Knowledge Extraction from the Language Extended Lexicon Glossary Using Natural Language Processing

Extracción de conocimiento a partir del glosario Léxico Extendido del Lenguaje utilizando Procesamiento de Lenguaje Natural

Leandro Antonelli[1];
Mario Lezoche[2];
Juliana Delle Ville[3]

[1] Universidad Nacional de La Plata, Buenos Aires-Argentina, leandro.antonelli@lifia.info.unlp.edu.ar
[2] Université de Lorraine, Nancy-France, mario.lezoche@univ-lorraine.fr
[3] Universidad Nacional de La Plata, Buenos Aires-Argentina, juliana.delleville@lifia.info.unlp.edu.ar

ISSN-p: 0123-7799
ISSN-e: 2256-5337

Vol. 27, **no.** 59, e2917, 2024

Recibido: 12 noviembre 2023
Aceptado: 22 abril 2024
Disponible: 29 abril 2024

How to cite / Cómo citar
L. Antonelli, M. Lezoche, and J. Delle Ville, "Knowledge Extraction from the LEL Glossary Using Natural Language Processing," *TecnoLógicas*, vol. 27, no. 59, e2917, Apr. 2024. https://doi.org/10.22430/22565337.2917

Abstract

Successful software development requires good specifications, that is, correct, consistent, and unambiguous requirements. Natural language artifacts are the most used tools for writing specifications; nevertheless, the use of natural language can introduce many defects such as ambiguity, vagueness, and generality. In light of this, the purpose of this paper is to propose an approach for writing good specifications and extracting knowledge from them. The proposed approach uses a particular lexicon, the Language Extended Lexicon (LEL) glossary, and suggests rules for extracting knowledge (concepts, attributes, and relationships) from it. This paper also presents a prototype that implements and supports the proposed approach, as well as a preliminary evaluation of the approach. In conclusion, the proposed approach represents a theoretical contribution, which was evaluated using the System Usability Scale, and its evaluation results are promising, as it obtained the maximum score.

Keywords

Requirement specification, knowledge representation, natural language, extended lexicon, software development.

Resumen

Un desarrollo de software exitoso necesita buenas especificaciones, es decir, requerimientos correctos, consistentes y no ambiguos. Los artefactos en lenguaje natural son la herramienta más utilizada para escribir especificaciones; sin embargo, el uso del lenguaje natural puede introducir muchos defectos: ambigüedad, vaguedad y generalidad. Por lo tanto, el objetivo de este artículo es brindar un enfoque para escribir buenas especificaciones y extraer conocimiento de ellas. El enfoque propuesto utiliza un léxico particular, el glosario Léxico Extendido del Lenguaje (LEL). Y sugiere reglas para extraer conocimientos (conceptos, atributos y relaciones) de él. Este artículo también presenta un prototipo que implementa y apoya el enfoque propuesto, así como también una evaluación preliminar del enfoque. En conclusión, el enfoque propuesto representa una contribución teoría, que fue evaluada utilizando el System Usability Scale y sus resultados son prometedores, ya que obtuvo la calificación máxima.

Palabras clave

Especificación de requerimientos, representación del conocimiento, lenguaje natural, léxico extendido, desarrollo de software.

1. INTRODUCTION

Good requirements are crucial to successful software development. If requirements are not well-defined, the resulting software application will not satisfy the needs, desires, and expectations of the stakeholders. Moreover, poor requirements can lead to a lot of reworks to modify the software application to truly meet such needs. According to estimates, errors made during the requirements stage can cost up to 200 times more if they are discovered after the software is delivered to the client [1].

The expression "good requirements" encompasses a broad range of characteristics that are desirable for software specifications. Development processes may be agile with a minimum of documentation or classic with orthodox software requirements specifications spanning hundreds of pages. In either case, the specifications must be correct, consistent and unambiguous. Although satisfying all these characteristics, known as quality attributes, can be challenging due to the huge effort involved in correcting errors, striving for the best possible specification is important.

Natural language is the most common tool for writing requirements [2]. Usually, requirements are specified by requirements engineers or business analysts, who must collaborate with stakeholders to gather the requirements and extract the necessary knowledge for the software application. The software specifications must be easy to understand and precise so that the development team can use them effectively. Natural language specifications are understandable for both sides [3]. However, the use of natural language can introduce many defects, such as ambiguity, vagueness, and generality [4].

Formal reasoning involves inferring knowledge from a specification. For example, considering a specification in the agricultural field that states that "plants need water, and tomato is a plant," it can be inferred that "tomatoes need water." To allow formal reasoning, the requirements must be free from defects introduced by natural language [5]. Moreover, formal reasoning makes it possible to automate many tasks to obtain good requirements [6], [7]. Consistency is one of the quality attributes of specifications, meaning that there is no contradiction in any given pairs of requirements. Hence, from a specification that states that "plants need water, tomato is a plant, and tomato does not need water," it can be inferred that "tomato needs water, and tomato does not need water." This contradiction indicates that the specification is not consistent.

The Language Extended Lexicon (LEL) is a semi-structured artifact designed to describe the language of a domain. In their study, [8] reported the use of LEL in complex domains such as healthcare. This glossary uses natural language to describe a domain's language and categorizes the vocabulary into four categories: subjects, objects, verbs, and states. Each term is described through two attributes: notion and behavioral responses. Thus, the LEL is more than a simple glossary; it represents a domain's knowledge captured through its language.

The notion of a kernel sentence first emerged in 1957 through the work of the linguist [9] and was further explored by the linguist [10]. Kernel sentences, also referred to as basic sentences, are declarative constructions that always use active voice and affirmation and contain only one verb. In this study, it is argued that the use of kernel sentences in the description of the LEL glossary helps obtain good specifications.

