A Method to obtain a Knowledge Representation from a Natural Language Specification of the Domain using the Glossary LEL

Leandro Antonelli[1], Mario Lezoche[2], Juliana Delle Ville[1]

[1]Lifia, Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[2] Université de Lorraine, Nancy, Lorraine, France

{leandro.antonelli, juliana.delleville}@lifia.info.unlp.edu.ar
mario.lezoche@univ-lorraine.fr

Abstract— Good requirements (correct, consistent, unambiguous, etc.) are crucial to software development success. Errors made in the requirements stage can cost up to 200 times if they are discovered once the software is delivered to the client. Natural language artifacts are the most used tool to write requirements, since they are understandable by the both parties that participate in the software development: the stakeholders and the development team. Nevertheless, natural language can introduce many defects (ambiguity, vagueness, generality, etc.). Formal reasoning is a good strategy to check whether requirements satisfy the attributes of good requirements or not, but formal reasoning cannot be applied to natural language specification with defects. Thus, this paper proposes an approach to write a good specification and obtain knowledge from it. The approach uses a particular lexicon, the glossary LEL, and it suggest guidelines to write good specification, and it also suggest rules to obtain knowledge (concepts and relations) from the glossary LEL. The paper also presents a prototype to assist to this approach, and a preliminary evaluation of the approach.

Keywords- Requirements specification, knowledge representation, natural language, artificial intelligence

1. Introduction

Good requirements are crucial to software development success. If requirements are not good, the software application developed from them will not satisfy the need, wishes and expectations of the stakeholders. Moreover, if requirements are not good, a lot of work will need to be done to repair the software application to satisfy their needs. It is estimated that errors made in requirements stage can cost up to 200 times if they are discovered when the software is delivered to the client [7].

The expression “good requirements” has a broad meaning. There are many characteristics that are desirable that a software specification has. Does not matter if the development process is agile with minimum of documentation or if it classic with an orthodox software requirements specification of hundreds of pages. It is desirable that the specification would be correct, consistent, unambiguous, etc. These are called quality attributes. Although it is impossibly to satisfy all of them, since there is a huge effort of rework to correct errors, it is necessary to invest the effort in trying to produce the best specification as possible.

Natural language is the most used tool to write requirements [21]. Requirements are specified usually requirements engineering or business analysts, and they should interact with the stakeholder who has the requirements and the knowledge to include in the software application. And the specification also needs to be understandable and precise enough so the software development team can develop the application from it. Natural language specifications are understandable for these both sides [26]. Nevertheless, natural language can introduce many defects: ambiguity, vagueness, generality, etc. [6].

Formal reasoning means infer knowledge from a specification. For example, considering an agriculture specification that states that “plants need water, and tomato is a plant”, it can be inferred that “tomatoes need water”. It is necessary to have requirements without defects originated by the natural language in order to make possible the forma reasoning [25]. Moreover, with forma reasoning it will be possible to automatize many tasks in order to obtain good requirements [15] [17]. For example, consistency is one of the quality attributes of the specification. And consistency means that there is no contradiction in any given pairs of requirements. Having the specification “plants need water, tomato is a plant, and tomato does not need water” it can inferred that “tomato needs water and tomato does not need water”. Since it is a contradiction, it can be concluded that the specification is not consistent.

The glossary LEL is a semi structured artifact with the goal of describing the language of the domain. The glossary LEL is easy to learn, it is easy to use, and it has good expressiveness. We have used the glossary LEL in many domains, some of them very complex, and we had good results. Cysneiros et al. [11] report the use of LEL in a complex domain as the health domain. The glossary LEL uses natural language to describe the language, but it also categorize the vocabulary in 4 categories: subjects, objects, verbs and states. The description of each term is done through two attributes: notion and behavioral responses. Thus, the glossary LEL is more than a simple glossary, it represents the knowledge of a domain (captured through its language) in natural language.

The concept of the kernel sentence was introduced in 1957 by linguist Z.S. Harris [16] and featured in the early work of linguist Noam Chomsky [10]. Kernel sentences are also known as basic sentences. They are declarative constructions, in active voice, always affirmative with only one verb. We argue that the use of kernel sentences in the description of the glossary LEL, as well as some other guidelines to make the natural language near a controlled language, allow us to obtain a good specification.

