UCD-Generator – A LESSA Application for Use Case Design

Imran Sarwar Bajwa¹, Irfan Hyder ²

¹ CISUC – Departament of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal
² PAF-KIET, Karachi, Pakistan.
imran@dei.uc.pt, hyder@pafkiet.edu.pk

Abstract

In object-oriented design, use cases are one of the important but complex components to design. UCD-Generator is a LESSA based application that makes this process more simple and easy. UCD-Generator is an automated system that has a vigorous ability to directly interact with the user. This research paper presents a natural language processing based approach LESSA that is used for automatically understanding the natural language text and extract required information. This information is used to draw the Use Case diagrams. User writes his interface based preferences in simple English in a few paragraphs and the designed system has conspicuous ability to analyze the given script. After compound analysis and extraction of associated information, the designed system actually draws the Use Case class diagrams. Other conventional CASE tools require a complete understanding of the intended business scenario and lot of extra time and efforts from the system analyst during the process of creating, arranging, labeling and finishing the USE Case diagrams. The designed system provides a quick and reliable way to generate Use Case diagrams to save the time and budget of both the user and system analyst.

Keywords: Information Extraction, Use Case Design, Natural Language Processing, Automatic Diagrams Generation, Text Understanding, UML Diagrams.

1. Introduction

UML (Unified Modeling Language) provide several types of diagrams that are used to increase the simplicity and conception of an application under development. Use case diagrams are one of the UML diagrams. A use case diagram is used to exemplify the supported functionalities the system. The software development teams visualize the functional requirements of a system by use case diagrams. A use case diagram describes the relationship of the actors to main processes in the system and also the relationships among different use cases. An actor is a human being that interacts with the designed system [1]. Each use case provides one or more scenarios that convey how the system should interact with the end user or another system to achieve a specific business goal. Use cases are a tool for capturing functional requirements related to a designed system applications. The functional requirements principally represent the intended behavior of a system. This complementary behavior may be expressed as services, tasks or functions provided by the specific system [2]. The patterns of modeling and mapping of business logic in software engineering have been completely changed in the recent times. These days all major phases of software engineering follow the rules defined by the Object Oriented design paradigm. All phases of software engineering are deviating from the conventions and new paradigms are more popular these days [3]. Same the case is with Software analysis process which uses Unified Modeling Language to map and model the user requirements. Analysis is the key process of building modern information system applications and base for the robust and vigorous software application’s design and development.

Use case design is the major part of UML documentation and is also a difficult phase of object oriented design of software applications as use cases are typically generated according to the requirements and preferences of a particular user. A system analyst has to spend a lot of time in correspondence with the user and understand his specific preferences. The other problem specifically addressed in this research is software analysis and design using available CASE tools which are just paint like tools as Visual UML, GD Pro, Smart Draw, Rational Rose etc [5]. To use their extensively overloaded interface of these CASE tools is a vexing problem. The process of generating the UML diagrams through these software engineering tools is very difficult, time consuming and lengthy process to perform. Consequently, an automated software development was required that may acquire the preference directly from user and generate corresponding USE Cases with improved output in minimum time consumed.

In this paper, a natural language processing approach is presented for semantic analysis of the text and its conversion of text into the use case diagrams. The module that implements UCD-Generator is called
LESSA (Language Engineering System for Semantic Analysis). LESSA is an automated rule-based Approach that reads the natural language text, understands its meanings and extracts the required information. LESSA performs lexical, syntax and semantic analysis of the natural language text.

In the next section the, the basics of LESSA have been briefly described. In the next section, the processing steps of the UCD-Generator system have been illustrated. In the next section, some experiments have been illustrated and in the last section the used approach is compared with other used approaches for text understanding and furthermore the conclusion of the research has been illustrated.

2. Proposed Approach

In order to address the intended problem of this research a useful and supportive system was required, which has sound ability to facilitate and assist both the users and software engineers. The functionality of the conducted research was domain specific but it can be enhanced easily in the future according to the requirements. Current designed system incorporate the capability of understanding the mapping user preferences after reading the given business scenario in plain text and then ultimately drawing the Use Case diagrams. An Integrated Development Environment has also been provided for User Interaction and efficient Input and output.

