An approach for Reverse Engineering from Web Applications into the Language of the Domain using the LEL Glossary

Angela Verónica Granizo Rodríguez[1,3][0000-0001-8117-275X], Leandro Antonelli[1,2][0000-0003-1388-0337], Sergio Firmenich[1,4][0000-0001-9502-2189], and Diego Firmenich[5][0000-0002-7212-4454]

[1] LIFIA, Facultad de Informática, UNLP. La Plata. Calle 50 y 120, S/N. Buenos Aires. Argentina.
[2] CAETI, Facultad de Tecnología Informática, Universidad Abierta Interamericana. Buenos Aires. Argentina.
[3] Escuela Superior Politécnica de Chimborazo. Panamericana Sur km 1 y 1/2 . Riobamba. Ecuador
[4] CONICET. Godoy Cruz 2290. Ciudad Autónoma de Buenos Aires. Argentina.
[5] Departamento de Informática, Facultad de Ingeniería, Universidad Nacional de la Patagonia San Juan Bosco. Argentina.

{vgranizo, lanto, sergio.firmenich}@lifia.info.unlp.edu.ar
dafirmenich@ing.unp.edu.ar

Abstract. Requirement engineering plays a crucial role in the software lifecycle, since errors made in the requirements require significant effort to be corrected in later stages. The main source of requirements is people; however, it is common to analyze existing applications when developing new software. This is particularly the case in the process of reengineering. On the other hand, the language of the domain is essential to understanding the domain and thus comprehending the requirements. Language Extended Lexicon (LEL) is a structured glossary designed to capture this language. This paper proposes an approach for obtaining the language of an application domain from a web application using the LEL glossary. The process comprises three main activities: general analysis of the web application, domain language capture, and the verification of the generated domain language. Additionally, this paper describes a web browser extension tool designed to support the process. Finally, the paper presents the results of a preliminary evaluation with promising outcomes regarding the applicability of the approach.

Keywords: Reverse engineering · Software web applications · Domain language · LEL · Software requirements · Application domain · Software application.

1 Introduction

Requirement engineering is a very important stage in the software lifecycle, as errors made in the requirements need a great effort to be corrected in the subsequent stages [10]. Furthermore, it is a critical stage in software development because if the requirements are not correct, the software development team will create an artifact that will not meet the customer's expectations [17]

Understanding the language of the application to gather requirements is crucial because, essentially, if the language is not understood, it will not be possible to write requirements, or they will not meet the necessary quality. The LEL, which stands for "Language Extended Lexicon", is a glossary [21] to understand the language of an application domain without the need to be concerned about the application's software. The LEL categorizes terms into four categories: subjects, objects, verbs, and states, and employs two attributes: notion and behavioral responses, to describe these terms. It has been demonstrated that the LEL glossary is an appropriate method for capturing language. There are some early reports about three significant characteristics of an LEL: it is easy to learn, it is easy to use, and it has good expressiveness [15]. Besides, the LEL can be used to obtain requirements [7]; thus, it can be considered as a first step in requirements elicitation.

Traditional software engineering involves starting with requirements to develop an application. Typically, requirements are gathered from people or documentation, but at times, people's involvement may not be available, being the main source of requirements. With the growth of the software industry, it has become common to analyze existing applications to obtain requirements when developing new software. Extracting requirements from other systems for subsequent development is known as reverse engineering [16].

Applications are a suitable source to understand language, as they contain packaged knowledge about the domain [14]. Since capturing language requires significant effort, it is crucial to have tools that simplify this task. Web browser extension technologies allow adding applications to any web page. It would be desirable to have a web browser extension that enables capturing the language of an application directly from it.

The goal of this paper is to propose a method for reverse engineering to obtain the language of an application domain from a web application. The method uses the glossary LEL as a template to capture the language. The method is partially supported by a tool that is a web browser extension during the domain language capture stage. The paper also shows a preliminary validation of the method using the SUS [12] survey.

The rest of the paper is organized in the following way. Section 2 describes the background necessary to understand the proposed approach. Section 3 describes the proposed approach. Sections 4 and 5 provide evidence about the applicability of the approach. Section 6 discusses some related works. Finally, section 7 presents some conclusions.