Having a good specification it is feasible to perform a formal reasoning to infer new knowledge. Nevertheless, before performing formal reasoning, it is necessary to synthetize the knowledge captured in the language through the glossary LEL. Thus, this paper proposes an approach to obtain the knowledge (concepts and relationships) from a glossary LEL. The inference of the new knowledge is outside the scope of this proposal. This paper also presents a prototype, that is, a tool to assist during the application of the proposed approach. Moreover, a preliminary evaluation of the approach is also described.

The rest of the paper is organized in the following way. Section 2 describes some background about glossary LEL and kernel sentences. Section 3 reviews some related work. Section 4 details our contribution namely, the proposed approach. Section 5 describes the tool to support the process. Section 6 presents the preliminary evaluation. Finally, Section 7 discusses some conclusions.

2. Background

This section describes basic concepts of the glossary LEL and the kernel sentences.

2.1. Glossary LEL

The Language Extended Lexicon (LEL) is a glossary that describes the language of an application domain, where not necessarily there is a definition of a software application. The glossary LEL is tied to a simple idea: "understand the language of a problem without worrying about the problem" [19]. The language is captured through symbols that can be terms or short expressions. They are defined through two attributes: notion and behavioral responses. Notion describes the denotation, that is, the intrinsic and substantial characteristics of the symbol, while behavioral responses describe symbol connotation, that is, the relationship between the term being described and other terms (Fig. 1). Each symbol of the LEL belongs to one of four categories: subject, object, verb or state. This categorization guides and assists the requirements engineer during the description of the attributes. Table 1 shows each category with its characteristics and guidelines to describe them.

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

Fig. 1. Template to describe a LEL symbol

Table 1. Template to describe LEL symbols according to its category

Category	Notion	Behavioral Responses
Subject	Who is he?	What does he 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 this situation is checked?

Let's consider a bank domain where there are clients that open account in order to save money. The client can deposit money into his account, and he can also withdraw money if the account has a positive balance. In this simple example there are many terms that should be defined into the glossary LEL to capture the language. For example, "client" should be a symbol of subject category. Then, "account" should be a symbol of object category. “Deposit” and “withdraw” should be symbols of verb category. Finally, “positive balance” should be a symbol of state category. According to the template described in Table 1, the symbols “client”, “account”, “withdraw” and “positive balance” are described in Table 2.

The behavioral responses of symbol “client” describes just two responsibilities: to open and to withdraw money, but there are other for example to deposit that is not mentioned. Then, the behavioral responses of the account only describes the responsibility of the bank to calculate interest, but the responsibilities of the client: open, deposit and withdraw should also be mentioned if the glossary LEL is complete. Then, the behavioral responses, of the verb “withdraw” describe how the task if performed by the bank, who first reduces the balance, then check if the resulting balance is positive, and in that case provide the money to the client. Otherwise, the balance is increased to its original value and no money is provided to the client. Finally, the behavioral responses of the state “positive balance” describe in detail what “positive” means. In this cases, it means greater or equal to zero.

Table 2. Glossay LEL symbols for the banking example

Category	**Notion**	**Behavioral Responses**
Subject: Client	Person that saves money in the bank.	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 interest for the account.
Verb: Withdraw	Act of extracting money from an account.	The bank reduces the balance of the account. If the account has a positive balance the bank provides the money to the client. If the account has not a positive balance the bank increases the balance of the account.
State: Positive balance	Situation where the account has money.	The balance is greater or equal to zero.

2.2. Kernel Sentences

A kernel sentence is a simple construction with only one verb. It is also active, positive and declarative. This basic sentence does not contain any mood. It is termed as “kernel” since it is the basis upon which other more complex sentences are formed.

For example, Fig. 2 describes two kernel sentences. The first one states that a subject (“client”) performs an action (“opens”) on a certain object (“account”). The second sentence has the same structure while it describes a different action (“deposits”) and the object is a little bit more complex because it states that the object “money” is deposited into the object “account”. Fig. 3 shows two sentences that are not kernel. The first sentence (“The client deposits money and calculates interest into the account”) has two verbs, that is why it is not a kernel sentences. Moreover, the sentence is partially correct, since it is true that the client deposits money into the account. But, it is not true that the client calculates the interests. It is the bank who performs this calculation. That is why simple (kernel sentences) are suggested to describe specification. A simple sentence is more probable to be completely true or false. The second sentence (“The account is opened”) is written in passive voice, while kernel sentences should be written in active voice. Since it is written in passive voice, the subject of the sentence (“the account”), is not the subject who performs the action. In fact, the real subject who performs the action is not mentioned. According to the example provided, the real subject should be “the client”, but it could also be the company that the client work for and the company is the one who decided to open an account to pay his salary. These are two simple examples about how can kernel sentences writing style help to improve specifications.

