Methodology supported by a tool for Capturing Domain Language in Web Applications

Angela Verónica Granizo Rodríguez[1,3], Leandro Antonelli[1,2], Sergio Firmenich[1,4], Diego Firmenich[5]

[1]LIFIA, Facultad de Informática – Universidad Nacional de la Plata (UNLP)
Calle 50 y 120, S/N – La Plata – Argentina

[2]CAETI, Facultad de Tecnología Informática – Universidad Abierta Interamericana
Buenos Aires. Argentina

[3]Escuela Superior Politécnica de Chimborazo (ESPOCH). Panamericana Sur km 1 y ½
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. *Requirements engineering is crucial in the software life cycle, as errors in requirements can be costly to correct in later stages. While people are the main source of requirements, analyzing existing applications is common, especially in reengineering. The Extended Lexicon of Language (LEL) is a glossary that helps capture domain language, essential for understanding both the domain and requirements. This article presents an approach to extract domain language from a web application using the LEL glossary. It involves three stages: general analysis, iterative language capture, and verification. A web browser extension tool supports this approach. Preliminary results show positive applicability.*

1. Introduction

An important stage in the software lifecycle is requirements engineering, as errors made in this phase can be difficult and costly to correct in later stages (BOEHM, 1997). Thus, if the requirements are not correct, the development team may create a product that does not meet the customer's expectations (FORSBERG; MOOZ, 1991).

It is essential to understand the language of the application domain in order to gather high-quality requirements. The Language Extended Lexicon (LEL) is a glossary (MESERVY et al., 2012) that helps to understand this language without focusing on the software itself. The LEL organizes terms into four categories: subjects, objects, verbs, and states, and describes them using two attributes: notion and behavioral responses. It has been shown to be an effective method for capturing domain language, standing out for being easy to learn and use, as well as having good expressive capacity (CYSNEIROS; DO PRADO LEITE, 2001). Therefore, it is considered a useful tool for obtaining requirements in early stages (ANTONELLI et al., 2012).

Traditionally, in software engineering, requirements are gathered from people or documentation, but in some cases, the involvement of people may not be available. In such cases, with the growth of the software industry, it has become common to analyze existing applications to gather requirements through reverse engineering (FAHMI; CHOI, 2007). Applications are a valuable source for understanding the domain language as they contain encapsulated knowledge (BROOKS, 1997). Since capturing the language requires considerable effort, it is important to have tools that facilitate this task. Web browser extensions offer a convenient way to add functionalities to any web page, making it ideal to have an extension that allows the language of an application to be directly captured.

This paper proposes a reverse engineering method to extract the language of a domain from a web application using the LEL as a template. This process is supported by a browser extension tool to facilitate the language capture. Furthermore, the paper presents a preliminary validation of the method through the SUS survey (BROOKE, 1996). The structure of the document is organized as follows: Section 2 reviews the background, Section 3 describes the proposed approach, Sections 4 and 5 provide evidence of the approach's applicability, Section 6 discusses related work, and finally, Section 7 presents the conclusions and futute work.

2. Language Extended Lexicon

The Language Extended Lexicon (LEL) is a glossary used to describe the language of an application domain, without necessarily requiring the definition of a software application. Its purpose is to record the definitions of the terms specific to a domain in order to "understand the language of a problem without worrying about the problem" (LEITE; FRANCO, 1993). Understanding a domain involves learning the language used in it, which highlights the importance of building an LEL.

Within an organization, experts, end-users, and customers possess knowledge about the domain, but they view it from different complementary perspectives. Therefore, LELs must provide a unified and coherent representation of the language being used. Language is expressed through symbols, which can be terms or short expressions, and are defined by two attributes: notion and behavioral responses. Notion refers to the denotation, that is, the essential characteristics of the symbol, while behavioral responses describe its connotation, or the relationship between the described term and other terms (Table 1 (ANTONELLI et al., 2023)). Each symbol in the LEL is classified into one of four categories: subject, object, verb, or state. Table 2 (ANTONELLI et al., 2023) presents the characteristics of each category and how to describe them.

Table 1. LEL Symbol Description Template

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

Table 2. Template for Describing LEL Symbols by Category

Category	**Notion**	**Behavioral response**
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 approach presented aims to analyze a web application (the input of this approach) and extract the knowledge and requirements associated with it, which are described through its own language, thus representing the output of the approach. Specifically, this method uses an LEL glossary to describe the application's language. Therefore, it is a reverse engineering method, as it derives a specification of knowledge and requirements from an existing application, which could facilitate the development of a new one.

The 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 phase. Each stage is organized into steps, and each step involves the execution of one or more activities.

The first stage, corresponding to the general analysis of the web application, focuses on conducting a preliminary study to understand its purpose, data, and functionalities. The second stage, related to domain language capture, conducts a more detailed analysis of the web application to identify essential components and define glossary expressions based on them. The third stage, which is the verification of the generated domain language, aims to review the definitions of the expressions, both individually and for consistency between terms. It should be noted that this approach is not strictly sequential; stages two and three can be repeated iteratively. Figure 1 summarizes this approach.

