# Defining the language of the software application using the vocabulary of the domain

Leandro Antonelli[1], Julio Leite[2], Alejandro Oliveros[3], Gustavo Rossi[1]

[1]Lifia, Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[2]Dep. Informática, PUC-Rio, Gávea, RJ, Brasil
[3]INTEC-UADE, Bs As, Argentina

{lanto,gustavo}@lifia.info.unlp.edu.ar
julio@inf.puc-rio.br
oliveros@gmail.com

**Abstract.** Requirements engineering is one the most critical stages in software development. If the requirements are not correct the software development team will produce an artifact that will not satisfy the needs, wishes and expectations of the client. Requirements (and knowledge in general) are spread among many stakeholders. Natural language is widely used since it is an adequate tool considering non-technical stakeholder. Nevertheless communication problems arise with the use of natural language. The software development team members need to learn about the application domain and this process of learning means focusing on the features to be included in the software application, while leaving apart the elements out of the boundaries of the application. This process is not easy when people face a new application domain. Thus, this paper proposes an approach to define the software application language from a vocabulary of the application domain.

**Keywords:** LEL, vocabulary, Requirements, application domain, software application.

## 1 Introduction

Requirements engineering is a critical stage of software development. Errors made at this stage can cost up to 200 times to repair when the software is delivered to the client [7]. Requirements described as Use Cases or as User Stories define the goals, the scope and the functionality of the software system. Nevertheless, software applications are “packed knowledge about the domain” [11]. This knowledge needs to be captured in a complementary artifact to Use Cases and User Stories, for example in business rules [23] or given-then-when scenarios [27]. While goals and requirements for the software application can be elicited from a small group of people (the client or the sponsor) the knowledge of the domain relies in a wider group of stakeholder (the domain experts) who generally has a different and complementary point of view of

Received April 2023 Accepted July 2023; Published August 2023

the domain. Thus, it is important to involve as many experts as possible to collaboratively [20] acquire their knowledge.

Experts and development team belong to different worlds and use different languages [26]. The experts use the language of the domain while development team uses a computer science language. In order to cope with this communication gap it is important to use artifacts in natural language that are readable by both parties [20]. Nevertheless, the use of natural language is not enough since both parties need to share a common language. Particularly the development team should adopt the language used in the application domain. This adoption is not easy because the application domain is broader than the software application. Hence, the knowledge in the application domain (and its representation through its language) sometimes is overwhelming for the development team members. Moreover, considering the amount of stakeholder in application domain (clients, users, sponsor, experts, etc.). Thus, it is hard for the development team to decide what is important regarding the boundaries of the software application. This paper proposes an approach to consider the language of the application domain (captured through its vocabulary) in order to reduce it to obtain the language limited to the boundaries of the software application.

The LEL is glossary [23] that has the aim of understanding the language of an application domain without worrying about the application software. The LEL categorizes terms in four categories (subjects, objects, verbs and states) and uses two attributes (notion and behavioral responses) to describe the terms. We believe that the LEL is a convenient tool because of three characteristics that we found in our experience: it is easy to learn, it is easy to use, and it has good expressiveness. We have used the LEL in many domains, some of them very complex, and we had good results. Cysneiros et al. [12] report the use of LEL in a complex domain as the health domain.

Our proposed approach uses the glossary LEL as input and obtains a new glossary LEL as output. Although the language used in the software application can omit, change or add concepts from the application domain, our proposed strategy only consider removing the elements of the application domain that are not relevant for the software. Thus, the proposed approach is a kind of “filter” with the aim to reduce the language of the domain to a subset that belong the boundaries of the application software. Hence, the proposed approach can also be considered as a process to define the scope of the application.

The rest of the paper is organized in the following way. Section 2 describes some preliminary knowledge needed to understand the approach. Section 3 describes the proposed approach. Section 4 provides evidence about the applicability and usability of the approach. Section 5 discuses some related works. Finally, section 6 presents some conclusion and future work.

## 2 Language Extended Lexicon

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 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

Received April 2023 Accepted July 2023; Published August 2023

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? | What other situations can be reached? |

## 3 The proposed approach

This section describes the proposed approach in a general way, and after that it describes every step.