A well-defined specification provides a global overview and, through formal reasoning, makes it possible to obtain complete and consistent specifications and infer new knowledge. However, prior to engaging in formal reasoning, it is imperative to synthesize the knowledge encapsulated within the language. This knowledge obtained can be used in various ways. In the case of the LEL, it synthesizes the glossary, offering a global overview and facilitating the inference of new knowledge. This is very important for developing complete and consistent specifications, especially when using natural language.

In light of the above, this paper proposes an approach to extract knowledge (concepts and relationships) from the LEL glossary. Importantly, the inference of new knowledge is beyond the scope of this proposal. It also introduces a prototype tool that implements the natural language algorithms described in the proposed approach to facilitate its application. Additionally, a preliminary evaluation of the approach is provided. Although this paper is an extension of a previous study [11], its novelty lies in the modifications made to the entire approach (including the removal of one step) and the incorporation of seven new rules for knowledge extraction (compared to the four rules proposed in the previous version of the approach).

The rest of this paper is organized as follows. Section 2 provides some background on the LEL glossary and kernel sentences. Section 3 reviews some related studies. Section 4 outlines the proposed approach in detail. Section 5 describes the tool that supports the process. Section 6 presents the approach's preliminary evaluation. Finally, Section 7 offers some concluding remarks.

2. BACKGROUND

This section provides an overview of the LEL glossary and kernel sentences.

2.1 The Language Extended Lexicon (LEL)

The Language Extended Lexicon (LEL) serves as a glossary delineating the language specific to an application domain, which should be distinguished from the software application itself. It operates on a simple principle: "understand the language of the problem without worrying about the problem" [12]. In this glossary, language is encapsulated through symbols, which may take the form of words or short expressions. Each symbol must be defined by two attributes: notion and behavioral responses. Notion captures the denotation, encompassing the intrinsic and substantial traits of the symbol, while behavioral responses describe its connotation, showing the relationship between the symbol and other symbols (see Figure 1). Each symbol in the LEL glossary falls into one of the following four groups: subject, object, verb, or state. This categorization helps requirement engineers in describing attributes. Table 1 presents each category, along with its characteristics and descriptions guidelines.

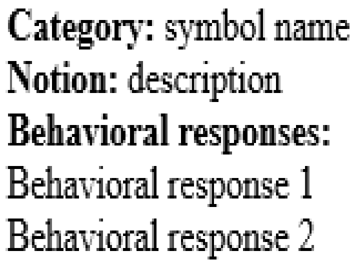
Category: symbol name
Notion: description
Behavioral responses:
Behavioral response 1
Behavioral response 2

Figure 1. Template for describing a symbol in the LEL glossary. Source: [12].

Table 1. Template for describing symbols in the LEL glossary according to their category. Source: [12].

Category	Notion	Behavioral responses
Subject	Who is he/she?	What does he/she do?
Object	What is it?	What actions does it receive?
Verb	What goal does it pursue?	How is the goal achieved?
State	What situation does it represent?	How is this situation checked?

As an example, consider a banking domain where clients open accounts to save money. Clients can deposit money into their accounts and also withdraw it as long as their account balance is positive. In this simple example, multiple symbols need to be defined in the LEL glossary to capture the language of the banking domain. For instance, “client” should be categorized as a subject, “account” as an object, “deposit” and “withdraw” as verbs, and “positive balance” as a state. Following the template in Table 1, the symbols “client,” “account,” “withdraw,” and “positive balance” are described in Table 2.

As observed in Table 2, the behavioral responses of the symbol “client” describe only two responsibilities: opening an account and withdrawing money. Nonetheless, depositing money is also a responsibility that is not included. Similarly, the behavioral responses of the symbol “account” only describe the bank’s responsibility to calculate the interest. To complete the LEL glossary, the client’s responsibilities—i.e., opening, depositing, and withdrawing—should also be mentioned. In the case of the verb “withdraw,” the behavioral responses describe the process performed by the bank, which involves first reducing the balance and then checking if the resulting balance is positive. If so, the bank provides the money to the client; otherwise, the balance is reset to its original amount, and no money is given to the client. Finally, the behavioral responses of the state “positive balance” describe in detail what “positive” means in this context—it means that the account balance is greater than or equal to zero.

Table 2. Symbols in the LEL glossary for the banking example. Source: Own work.

Category	Notion	Behavioral responses
Subject: Client	Person who saves money in the bank. A client holds accounts. A client may be regular or premium.	The client opens an account. The client withdraws money from the account.
Object: Account	Instrument used by the client to save money.	The bank calculates the interest on the account.
Verb: Withdraw	Act of extracting money from an account.	The bank reduces the account balance. If the account has a positive balance, then the bank provides the money to the client. If the account does not have a positive balance, then the bank resets the account balance to its original amount.
State: Positive balance	Situation in which the account has money in it.	The account balance is greater than or equal to zero.

2.2 Kernel sentences

A kernel sentence is a simple, active, affirmative, and declarative sentence that contains only one verb. Devoid of any mood, this basic sentence is termed “kernel” because it serves as the foundation for more intricate sentence structures. Figure 2 illustrates two examples of kernel sentences. The first sentence states that a subject (“client”) performs an action ("opens") on a specific object ("account"). The second sentence, although following the same structure, albeit describing a different action ("deposits"), is slightly more complex because it states that the object "money" is deposited into the object "account."

Figure 3, for its part, shows two sentences that are not kernel sentences. The first sentence ("The client deposits money and calculates the interest on the account") contains two verbs, which disqualifies it as a kernel sentence. Moreover, the sentence is partially correct because although it is true that the client deposits money into the account, it is the bank, not the client, that calculates the interest. This is why simple (kernel) sentences are suggested to describe specifications; a simple sentence is more likely to be either completely true or false. The second sentence ("An account is opened") is written in passive voice, and the kernel sentences should use active voice. Since it is written in passive voice, the subject of the sentence ("the account") is not the agent performing the action. In fact, the real subject performing the action is not mentioned. According to the provided example, the real subject should be "the client," though it could also be the company for which the client works, which decided to open an account to pay salaries. These two simple examples illustrate how the writing style of kernel sentences can help improve specifications.