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

Fig. 2 Kernel Sentences

The client deposits money and calculates interest into the account.
The account is opened.

Fig. 3 No Kernel Sentences

3. Related works

Vu et al. [34] propose a method to transform specification to source code. Their approach systematically extracts the formal semantics of ARM instructions from their natural language specifications. Although ARM is based on RISC architecture and the number of instructions is relatively small, there are various series including Cortex-A, Cortex-M, and Cortex-R. Their approach applies translation rules enriched by the sentences similarity analysis.

Some other proposal deal with more abstract representation of the knowledge, for example data base table, schemas or object oriented designs. Geetha et al. [14] identify the schema and relationship of a data base from the natural language requirements specification. Their work starts with identifying the table schema and their properties. Then the Primary Key attribute is identified based on adjectives. It is done with rules and also with a machine learning component trained with statistical data. The relationships are identified using the primary key to identify the Foreign Key.

Bargui et al. [5] propose and approach to specify requirements through a template that represents the concepts of a decision making process. They also provide a linguistic pattern for the acquisition of analytical queries. This pattern facilitates the automatic analysis of the queries without being too restrictive to the writing styles.

Kuchta et al. [18] propose a method and a tool to extract concepts based on a grammatical analysis of requirements written in English without the need to refer to specialized ontology. These concepts can be further expressed in the class model, which then can be the basis for the object-oriented analysis of the problem. The method uses natural language processing techniques to recognize parts of speech and to divide sentences into phrases and also the WordNet dictionary to search for known concepts and recognize relationships between them.

Rigou et al. [28] are concerned about a formal or a semi-formal model description of functional requirements in the context of a model-driven engineering approach, where a platform-independent model, is used to derive automatically or semi-automatically the source code of a system. They claim that generally, the approaches use a predefined set of rules which impose several restrictions on the way a specification is written. Thus, they propose a deep learning approach.

Then, there are proposal that deals with conceptual model earlier in the software development life cycle. For example domain models, concepts, and entities, some approaches are based on syntactic rules and some other are enriched with semantic. Shen et al. [31] propose an approach to assist the phases of requirements elicitation and analysis using masked language models, which have been used to learn contextual information from natural language sentences and transfer this learning to natural language processing (NLP) tasks. They claim that the masked language models can be used to predict the most likely missing word in a sentence, and thus be used to explore domain concepts encoded in a word embedding. Their approach extracts domain knowledge from user-authored scenarios using typed dependency parsing techniques. They also explore the efficacy of a complementary approach of using a BERT-based masked language models to identify entities and associated qualities to build a domain model from a single-word seed term.

Szwed et al. [33] propose an approach to extract concepts from unstructured Polish texts with special focus the morphological forms. Since Polish is a highly inflected language, detected names need to be transformed following Polish grammar rules. They propose a method for specification of transformation patterns, which is based on a simple annotations language. Annotations prepared by a user are compiled into transformation rules. During the concept extraction process the input document is split into sentences and the rules are applied to sequences of words comprised in sentences. Recognized strings forming concept names are aggregated at various levels and assigned with scores.

Lit et al. [20] presents an approach that employs LSTM-CRF model for requirement entity extraction. Their proposal consist of four phases: (i) model construction, where it is built a LSTM-CRF model and an isomorphic LSTM language model for transfer learning; (ii) LSTM language model training, where it is captured the general knowledge and adapt it to requirement context; (iii) LSTM-CRF training, where it is trained the LSTM-CRF model with the transferred layers; (iv) requirement entity extraction, where it is applied the trained LSTM-CRF model to a new-coming requirement, and automatically extracts its requirement entities.

Wang et al. [35] propose an approach that selects frequent verbs from software requirement specification documents in the e-commerce domain, and built the semantic frames for those verbs. Then the selected sentences are labeled manually and the result is used as training examples for machine learning. They also correct the parsing result of the Stanford Parser. During the labeling process, they adopt a sequential way in which the previous labeled results will be used to construct dynamic features for the identification of the subsequent semantic roles.