2.1. Automated Object-Oriented Design

From various projected phases of software application engineering, analysis and design of an information system actually relates to understand the system and its primary components [2]. Typically, design relates to manage and control the complexity parameter in a domain. In software engineering, design methods provide various notation usually graphical ones. These notations allow to store and communicate the perpetual design decisions. Object-oriented design has overruled the typical analysis and design techniques as structured design and data-driven design [5]. As compared to old style design paradigms, object-oriented design models the every active entity of the problem domain using concept of objects. Objects have:

- Object State (shape and condition)
- Object Behaviour (What they perform)

The major emphasis of the conducted research was automatic identification of objects from a problem domain. User provides the input text in English language related to the business domain. After the lexical analysis of the text, syntax analysis is performed on word level to recognize the word category [6]. First of all the available lexicons are categorized into nouns, pronouns, prepositions, adverbs, articles, conjunctions, etc. The syntactic analysis of the programs would have to be in a position to isolate subject, verbs, objects, adverbs, adjectives and various other complements. It is little complex and multipart procedure.

A procedure is defined that understands these sentences and discover the actor, finds interaction, goals and intended objects as following:

**Actor:** The actor causes the action to occur as in "user fills the form" user is actor who performs the task of form filling.

**Action:** This component defines the ultimate goal of a certain use case. For example, user fills form to get registered. Here registered is goal of the intended Use Case.

**Object:** The thematic object is the object the sentence is really all about—typically the object, undergoing a change. Often the thematic object is the same as the syntactic direct object, as "User presses the key." Here the key is thematic object.

2.2. LESSA

LESSA (Language Engineering Systems for Semantic Analysis) is a natural language processing approach. That is primarily used to understand and analyze the natural language text. LESSA is based on a rule-based algorithm. This rule-based algorithm has a number of rules that are defined according to the English grammar. LESSA bases on a rule based algorithm that is used for understanding and analyzing the natural languages text. LESSA has following major modules:

2.2.1. Tokenization

This module reads the natural language text and converts the text into lexical tokens. These lexical tokens are processing elements of LESSA. These lexical tokens are further passed to the next phase for further processing.

2.2.2. POS Tagging

This module receives the lexical tokens from the previous module and applies a rule-based algorithm on the text. The major function of this module is to identify and extract the various parts of speech in the given text. This phase is also called parts of speech tagging [13].

2.2.3. Meaning Understanding

This last module understands the semantics of the given text and on the basis of extracted semantics, the object, subject and adverb parts of the sentences are identified. Following is an example to describe the working of LESSA module.

"User fills the form."

Typically, LESSA reads this text and performs the lexical analysis to find the parts of speech and then further analyze to find the actor, action and the object in the given course of text. For this example, following is the output.
Lexicons | Phase-I | Phase –II
--- | --- | ---
User | Noun | Actor
fills | Verb | Action
the | Article | ------
form | Noun | Object

This is the final output of syntax assessment phase and all subject nouns are marked as objects and verbs are marked as actions and all object nouns s are marked as the objects in the scenario. In the above example, there is one actor ‘User’ in the example. Besides the actor, there is also an action ‘fills’ and finally there is an object “form”, on which certain action has been performed.

3. UCD-Generator

The designed system UCD-Generator (Use case Diagram Generator) uses natural language processing techniques to generate the Use Case diagrams. The process of generation of the use case diagram is a multi-phase process. There are two main phases as following

a. Information Extraction
b. Diagram Generation

3.1. Information Extraction

In the information extraction phase the text input given by the user is analyzed to extract the parts of speech. This phase is also called parts of speech tagging in which pronouns, verbs, adjectives, articles, etc are identified in a sentence. LESSA performs this action that typically uses a rule-based algorithm that has been used to perform syntax analysis. After Syntax analysis, the sentence level categorization of a sentence (string) as actor, object is performed. In the last the categorized string of a sentence is represented in to the respective Use Case diagrams

3.2. Diagram Generation

In this phase, first the extracted information from the previous phase is analyzed and then according to this information, the use case diagrams are generated. In this phase, small diagrams are combined to generate the complete use case diagram. The designed system follows following four steps to generate a use case diagram.