### 3.1 The approach in a nutshell

The proposed approach has the goal to analyze the glossary LEL used as input and select a subset of symbols and their descriptions, in order to provide a new glossary LEL as output. This output glossary LEL will describe the elements that would be inside the boundaries of a new software application that would be developed to provide support to the application domain.

It is important to mention that this proposed approach only considers the reduction of symbols and their description from the input glossary LEL to the output glossary LEL. And the proposed approach does not consider the modification of the descriptions or the additions of new ones.

The approach is based mainly in the relationship of the categories of the glossary LEL and some key elements in a software application design. The categories of the glossary LEL are: subject, objects, verbs and states. For the proposed approach, states are not used. Thus, symbols of category subject of the LEL are related with user roles in a software application, verbs of the LEL are related with functionality of the application, and finally, objects of the LEL are related with databases of the application [2] [16].

Received April 2023 Accepted July 2023; Published August 2023

The proposed approach consists basically of a succession of four steps: (i) classification of subjects, (ii) classification of behavioral responses of subjects, (iii) classification of behavioral responses of verbs, and (iv) classification of objects. Fig. 2 summarizes the steps.

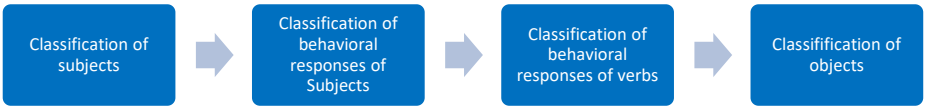


**Fig. 2.** Our approach in a nutshell

### 3.2 Classification of subjects

The classification of subjects pursues the goal of dividing the universe of discourse in three parts: (i) the software application to be developed, (ii) other software applications already developed, and (iii) the information system or behavior of the application domain that will remain manual, that is with no automatization.

Our approach is concerned about (ii) the software application to be developed. Nevertheless, the others two subsets are important because they will interact with the software application to be developed.

Thus, subjects of the glossary LEL should be categorized in one of the following categories: (i) subjects / users of the intended software application, (ii) subjects / users of another software application, and (iii) subjects / actors that will keep performing activities manually.

The rest of the paper will use an agriculture domain in order to provide examples of the proposed approach. Thus, we consider a farm that has the objective of growing fruits as business. The farmer is the person who has the technical agriculture knowledge. There are many field laborers who help the farmer. And there is an administrator who is in charge of taking the strategic decisions for the business.

The glossary LEL for this situation includes three subjects: farmer (Fig. 3), administrator (Fig. 4), and field laborer (Fig. 5). The farmer should be categorized as (i) subject / user of the intended software application, because the goal is to automatize some of their tasks. The administrator should be categorized as (ii) subject / user of another software application, because he already has a software application to manage the needs of the markets, the sales, the cash flow, etc. Finally, the field laborer should be categorized as (iii) subjects / actors that will keep performing activities manually, since he is in charge of cultural activities that consist in activities that cannot be automatized with machines. Fig. 6 summarizes this procedure.

**Subject:** farmer (user of the intended software application)
**Notion:** responsible to grow the fruits.
**Behavioral responses**
The farmer fertilizes spraying.
The farmer fertilizes watering.

**Fig. 3.** Subject farmer

Received April 2023 Accepted July 2023; Published August 2023

**Subject:** administrator (user of another software application)
**Notion:** responsible to maintain a positive balance in the cash flow of the business.
**Behavioral responses**
The administrator decides the fruits to plant.

**Fig. 4.** Subject administrator

**Subject:** field laborer (actor that will keep performing activities manually)
**Notion:** responsible to labor tasks in the field.
**Behavioral responses**
The field laborer performs cultural activities.

**Fig. 5.** Subject field laborer

```
for each subject s with the glossary LEL
   categorize s as
   (i) user of the intended software application
   (ii) user of another software application
   (iii) actors that will keep performing activities manual-
ly
```

**Fig. 6.** Procedure for Subject categorization

### 3.3 Classification of behavioral responses of subjects

The behavioral responses of the subjects denote the actions (activities, tasks) that subjects perform within the application domain. Thus, subjects categorized as "subject / user of the intended software application" will be users of the software application and some of their behavioral responses would be functionality that will be included in the software application to be developed. This second steps of the approach, consists in analyzing the behavioral responses of the subject previously categorized as "subject / user of the intended software application", and each behavioral response should be categorized as: (i) functionality of the intended software application, (ii) functionality of another software application, and (iii) activities to keep performing manually. In some situations, all the activities of the subjects selected could be included in the new software application. But, some other times it is necessary this second step to analyze every activity in order to define the scope of the new software application.