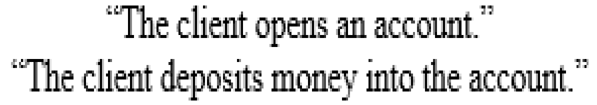
"The client opens an account."
"The client deposits money into the account."

Figure 2. Kernel sentences. Source: Own work.

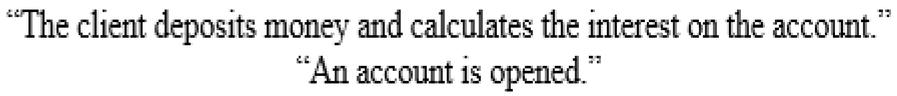
"The client deposits money and calculates the interest on the account."
"An account is opened."

Figure 3. Non-kernel sentences. Source: Own work.

3. RELATE WORKS

These authors [13] proposed a method for transforming specifications into source code. Their approach systematically extracts the formal semantics of the ARM instructions from their natural language specifications. Although ARM is based on a RISC architecture with a relatively small number of instructions, there are various series, including Cortex-A, Cortex- M, and Cortex-R. Moreover, the approach proposed by the authors uses translation rules enriched by sentence similarity analysis.

Other proposals have delt with more abstract representations of knowledge, such as database tables, schemas, or object-oriented designs [14], for instance, identifying the schema and relationship in a database from natural language requirements specifications. Their proposed approach starts with identifying table schemas and their properties and then uses adjectives to determine the Primary Key attribute. In addition, it uses rules as well as a machine learning component.

For their part [15], suggested specifying requirements using a template that embodies the concepts of a decision-making process and provided a linguistic framework for acquiring analytical queries. This proposed framework simplifies the automated analysis of queries without overly constricting writing styles.

In their study, [16] introduced an approach and tool for extracting concepts by grammatically analyzing English requirements, eliminating the need for specialized ontology references. These concepts can be subsequently represented in a class model, serving as a foundation for object-oriented problem analysis. The proposed method employs Natural Language Processing (NLP) techniques to identify parts of speech and segment sentences into phrases. Additionally, it uses the WordNet dictionary to identify familiar concepts and determine relationships among them. These authors [17] focused on formal or semi-formal model descriptions of functional requirements in the context of a model-driven engineering approach. In this approach, a platform-independent model is used to derive the source code of a system either automatically or semi-automatically. According to the authors, most approaches use a predefined set of rules that impose several restrictions on how specifications are written. Thus, they proposed using a deep learning approach.

Furthermore, some authors have addressed conceptual models earlier in the software development lifecycle, with some approaches relying on syntactic rules, while others incorporating semantic enrichment. [18] for example, proposed an approach to support requirements elicitation and analysis employing masked language models. These models excel at capturing contextual information from natural language sentences and transferring this knowledge to NLP tasks. The authors stated that masked language models can predict the most probable missing word in a sentence, enabling the exploration of domain concepts embedded within word embeddings. Their proposed methodology involves extracting domain knowledge from user-generated scenarios through typed dependency parsing techniques. Additionally, they investigated the effectiveness of a supplementary approach using BERT-based masked language models to identify entities and their associated attributes, thereby constructing a domain model from a single-word seed term.

In their study [19] introduced a method for extracting concepts from unstructured Polish texts, placing particular emphasis on morphological forms. Given the highly inflected nature of Polish, the identified names had to undergo transformation in accordance with Polish grammar rules. The proposed approach involves specifying transformation patterns using a straightforward annotation language. Users prepare annotations, which are then compiled into transformation rules. During the concept extraction process, the input document is segmented into sentences, and the rules are applied to word sequences within these sentences. The identified strings forming concept names are subsequently aggregated at different levels and assigned scores.

These authors [20], for their part, presented an approach that uses an LSTM-CRF model to extract requirement entities. Their methodology comprises four phases: (i) model construction, which involves building an LSTM-CRF model and an isomorphic LSTM language model for transfer learning; (ii) LSTM language model training, which focuses on capturing general knowledge and adapting it to the requirement context; (iii) LSTM-CRF training, which entails training the LSTM-CRF model with transferred layers; and (iv) requirement entity extraction, which involves applying the trained LSTM-CRF model to new requirements to automatically extract their respective entities. [21] proposed an approach that selects common verbs from software requirement specification documents in the e-commerce domain and constructs semantic frames for these verbs. The author manually labeled selected sentences using the results as training examples for machine learning and refined the parsing output from the Stanford Parser. Throughout the labeling process, the author adopted a sequential approach, using previously labeled results to dynamically construct features for identifying subsequent semantic roles.

In their study, [22] introduced a method that models domain knowledge through an ontology. Their proposed method of ontology population focuses on identifying instance properties from texts. The authors employed extraction rules automatically acquired from a training corpus and bootstrapping terminology to identify instance properties represented by triples of terms. These rules use lexical, syntactic, and semantic levels of analysis and are generated from recurrent syntactic paths between terms denoting instances of concepts and properties.

Finally, some approaches in the literature aim to obtain knowledge representations in a manner similar to our proposal. For example [23], proposed a knowledge extraction method that relies on an explicit representation of the content of natural language requirements in the form of a semantic relation graph. This approach is entirely automated and includes an NLP pipeline to convert unrestricted natural language requirements into a graph. Natural language is segmented into distinct parts and relationships are established between them based on their semantic connections.

For their part [24], explored a set of state-of-the-art pre-trained general-purpose and domain-specific language models to extract knowledge triples for metal-organic frameworks. The authors created a knowledge graph benchmark with seven relationships for 1248 published synonyms of metal-organic frameworks.

