Automatic Use Case & Class Diagram Generation

Sai Sruthi Mallineni[1] Sai Keerthana Goli[2]
[1,2]Department of Computer Science
[1,2]CBIT, Hyderabad, India

Abstract— Unified Modeling Language (UML) is the most popular modeling language for analysis, design and development of the software system. There has been a lot of research interest in generating these UML models, especially Use Case and Class diagrams, automatically from Natural Language requirements. System specifications and facilitating systems development is best described by use case modelling which is referred as an important requirements engineering technique. Generating Use Case models and class diagrams from linguistic representations of system requirements as a source of information is a challenging task and is also considered as a new field. We try to solve the problem of extracting the required elements that are needed to automatically generate Use Case diagrams and class diagrams from specification documents which are written in common natural language. Therefore, we are developing an automated system which employs the Natural Language Processing (NLP) techniques to parse specifications syntactically based on a predefined set of heuristic rules. Furthermore, our system incorporates the capability of analyzing and understanding the English text as a semantic unit to infer some important linguistic features such as reference, comparing and additive cohesive devices. The extracted information is then mapped into actors and use cases, which are the basic elements of Use Case diagrams. The interest in class diagrams can be attributed to the fact that classes represent the abstractions present in the system to be developed. However, automated generation of UML class diagrams is a challenging task as it involves lot of pre-processing or manual intervention at times. The knowledge extracted for the Use Case diagrams is used to derive class diagrams. Our approach has generated similar class diagrams as reported in earlier works based on linguistic analysis with either annotation or manual intervention.

Key words: UML Diagrams, Heuristic Rules, NLP Techniques

I. INTRODUCTION

The main objective of this project is to implement automation of Use Case and Class diagrams with a user friendly interface. Stemming and lemmatization algorithms are used for proper use cases, actors, classes, functions and attributes.

In this project, we introduce a new automated structured approach to acquiring, analyse and then transform natural language descriptions into UML diagrams. The user writes a few paragraphs in simple English language to describe the system requirements. Our approach uses NLP techniques to extract Use Case diagram elements via translating the textual specifications to words, these words are categorized into several Parts Of Speech (POS), and then we apply stemming algorithm and a set of syntactic heuristic rules to identify the actors and use cases of the target software system. After the compound analysis and extraction of associated information, the designed system draws the desired Use Case diagram. From the information obtained from the Use Case diagrams, Class diagram can be generated taking into consideration the classes, functions and attributes.

The automation of structured information extraction (IE) from natural language text using NLP is a relatively new field, which needs a large amount of domain knowledge. Use Case and Class diagrams play a broader role in describing systems specifications, and facilitating systems analysis, design, and implementation. However, building such diagrams is very important and time-consuming task which requires a complete understanding of the system requirements.

II. EXISTING SYSTEM

The existing system do not have the capability of analysing the natural language specifications. It generates the diagram by taking the input from the user in terms of a dictionary key and values.

III. PROPOSED SYSTEM

The proposed approach incorporates the capability of automatically generating accurate and complete Use Case diagrams and Class diagrams. An integrated development environment is also provided for user interaction and efficient input of system requirements and output of the diagrams. The proposed automated system for diagrams generation has the ability to draw Use Case and Class diagrams after analysing the text scenario provided by the user.

IV. LITERATURE SURVEY

A. Use Case Diagrams

There are various products developed for visualizing Use Case models such as Visual UML, GD Pro, Smart draw, Rational Rose, Microsoft Visio etc. 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. 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 complete. 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 [2] an approach has been proposed to extract the basic elements for generating the class diagram such as classes, data members and member functions from user requirements. Here, a rule-based system is proposed for NL-based OO Software Modelling. Proposed system automates

All rights reserved by www.ijsrd.com

the building of UML diagrams from free text requirement document. The approach of natural language processing (NLP) is used. NLP approach consists of understanding the written requirements and then use the domain knowledge to improve the identification of classes, actors & messages between the actors. There are three basic modules of the system;

1) Generation of Class Diagram
2) Generation of Sequence Diagram
3) JAVA code generation.

The methodology works by analysing the requirement specification. It is then parsed by using NLP parsing techniques to get the nouns & verbs which helps in finding out the candidate elements. With the help of domain expert knowledge results are then improved. This project makes use of Word net as ontology to get the semantic relation between the classes and attributes. POS Tagger is used to do the Part Of Speech analysis. Stemming algorithm is used to get the stem (root) of the word. This analysis helps in finding and identifying classes, attributes and methods. This approach was implemented as a software tool to generate the class diagrams. Approach proposed in [3] and [4] that take unstructured requirements in form of plain text to derive activity diagrams. The proposed approach has following steps:

1) Building of semantic network from the given scenario

This phase takes the input scenario in a textual format and does semantic analysis and builds a semantic network between the users and the objects including the actions

2) Translation from Semantic Network into models

Use Case Path model – generation rules: The Use Case Path (UCP) presents, consecutively, the route of one action through the different actors responsible for its implementation.