Regarding the example, the farmer fertilizes in two different ways. One technique consists in using a spraying back pack, and another one consists in using an irrigation pipe. The procedure to fertilize using the backpack is manual, so it will be outside the new software application. Nevertheless, the irrigation pipe can be adapted in order to automatize the fertilization. Thus, the behavioral impact "The farmer fertilizes spraying" is categorized as (iii) activities to keep performing manually. While the behavioral impact "The farmer fertilizes watering" is categorized as (i) functionality of the intended software application (Fig. 7). The procedure is summarized in Fig. 8.

Received April 2023 Accepted July 2023; Published August 2023

**Subject:** farmer
**Notion:** responsible to grow the fruits
**Behavioral responses**
The farmer fertilizes spraying. (activities to keep performing manually)
The farmer fertilizes watering. (functionality of the intended software application)

**Fig. 7.** Categorization of behavioral responses of subject farmer

```
for each subject s categorized as user of the intended soft-
ware application
   for each behavioral response b that belong to s
       categorize b as
       (i) functionality of the intended software applica-
tion
       (ii) functionality of another software application
       (iii) activities to keep performing manually
```

**Fig. 8.** Procedure for behavioral responses of the subject categorization

### 3.4 Classification of behavioral responses of verbs

The behavioral responses of the verbs describe how the activity represented by the verb should be carried out. The behavioral responses are a kind of work breakdown of the verb that describes. Although the step 2 of the approach classifies the behavioral responses of the subjects according to their inclusion in the intended software application, it could happen that some activities will not be completely automatized. Thus, each behavioral response of the subjects (categorized as "functionality of the intended software application") that in turn are described as verbs should be analyzed. In this step, the behavioral responses of these verbs should be categorized as (i) functionality of the intended software application, (ii) functionality of another software application, and (iii) activities to keep performing manually.

The process of fertilizing through watering with the irrigation pipe is composed of several steps. First, some calculus of the mixture of the minerals to use to fertilize should be done. Then, the mixture should be prepared. After that, the mixture should be poured into the irrigation pipe. Finally, it should be decided which sectors of the layout of the field should be fertilized. Thus, Fig. 9 summarizes the categorization of every behavioral response, and the procedure is summarized in Fig. 10.

**Verb:** fertilize watering
**Notion:** activity that pursue the aim of adding nutrient to the plant.
**Behavioral responses:**
The farmer plans the mixture of minerals. (functionality of another software systems)
The farmer prepares the mixture of minerals. (activities to keep performing manually)

Received April 2023 Accepted July 2023; Published August 2023

The farmer pours the mixture into the irrigation pipe. (activities to keep performing manually)
The farmer plans the layout to fertilize. (functionality of the intended software application)
The farmer activates the irrigation pipe. (functionality of the intended software application)

**Fig. 9.** Categorization of the verb "Fertilize watering"

```
for each subject s categorized as user of the intended soft-
ware application
  for each behavioral response b that belong to s catego-
rized as functionality of the intended software application
      for each behavioral response v of the verb that de-
scribes b
      categorize v as
      (i) functionality of the intended software applica-
tion
      (ii) functionality of another software application
      (iii) activities to keep performing manually
```

**Fig. 10.** Procedure for behavioral responses of the verb categorization

### 3.5 Classification of objects

The behavioral responses should have the structure: subject + verb + object [3] where the object describes the element (material, resource, data) on which relies the action of the verb. Thus, if the verb (that is the behavioral response of the previous step) is categorized as "functionality of the intended software application", the object that receive the action, is probably an "object of the intended software application". Nevertheless, it could happen that the object is within the border of two different software applications. Thus, every object should be analyzed an categorized as (i) object within the boundaries of the intended software application, (ii) object within the boundaries of another software application, and (iii) object shared by several software applications.