4. PROPOSED APPROACH

This section outlines the proposed approach. It begins with an overview of its two steps: (i) LEL glossary description and (ii) knowledge extraction. Then, each step is discussed in detail.

4.1 Our approach in a nutshell

The proposed approach aims to provide a synthesized representation of the knowledge captured through the language of the application domain in the LEL glossary. It uses knowledge as input and produces a knowledge representation as output. The necessary knowledge can be obtained from different sources, such as directly from stakeholders through interviews or other types of techniques (e.g., focus groups and questionnaires).

The approach is intended for use by a requirements engineer or a business analyst who initially describes the LEL glossary and then synthesizes the knowledge. Importantly, the approach is not limited to a single person (one requirements engineering or one business analyst); multiple individuals can participate in the process. In other words, many requirements engineers can collaborate in describing the LEL glossary. Additionally, domain experts among stakeholders can directly contribute to writing symbols for the LEL glossary. Although different individuals can have varied points of view and writing styles, the guidelines provided in the first step of the approach help ensure a consistent and coherent LEL glossary.

In a previous version of this approach, a step was included between LEL glossary description and knowledge extraction to guide the revision of the LEL glossary. This revision was initially considered vital when more than one person participated in the construction of the LEL. Nevertheless, based on recent experience, the LEL glossary produced in the first step did not require extensive revision. As a result, the revision step has been removed in this new version of the approach.

The proposed approach can be applied in various situations: (i) to summarize documents created by others, (ii) to consolidate and summarize documents created by only one analyst from multiple sources, and (iii) to collaboratively consolidate and summarize documents created by a group of analysts or experts. It consists of two main steps: (i) LEL glossary description, and (ii) knowledge extraction. The first step offers guidelines to help individuals describe the LEL glossary. The second step outlines rules for extracting knowledge from the specified LEL glossary, which is described in terms of concepts, relationships, cardinality, and attributes. This represents the main difference from the previously published approach [11]. Figure. 4 provides an overview of the proposed approach.

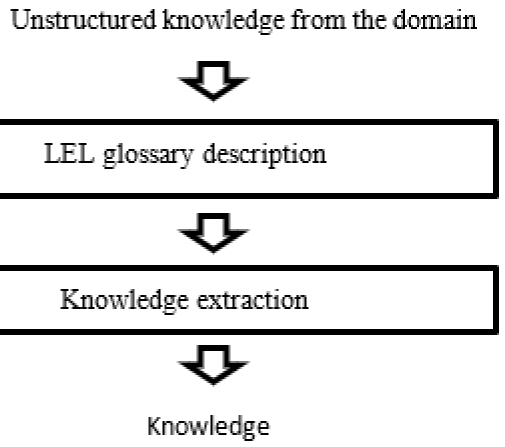

Figure. 4. Overview of the proposed approach. Source: Own work.

Considering the LEL glossary described in Table 2, it can be assumed that it was described in Step 1 of the proposed approach. If Step 2 is applied to that LEL, the concepts “client” and “account” will be connected through the relationship “withdraw.” Then, the entire relationship (“client,” “withdraw,” “account”) is restricted by the condition “positive balance.”

4.2 Step 1: Guidelines for writing the LEL glossary

This section outlines the six proposed guidelines for describing the LEL glossary. It is important to note that the construction of the LEL glossary is an iterative and incremental process. Initially, some symbols are identified and described, and then new symbols emerge because they are used in the description of the previously defined symbols. Thus, the proposed guidelines do not encompass the entire process of constructing the LEL glossary; rather, they complement the process.

Basically, the proposed guidelines emphasize the importance of using kernel sentences, referencing symbols that have already been defined, suggesting how to write compound sentences (using subordinate clauses like “if-then,” negation, and conjunctions like “and”), and incorporating state symbols in subordinate expressions. The rest of this section elaborates on each guideline and provides a series of examples.

Guideline 1. Sentences for describing behavioral responses must follow the principles of kernel sentences. Behavioral responses across all categories (subject, object, verb, and state) must be written according to the principles of kernel sentences, that is, featuring an explicit subject, only one verb, in turn followed by and an object that receives the action. Figure 2 shows examples of sentences written following this guideline; they describe the behavioral responses of the symbol “client” in Table 2.

Guideline 2. Terms used to describe behavioral responses must be defined in the LEL glossary. This guideline advises that the subject, verb, and object mentioned in the previous guideline should also be symbols already defined in the LEL glossary. For example, the symbol “client” defined in Table 2 has the following behavioral responses: “The client opens an account” and “The client withdraws money from the account.” If Table 2 is assumed to contain a complete description of the LEL glossary, the symbols “client,” “account,” and “withdraws” are defined. Nonetheless, as observed, the symbol “opens” is not defined, and, according to this guideline, it should be defined.

Guideline 3. Use subordination (if-then sentences) in the behavioral responses of verb symbols to describe conditions. For instance, consider the verb “withdraw” in Table 2. One of its behavioral responses states: “If the account has a positive balance, then the bank provides the money to the client.” This means that the bank only provides the money to the client when the account has a positive balance after considering the withdrawal amount. Thus, the balance is checked to ensure that it remains positive after the transaction. Importantly, this guideline does not contradict Guideline 1, as the use of subordination involves the use of kernel sentences. Although two kernel sentences are employed in the same sentence, they are properly associated.

Guideline 4. State symbols should be used in the condition of a subordinate sentence. Since state symbols describe situations, these situations should be used as conditions in subordinate sentences. Moreover, these conditions should follow the principles of a kernel sentence. For example, consider the verb “withdraw” in Table 2. One of its behavioral responses states: “If the account has a positive balance, then ...” Here, “positive balance” is a state symbol described in Table 2, and the expression “the account has a positive balance” conforms to the rules of the kernel sentences.