Sadoun et al. [29] presents an approach that model the domain knowledge through an ontology and to formally represent user requirements by its population. Their approach of ontology population focuses on instance property identification from texts. They use extraction rules automatically acquired from a training corpus and a bootstrapping terminology. These rules aim at identifying instance property mentions represented by triples of terms, using lexical, syntactic and semantic levels of analysis. They are generated from recurrent syntactic paths between terms denoting instances of concepts and properties.

Finally, some approaches are concerned in obtaining knowledge representation similar to our proposal.

Schlutter et al. [30] propose a knowledge extraction approach based on an explicit knowledge representation of the content of natural language requirements as a semantic relation graph. Their approach is fully automated and includes an NLP pipeline to transform unrestricted natural language requirements into a graph. They split the natural language into different parts and relate them to each other based on their semantic relation.

An et al. [1] explore a set of state-of-the-art pre-trained general-purpose and domain-specific language models to extract knowledge triples for metal-organic frameworks. They created a knowledge graph benchmark with 7 relations for 1248 published metal-organic frameworks synonyms.

4. The proposed approach

This section describes the proposed approach. First, it describes the generality of the approach, that consist of three steps: (i) glossary LEL description, (ii) glossary LEL revision, and (iii) knowledge extraction. Then, every of the step is described in a different subsection.

4.1. Our approach in a nutshell

The proposed approach has the objective of obtaining a synthetic representation of the knowledge captured in the language of the application domain captured through the glossary LEL.

The proposed approach needs the knowledge as input and produces a knowledge representation as output. The knowledge can be obtained from different sources. It can be obtained mainly directly from the stakeholders, though interviews or other type of technic (focus group, questionnaires, etc.). But the knowledge can be obtained from documentation.

Thus, the proposed approach is thought to be applied by a requirements engineer or a business analyst who describe the glossary LEL, review it and finally synthetize the knowledge. Moreover, the approach can be used not only by one persona, many people can participate. That is, many requirements engineering can participate describing the glossary LEL in a collaborative way. Moreover, stakeholder experts of the domain can participate writing symbols of the glossary LEL. That is why, the approach consists of three steps: (i) glossary LEL description, (ii) glossary LEL revision, and (iii) knowledge extraction. When more than one person participate of the construction of the LEL, the step (ii) is vital. Nevertheless, when only one person participate, and the volume of information is huge, the step (ii) is also vital. Moreover, the need of a tool to automatize the revision is crucial.

Summing up, the proposed approach can be applied in different situations: (i) to summarize documents produced by other people, (ii) to consolidate and summarize documents produced by only one analyst eliciting from multiple sources, and (iii) to consolidate and summarize documents produced collaborative by a group of analyst or experts.

The three steps that compose the proposed approach are: (i) glossary LEL description, (ii) glossary LEL revision, and (iii) knowledge extraction. The first step proposes a set of guidelines to help people to describe the glossary LEL. The second step proposes a set of guidelines to review some aspects of the description. Some revision (not all of them) are linked with guidelines of the step (i). Finally, the step (iii) provides rules to obtain knowledge from the glossary LEL specified, which is described in terms of concepts and relations. Fig. 4 summarizes the approach.

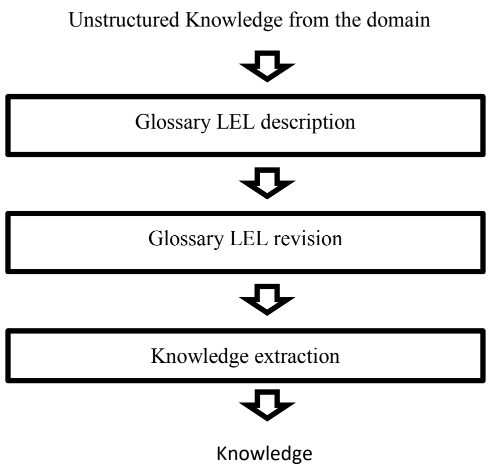


Fig. 4. Our approach in a nutshell

Let's consider the glossary LEL described in Table 2. We can assume that is was described in the step 1 of the proposed approach, and reviewed in the step 2. In fact, the example of the Table 2 satisfies all the guidelines suggested. If step 3 is applied to that LEL it will obtained that the concept "client" and the concept "account" are linked by the relation "withdraw". Then, the whole relation ("client", "withdraw", "account") is restricted by the condition "positive balance".