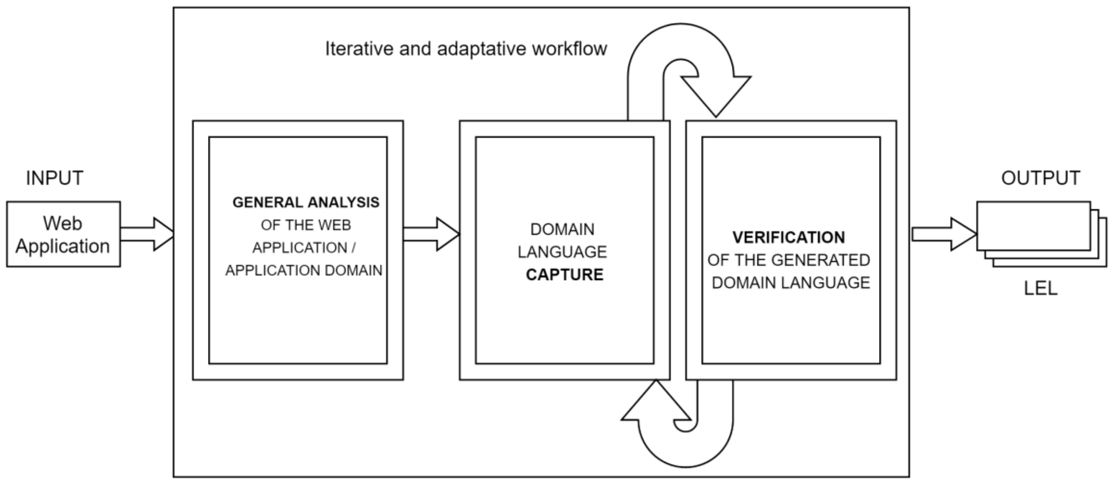

Iterative and adaptative workflow
INPUT
Web Application
GENERAL ANALYSIS OF THE WEB APPLICATION / APPLICATION DOMAIN
DOMAIN LANGUAGE CAPTURE
VERIFICATION OF THE GENERATED DOMAIN LANGUAGE
OUTPUT
LEL

Figure 1. The approach

The subsections below provide a detailed description of each stage. Each one is illustrated using a website for the sale of agricultural products. This website offers plant protection products for crop nutrition, pest control, and agricultural consultancy, both general and specific. To support these activities, it includes a categorized knowledge base on crop solutions, along with information about local suppliers. It also allows users to contact agricultural professionals for consultations. It is important to note that the website includes information on professionals in Africa.

3.1. General Analysis of the Web Application

This stage, which is the first in the approach, consists of two main steps: conducting a (i) general analysis of the web application to be studied, and performing an (ii) exploratory study of the application's domain. In other words, both the web application itself and the application's domain are analyzed.

To execute step (i) the general analysis of the application to be studied, three activities must be carried out. First, the application should be explored to understand its overall purpose, which should be summarized in a concise sentence that begins with a verb. For example, the purpose of the web application "Greenlife Crop Protection Africa" is to "provide phytosanitary products for crop protection, nutrition, pest control, and general agricultural consultancy in Africa."

Second, a more detailed exploration of the application is necessary to create its navigation map. This map should use squares to represent the pages and arrows to indicate the direction of navigation. It's important to note that the pages should be identified conceptually. For instance, in an e-commerce application, the product description page should be represented by one square, regardless of how many products are available.

For example, the Greenlife Crop Protection Africa website has a main page that provides access to three sections: crop solutions, products, and services. From the crop solutions page, users can browse different categories to find specific solutions. In the products section, various product types are displayed, and users can select a product and locate a distributor. Lastly, the services section offers three options: submitting questions, finding an agricultural professional, and locating purchase points for products. The navigation map for this website is shown in Figure 2, while Figure 3 provides snapshots of the site's different pages.