Guideline 5. Use negation to invert a condition or action. For example, the verb “withdraw” in Table 2 includes a behavioral response that states: “If the account does not have a positive balance, then ...” The use of negation can prevent certain behavioral responses. For instance, the behavioral responses of a subject describe all the actions that the subject can perform. In Table 2, the symbol “client” is described with the following behavioral responses: “The client opens an account” and “The client withdraws money from the account.” Generally, according to Guideline 1, which recommends the use of positive kernel sentences), behavioral responses should be affirmative. Nevertheless, it may sometimes be very important to explicitly state that a certain action is not possible, and negation can be used for such purpose. For example, if a bank only works with organizations and not with private clients, the responsibility of opening accounts falls to the organization, not the client. In such scenario, a behavioral response for a client might read: “The client does not open an account.”

Guideline 6. Use coordinating conjunctions (and, or) to join two subjects, objects, or kernel sentences. Although it is possible to write multiple kernel sentences, simpler and more natural sentences can be generated using conjunctions. For instance, instead of writing “The client deposits money into the account. The bank deposits money into the account,” this could be rewritten as “The client and the bank deposit money into the account.”

4.3 Step 2: Knowledge extraction

This section outlines a set of rules (inspired by [25]) for extracting knowledge from the LEL glossary following the guidelines described in the previous section.

- *Rule 1. Concepts are derived from symbols of the subject and object categories.*

From the LEL glossary described in Table 2, two concepts are identified: "client" and "account." The first is a subject, and the second is an object, but both are concepts.

- *Rule 2. Relationships are derived from symbols of the verb category.*

From the LEL glossary described in Table 2, one relationship is identified: "withdraw." According to the behavioral response of the symbol "client," this action ("withdraw") connects the concepts of "client" and "account."

- *Rule 3. Conditions are derived from symbols of the state category.*

From the LEL glossary described in Table 2, one condition is identified: "positive balance."

- *Rule 4. Relationships obtained through Rule 2 are restricted by conditions.*

In the LEL glossary described in Table 2, the state: "positive balance" establishes a condition. This state is mentioned in the verb "withdraw," which creates a relationship between "client" and "account." Therefore, the entire relationship ("client," "withdraw," "account") is restricted by the condition "positive balance."

- *Rule 5. Aggregations are derived from verbs like contains, has, includes, is composed by, and is defined by.*

In Table 2, the subject "client" is described as "A client holds accounts." The concepts "client" and "accounts" are obtained through Rule 1. Then, through Rule 2, the verb "has" establishes a relationship between "client" and "accounts." This, nonetheless, is an aggregation relationship rather than a general one.

- *Rule 6. Generalizations are derived from verbs such as is and may be.*

In Table 2, the subject "client" is described as "A client may be regular or premium." The concepts "client," "regular," and "premium" are obtained through Rule 1. Then, through Rule 2, the verb "may be" creates two relationships: one between "client" and "regular" and another between "client" and "premium." These, nevertheless, are generalization relationships rather than general ones.

- *Rule 7. Relationships with the structure "concept A" of "concept B" is "concept C."*

This rule is similar to Rule 6 in that it includes two concepts ("concept A" and "concept B") connected through the verb "is." However, the first part of the phrase ("concept A of") establishes the real relationship between "concept B" and "concept C." For instance, in the expression "The bank of the client is a federal bank," "bank," "client," and "federal bank" are concepts identified through Rule 1. Using Rule 7, the concepts "client" and "federal bank" are connected by means of the concept "bank."

- *Rule 8. Cardinality is defined by the target concept.*

This rule is related to Rules 2, 5, and 7. For instance, in the example provided in Rule 2, "The client withdraws money from the account," the target concept is "account," indicating that the client is connected to only one account. In the example provided in Rule 5, "A client holds accounts," the client is connected to multiple accounts. Finally, in the example provided in Rule 7, "The bank of the client is a federal bank," the target concept is singular ("federal bank"), suggesting that the client is connected to only one federal bank.

- *Rule 9. Multiple cardinality can be defined by indicators like many, each, all, every, some, and any.*

Rule 2 (and Rule 8) uses the example "The client withdraws money from the account," indicating that a client withdraws money from a single account. Conversely, the expression "The client withdraws money from each account" suggests that the client has multiple accounts. Thus, these indicators can be used to determine multiplicity in the relationships.

- *Rule 10. Attributes are derived from verbs like identified by, recognized by, contains, has, includes, is composed by, and is defined by.*

This rule may seem similar to Rule 5, but the difference is that, in Rule 5, the elements involved in the relationship (both sides of the verbs) should be identified as concepts through Rule 1. In Rule 10, the expressions should follow the structure "concept-verb-noun," where the concept is identified by means of Rule 1, the verb is one of the verbs listed in Rule 10, and the noun is not a concept. For instance, in the example "The client is identified by his/her social security number," "client" is a concept, but "social security number" is not, making it an attribute of "client."

- *Rule 11. Genitive cases suggest attributes.*

Expressions with the structure "noun A's noun B," where "noun A" is a concept identified by means of Rule 1 and "noun B" is not a concept, suggest that "noun B" is an attribute of "noun A." For example, in the expression "client's address," "client" is a concept, while "address" is not, making it an attribute of "client."

5. SUPPORTING TOOL

A software prototype was developed to facilitate the application of the proposed approach. The tool incorporates the rules outlined in Step 2 through algorithms based on natural language. It is a web application designed following a service-oriented architecture. Its core functionality and services were written in Python [26], while the web interface was developed with Django [27], and the APIs were constructed using Flask [28]. Python [26] was also employed for communication with the Spacy [29] and NLTK libraries [30] used for NLP. The application is responsive, allowing it to adapt to the device being used, including computers (desktop or laptop) or mobile devices (phones or tablets). Consequently, it offers versatility across different platforms and provides experts with various options to contribute to knowledge acquisition.