4.2. Step 1, guidelines to write the glossary LEL

This section describes every one of the six guidelines proposed to describe the glossary LEL. There are some other proposals about how to describe the glossary LEL. Some of them suggest the basic template described in Table 1 and some other suggests specific structures [2] [3] [Antonelli 2023]. These guidelines reinforce the need to use kernel sentences and reference to symbol already defined, and moreover, this guidelines provides suggestion about how to write compound sentences using subordination (if then), negation (no) and coordination (and). Finally, these guidelines also consider the use of state symbols in subordination expressions. The rest of the section describes and provides example for every one of the guidelines.

Guideline 1. Sentences to describe behavioral responses must suit kernel sentences philosophy. Behavioral responses of every category (subject, object, verb and state) must be written according to the philosophy of kernel sentence, that is, an explicit subject followed by only one verb in turn followed by an object that receives the action. Fig. 2 shows examples of sentences well written according to this guideline that are used to 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 suggest that the subject, the verb and the object mentioned in the previous guidelines, should also be symbols already defined in the glossary LEL. For example, the symbol "client" defined in Table 2, has the following description in the behavioral responses "The client opens an account. The client withdraws money from the account." If it is considered that Table 2 has a complete description of a glossary LEL, the symbols "client", "account" and "withdraws" are defined. But the symbol "opens" is not defined and it should be defined according to this guideline.

Guideline 3. Use subordination (if then) in behavioral responses of verb symbols to describe conditions. Let's consider the example of verb "Withdraw" in Table 2. One of the behavioral responses states "If the account has a positive balance then bank provides the money to the client.". That means that the bank only provides the money to the client, when the account has positive money after reducing the balance with the amount that the client want to withdraw. That is, it is checked whether the balance remains positive after the extraction. It is important to mention that the application of this guideline is not a contradiction to the guideline 1, since the use of a subordination requires kernel sentences, although there are two kernel sentences in the same sentence, there are still kernel sentences associated.

Guideline 4. State symbols should be used in the condition of a subordinated sentence. Since the state symbols describes situation, this situation should be used as a condition of the subordinated sentence. Moreover, the condition should be included in the kernel sentence. For example, let's consider the verb "Withdraw" in Table 2. One of the behavioral responses states "If the account has a positive balance then ...", "positive balance" is a state symbol describe in Table 2, and the expression "the account has a positive balance" suit the rules of kernel sentences.

Guideline 5. Use negation (not) to invert a condition or an action. For example, the verb "Withdraw" in Table 2 contains a behavioral responses that states "If the account has not a positive balance then ...". Nevertheless, the negation can be used to deny some behavioral response. That is, behavioral response of subject for example, describe all the activities that subject can perform. The symbol "client" defined in Table 2, has the following description in the behavioral responses "The client opens an account. The client withdraws money from the account." Generally, (and according to guideline 1 that recommends the use of kernel sentences that should be positive by definition), behavioral responses should be positive. Nevertheless, some times, it is very important to make explicitly definition that something cannot be done. In this case, the negation can be used. For example, considering a bank that only works with organization and not with private clients, the responsibility of opening account relies on the organization and not into the client. Thus, a behavioral response of a client could state "The client does not open an account".

Guideline 6. Use coordination (and) to join two subjects, objects or kernel sentences. Although multiple kernel sentences can be written, using the coordinator "and" more simple and natural sentences can be written. For example, instead of writing "The client deposits money in the account. The bank deposits money in the account", it could be written "The client and the bank deposits money in the account".

4.2.- Step 2, guidelines to review the glossary LEL

This section describes some guidelines to review the glossary LEL described with the guidelines of the previous sections. These guidelines are simply a way of checking whether the previous guidelines were used correctly. Although these guidelines can be used as a checklist after some description was done, the following section describes a prototype that can perform these checks automatically.

Guideline 1. Check for explicit subject and only one verb written in active voice in a behavioral response. This guideline has the objective to be sure that kernel sentences are used to describe behavioral responses. Thus, the use of only one verb, written in active voice should be used. Moreover, the sentence should have at least one explicit subject.

Guideline 2. Check for the definition of every subject, verb and object in a behavioral response. This guideline complements the guideline number 2 of the previous section. The objective is to check that every element (with semantic meaning) used in a sentence is describes within the glossary LEL.

Guideline 3. Check for the definition of every condition of a subordination as a state. This guideline complements guideline number 4 of the previous section. The objective is similar to the objective of guideline 2 of this section.