2 Language Extended Lexicon

The Language Extended Lexicon (LEL) is a glossary that describes the language of an application domain, where a definition of a software application is not necessarily present. LEL aims to record the definitions of terms that belong to a domain to "understand the language of a problem without worrying about the problem" [20].

A prerequisite for understanding the domain involves learning the language used in that domain (LEL), which is why building an LEL is so important. Within an organization, experts, end-users, and customers have some knowledge of the domain, but they have different and complementary perspectives. Therefore, LELs must provide a unified and coherent presentation of the language used by them. Language is represented through symbols, which can be terms or short expressions, defined through two attributes: notion and behavioral responses. Notion describes denotation, i.e., the intrinsic and substantial characteristics of the symbol, while behavioral responses describe the connotation of the symbol, i.e., the relationship between the term being described and other terms (Table 1 [4]).

Table 1. Template to describe an LEL symbol

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

Each symbol in 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 attributes. Furthermore, it allows for the organization of domain concepts and provides a template for describing attributes. Table 2 [4] outlines each category along with its characteristics and how to describe them.

Table 2. 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 Approach

The proposed approach aims to analyze a web application (this is the input of the approach) and obtain the knowledge and requirements from it described through its language, which constitutes the output of the approach. Particularly, the proposed approach uses a glossary LEL to describe its language. Thus, the proposed approach is a reverse engineering method, since from an application, it obtains a specification of the knowledge and requirements where both can be used to develop another application.

The proposed approach consists of three main stages carried out by the requirements engineer: (A) general analysis of the web application, (B) domain language capture, and (C) verification of the generated domain language. The method is partially supported by a web browser extension during the domain language capture stage.

Each of these stages is organized into steps, with each step involving the execution of one or more activities. This structured approach ensures thoroughness and effectiveness in the implementation of each stage.

The first stage (general analysis of the web application) aims to conduct an initial and preliminary study of the web application in order to understand its goal, data, and functionality.

The second stage (domain language capture) aims to conduct a detailed study of the web application in order to identify essential components and define glossary expressions based on these components.

The third stage (verification of the generated domain language) aims to review the definitions of the expressions, both individually and also the consistency among different terms. Figure 1 summarizes the proposed approach. It is important to mention that the proposed approach is not strictly a sequential pipeline, in fact stages two and three can be performed iteratively.

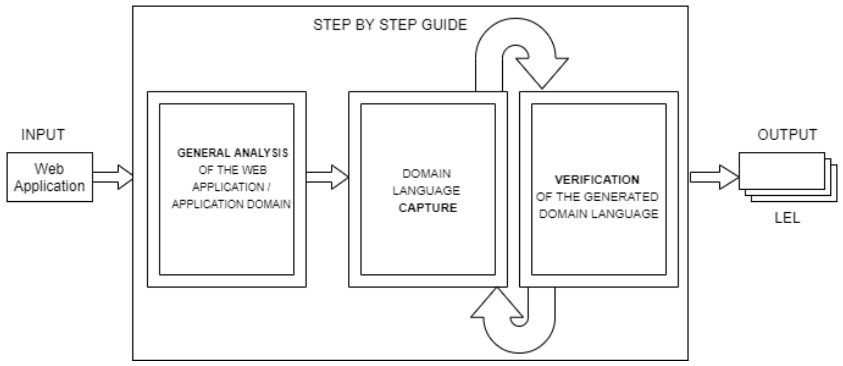

STEP BY STEP GUIDE
INPUT
Web Application
GENERAL ANALYSIS OF THE WEB APPLICATION / APPLICATION DOMAIN
DOMAIN LANGUAGE CAPTURE
VERIFICATION OF THE GENERATED DOMAIN LANGUAGE
OUTPUT
LEL

Fig. 1. The proposed approach

The following subsections describe in detail each one of the stages. Every stage is exemplified using a website to sell agricultural products. It provides phytosanitary products for crop protection and nutrition, pest control, and both general and specific agricultural consultancy. To facilitate this, it offers a categorized knowledge base of crop solutions, as well as information about local vendors. They also offer the opportunity to contact agricultural professionals and make inquiries. It is important to mention that the website provides information about African professionals.