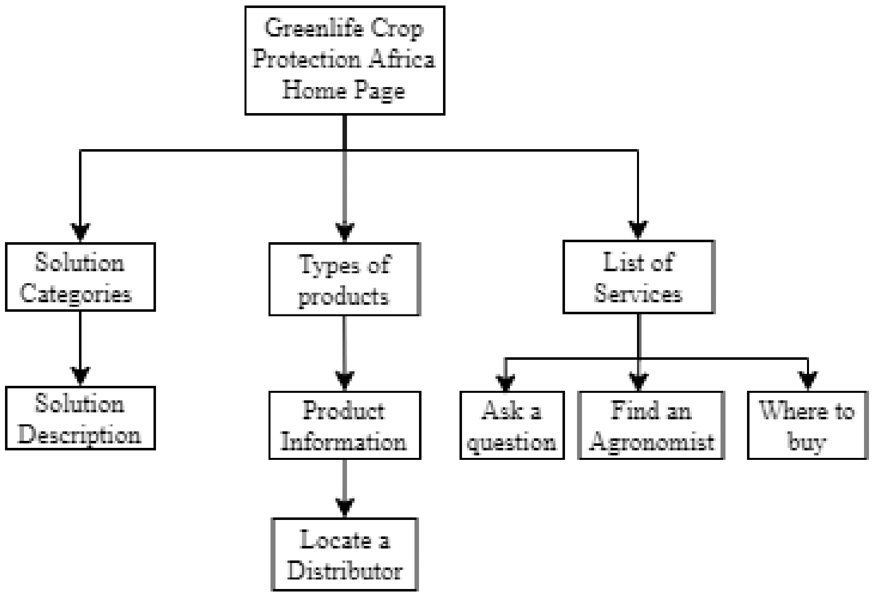


Figure 2. Navigation Map for Greenlife Crop Protection

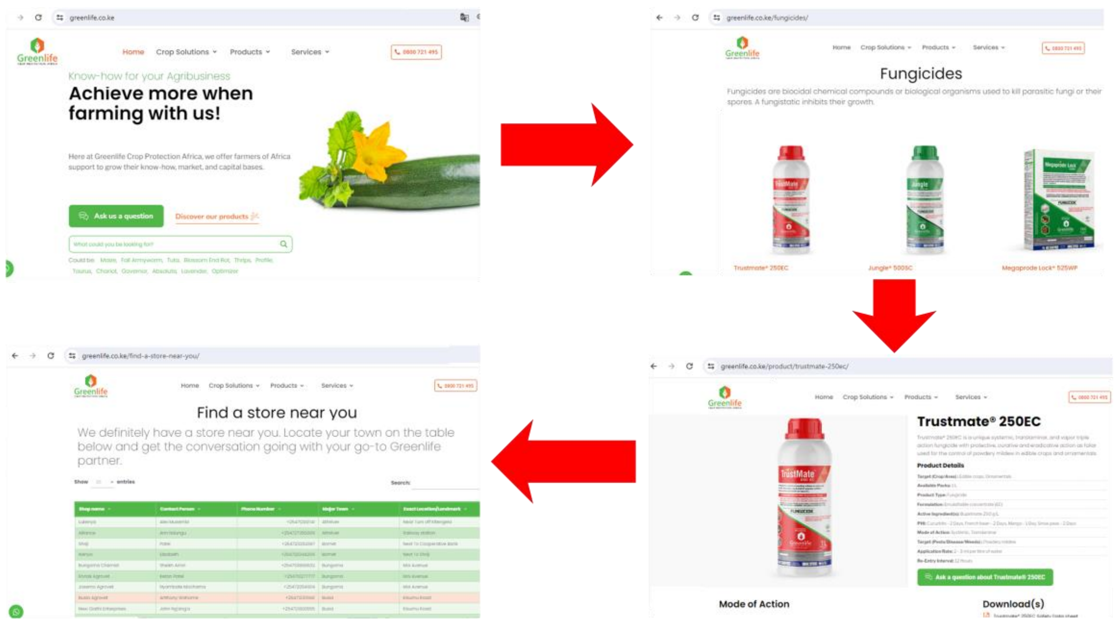


Figure 3. Greenlife Crop Protection Website example

Figure 3 illustrates the home page as the root of the navigation map. The second webpage, representing the "types of products" section, is the next node at the second level of the map. The third page, showing product information, is located at the third level. The fourth webpage, which helps users find a distributor, appears at the fourth level of the navigation map.

Third, a list of the application's general functionalities should be compiled. For instance, the Greenlife web application for Africa provides phytosanitary products for crop protection and nutrition, offers both general and specialized agricultural consulting, contains a categorized knowledge base on crop solutions, and supplies information on local vendors and agricultural professionals.

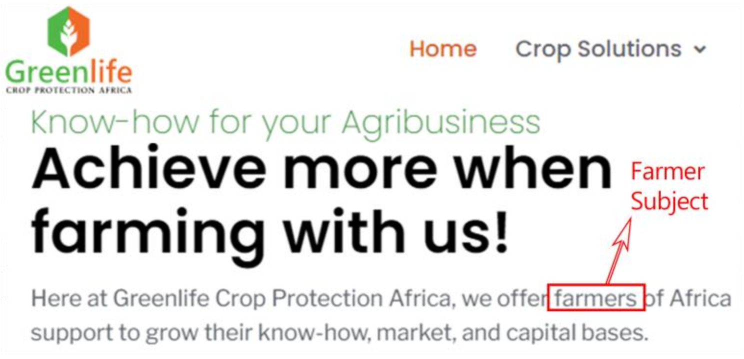
To perform step (ii) the exploratory study of the domain, additional documentation should be consulted, based on the findings from step (i).