Guideline 4. Replace modal verbs with a subordination. The expression "The client withdraws money from the account" states that the client is allowed to do that and he will do it when he need. If a modal verb is used "The client might withdraw money from the account" it means that there is certain condition. Thus, the sentence should be rewritten as "If (some condition related to the client) then the client withdraw money from the account"

Guideline 5. Rephrase expressions with "no" to be sure of expressing prohibition. Guideline 5 of the previous section suggest the use of negation to invert a condition or an action. When it is used to invert an action, that is, to state that some subject does not do some action, it is important to be sure that the description states a prohibition to perform that action. For example, the expression "The client does not open an account" can be rephrased as "The client is not allowed to open an account" to be sure that a prohibition is described.

Guideline 6. Check for unambiguous terms used in notion. When technical specification are written, the goal to be precise and unambiguous is naturally pursued, that is why it is not added a complementary guideline in the previous section. Nevertheless, it is important to pay attention to this issue and to consider this guideline for the revision. Although this guideline can be applied manually, maybe the tool support using some tool of word-sense disambiguation is more useful.

4.3.- Step 3, knowledge extraction

This section describes some rules to extract knowledge form the glossary LEL described and reviewed with the guidelines of the previous sections.

Rule 1. Concepts are obtained from symbols of subject and objects categories. Considering the glossary LEL described in Table 2 two concepts are obtained: "client" and "account", the first one is a subject and the second one is an object, but both of them are concepts.

Rule 2. Relations are obtained from symbols of verbs categories. Considering the glossary LEL described in Table 2 one relation is obtained: "withdraw". According to the behavioral response of the symbol "client", this action ("withdraw") relates the concepts "client" and "account".

Rule 3. Conditions are obtained from symbols of state categories. Considering the glossary LEL described in Table 2 one condition is obtained: "positive balance".

Rule 4. Relations obtained by rule 2 are restricted by conditions. Considering the glossary LEL described in Table 2 the state: "positive balance" gives origin to a condition. This state is mentioned in the "withdraw" verb that gives origin to the relation between "client" and "account". Thus, the whole relation ("client", "withdraw", "account") is restricted by the condition "positive balance".

5. Tool support

A software prototype was implemented that can be used to support the application of the proposed approach. The prototype is a web application implemented following a service-oriented architecture. The core of the application and its services are implemented in Python [27], while the web component is implemented with Django [12], and the APis are implemented with Flask [13]. Python [27] is also used to communicate to the Spacy [32] and NTLK libraries [23] used to deal with natural language processing.

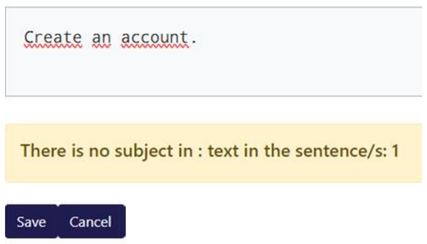
The application is responsive, that means, the interface of the application adapts itself to the device used: a computer (desktop or laptop) or mobile device (phone or tablet). Thus, the application provides a variety of platforms to be used and experts will have a wide range of possibilities to contribute with knowledge acquisition.

The prototype implements two roles of users: (i) administrators of the projects and (ii) experts. The administrator of the tool can create project and the experts contribute with knowledge to the projects. The prototype is in fact a framework, since it can manage different type of artifacts based on natural language (Fig. 5). And the prototype also is easily extended to perform different process to derive more information from the artifacts.

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

Fig. 5. Configuration of artifacts

The prototype uses Spacy [32] and NLTK [23] libraries to perform natural language processing. The POS (part of speech) tagging is an appropriate tool to determine the revisions suggested by guideline 1 as well as the identification of modal verbs mentioned in guideline 4. Then, word disambiguation is another powerful tool mentioned in guideline 6. Then, guidelines 2 and 3 requires to check the presence of some words as symbols of the glossary LEL. In order to do that, the stemming and lemmatization tools are very useful, since they convert words to their root form in order to make the comparison. Fig. 6 shows how the tool check kernel sentences.


Create an account.

There is no subject in : text in the sentence/s: 1

Fig. 6. Kernel sentence checking

The framework Plant UML [24] is used to display the model (Figure Fig. 7).