3.1 General Analysis of the Web Application

This stage (the first one of the approach), consists in turn of two steps. On one hand, conducting a (i) general analysis of the web application to be studied, and on the other hand, conducting an (ii) exploratory study of the application's domain. That is, both elements are studied: the web application and the application's domain.

To carry out step (i) (the general analysis of the application to be studied), it is necessary to perform three activities. First, it is necessary to navigate the application exploratively to understand its purpose. This, in turn, should be described in a short sentence that starts with a verb. For example: the objective of the web application Greenlife Crop Protection Africa is "supplying phytosanitary products for crop protection, nutrition, pest control, and general agricultural consultancy in agricultural domains in Africa".

Second, it is necessary to navigate the application in more detail in order to create its navigation map. To create this map, squares should be used to identify the pages, and lines with arrows to indicate the direction of the navigation. It is important to note that pages should be identified conceptually. As an example for clarification, consider an application that sells products; it should be identified one square as the product description page does not matter how many products are sold.

The Greenlife Crop Protection Africa website features a main page from which it is possible to access three different sections: crop solutions, products, and services. From the crop solutions page, various categories can be viewed, ultimately allowing users to find a specific solution. In the products section, different types of products are presented, making it easy to select a particular one, from where it is also possible to locate a distributor. Finally, the services section offers three options: asking a question, finding a professional agronomist, and locating places to purchase the products. The navigation map for the website depicted in Figure 3 is shown in Figure 2.

Figure 3 shows some snapshots from the website used. The first webpage is the home page. Next, the webpage on the right shows the types of products, followed by the product information. Finally, the webpage on the left presents the page for locating a distributor. These pages can be found in the navigation map described in Figure 2. Thus, first webpage is the root of the navigation map. The second web page is the node "types of products" in the second level. Then, the third web page is the node "product information" in the third level, and the fourth web page is the node "locate a distributor" in the fourth level.

Third, it is necessary to list the general functionality that the application provides. For example, the previously mentioned Greenlife web application is an African website that offers the following functionalities: it provides phytosanitary products for crop protection and nutrition. It offers both general and specific agricultural consulting services. It includes a knowledge base of crop solutions

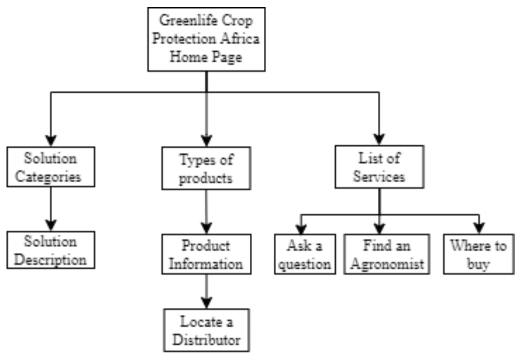


Fig. 2. The navigation map for Greenlife Crop Protection web application sections

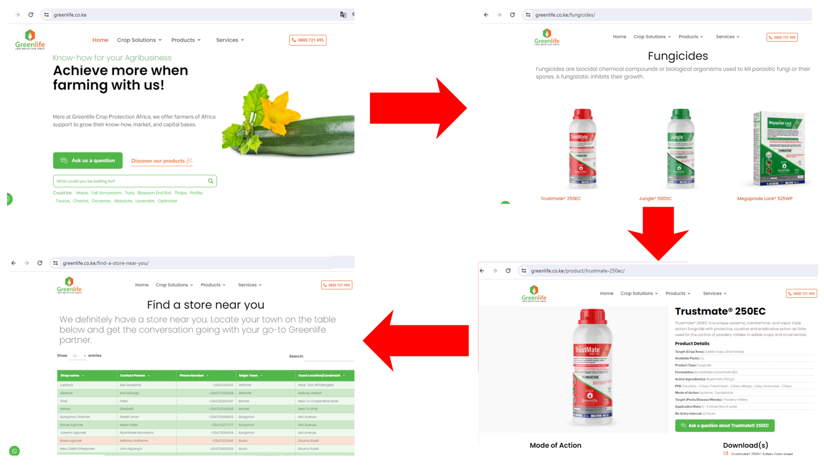


Fig. 3. Website example

organized by categories. It provides information about local vendors and agricultural professionals.