3.2. Domain Language Capture

In this second stage of the approach, symbols are identified, categorized (as subject, object, verb, or state), and described. This process involves defining the terms (symbols) of the LEL and linking them to the web application. The notion and behavioral responses of each symbol should also be described. It's important to explore all pages of the site to comprehensively capture the domain language. This stage consists of three steps: (i) identifying symbols and their categories, (ii) describing the notion, and (iii) describing the behavioral responses.

The first step, identifying symbols and their categories, requires navigating the entire web application using the navigation map to identify and categorize symbols. As a result, a list of symbols, their categories, and their corresponding locations within the HTML will be created. Note that a symbol may be linked to multiple elements across different pages. The second step, describing the notion, defines the essence of the symbols. This step comes after symbol identification to ensure a clearer understanding of the symbols, making the descriptions more thorough. The third step, describing behavioral responses, involves writing sentences like "a subject performs an action on an object," where the subject, action, and object are symbols identified in the LEL.

To carry out the activities in the first step (identification of symbols and their categories), it is important to analyze and categorize each element of the web application as a subject, object, verb, or state. Identifying and categorizing subjects involves recognizing one of the following situations: (i) each user role or (ii) any element in the web application (text, images, or any piece of information from any medium) that represents a person or organization, associating it with a subject symbol. The corresponding title will be the text describing the user role, person, or organization, such as "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.

Figure 4. Farmer Subject Linked in the Greenlife Crop Protection Web Application – Home Page

For objects, it is essential to identify any element in the web application that represents resources, tools, or data. Its title should be a noun or a short phrase that describes the passive element, such as “phytosanitary product,” “best product,” “question,” “crop solution,” “store,” or “technical assistant.”

For verbs, it is necessary to recognize (i) each button or (ii) any element in the web application (text, images, or any piece of information from any medium) that implies an action, which should be linked to a verb symbol. The title will be the element's text, using an infinitive verb. For example, “ask a question” on the Greenlife application’s homepage.

States represent situations in which subjects, objects, or verbs can be found. To identify them, it is necessary to locate any element in the web application (text, images, or any information from any medium) that can be in a specific state. Its title describes the transition of the identified subject, object, or verb. For example, “ pending receipt of response to a question state.”

The second step (description of the notion) involves defining the notion of each symbol identified in the first step. The notion of the identified subject corresponds to its characteristics or conditions and can be expressed using terms such as “is” or “has.” Table 3 provides an example of the subject “farmer” and its notion. The notion of the identified object refers to its attributes and is described similarly, as in the case of an object “question,” which includes location, crop type, full name, email, phone number, agricultural county, and question text.

Table 3. Farmer Subject - Notion

Subject	Farmer
Notion	It is anonymous use of the web applicaton They navigate the website to obtain agricultural information

The notion of the identified verb is related to the action's objective, described with phrases that answer questions like “for what purpose?” or “why?” it is performed. For example, “Action to ask a question to an agronomist or technical assistant.” Regarding states, their notion is the represented situation; for example, the state “awaiting response” to a question posed by the farmer implies that they are waiting for an expert’s response on the website.

The third step (description of behavioral responses) focuses on defining the behavioral responses of the symbols identified. For subjects, these are the actions they perform, as exemplified in Table 4 with the subject “farmer.” A subject may lack a notion or behavioral responses, as in the case of “company.” The behavioral responses of the identified object refer to the actions taken upon it. For verbs, this involves describing the necessary steps to carry out the corresponding action, explaining how these actions are performed when interacting with the element. Finally, the behavioral responses of the identified state are the actions required to transition to another state. The next state derives from the behavioral responses of the previous state.

Table 4. Farmer Subject – Behavioral responses

Subject	Farmer
Notion	It is anonymous use of the web applicaton 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

This process ensures that the domain language is fully captured and linked to specific elements of the web application.

3.3. Verification of the Generated Domain Language

This stage involves reviewing the symbols from the domain language capture phase and making any necessary adjustments. The outcome is a set of verified symbols. The process consists of three steps: (i) verifying the description of each symbol based on the template provided in tables 1 and 2 from the background section, (ii) checking for duplicates, and (iii) identifying any new symbols.

The first step, which focuses on verifying the description of each symbol according to the proposed template (internal consistency), entails reviewing both the accuracy of the symbol's notion and its behavioral responses to ensure a clearer understanding of the symbol. Additionally, this step allows for the revision of the symbol's title to ensure it is correctly defined.

In the second step, checking for duplicates (external consistency), it is possible to find the same symbol in different parts of the website with different titles or definitions. In such cases, the repeated symbols are consolidated, creating a new symbol that includes both titles and combines the definitions. Alternatively, if the same title is used for the repeated symbol, adjustments are made accordingly.