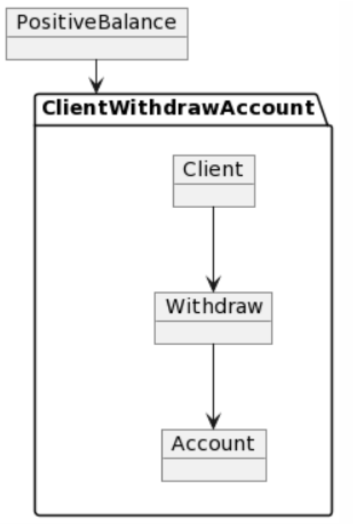


Fig. 7. Knowledge represented graphically

6. Evaluation

The proposed approach was evaluated, although the prototype was not used in the evaluation because the goal of the evaluation was to assess the perceived usability of the proposed approach instead of evaluating the tool.

The Systems Usability Scale (SUS) was used to perform the evaluation [8] [9] in terms of the usability of the proposed approach. Although SUS is mainly used to assess usability of software systems, it was probe to be effective to assess products and processes [Bangor 2008].

The System Usability Scale (SUS) consists of a 10-item questionnaire; every question must be answered in a five-options scale, ranging from ”1” (”Strongly Disagree”) to ”5” (”Strongly Agree”). Although there are 10 questions, they are related by pairs, asking the same question but in a complementary point of view in order to obtain a result of high confidence. The questions are the following:

1.- I like to use this approach.
2.- I find this approach to be more complicated than it should be.
3.- I think the approach is simple and easy to use.
4.- I need technical support to use this approach.
5.- I find the approach functioning smoothly and is well integrated.
6.- I think there are a lot of irregularities in the approach.
7.- I think most people can learn this approach quickly.
8.- I find this approach to be time-consuming.
9.- I feel confident using this approach.
10.- I think there are a lot of things to learn before I can start using this approach.

The participants of the evaluation were 5 students (with an age average of 24.8 years old) that participate in research project. It is important to mention that all of the students have experience in industry since in Argentina, students generally begin to work in industry in second year of their undergraduate studies. Thus, participants are suitable to this evaluation since they have experience in reading specification and deciding about the understandability of it. So, their opinion about whether the knowledge summarization is good or not to is important to us. They received training on the proposed approach and they were requested to answer the questionnaire.

The calculation of the SUS score is performed in the following way. First, items 1, 3, 5, 7, and 9 are scored considering the value ranked minus 1. Then, items 2, 4, 6, 8 and 10, are scored considering 5 minus the value ranked. After that, every participant's scores are summed up and then multiplied by 2.5 to obtain a new value ranging from 0 to 100. Finally, the average is calculated. The approach can have one of the following results: "Non acceptable" 0-64, "Acceptable" 65-84, and "Excellent" 85-100 [22]. The score obtained was 88.5. Thus, the approach can be considered as "Excellent".

6. Conclusions

This paper has presented an approach to obtain knowledge specification from a natural language specification. Particularly, we proposed the use of the glossary LEL to capture the language of the domain, hence, its knowledge. The proposed approach consists in a set of guidelines to help to capture and organize the natural language specification, in order to obtain the knowledge. The knowledge obtained can be used in different way. It synthetizes the glossary LEL, so it can be used to provide a global view. But it also make possible to infer new knowledge, and this is very important to provide complete and consistent specifications, moreover if we consider that they use natural language. This is our main further work. We plan to extend this proposal to infer knew knowledge to be used to add to the glossary LEL so stakeholder can validate and use them. And also to identify inconsistencies and conflict, so they can be solved. Another further work is related to improve the glossary LEL with a categorization of subjects as countable and uncountable. This categorization will make possible to be precise in certain descriptions involving them. We plan to enrich the LEL with magnitudes (numbers with a unit) and categorical elements (a list of possible value). We think that both characteristics are important to write precise specification and obtaining inferred knowledge. Particularly, this will make possible to verify requirements and designing tests from it.

Acknowledgment

This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

References

[1] An, Y. et al.: "Exploring Pre-Trained Language Models to Build Knowledge Graph for Metal-Organic Frameworks (MOFs)" 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, pp. 3651-3658 (2022)

[2] Antonelli, L., Leite, J.C.S.P., Oliveros, A., Rossi, G.: "Specification Cases: a lightweight approach based on Natural Language", Workshop in Requirements Engineering (WER), Brasilia, Brasil, Agosto 23 – 27 (2021)

[3] Antonelli, L., Fernandez, A., Ruffolo, N., Sansone, E., Torres, D.: "A Collaborative Approach to specify Kernel Sentences using Natural Language", Workshop in Requirements Engineering (WER), Natal, Brasil, Agosto 23 – 26 (2022).