To carry out (ii) the exploratory study of the domain, it is necessary to refer to additional documentation based on what was obtained in (i).

3.2 Domain Language Capture

In this stage (the second one of the approach), the identification, categorization (subject, object, verb, and state) and description of symbols are carried out. The process involves defining the terms of the LEL (symbols) and linking them to the web application. The notion and behavioral response for each identified symbol should be described. It is important to note that exploring all pages of the site is necessary to capture the domain language comprehensively.

This stage (domain capture language) is composed of three steps: (i) identify symbols and their categories, (ii) describe the notion, and (iii) describe the behavioral responses.

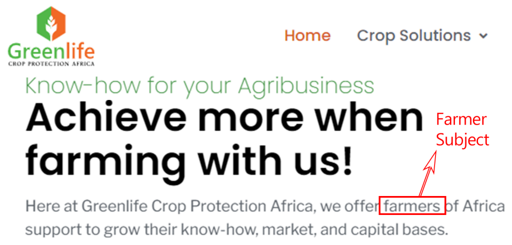
The first step (identifying symbols and their categories) involves navigating through the entire web application, using the navigation map, to identify and categorize symbols. As a result, there will be a list of symbols, along with their categories, and a "link" to the HTML where the symbol appears. It is worth noting that the same symbol could be "linked" to several elements on different pages.

The second step (describe the notion) involves defining the notion of the symbols. It is advisable to do this as a second step because, after navigating through the entire web application and "linking" the symbols to the HTML, there is a better understanding of the symbols. In this way, the description of the notion becomes more comprehensive.

The third step (describe the behavioral responses) involves writing sentences like "a certain subject performs an action on a certain object," where the subject, action, and object should be symbols identified in the LEL.

Regarding the activities to carry out the first step (identifying symbols and their categories), each element of the web application should be analyzed and categorized as a subject, object, verb, or state.

For the identification and categorization of subjects, it is necessary to recognize one of the following situations: (i) each user role, (ii) any element in the web application (text, images, or any piece of information from any medium) representing a person or organization must correspond to a subject symbol. Its title is the text of a user role, person, or organization. For example, farmer, agronomist, or company. See Figure 4.



Know-how for your Agribusiness

Achieve more when farming with us!

Here at Greenlife Crop Protection Africa, we offer farmers of Africa support to grow their know-how, market, and capital bases.

Fig. 4. Farmer Subject linked in the Greenlife web application - home page

For the identification and categorization of objects, it is necessary to recognize any element in the web application (text, images, or any piece of information from any medium) that represents resources, tools, or data. Its title is a common noun or a short phrase representing the passive element. For example, 'phytosanitary product,' 'question,' 'best product,' 'store,' 'crop solution,' or 'technical assistant.'

For the identification and categorization of verbs, it is necessary to recognize one of the following situations: (i) each button, (ii) any element of the web application (text, images, or any piece of information from any source) that performs an action must correspond to a verb symbol. Its title is the text of the element, and a verb name in infinitive is chosen. For example, the verb "ask a question" on the home page of the Greenlife web application.

States are situations in which subjects, objects, or verbs can find themselves. For the identification and categorization of states, it is necessary to recognize any element of the web application (text, images, or any piece of information from any source) that can be in a certain state. Its title is the name of the transition of the identified subject, object, or verb. For example, "pending receipt of response to a question state".

The second step (describe the notion) involves gradually defining the notion of the symbols identified in the first step.

The notion of the identified subject is the characteristics or conditions that the subject satisfies. It can be specified with words such as "is," "has," or any other characteristic. Table 3 provides an example of the farmer subject and its notion.

Table 3. Farmer Subject - Notion

Subject:	Farmer
Notion:	It is an anonymous user of the web application
	They navigate the website to obtain agricultural information

The notion of the identified object refers to its characteristics or attributes, and it can be specified using words such as 'is,' 'has,' or 'is characterized by.' For example, a question object is composed of a location, crop type, full name, email, phone number, farming county, and the text of the question.

The notion of the identified verb refers to the goal it pursues and can be specified through phrases that answer questions such as "what for" or "why" the verb exists. For example, "Action to ask a question to an agronomist or technical assistant."