The process of fertilizing through watering with the irrigation pipe should have access and control of the layout of the field. This is necessary to open and close the hatches to make the water (with the fertilizes) flows to the desired sector of the field. And of course, it should also have access to the pump to activate the irrigation pipe. Fig. 11 summarizes the categorization of every object, and Fig. 12 summarizes the procedure.

**Verb:** fertilize watering
**Notion:** activity that pursue the aim of adding nutrient to the plant.
**Behavioral responses:**
The farmer plans the mixture of minerals.
The farmer prepares the mixture of minerals.
The farmer pours the mixture into the irrigation pipe.

Received April 2023 Accepted July 2023; Published August 2023

The farmer plan the layout to fertilize. (object within the boundaries of the intended software application)
The farmer activates the irrigation pipe. (object within the boundaries of the intended software application)

**Fig. 11.** Categorization of the objects belonging to the “Fertilize watering”

```
for each subject s categorized as user of the intended soft-
ware application
   for each behavioral response b that belong to s catego-
rized as functionality of the intended software application
      for each behavioral response v of the verb that de-
scribes b
         for each object o of v
               categorize o as
               (i) object within the boundaries of the in-
         tended software application
               (ii) object within the boundaries of another
         software application
               (iii) object shared by several software ap-
         plications.
```

**Fig. 12.** Procedure for objects categorization

This step could also be used as a revision phase, since if a verb is considered to be “functionality of the intended software application”, and the object is considered “object within the boundaries of another software application”, it could be analyzed why the verb (functionality) is within the boundaries while the object (data) is outside.

In the example, as a result of the process applied, it is stated that the farmer will be the user of the software application that will provide the functionality to control the hatches of the field to fertilize as well as switching on and off the pump.

## 4 Evaluation

The framework proposed was applied to an application to manage sanitary resources related to covid-19. The system manages doctors, rooms, beds and patients. The system also manages the evolution of a patient and provides alerts according to certain workflow to follow the evolution of the patient.

Participants were 25 students of a degree course divided in 11 groups. The objective of the course is to provide a realistic experience in software development. In particular, the course emphasizes requirements practices. It is important to mention that most of the students have experience in industry since in Argentina, students generally begin to work in industry in second year of their undergraduate studies.

Participants received a glossary LEL already prepared and they had to apply the proposed approach. One of the professors of the course is a Medical Doctor, and he played the role of the client providing the information about what should be included in the intended software application. Another professor of the course checked the categorization of the elements of the glossary LEL.

Received April 2023 Accepted July 2023; Published August 2023

The evaluation was focused on the applicability of the approach. The System Usability Scale (SUS) [9] [10] was used to assess the usability and applicability of the approach. Although SUS is mainly used to assess usability of software systems, it was probe to be effective to assess products and processes [6]. The System Usability Scale (SUS) consists of a 10-item questionnaire; every question must be answered in a five-option 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 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 71,17. Thus, the approach can be considered as "acceptable".

## 5 Related work

Lee et al. [18] use domain knowledge information (and requirements documents) in natural language to create a richer knowledge base used to produce artifacts specific of the application domain. Their approach produces UML diagrams and source code. They propose a broader approach that ours. Nevertheless, they do not describe how to define the boundaries of the application to produce artifacts so specific like source code. Voelter et al. [28] use domain specific languages in product line engineering as a middle ground between feature modeling and programming. This approach pursues the same concern of our approach to define the boundaries of the software application even in a specific domain as software product lines.

Wang et al. [29] are concerned about reducing the gap between natural language requirements and Architecture Analysis and Design Language models. Although their objective is different from the one of our approach, in some way both concerns are related. Their approach is mainly based on data dictionaries and glossaries, which are the elements used by us. Borelli et al. [8] also propose an approach for architectural design. In particular, they work with IoT, and they propose a tool to analyze a Domain Specific Language (DSL) in order to obtain a new language called: A Buildout IoT Application Language (BIoTA).

Mukhtar et al. [24] consider the importance of identifying the vocabulary of a software application. They use general dictionaries to identify compound words that contain some atomic words. Then, experts analyze the terms and their definition to finally consider those terms. This approach could be used in a previous stage of our approach, where relevant concept analysis should be identified.