[4] Antonelli, L., Delle Ville, J., Dioguardi, F., Fernandez, A., Tanevitch, L., Torres, D.: "An Iterative and Collaborative Approach to Specify Scenarios using Natural Language", Workshop in Requirements Engineering (WER), Natal, Brasil, Agosto 23 – 26 (2022).

[5] Bargui, F., Ben-Abdallah, H., Feki, J.: "Multidimensional concept extraction and validation from OLAP requirements in NL," 2009 International Conference on Natural Language Processing and Knowledge Engineering, Dalian, China, pp. 1-8 (2009)

[6] Berry, D., Kamsties, E. and M. Krieger. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. University of Waterloo. 2003

[7] Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.

[8] Brooke, J.: "SUS-A quick and dirty usability scale" Usability evaluation in industry, 189(194), pp. 4-7, 1996.

[9] Brooke, J: "SUS: a retrospective", Journal of usability studies 8.2, pp.29-40, 2013.

[10] Chomsky, N.: The Logical Structure of Linguistic Theory. Plenum Press, New York, 1975.

[11] Cysneiros, L.M., Leite, J.C.S.P.: Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation, in proceedings of the Workshops de Engenharia de Requisitos, Wer'01, Buenos Aires, Argentina, 2001.

[12] Django, https://www.djangoproject.com/, accessed: 2023-03-05

[13] Flask, https://flask.palletsprojects.com/, accessed: 2023-03-05

[14] Geetha, S., Ganapathy S. A. M.: "Extraction of key attributes from natural language requirements specification text." Computer science, DOI:10.1049/IC.2013.0341 (2013).

[15] A. Hall, Seven Myths of Formal Methods, IEEE Software, 9, pp. 11-19, September 1990

[16] Harris, Z. S.: Co-occurrence and transformation in linguistic structure. (Linguistic Society of America) pp. 390- 457, 1957.

[17] C. A. R. Hoare, An overview of some formal methods for program design, in: C. A. R. Hoare, C. B.

[18] Kuchta, J., Padhiyar, P.: "Extracting Concepts from the Software Requirements Specification Using Natural Language Processing," 2018 11th International Conference on Human System Interaction (HSI), Gdansk, Poland, 2018, pp. 443-448 (2018)

[19] Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition, in Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, pp 243-246 (1993)

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

[21] Lim, S. L., Finkelstein, A.: "StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation", IEEE transactions on software engineering, Volume 38, Issue 3, May-Jun 2012, DOI 10.1109/TSE.2011.36, pp 707-735, 2012

[22] McLellan, S., Muddimer, A., Peres, S. C.: "The effect of experience on System Usability Scale ratings." Journal of usability studies 7.2, pp. 56-67, 2012.

[23] NLTK, https://www.nltk.org/, accessed: 2023-03-05

[24]PlantUML, available at: https://plantuml.com/, accessed on 27th February 2023

[25] K. Pohl, The Three Dimensions of Requirements Engineering, in: Proc. 5th Int. Conf. of Advanced Information Systems Engineering 1993, Paris, Springer, Berlin, pp. 275-292, 1993

[26] Potts, C.: "Using schematic scenarios to understand user needs," in Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, 1995

[27] Python, https://www.python.org/, accessed: 2023-03-05

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

[29] Sadoun, D., Dubois, C., Ghamri-Doudane, Y., Grau, B.: "From Natural Language Requirements to Formal Specification Using an Ontology," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, pp. 755-760 (2013).

[30] Schlutter, A., Vogelsang, A.: "Knowledge Extraction from Natural Language Requirements into a Semantic Relation Graph". In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, pp 373–379 (2020)

[31] Shen, Y., Breaux, T.: "Domain Model Extraction from User-authored Scenarios and Word Embeddings" 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW), Melbourne, Australia, pp. 143-151 (2022)

[32] Spacy https://spacy.io/, accessed 2023-03-05

[33] Szwed, P.: "Concepts extraction from unstructured Polish texts: A rule based approach," 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), Lodz, Poland, pp. 355-364 (2015.

[34] Vu, A.V., Ogawa, M.: "Formal Semantics Extraction from Natural Language Specifications for ARM". In: ter Beek, M., McIver, A., Oliveira, J. (eds) Formal Methods – The Next 30 Years. FM 2019. Lecture Notes in Computer Science(), vol 11800. Springer, Cham (2019)

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