The notion of the identified state is the represented situation; for example, the pending state of receiving a response via email or phone to a question asked by the farmer. For example, it is a situation where a farmer is waiting for the answer of some expert from the website.

The third step (describe the behavioral responses) consists of describing the behavioral responses of the identified symbols.

The behavioral responses of the identified subject are the actions it performs. Table 4 provides an example of the farmer subject and its behavioral responses. A subject may have neither notion nor behavioral responses, as in the case of the company subject.

Table 4. Farmer Subject - Behavioral responses

Subject:	Farmer
Notion:	It is an anonymous user of the web application
	They navigate the website to obtain agricultural information
Behavioral responses:	The farmer asks a question
	The farmer consults the best phytosanitary product
	The farmer searches for an agricultural topic
	The farmer contacts an agronomist
	The farmer consults a store

The behavioral responses of the identified object are the actions that are performed on the object.

The behavioral responses of the identified verb involve describing the necessary steps to carry out the action that the element performs. In other words, it entails explaining how the steps or actions associated with the verb are executed. The behavioral responses are obtained by clicking or interacting with the element and observing the actions that are taken.

The behavioral responses of the identified state are the actions that must be carried out to transition to another state. The subsequent state is derived from the behavioral responses of the preceding state.

3.3 Verification of the Generated Domain Language

This involves reviewing the symbols from the stage of capturing the domain language and making adjustments if necessary. As a result, we obtain the verified symbols. It consists of three steps: (i) verification of the description of each symbol according to the proposed template in tables 1 and 2 of the background section, (ii) searching for repetitions, and (iii) identification of new symbols.

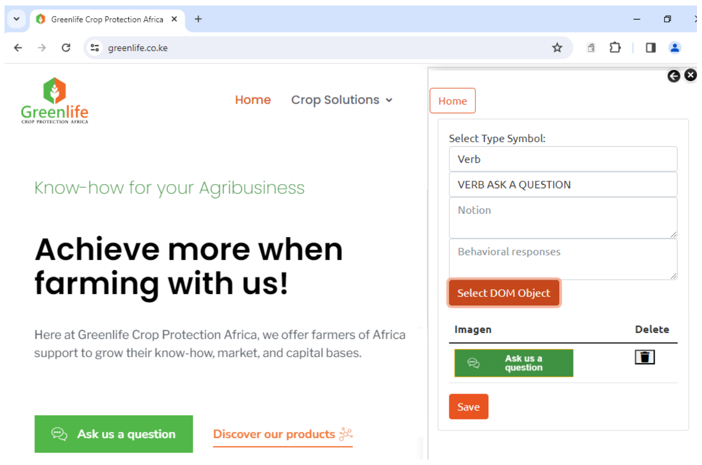
The first step, involving the verification of the description of each symbol according to the proposed template (internal consistency), consists of reviewing both the accuracy of the notion and its behavioral responses, thus achieving a better understanding of the identified symbol. It is also possible to review the correct definition of the symbol's title.

In the second step, searching for repetitions (external consistency), it may happen that the same symbol is found on the website in two different places (with a different title) and is defined twice. In such a case, the symbol is repeated, so a new symbol is created with both titles and the sum of the definitions. Another case is having the same symbol repeated, and the same title is used.

The third step, the identification of new existing symbols, involves reviewing the definition of a symbol and realizing that there is a symbol worthy of being defined. This new symbol also has to appear on the website being analyzed. It is added to the LEL glossary, following the steps of capturing the domain language. Subsequently, these added symbols would be verified according to the first and second activities of this verification stage.

4 Tool Support

A browser extension tool for the Google Chrome web browser was implemented to support the application of the proposed approach during the domain language capture stage. Figure 5 shows an example of the Greenlife Crop Protection Africa website that has been worked on. To the right is a box with a view of the developed web browser extension.



Know-how for your Agribusiness

Achieve more when farming with us!

Here at Greenlife Crop Protection Africa, we offer farmers of Africa support to grow their know-how, market, and capital bases.