The developed prototype features two user roles: (i) project administrators and (ii) experts. Administrators can create projects, while experts actively contribute their knowledge to such projects. Moreover, the prototype acts as a framework because it can manage different types of artifacts based on natural language. Figure 5 illustrates how to configure the LEL glossary artifacts. The prototype can also be easily extended to perform an additional process to derive more information from the artifacts.

The rules proposed in Step 2 were implemented using the Spacy [29] and NLTK [30] libraries for NLP. Rules 1 through 4 involve checking the LEL glossary symbols and their usage in the descriptions using basic NLP tools like stemming and lemmatization functions. Rules 5, 6, 9, and 10 rely on dictionaries to identify specific words or expressions. Rules 7 and 11 employ matchers to detect particular structures. Finally, Rule 8 uses POS tagging to recognize whether a noun is singular or plural.

```
Class LelElement ():
symbol (length = 255, label “Symbol”)
category (length = 10, label “Category”)
Notion (length = 1000, label “Notion”)
BehavioralResponses (length = 1000, label “Behavioral responses”)
```

Figure 5. Configuration of artifacts. Source: Own work.

PlantUML [31] was used to visualize the model. Figure 6 shows the knowledge extracted from Table 2 by applying the rules proposed in Step 2.

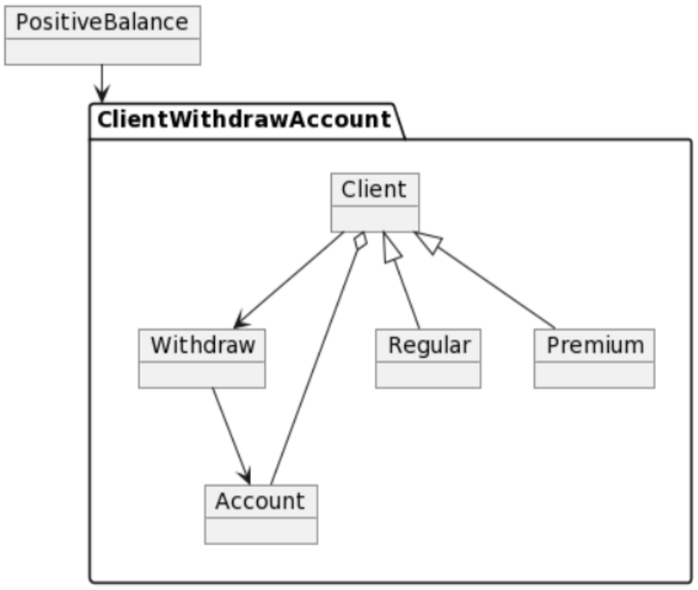

Figure 6. Graphical representation of knowledge. Source: Own work.

6. EVALUATION

The proposed approach was evaluated for its perceived usability rather than the effectiveness of the prototype, as the aim was to assess the approach itself rather than the tool. Moreover, the accuracy of the identified elements was not assessed in this study because the rules employed to identify the elements were inspired by a previous work [25]. Although assessing precision and recall is important for future research, usability evaluation was prioritized in this study.

The usability of the proposed approach was evaluated using the Systems Usability Scale (SUS) [32], [33]. While the SUS is primarily employed for assessing the usability of software systems, it has proven effective for evaluating products and processes [34]. This scale is a 10-

item questionnaire with each question requiring a response on a five-point scale, ranging from "1" (strongly disagree) to "5" (strongly agree). Despite containing 10 items, these are paired to ask the same question from complementary perspectives, ensuring high-confidence results. The questions, adapted to the purposes of this study, were as follows:

1. I think that I would like to use this approach frequently.
2. I found the approach unnecessarily complex.
3. I thought the approach was easy to use.
4. I think that I would need the support of a technical person to be able to use this approach.
5. I found the various functions in this approach were well integrated.
6. I thought there was too much inconsistency in this approach.
7. I would imagine that most people would learn to use this approach very quickly.
8. I found the approach very cumbersome to use.
9. I felt very confident using the approach.
10. I needed to learn a lot of things before I could get going with this approach.

The individuals who participated in the evaluation were five students with an average age of 24.8 years. They were part of a research project and were also working professionals with industry experience. Importantly, in Argentina, students typically begin working during their second year of university, so the participants had relevant professional experience. This made them suitable for the experiment, as they were familiar with reading specifications and assessing their understandability. Their opinion about the quality of knowledge summarization, therefore, was valuable. Furthermore, according to [35], a sample size of four participants (with a variation of plus or minus 1) is recommended to identify up to 80 % of usability problems in a test.

Participants received training on the proposed approach and were then asked to complete a questionnaire. They were a one-page specification of a real medical software application for managing cardio-assisted spaces and were asked to construct the LEL glossary according to the guidelines of Step 1 of the proposed approach. They could ask questions if they needed more information. The goal was to structure natural language in the LEL glossary. Then, they were instructed to apply the rules of Step 2 of the approach to create a conceptual model.

The SUS score was calculated as follows: First, items 1, 3, 5, 7, and 9 were scored by subtracting their ranked value from 1; and items 2, 4, 6, 8, and 10 were scored by subtracting their ranked value from 5. Then, the scores of all participants were summed and multiplied by 2.5, resulting in a value between 0 and 100. Finally, the average score was computed. The resulting score categorizes the approach into one of the following three groups: "Non-acceptable" (0–64), "Acceptable" (65–84), or "Excellent" (85–100) [36]. The obtained score was 88.5, rating the approach as "Excellent" according to the five participants.

Validity threats are generally grouped into four categories: conclusion, internal validity, construct validity, and external validity. Conclusion threats include the reliability of measures, which was mitigated using a well-tested approach (the SUS). Internal validity threats, like instrumentation, were addressed by selecting a real domain, specification, and expert. Construct validity was maintained by having participants describe an LEL glossary using provided templates. Lastly, external validity threats, such as artificial settings, were countered by using a real domain, limiting the experiment duration, and selecting participants with relevant industry experience.