Bai et al. [4] [5] propose a strategy to build domain specific language models from general domain data. They search for similar topics in documents of related domains while we work with a specific domain to provide a description of a specific application. Moreover, Bai et al. work with natural language documents while we work with a semi structured natural language model as Language Extended Lexicon.

Received April 2023 Accepted July 2023; Published August 2023

Demsky et al. [13] developed an application called Bristlecone that relate high-level specification with the low level application's conceptual operations, which can be considered as a kind of vocabulary.

Doorn et al [15] propose a strategy to understand the future universe of discourse through the use of Scenarios [16]. Their strategy relies on constructing future scenarios (and requirements) in order to obtain the future universe of discourse captured by LEL. This strategy is more complex that our approach since they try to define the future universe of course, while our approach only limit the application domain universe of discourse. We plan to develop an extension of our proposed approach to obtain the language of the future application system, but we believe that defining the limits of the universe of discourse beforehand is more important. Haj et al. [17] propose an approach based on natural language processing that to obtain Semantic of Business Vocabulary and Rules (SBVR). Lie et al. [21] proposed an approach to develop the software application language with a semantic support. They use User Stories as input and analyze them in order to obtain relevant concept which are linked with wordnet in order to have a semantic support. It is an interesting idea, but the wordnet dictionary is a general way, so it could be hard to relate the concept with the correct definition.

Dilshener et al. [14] performed an analysis about the relationship between the concept that appear in the source code and the use of those concepts in a more abstract and conceptual artefact. This works is a kind of verification that both vocabularies should be synchronized in order to make easy the software development process. Amatriain et al. [1] performed a similar analysis showing that they were able to develop a framework with minimal overhead thanks to the use of vocabularies, in particular a domain specific language (DSL). Nascimento et al. [25] performed an analysis of the vocabulary used in the source code in order to assess the level of understanding of the students that has written the source code. It is interesting to emphasize how the use of language can reduce the effort and it also can be used to assess the understanding of the domain.

## 6 Conclusions

This paper presented an approach to define the language of the software application from the language of the application domain. The process of acquiring and identifying the relevant requirements and knowledge to develop a new software application can be overwhelming in some situation of a complex domain where a lot of stakeholders are involved. Thus, this paper proposes an approach that uses the LEL glossary to capture the vocabulary of the application domain and obtain the LEL glossary of the software application in a straightforward way with different steps of analyzing the concepts defined by the language and categorizing them according to their situation in the boundaries of the software application. This process should be considered as a framework that filter the elements of the application domain and define the scope of the software application at the same time. The proposed approach only consider the reduction of the initial LEL glossary, but the analyst who applies the proposed approach will acquire more information during the process and he could add this new knowledge to the LEL glossary obtained as a result. This specific improvement of the

Received April 2023 Accepted July 2023; Published August 2023

proposed approach is a future work. We are currently analyzing scenarios where the language changes or is enriched in order to propose a new approach that also consider this situations.

## Acknowledgment

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

## References

1. Amatriain, X., Arumi, P.: "Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain," in IEEE Transactions on Software Engineering, vol. 37, no. 4, pp. 544-558, July-Aug., doi: 10.1109/TSE.2010.48 (2011).
2. Antonelli, L., Rossi G., Leite, J.C.S.P., Oliveros, A: “Deriving requirements specifications from the application domain language captured by Language Extended Lexicon”, Workshop in Requirements Engineering (WER), Buenos Aires, Argentina, April 24 – 27 (2012).
3. 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, Brazil, August 23 – 27 (2021).
4. Bai, S., Zhang, N., Li, H.: "Semi-supervised Learning of Domain-Specific Language Models from General Domain Data", in Proceeding of the International Conference on Asian Language Processing (IALP '09), ISBN 978-0-7695-3904-1, pp 273 - 279 (2009)
5. Bai, S., Huang, C.L.,Tan, Y.K., Ma, B.: Language models learning for domain-specific natural language user interaction, in proceeding of the IEEE International Conference on Robotics and Biomimetics (ROBIO), ISBN 978-1-4244-4774-9, pp 2480 - 2485 (2009).
6. Bangor, A., Kortum, P. T., Miller, J. T. "An empirical evaluation of the system usability scale." Intl. Journal of Human–Computer Interaction 24.6, pp. 574-594 (2008).
7. Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.
8. Borelli, F. F., Biondi, G. O., Kamienski, C. A.: "BIoTA: A Buildout IoT Application Language," in IEEE Access, vol. 8, pp. 126443-126459, doi: 10.1109/ACCESS.2020.3003694 (2020).
9. Brooke, J. “SUS-A quick and dirty usability scale. Usability evaluation in industry”, 189(194), 4-7, (1996).
10. Brooke, J. "SUS: a retrospective." Journal of usability studies 8.2, pp.29-40, (2013).
11. Brooks, F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition 1995.
12. 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.