Select Type Symbol: Verb

VERB ASK A QUESTION

Notion

Behavioral responses

Fig. 5. Web browser extension

The architecture of our tool has the following structure: a user interface layer through a web browser extension, a microservices layer, and a data layer. Additionally, our tool was developed using JavaScript. In the front end, we utilized the AngularJS [2] framework, while in the backend, we employed the Express [24] framework. In addition, the database we utilized is MongoDB [22]. Our tool operates within the web browser as an extension of it. The main functionality of the web browser extension developed focuses on identifying symbols from the LEL glossary within an already developed web application. This is achieved through web augmentation techniques, visually capturing each DOM element (image), its location on the site, and the XPATH of DOM object. All of this is stored in JSON format within the database.

5 Evaluation

The proposed approach was evaluated using the web browser extension tool designed to support the process. Specifically, the second stage of the approach was assessed, involving the capture of domain language related to the identification of symbols and categories, as well as the description of notions and behavioral responses.

The participants in the evaluation were 12 members of a research project at the University of the Plata, in Argentina. All participants have experience in software development and they played the role of requirements engineers to perform reverse engineering and had to identify an object symbol anywhere on the IMDb (Internet Movie Database) website and write down the notion and behavioral responses following the proposed approach. The mentioned website is a well-known online database for movies, television, and video games. A guide was provided, and participants carried out the activity explained above with the support of the web browser extension. It is important to note that participants had no experience using the LEL glossary. They received training on the proposed approach related to the domain language capture stage before the experiment.

The System Usability Scale (SUS) [12] [13] was used to assess the applicability of the proposed approach. Although the SUS is primarily employed to evaluate the usability of software systems, it has proven effective in assessing processes and products [9]. It consists of a 10-item questionnaire; each question must be answered on a five-point scale, ranging from "1" ("Strongly Disagree") to "5" ("Strongly Agree"). Despite having 10 questions, they are paired, asking the same question but from a complementary perspective to obtain a result of high confidence. The SUS score is calculated as follows. Firstly, items 1, 3, 5, 7, and 9 are scored considering the ranked value minus 1. Then, items 2, 4, 6, 8, and 10 are scored considering 5 minus the ranked value. Afterward, each participant's scores are summed and then multiplied by 2.5 to obtain a new value ranging from 0 to 100. Finally, the average is calculated. The approach can fall into one of the following categories: "Not acceptable" 0-64, "Acceptable" 65-84, and "Excellent" 85-100 [15]. The obtained score was 71.04. Therefore, the approach can be considered "acceptable."

6 Related works

The use of reverse engineering to derive requirements from previously developed systems has been explored from various perspectives. The approach by Hassan et al. [19] focuses on extracting requirements from the source code of a legacy system. Our approach aims to obtain and understand the language of a domain from a web application using the LEL. Both approaches use reverse engineering, but they apply to different contexts and use other methods to achieve their goals.

Similarly, with Aman et al. [1], we use reverse engineering to gather information about requirements. However, they present a framework based on XML that creates a UML framework to generate software requirements specifications. Fahmi et al. highlight the concept of employing reverse engineering in application renewal tasks [16]. This approach identifies retained functions, redundancies, and reusable elements, aligning with our goal of deeper domain language understanding through reverse engineering. Tramontana [27] introduces an approach for reverse engineering web applications, differing from our method in the specific details of the employed methodology, particularly in the use of reverse engineering along with UML diagram reconstruction.

Su et al. [26] propose an aspect-oriented software reverse engineering framework for understanding crosscutting properties in legacy systems at the requirements level. In contrast, our approach emphasizes understanding the domain-specific language through the LEL glossary and reverse engineering. Sabir et al. [25], propose a model-driven reverse engineering (MDRE) framework called "Source to Model Framework (Src2MoF)" to generate structural (class) and behavioral (activity) diagrams of the Unified Modeling Language (UML) from Java source code. Both approaches share the application of reverse engineering. However, their approach produces UML diagrams, unlike ours which generates an LEL. Bolchini et al. [11], introduces a lightweight methodology that combines goal-directed requirements engineering and scenario-based techniques. While our approach aims to extract the language of a domain from a web application, theirs focuses on conceptual tools and a lightweight methodology for requirements analysis in web applications.