The third step, identifying new symbols, involves reviewing the existing symbol definitions and recognizing that a new symbol should be defined. This new symbol must also appear on the analyzed website. It is added to the LEL glossary, following the process used for capturing domain language. Once added, these new symbols undergo the same verification steps as the previous ones, ensuring consistency and accuracy.

4. Tool Support

A browser extension tool for Google Chrome was created to assist with the proposed approach during the domain language capture stage. Figure 5 provides an example of the Greenlife Crop Protection Africa website that was analyzed. On the right side, a box displays the view of the developed browser extension. The architecture of the tool is structured as follows: a user interface layer through the web browser extension, a microservices layer, and a data layer. The tool was built using JavaScript, with the front end developed using the AngularJS framework (ANGULARJS, [s.d.]) and the backend utilizing the Express framework (NODEJS, [s.d.]). Additionally, MongoDB (MONGODB, [s.d.]) was used as the database. The tool functions as an extension within the browser, with its primary feature being the identification of symbols from the LEL glossary within an existing web application. This is accomplished using web augmentation techniques that visually capture each DOM element (image), its location on the website, and the DOM object's XPATH. All captured data is stored in JSON format in the database

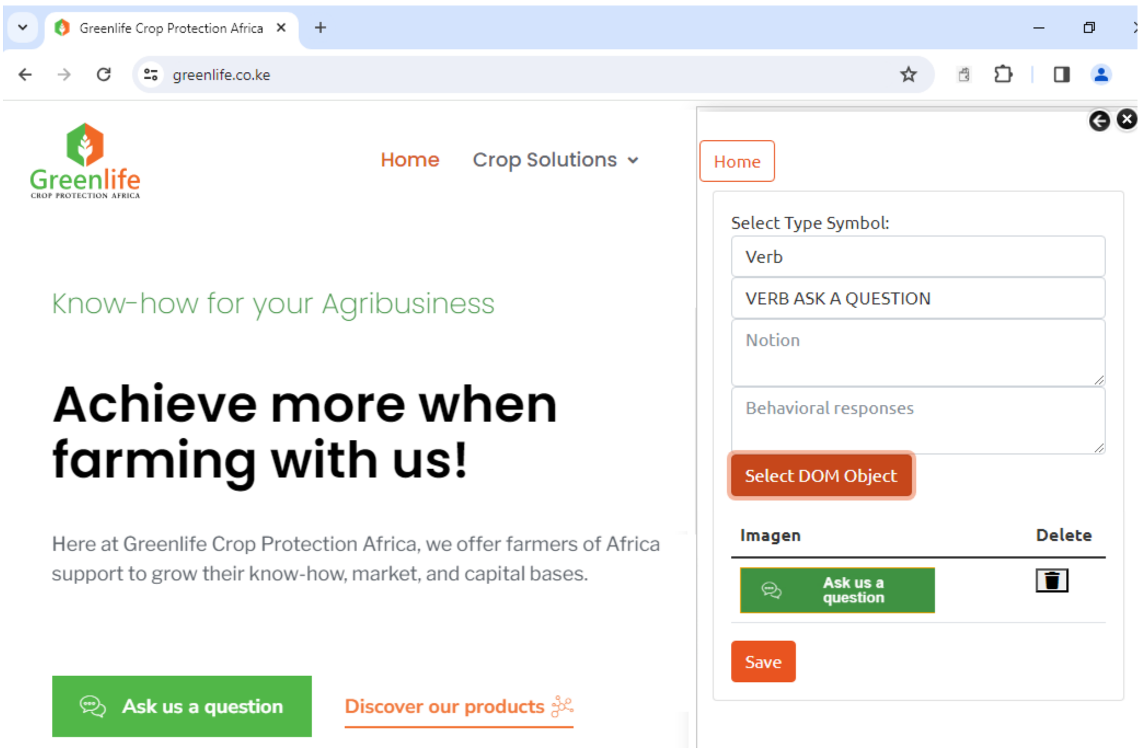


Figure 5. Web browser extension

4.1. Domain Language Capture Design

The domain language capture tool is based on the diagram shown in Figure 6, developed as a web browser extension utilizing web augmentation techniques. This tool allows for the creation of structures directly within the browser, temporarily storing them in local storage before transferring them to a non-relational database.

Figure 6 displays the package and class diagram that represents the tool's design, highlighting two main packages: (i) End-User Support and (ii) Metamodel, which are described below.

