[6] C. Palomares, C. Quer, and X. Franch, "PABRE-Man: Management of a requirement patterns catalogue," *2011 IEEE 19th International Requirements Engineering Conference*, pp. 341–342, Aug. 2011.

[7] R. T. Anchiêta, R. F. de Sousa, and R. S. Moura, "Identifying of user interface elements from use case descriptions," in *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI), Medellin, Colombia, October 1-5, 2012*, 2012, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/CLEI.2012.6427184

[8] R. Juarez-Ramirez, C. Huertas, and S. Inzunza, "Automated Generation of User-Interface Prototypes Based on Controlled Natural Language Description," *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pp. 246–251, Jul. 2014.

[9] R. Schwitter, "Controlled natural languages for knowledge representation," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1113–1121. [Online]. Available: http://dl.acm.org/citation.cfm?id=1944566.1944694

[10] M. Sayão, "Verificação e validação em requisitos: Processamento da linguagem natural e agentes," Ph.D. dissertation, PUC-Rio, 2007.

[11] R. T. Anchieta, "Uso de PLN para extrair elementos de interface de usuário a partir de descrições de requisitos de softwares," Master's thesis, Universidade Federal do Piauí, 2014.

[12] W. de Padua Paula Filho, *Engenharia de software: fundamentos, métodos e padrões*. Livros Técnicos e Científicos, 2001. [Online]. Available: http://books.google.com.br/books?id=7s_vAAAACAAJ

[13] R. T. Anchiêta, R. F. de Sousa, and R. S. Moura, "Using NLP techniques for identifying GUI prototypes and UML diagrams from use cases," in *The 25th International Conference on Software Engineering and Knowledge Engineering (SEKE), Boston, MA, USA, June 27-29, 2013.*, 2013, pp. 48–53.

[14] R. Young, *The requirements engineering handbook.* Boston: Artech House, 2004.

[15] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *International Conference on New Methods in Language Processing*, Manchester, UK, 1994, pp. 44–49.

[16] A. Ratnaparkhi, "A maximum entropy part-of-speech tagger," in *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania, 1996.

[17] S. E. Lee and S. S. Han, "Qtag: introducing the qualitative tagging system." in *Hypertext*, S. Harper, H. Ashman, M. Bernstein, A. I. Cristea, H. C. Davis, P. D. Bra, V. L. Hanson, and D. E. Millard, Eds. ACM, 2007, pp. 35–36.

[18] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*, ser. O'Reilly Series. O'Reilly, 2009.

[19] T. Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Bookshelf, 2007.

[20] V. M. Orengo and C. Huyck, "A Stemming Algorithm for Portuguese Language," in *Proc. of Eigth Symposium on String Processing and Information Retrieval (SPIRE 2001) - Chile*, 2001, pp. 186–193.

[21] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1983.

[22] H. Luhn, *The Automatic Creation of Literature Abstracts (auto-abstracts)*. IBM Research Center, International Business Machines Corporation, 1958. [Online]. Available: http://books.google.com.br/books?id=m59RGwAACAAJ

[23] G. Zipf, *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press, 1949. [Online]. Available: http://books.google.com.br/books?id=1tx9AAAAIAAJ

[24] R. Schwitter, "natural languages. technical report," *Centre for Language Technology, Macquarie University*, 2007.

[25] *ISO/IEC 14977:1996 Information Technology - Syntactic Metalanguage - Extended BNF*, Std., 1996.

[26] J. Leite and A. Franco, "A strategy for conceptual model acquisition," in *Proceedings of IEEE International Symposium on Requirements Engineering*. IEEE Computer Society Press, 1993, pp. 243–246.

[27] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1109/MC.2006.58

[28] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.

[29] H. Cunningham, "Information extraction, automatic," *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

[30] C. P. D. Team, "Cmmi for systems engineering/software engineering, version 1.02, staged representation (cmmi-se/sw, v1.02, staged)," Tech. Rep., 2000.