Mukhtar et al.[23] use general dictionaries to identify compound words that contain fundamental or atomic words. While our approach focuses on reverse engineering from web applications to the domain language, theirs focuses on understanding the specific vocabulary of a software application. The application's vocabulary is a subset of domain language. Antonelli et al. [6] propose and validate a strategy for collaboratively capturing the domain language using the LEL. Our approach focuses on obtaining the domain language from the web application. Garrido et al. [18] propose an agile methodology for building mathematical programming models using LEL and scenarios. Both approaches use LEL to capture the domain language but differ in the specific application, domain of interest, and methodology used. Also, Antonelli et. al [5] use Kernel sentences as input and use cases as output. These can be integrated into the LEL, resulting from our methodology. Besides Antonelli et al. [3] propose an approach that creates a multidimensional schema from the language of the domain captured through the LEL. The LEL from our web application can be used as input for this approach. In another research work, Antonelli et al. [4], propose an approach to consider the language of the application domain, captured through its vocabulary, to refine it and obtain a language limited to the boundaries of the software application. In our case, the input comes from a web application, and the output is an LEL, which could be used as input for their approach. Other research work from Antonelli et al. [8] proposes a collaborative approach to derive a conceptual model from specifications in natural language using kernel sentences. Although it differs from our approach, it could be integrated into the behavioral responses section of the LEL using kernel sentences.

7 Conclusions and future work

This paper proposes an approach for reverse engineering to obtain the language of an application domain from a web application using the LEL glossary. The approach consists of three main activities: general analysis of the web application, domain language capture, and the verification of the generated domain language. A preliminary evaluation was performed to show the applicability of the approach. This paper also presents a web browser extension tool designed to support the process.

The language of the domain is essential to understanding the domain and thus comprehending the requirements and if they have errors require significant effort to be corrected in later stages of the software development. Also, it is common to analyze existing applications for developing new software. LEL is a structured glossary designed to capture the language of the domain and it is the contribution proposed in this paper that generates an LEL to extract the language of a domain from a web application.

To refine and improve the proposed method, it is proposed to conduct a more comprehensive evaluation through a case study. Also, the effectiveness of the approach will be demonstrated and a baseline will be established to compare the tool's performance with human performance.

References

1. Aman, H., Ibrahim, R.: Reverse engineering: From xml to uml for generation of software requirement specification. In: 2013 8th International Conference on Information Technology in Asia (CITA). pp. 1–6 (2013). https://doi.org/10.1109/CITA.2013.6637575
2. AngularJS: https://angular.io/, accessed 2024-03-18
3. Antonelli, L., Bimonte, S., Rizzi, S.: Multidimensional modeling driven from a domain language. Automated Software Engineering **30**(1), 6 (2022). https://doi.org/10.1007/s10515-022-00375-5
4. Antonelli, L., Leite, J., Oliveros, A., Rossi, G.: Defining the language of the software application using the vocabulary of the domain. Electronic Journal of SADIO **22**(3), 2–14 (2023)
5. Antonelli, L., do Prado Leite, J.C.S., Oliveros, A., Rossi, G.: Specification cases: a lightweight approach based on natural language. In: Workshop em Engenharia de Requisitos (2021). https://doi.org/10.29327/1298728.24-5.
6. Antonelli, L., Rossi, G., Oliveros, A.: A collaborative approach to describe the domain language through the language extended lexicon. Journal of Object Technology **16**(3), 1–27 (Jun 2016). https://doi.org/10.5381/jot.2016.15.3.a3
7. Antonelli, L., Rossi, G., do Prado Leite, J.C.S., Oliveros, A.: Deriving requirements specifications from the application domain language captured by language extended lexicon. In: Anais do WER12 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, April 24-27, 2012 (2012)
8. Antonelli, L., Ville, J.D., Adorno, M., Ballestero, L., Cecconato, S., Fernández, A., Maclen, G., Maltempo, G., Mattei, J., Tanevitch, L., Torres, D.: An approach to extract a conceptual model from natural language specifications. In: Antonelli, L., Lucena, M., Portugal, R.L.Q. (eds.) Anais do WER23 - Workshop em Engenharia de Requisitos, Porto Alegre, RS, Brasil, Agosto 15-17, 2022. LFS (UFRN, Brasil) (2023). https://doi.org/10.29327/1298356.26-12
9. Bangor, A., Kortum, P.T., Miller, J.T.: An Empirical Evaluation of the System Usability Scale. Intl. Journal of Human-Computer Interaction **24**(6), 1–44 (2008). https://doi.org/10.1080/10447310802205776, https://doi.org/10.1080/10447310802205776