Step 1 – In step I, the actors are identified. Actors are those people who use the various functions of the intended system e.g. “User hits the key”. Here the actor is represented by the user

Step 2 – One Actor is picked at a time. Actor is named: actor is named “user” in this example.

Step 3 – In this step, particular actions of an actor with the system are defined. These are the actions that are required by an actor and these actions in a system are categorized as Use Cases.

Step 4 - For each of these Use Cases decide on the most usual course when that Actor is using the system. This is the basic course or description of the Use Case.

3.3. UCD-Generator Architecture

The complete architecture of UCD-Generator has been divided into three distinct parts. First part The natural language processing engine is module which performs lexical and syntax analysis and extracts information like who is actor, what is an object and what is the related description of the particular use case. The details are manifested below:

The designed system has an insightful ability to understand the natural language text and draw the intended use case diagrams. Current designed system
incorporate the capability of understanding the mapping user preferences after reading the given business scenario in plain text and then ultimately drawing the Use Case diagrams. An Integrated Development Environment has also been provided for User Interaction and efficient Input and output.

4. Experiments and Analysis

An application UCD-Generator was designed to test the results of the proposed approach. This application has two parts. First part is the input acquisition part in which an input window is available. Input text can be provided here for further processing as shown in the figure 4.1.

![Figure 4.1 – Use Case Input Dialog Box](image)

The input dialog box of UCD-Generator has two tabs. First tab is ‘UseCase Input’ and second tab is ‘Samples’. In first tab common tips to generate the system use cases are provided. For better accuracy, user is intended to write only one or two use cases at a time. User is also expected to write grammatically correct English as this will finally enhance the system accuracy. The complete information about any use case should be provided: actor, action and action description. In ‘UseCase Input’ window ‘Scenario Help’ is also provided for assisting the user in providing precise input. ‘Sample’ tab has some sample examples of input and their generated output by the UCD-Generator.

In this example the input text was “First, the user fills the form. Then he submits the form for the further processing.” In this input there are two use cases of single functionality. When the user clicks the ‘Draw Diagram’ button, the text is handed over to LESSA for semantic analysis. LESSA module first reads the input text and converts the text into lexical tokens. These tokens are further processed to find the parts of speech. After parts-of-Speech tagging, the text is further analyzed to extract its meanings. The extracted information is shown in the figure 4.2.

![Figure 4.2 – Use Case Information Dialog Box](image)

The use case information is displayed in the ‘Use Case View’ window. This window has two abs. The ‘Use Cases’ tab shows the generated use cases. Other tab ‘Use Case Info’ shows the extracted information from the input text. The information available is the use case actor, use case action and use case object. After extracting this information, a use case diagram can be generated successfully.

Figure 4.3 shows the final output window which displays the generated use case in ‘Use Case’ window. The generated use case is based on the information displayed in the ‘Use Case Info.’ window. An additional facility is provided for the user of the system, that he can modify the extracted information for accurate use case diagram generation. User can select the accurate information from ‘UseCase Info’ tab to improve the accuracy.

![Figure 4.3 – Use Case final Output Dialog Box](image)

User can improve the diagrams accuracy by repeating this process with technically and grammatically improved input text. About 50 different real-time scenarios were provided to the system to check its accuracy. 85% of the scenarios were accurately translated into use case diagrams. Accuracy if further improves if the extracted information is filtered before generating the diagrams.
5. CONCLUSION

This conducted research is all about the automatic generation of the Use Case diagrams after reading and analyzing the given scenario in English language provided by the user. The designed system can find out the classes and objects and their attributes and operations using an artificial intelligence technique such as natural language processing. Then the Use Case diagrams are drawn. The accuracy of the software is up to about 85% with the involvement of the software engineer provided that input is properly formatted. The given scenario should be complete and written in simple and correct English. Under the scope of our project, software will perform a complete analysis of the scenario to find the classes, their attributes and operations. A well-designed graphical user interface has also been provided to the user for entering the Input scenario in a proper way and generating Use Case diagrams to ultimately make them the part of the object oriented documentation.

REFERENCES