Received April 2023 Accepted July 2023; Published August 2023

13. Demsky B., Sundaramurthy, S.: "Bristlecone: Language Support for Robust Software Applications," in IEEE Transactions on Software Engineering, vol. 37, no. 1, pp. 4-23, Jan.-Feb., doi: 10.1109/TSE.2010.27 (2011)
14. Dilshener, T. Wermelinger, M.: "Relating developers' concepts and artefact vocabulary in a financial software module," 2011 27th IEEE International Conference on Software Maintenance (ICSM), pp. 412-417, doi: 10.1109/ICSM.2011.6080808 (2011).
15. Doorn, J.H., Hadad, G.D.S., Kaplan, G.N.: "Comprendiendo el Universo de Discurso Futuro", WER'02 - Workshop en Ingeniería de Requisitos, Valencia, España, pp.117-131, Noviembre 2002.
16. Hadad, G., Kaplan, G., Oliveros, A., Leite, J.C.S.P.: Construcción de Escenarios a partir del Léxico Extendido del Lenguaje, in Proceedings SoST, 26JAIIO, Sociedad Argentina de Informática y Comunicaciones, Buenos Aires (1997)
17. Haj, A., Jarrar, A., Balouki, Y. Gadir, T.:"The Semantic of Business Vocabulary and Business Rules: An Automatic Generation From Textual Statements," in IEEE Access, vol. 9, pp. 56506-56522, doi: 10.1109/ACCESS.2021.3071623 (2021).
18. Lee, B. S., Bryant, B.R.: Automation of software system development using natural language processing and two level grammar, In Proceeding of the Workshop Radical Innovations of Software and Systems Engineering in the Future, Monterey, 244-257 (2002)
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. 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
21. Liu, Y., Lin, J., Cleland-Huang, J., Vierhauser, M., Guo, J., Lohar, S.: "SENET: A Semantic Web for Supporting Automation of Software Engineering Tasks," 2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), pp. 23-32, doi: 10.1109/AIRE51212.2020.00011 (2020).
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. Meservy, T. O., Zhang, C., Lee, E. T. and Dhaliwal, J.: "The Business Rules Approach and Its Effect on Software Testing," in IEEE Software, vol. 29, no. 4, doi: 10.1109/MS.2011.120, pp. 60-66, July-Aug, 2012.
24. Mukhtar, T., Afzal, H. Majeed, A.: "Vocabulary of Quranic Concepts: A semi-automatically created terminology of Holy Quran," 2012 15th International Multitopic Conference (INMIC), pp. 43-46, doi: 10.1109/INMIC.2012.6511467 (2012).
25. Nascimento, M., Araújo, E., Serey D., Figueiredo, J.: "The Role of Source Code Vocabulary in Programming Teaching and Learning," 2020 IEEE Frontiers in Education Conference (FIE), pp. 1-8, doi: 10.1109/FIE44824.2020.9274137 (2020).
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. Rose, S., Nagy, G.: Formulation: Document examples with Given/When/Then, Independently published, 979-8723395015, 2021.

Received April 2023 Accepted July 2023; Published August 2023

28. Voelter, M., Visser, E.: Product Line Engineering Using Domain-Specific Languages in proceeding of the 15th International Software Product Line Conference (SPLC) ISBN 978-1-4577-1029-2, pp 70-79 (2011)
29. Wang, F. et al.: "An Approach to Generate the Traceability Between Restricted Natural Language Requirements and AADL Models," in IEEE Transactions on Reliability, vol. 69, no. 1, pp. 154-173, March, doi: 10.1109/TR.2019.2936072 (2020).

Received April 2023 Accepted July 2023; Published August 2023