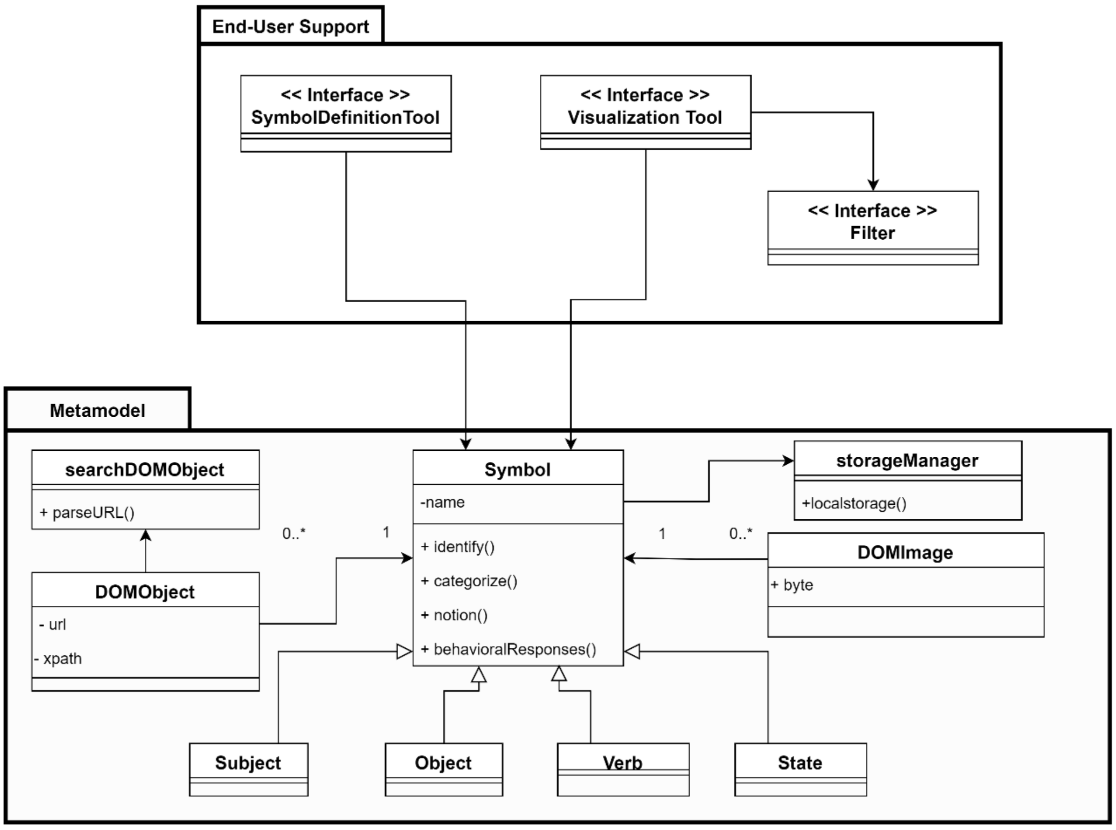


Figure 6. Package and Class Diagram of the Domain Language Capture Tool

The End-User Support package contains classes that function as data transfer objects and provide support for user functionalities. The SymbolDefinitionTool class manages the complete symbol definition and DOM object capture within the application in use, while VisualizationTool processes user queries on the defined LEL elements, which can be filtered via the Filter class.

The Metamodel package provides the main structure for storing the application's domain language. The Symbol class is responsible for storing each defined symbol, first in local storage and then in the database through the StorageManager class. Additionally, symbol classification is organized within the classes Subject, Object, Verb, and State.

The DOMObject and SearchDOMObject classes are responsible for locating the defined symbols along with the DOM objects on the website under analysis, using XPATH. Finally, the `DOMImage` class manages images captured from the DOM objects previously selected by the user within the application from which the domain language is captured.

5. Evaluation

The proposed approach was evaluated using a web browser extension tool, specifically designed to support the process. In particular, the second phase of the approach was evaluated, which focuses on capturing domain language related to the identification of symbols and categories, as well as the description of notions and behavioral responses.

The evaluation involved 12 members of a research project at the University of La Plata, Argentina. All participants had experience in software development and took on the role of requirements engineers to perform reverse engineering. Their task was to identify an object symbol anywhere on the IMDb (Internet Movie Database) website and record the corresponding notion and behavioral responses, following the proposed approach. IMDb is a well-known platform that hosts an online database for movies, television, and video games.

Participants were provided with a guide, and they carried out the activity with the support of the browser extension. It is important to note that the participants had no prior experience with the LEL glossary. They also received training on the proposed approach, particularly on the domain language capture stage, before beginning the experiment.

To assess the applicability of the approach, the System Usability Scale (SUS) (BROOKE, 1996) (BROOKE, 2013) was used. Although the SUS is primarily used to evaluate the usability of software systems, it has also proven effective for evaluating processes and products (BANGOR; KORTUM; MILLER, 2008). This scale consists of a 10-item questionnaire, with responses recorded on a five-point scale, ranging from "1" ("Strongly disagree") to "5" ("Strongly agree"). Although the questionnaire includes 10 questions, they are paired, meaning the same question is asked from a complementary perspective to ensure more reliable results.

The SUS score is calculated as follows: first, items 1, 3, 5, 7, and 9 are scored by subtracting 1 from the assigned value. Then, items 2, 4, 6, 8, and 10 are scored by subtracting the assigned value from 5. Afterward, the scores for each participant are summed and multiplied by 2.5 to obtain a final value on a scale from 0 to 100. Finally, the average score is calculated.