10. Boehm, B.W.: Software Engineering. Computer society Press, IEEE (1997)
11. Bolchini, D., Paolini, P.: Capturing web application requirements through goal-oriented analysis. In: WER. pp. 16–28 (2002)
12. Brooke, J.: "SUS-A quick and dirty usability scale." Usability evaluation in industry. CRC Press (June 1996), https://www.crcpress.com/product/isbn/9780748404605, iSBN: 9780748404605
13. Brooke, J.: SUS: a retrospective. Journal of usability studies **8**(2), 29–40 (2013)
14. Brooks, F.P.: The Mythical Man-Month. Addison-Wesley Professional, 2 edn. (1997)
15. Cysneiros, L.M., do Prado Leite, J.C.S.: Using the language extended lexicon to support non-functional requirements elicitation. In: Proceedings of the Workshop em Engenharia de Requisitos. pp. 139–153. Buenos Aires, Argentina (2001)
16. Fahmi, S.A., Choi, H.J.: Software reverse engineering to requirements. In: 2007 International Conference on Convergence Information Technology (ICCIT 2007). pp. 2199–2204 (2007). https://doi.org/10.1109/ICCIT.2007.228
17. Forsberg, K., Mooz, H.: The relationship of system engineering to the project cycle. In: Proceedings of the First Annual Symposium of National Council on System Engineering. pp. 57–65 (1991)
18. Garrido, A., Antonelli, L., Martin, J., Alemany, M., Mula, J.: Using lel and scenarios to derive mathematical programming models. application in a fresh tomato packing problem. Computers and Electronics in Agriculture **170**, 105242 (2020). https://doi.org/https://doi.org/10.1016/j.compag.2020.105242, https://www.sciencedirect.com/science/article/pii/S0168169919317338
19. Hassan, S., Qamar, U., Hassan, T., Waqas, M.: Software reverse engineering to requirement engineering for evolution of legacy system. In: 2015 5th International Conference on IT Convergence and Security (ICITCS). pp. 1–4 (2015). https://doi.org/10.1109/ICITCS.2015.7293021
20. Leite, J., Franco, A.: A strategy for conceptual model acquisition. In: Proceedings of the IEEE International Symposium on Requirements Engineering. pp. 243–246 (1993)
21. Meservy, T.O., Zhang, C., Lee, E.T., Dhaliwal, J.: The business rules approach and its effect on software testing. IEEE Software **29**(4), 60–66 (2012). https://doi.org/10.1109/MS.2011.120
22. Mongodb: https://www.mongodb.com/es, accessed 2024-03-18
23. Mukhtar, T., Afzal, H., Majeed, A.: Vocabulary of quranic concepts: A semi-automatically created terminology of holy quran. In: 2012 15th International Multitopic Conference (INMIC). pp. 43–46 (2012). https://doi.org/10.1109/INMIC.2012.6511467
24. Nodejs: https://nodejs.org/en, accessed 2024-03-18
25. Sabir, U., Azam, F., Haq, S.U., Anwar, M.W., Butt, W.H., Amjad, A.: A model driven reverse engineering framework for generating high level uml models from java source code. IEEE Access **7**, 158931–158950 (2019). https://doi.org/10.1109/ACCESS.2019.2950884
26. Su, Y., Zhou, X.W., Zhang, M.Q.: Approach on aspect-oriented software reverse engineering at requirements level. In: 2008 International Conference on Computer Science and Software Engineering. vol. 2, pp. 321–324 (2008). https://doi.org/10.1109/CSSE.2008.834
27. Tramontana, P.: Reverse engineering web applications. In: 21st IEEE International Conference on Software Maintenance (ICSM'05). pp. 705–708 (2005). https://doi.org/10.1109/ICSM.2005.77