7. DISCUSSION

The approach proposed in this study focuses on generating a conceptual model based on rules inspired by a previous work [25]. While the approach itself may not be entirely new, its novelty lies in using the LEL glossary as input to obtain a conceptual model. The LEL glossary adds some structure by identifying subjects, objects, verbs, and states, which the approach then uses. Although building the LEL glossary requires effort, it yields high-quality results.

The rules in the proposed approach incorporate different strategies. Some rules rely on dictionaries to find specific words or expressions, whereas the NLP framework helps identify the elements related to specific expressions. This is known as dependency relations, and it is very important to identify not only these relationships, but also the elements involved in them.

In their study, [16] also used NLP techniques to recognize parts of speech and divide sentences into phrases. They employed the WordNet dictionary to search for known concepts and identify relationships between them, but it is unclear how they determined the elements involved in these relationships. WordNet is a traditional dictionary; however, specific domains often have their own vocabulary.

For their part [18], extracted domain knowledge from user-authored scenarios using typed dependency parsing techniques. Additionally, they investigated the effectiveness of a complementary approach that employs BERT-based masked language models to identify entities and their associated qualities, enabling the construction of a domain model from a single-word seed term. Their work aligns with the present study, as they used structured elements (scenarios) but based their approach on single words. [23] proposed a similar approach to the one proposed in this study, since they used semantic relations to transform unrestricted natural language requirements into a graph.

8. CONCLUSIONS

This paper presented an approach for extracting knowledge from natural language specifications, particularly those captured through the LEL glossary. The proposed approach incorporates a set of guidelines to help capture and organize natural language specifications for the extraction of knowledge. This knowledge can then be used in different ways, including providing a global overview of the LEL glossary and allowing the inference of new knowledge. This is very important for creating complete and consistent specifications.

Future work will focus on extending this proposal to infer new knowledge that can be added to the LEL glossary for stakeholders to validate and use it effectively. Additionally, efforts will be made to identify inconsistencies and conflicts so that stakeholders can resolve them. Another area of future research involves improving the LEL glossary by categorizing subjects as countable and uncountable. This categorization will make it possible to be precise in certain descriptions. In addition, future work will include enriching the LEL glossary with magnitudes (quantities in units) and categorical elements (lists of possible values). Both characteristics are important for writing precise specifications and inferring knowledge. Particularly, this will facilitate the verification of requirements and the design of tests based on them. Finally, efforts will be directed toward improving both the tool and the experiment, with plans to analyze the precision of the proposed approach using new participants from the industry.

9. ACKNOWLEDGEMENT AND FUNDING

This study was partially supported by funding from the STIC AmSud program through project 22STIC-01.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest.

AUTHORS CONTRIBUTIONS

Mario Lezoche has contributed with the ideas discussing and planning them, as well as the revision of the manuscript. Juliana Delle Ville has contributed with the strategy discussing it as well as with the implementation of the prototype. And Leandro Antonelli has contributed discussing the ideas, the strategy, the implementation of the prototype and the writing of the manuscript.

10. REFERENCES

[1] Boehm, "Software Engineering," *IEEE Transactions on Computers*, vol. C-25, no. 12, pp. 1226-1241, Dec. 1976. https://doi.org/10.1109/TC.1976.1674590

[2] S. L. Lim, and A. Finkelstein, "StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 707-735, May-June 2012. https://doi.org/10.1109/TSE.2011.36

[3] C. Potts, "Using schematic scenarios to understand user needs," in *Proc. of the confe. on Desig. Inter. Syst. processes, practices, methods, & techniques - DIS '95*, Association for Computing Machinery, New York, NY, USA, 1995, 247–256. https://doi.org/10.1145/225434.225462

[4] D. M. Berry, E. Kamsties, and M. M. Krieger, *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity, v1.0.* (2003). Accessed: Sep. 25, 2023. [Online]. Available: https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf

[5] K. Pohl, "The three dimensions of requirements engineering: A framework and its applications," *Inf. Syst.*, vol. 19, no. 3, pp. 243–258, Apr. 1994. https://doi.org/10.1016/0306-4379(94)90044-2

[6] A. Hall, "Seven myths of formal methods," *IEEE Software*, vol. 7, no. 5, pp. 11-19, Sep. 1990. https://doi.org/10.1109/52.57887

[7] Hoare, "An overview of some formal methods for program design," *Computer (Long Beach Calif.)*, vol. 20, no. 9, pp. 85–91, 1987. https://doi.org/10.1109/MC.1987.1663697

[8] L. M. Cysneiros, and J. C. Sampaio do Prado Leite, "Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation," in *Anais do WER01 - Workshop em Engenharia de Requisitos*, Buenos Aires, Argentina, 2001. https://www.researchgate.net/publication/221235247_Using_the_Language_Extended_Lexicon_to_Support_Non-Functional_Requirements_Elicitation

[9] Z. S. Harris, "Co-Occurrence and Transformation in Linguistic Structure," *Language,* vol. 33, no. 3, p. 283, Apr. 1957. https://doi.org/10.2307/411155

[10] N. Chomsky. "The Logical Structure of Linguistic Theory," *Plenum Press,* p. 573, 1975. https://dingo.sbs.arizona.edu/~langendoen/ReviewOfChomskyLSLT.pdf

[11] L. Antonelli, M. Lezoche, and J. Delle Ville, "A Method to obtain a Knowledge Representation from a Natural Language Specification of the Domain using the Glossary LEL," Presented at the *Decissioning 2023*, Popayan, Jun. 2023. https://host170.sedici.unlp.edu.ar/server/api/core/bitstreams/41a01f6d-54e6-4039-9056-996a6b47f50a/content