The approach is classified into one of the following categories: "Not acceptable" (0-64), "Acceptable" (65-84), and "Excellent" (85-100) (CYSNEIROS; DO PRADO LEITE, 2001). The obtained score was 71.04, placing the approach in the "acceptable" category.

6. Related works

Reverse engineering has been explored from various perspectives as a tool for extracting requirements from already developed systems. Hassan et al. (HASSAN et al., 2015) focus on extracting requirements directly from the source code of a legacy system. In turn, our method seeks to capture and understand domain language through a web application using the LEL. Although both approaches use reverse engineering, each is applied in different contexts and employs different methods to achieve its goals. We use reverse engineering to gather information about requirements, similarly to Aman et al. (AMAN; IBRAHIM, 2013). However, they propose an XML-based framework that uses UML to generate software requirements specifications.

On the other hand, Fahmi et al. highlight the application of reverse engineering in application renewal (FAHMI; CHOI, 2007), focusing on identifying retained functions, redundancies, and reusable elements, which aligns with our goal of gaining a deep understanding of domain language through reverse engineering. As for Tramontana (TRAMONTANA, 2005), this author suggests a reverse engineering approach specifically for web applications, differing from ours in methodological aspects, particularly in the combination of reverse engineering with UML diagram reconstruction.

Su et al. (SU; ZHOU; ZHANG, 2008) propose an aspect-oriented software reverse engineering framework to understand cross-cutting properties in legacy systems at the requirements level. In contrast, our approach emphasizes understanding domain-specific language through the LEL glossary and reverse engineering. Sabir et al. (SABIR et al., 2019) 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 apply reverse engineering. However, their approach produces UML diagrams, whereas ours generates an LEL. Bolchini et al. (BOLCHINI; PAOLINI, 2002) introduce a lightweight methodology that combines goal-oriented requirements engineering and scenario-based techniques. While our approach seeks to extract domain language from a web application, theirs focuses on conceptual tools and a lightweight methodology for requirements analysis in web applications.

Mukhtar et al. (MUKHTAR; AFZAL; MAJEED, 2012) use general dictionaries to identify compound words that include essential or atomic words. While our approach focuses on reverse engineering from web applications to obtain domain language, theirs concentrates on analyzing the specific vocabulary of a software application, which constitutes a subset of domain language.

Antonelli et al. (ANTONELLI; ROSSI; OLIVEROS, 2016) propose and validate a strategy to collaboratively capture domain language using the LEL. Our approach focuses on obtaining domain language from the web application. Garrido et al. (GARRIDO et al., 2020) propose an agile methodology for building mathematical programming models using LEL and scenarios. Both approaches use LEL to capture domain language but differ in specific application, domain of interest, and methodology employed. Antonelli et al. (ANTONELLI et al., 2021) also employ kernel sentences as input and generate use cases as output, which can be incorporated into the LEL produced by our approach.

On the other hand, Antonelli et al. (ANTONELLI; BIMONTE; RIZZI, 2022) present a method that builds a multidimensional schema from the domain language obtained through the LEL. The LEL from our web application could serve as input for this method. In a different study, Antonelli et al. (ANTONELLI et al., 2023) propose an approach to consider the application domain language, captured through its vocabulary, to refine it and obtain a language restricted to the boundaries of the software application. In our case, we obtain an LEL from a web application, which can be used as input in this approach. In another study, Antonelli et al. (ANTONELLI et al., 2022) propose a collaborative method for generating a conceptual model from natural language specifications using kernel sentences. Although this method is different from ours, it could be incorporated into the behavioral responses section of the LEL using these kernel sentences.

7. Conclusions and future work

This article proposes a reverse engineering approach to extract the language of a specific domain from a web application, using the LEL glossary. This method is structured into three main stages: a general analysis of the web application, the capture of domain language, and the verification of the generated language. A preliminary evaluation was conducted to validate the method's applicability, and a browser extension is presented to facilitate the process.

Domain language is essential for understanding both the domain context and its requirements; if these contain errors, correcting them in later stages of software development requires considerable effort. It is also common practice to analyze existing applications when developing new systems. The LEL serves as a structured glossary for capturing domain language, and the main contribution of this article is the generation of an LEL from a web application to represent this language.

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

References

AMAN, H.; IBRAHIM, R. Reverse Engineering: From Xml to Uml for generation of software requirement specification. 2013 8th International Conference on Information Technology in Asia - Smart Devices Trend: Technologising Future Lifestyle, Proceedings of CITA 2013, p. 1–6, 2013.

ANGULARJS. No Title. , [s.d.]. Disponível em: <https://angular.io/>. Acesso em: 18 mar. 2024

ANTONELLI, L. et al. Deriving requirements specifications from the application domain language captured by Language Extended Lexicon. Anais do {WER12} - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, April 24-27, 2012. Anais...2012.

ANTONELLI, L. et al. Specification Cases: a Lightweight Approach based on Natural Language. Workshop em Engenharia de Requisitos. Anais...2021. Disponível em: <https://doi.org/10.29327/1298728.24-5.>

ANTONELLI, L. et al. An approach to extract a conceptual model from natural language specifications. (L. Antonelli, M. Lucena, R. L. Q. Portugal, Eds.)Anais do {WER23} - Workshop em Engenharia de Requisitos, Porto Alegre, RS, Brasil, Agosto 15-17, 2022. Anais...{LFS} (UFRN, Brasil), 2022. Disponível em: <https://doi.org/10.29327/1298356.26-12>

ANTONELLI, L. et al. Defining the language of the software application using the vocabulary of the domain. Electronic Journal of SADIO, v. 22, n. 3, p. 2–14, 2023.

ANTONELLI, L.; BIMONTE, S.; RIZZI, S. Multidimensional modeling driven from a domain language. Automated Software Engineering, v. 30, n. 1, p. 6, 2022.

ANTONELLI, L.; ROSSI, G.; OLIVEROS, A. A Collaborative Approach to Describe the Domain Language through the Language Extended Lexicon. Journal of Object Technology, v. 16, n. 3, p. 1–27, 2016.

BANGOR, A.; KORTUM, P. T.; MILLER, J. T. An Empirical Evaluation of the System Usability Scale. Intl. Journal of Human-Computer Interaction, v. 24, n. 6, p. 1–44, 2008.

BOEHM, B. W. Software Engineering. [s.l.] Computer society Press, IEEE, 1997.

BOLCHINI, D.; PAOLINI, P. Capturing Web Application Requirements through Goal-Oriented Analysis. WER. Anais...2002.

BROOKE, J. “SUS-A quick and dirty usability scale.” Usability evaluation in industry. [s.l.] CRC Press, 1996.

BROOKE, J. SUS: a retrospective. Journal of usability studies, v. 8, n. 2, p. 29–40, 2013.

BROOKS, F. P. The Mythical Man-Month. 2. ed. [s.l.] Addison-Wesley Professional, 1997.

CYSNEIROS, L. M.; DO PRADO LEITE, J. C. S. Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation. Proceedings of the Workshop em Engenharia de Requisitos. Anais...Buenos Aires, Argentina: 2001.

FAHMI, S. A.; CHOI, H.-J. Software Reverse Engineering to Requirements. 2007 International Conference on Convergence Information Technology (ICCIT 2007). Anais...IEEE, 2007. Disponível em: <https://ieeexplore.ieee.org/document/4420580/>. Acesso em: 5 mar. 2022

FORSBERG, K.; MOOZ, H. The Relationship of System Engineering to the Project Cycle. Proceedings of the First Annual Symposium of National Council on System Engineering. Anais...1991.

GARRIDO, A. et al. Using LEL and scenarios to derive mathematical programming models. Application in a fresh tomato packing problem. Computers and Electronics in Agriculture, v. 170, p. 105242, 2020.

HASSAN, S. et al. Software Reverse Engineering to Requirement Engineering for Evolution of Legacy System. 2015 5th International Conference on IT Convergence and Security (ICITCS). Anais...IEEE, 5 ago. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7293021/>. Acesso em: 29 out. 2022

LEITE, J. C. S. DO. P.; FRANCO, A. P. M. A strategy for conceptual model acquisition. Proceedings of the {IEEE} International Symposium on Requirements Engineering. Anais...1993.

MESERVY, T. O. et al. The Business Rules Approach and Its Effect on Software Testing. IEEE Software, v. 29, n. 4, p. 60–66, 2012.

MONGODB. No Title. , [s.d.]. Disponível em: <https://www.mongodb.com/es>

MUKHTAR, T.; AFZAL, H.; MAJEED, A. Vocabulary of Quranic Concepts: A semi-automatically created terminology of Holy Quran. 2012 15th International Multitopic Conference (INMIC). Anais...2012. Disponível em: <https://doi.org/10.1109/INMIC.2012.6511467>

NODEJS. No Title. , [s.d.]. Disponível em: <https://nodejs.org/en>

SABIR, U. et al. A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code. IEEE Access, v. 7, p. 158931–158950, 2019.

SU, Y.; ZHOU, X.-W.; ZHANG, M.-Q. Approach on Aspect-Oriented Software Reverse Engineering at Requirements Level. 2008 International Conference on Computer Science and Software Engineering. Anais...IEEE, 2008. Disponível em: <https://ieeexplore.ieee.org/document/4722062/>. Acesso em: 6 mar. 2022

TRAMONTANA, P. Reverse engineering Web applications. 21st IEEE International Conference on Software Maintenance (ICSM’05). Anais...2005. Disponível em: <https://doi.org/10.1109/ICSM.2005.77>