[12] J. C. S. do P. Leite, and A. P. M. Franco, "A strategy for conceptual model acquisition," in *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, CA, USA, 1993, pp. 243-246. https://doi.org/10.1109/ISRE.1993.324851

[13] A. V. Vu, and M. Ogawa, "Formal semantics extraction from natural language specifications for ARM," in *Formal Methods – The Next 30 Years,* M. H. ter Beek, A. McIver, and J. Oliveira, Eds., Switzerland: Springer, Cham, 2019, pp. 465–483. https://doi.org/10.1007/978-3-030-30942-8_28

[14] S. Geetha, and G.S.A. Mala, "Extraction of key attributes from natural language requirements specification text," in *IET Chennai Fourth International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2013),* Chennai, India, 2013, p. 374-379. https://doi.org/10.1049/IC.2013.0341

[15] F. Bargui, H. Ben-Abdallah, and J. Feki, "Multidimensional concept extraction and validation from OLAP requirements in NL," in *2009 International Conference on Natural Language Processing and Knowledge Engineering*, Dalian, China, 2009, pp. 1-8. https://doi.org/10.1109/NLPKE.2009.5313769

[16] J. Kuchta, and P. Padhiyar, "Extracting Concepts from the Software Requirements Specification Using Natural Language Processing," in *2018 11th International Conference on Human System Interaction (HSI)*, Gdansk, Poland, 2018, pp. 443-448. https://doi.org/10.1109/HSI.2018.8431221

[17] Y. Rigou, D. Lamontagne, and I. Khriss, "A Sketch of a Deep Learning Approach for Discovering UML Class Diagrams from System's Textual Specification," in *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, Meknes, Morocco, 2020, pp. 1-6. https://doi.org/10.1109/IRASET48871.2020.9092144

[18] Y. Shen, and T. Breaux, "Domain Model Extraction from User-authored Scenarios and Word Embeddings," in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, Melbourne, Australia, 2022, pp. 143-151. https://doi.org/10.1109/REW56159.2022.00036

[19] P. Szwed, "Concepts extraction from unstructured Polish texts: a rule based approach," in *Federated Conference on Computer Science and Information Systems,* Lodz, Poland, 2015. https://doi.org/10.15439/2015F280

[20] M. Li *et al.*, "Automated Extraction of Requirement Entities by Leveraging LSTM-CRF and Transfer Learning," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Adelaide, SA, Australia, 2020, pp. 208-219. https://doi.org/10.1109/ICSME46990.2020.00029

[21] Y. Wang, "Semantic information extraction for software requirements using semantic role labeling," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, Nanjing, China, 2015, pp. 332-337. https://doi.org/10.1109/PIC.2015.7489864

[22] D. Sadoun, C. Dubois, Y. Ghamri-Doudane, and B. Grau, "From Natural Language Requirements to Formal Specification Using an Ontology," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, Herndon, VA, USA, 2013, pp. 755-760. https://doi.org/10.1109/ICTAI.2013.116

[23] A. Schlutter, and A. Vogelsang, "Knowledge extraction from natural language requirements into a semantic relation graph," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, New York, NY, USA, 2020, pp. 373-379. https://doi.org/10.1145/3387940.3392162

[24] Y. An *et al.*, "Exploring Pre-Trained Language Models to Build Knowledge Graph for Metal-Organic Frameworks (MOFs)," in *2022 IEEE International Conference on Big Data (Big Data)*, Osaka, Japan, 2022, pp. 3651-3658. https://doi.org/10.1109/BigData55660.2022.10020568

[25] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation", in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, Saint-Malo, 2016, pp. 250–260. https://doi.org/10.1145/2976767.2976769

[26] G. Van Rossum. *Python.* v3.12.3. Accessed: Mar. 05, 2023. [Online]. Available: https://www.python.org/

[27] A. Holovaty, and S. Willison. *Django.* v5.0.4. Accessed: Mar. 05, 2023. [Online]. Available: https://www.djangoproject.com/

[28] A. Ronacher. *Flask,* v3.0.3. Accessed: Mar. 05, 2023. [Online]. Available: https://flask.palletsprojects.com/

[29] M. Honnibal, and I. Montani. *Spacy.* v3.0. Accessed: Mar. 05, 2023. [Online]. Available: https://spacy.io/

[30] S. Bird, E. Klein, and E. Loper. *NLTK.* v3.11. Accessed: Mar. 05, 2023. [Online]. Available: https://www.nltk.org/

[31] A. Roques, *PlantUML,* v1.2024.4. Accessed: Feb. 27, 2023. [Online]. Available: https://plantuml.com/

[32] J. Brooke, "SUS-A quick and dirty usability scale," in *Usability evaluation in industry,* 1st ed. United Kingdom, CRC Press, 1996. https://www.taylorfrancis.com/chapters/edit/10.1201/9781498710411-35/sus-quick-dirty-usability-scale-john-brooke

[33] J. Brooke, "SUS: a retrospective," *Journal of user experience,* vol. 8, no. 2, pp.29-40, 2013. https://uxpajournal.org/sus-a-retrospective/

[34] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Int. J. Hum. Comput. Interact.*, vol. 24, no. 6, pp. 574–594, Jul. 2008. https://doi.org/10.1080/10447310802205776

[35] J. Nielsen, "Estimating the number of subjects needed for a thinking aloud test," *Int. J. Hum. Comput. Stud.*, vol. 41, no. 3, pp. 385–397, Sep. 1994. https://doi.org/10.1006/ijhc.1994.1065

[36] S. McLellan, A. Muddimer, and S. Camille Peres, "The effect of experience on System Usability Scale ratings," *Journal of user experience,* vol. 7, no. 2, pp. 56-67, 2012. https://uxpajournal.org/the-effect-of-experience-on-system-usability-scale-ratings/