– Merge

UCPs with similar parts are candidates for this operation. The objective is to make one UCP out of a number of them that have repeating parts. From all these identical parts, a single UCP is formed.

– Split

This operation is the opposite of merge. In a complex UCP scheme, we can follow all possible paths keeping the logic of the connections, whether parallel or not.

– Add

A new action or a sequence of actions can be added to any knot of the existing action sequence.

– Cut

This operation is the opposite of add. Incorrect or unnecessary parts can be cut if cutting does not affect the logic of the connections.

DMG [5] is the only existing work that proposes a large number of syntactic and semantic heuristics to be used in natural language into ER models. It provides a basis for the development of new heuristics applied in ER converter which extracts knowledge from requirements specifications. An Entity Relationship (ER) data model is a high level conceptual model that describes information as entities, attributes, and relationships. Entity relationship modelling designed to facilitate database design. The approach provides the opportunity of using natural language documents as a source of knowledge for generating ER data model. The structural approach is used to parse specification syntactically based a predefined set of on heuristics rules. Extracted words with its Part Of Speech (POS) mapped into entities, attributes and relationships, which are the basic elements of ER diagrams. Nouns in system's requirements can be identified as entities. A relationship is an association among two or more entities. Relationships can be derived from verbs.

This process includes

1) Text Acquisition

The business scenario is taken as an input from the client.

2) Text Segmentation

The given paragraph is divided into sentences.

3) Tokenization

The sentence is split into tokens.

4) Parts Of Speech tagging

The tokens are associated with their corresponding Parts of Speech.

5) Chunking

Chunking is the process of taking individual units of information (chunks) and grouping them into larger units. Tokens of a sentence are group together into larger chunks, each chunk corresponding to a syntactic unit such as a noun phrase (NP) or a verb phrase (VP). To perform the chunking, a POS tagged set of tokens is required with tokens itself. Part of speech tagging tells whether words are nouns, verbs, adjectives, etc.., but it doesn't give any indication about the structure of the sentence or phrases in the sentence.

6) Parsing

Sequences of words are transformed into structures that indicate how the sentence's units relate to each other. This step helps us in identifying the main parts in a given sentence such as object, subject...etc. Parsing analysis will be able to extract nouns that are playing the role of entities or attributes, and extract verbs that act as a relationship between entities. Also, cardinalities and multiplicities information may be extracted from determiners, adjectives, modal verbs and quantifies Memory-Based Shallow Parser (MBSP) as parser method. The following are the rule based approach that is used

– Identify Entities
– Identify Attributes
– Identify Relationships
– Identity Primary Key

7) Generate ER

Once all words have been assigned to its ER element type, relevant information consisting of which words are entities, relationships, cardinalities and attributes are stored in text files. These text files are then used to generate ER diagram.

The approach proposed in [6] formalizes use cases using Event table. Through this study, the event table will be used to derive the Use Case and class diagrams.

– Event Table

Event table is a list of actions that lists events in rows and the information about each event in columns. Analyst can use event table to define Use Case model and domain class model.

However, analyst has to make some decisions when building a Use Case model. The first one is to combined business events into one Use Case. Also, analyst makes

All rights reserved by *www.ijsrd.com*

decisions to split one business event into multiple use cases. Therefore this approach completely depends on the availability of a comprehensive event table which is built from system requirements.

An approach [7] is proposed to generate Use Case diagrams from software requirements. This approach does not deal with textual requirements directly. It depends on other models to obtain the Use Case features using combination of two technologies: Recursive Object Model (ROM) and Expert Comparable Contextual (ECC) Models.

An approach [8] called computer automated Use Case diagram generator (CAUse) which can generate Use Case diagrams from text written using special language called ADD.

A semi-automated approach [9] that can generate Use Case diagram from textual user requirements written in Arabic language. This approach relies on the use of Arabic NLP tool called MADA + TOKAN to parse the Arabic statements of the textual user requirements to obtain necessary information needed to determine the potential actors and use cases. In this approach, An Arabic natural language processing tool MADA+TOKAN is used in this research to help in splitting and tokenizing the Arabic user requirements text. MADA+TOKAN is used to parse different statements of the user requirements written in Arabic to obtain different components of a sentence like lists of nouns, noun phrases, verbs, verb phrases, etc. that aid in finding potential actors and use cases. Once this is performed, a set of heuristics are used to construct the use case model.

1) MADA + TOKAN is a versatile, highly customizable and freely available toolkit for Arabic NLP applications. It consists of two components. MADA is a utility that, given raw Arabic text, adds as much lexical and morphological information as possible by disambiguating in one operation part-of-speech tags, lexemes, diacritizations and full morphological analyses. TOKAN is a utility that, given the information MADA produces, can generate a tokenization (sometimes also called "segmentation") formatted exactly to user specifications. This tokenization also identifies the stem of the word. All user requirements are processed using the MADA+TOKAN.
2) To identify the actors from the user requirements written in Arabic, a set of heuristics are presented. These heuristics are used to extract the actors from the tagging of the user requirements generated from the MADA+TOKAN. These heuristics are presented as follows:

– If the statement is simple (i.e. it contains only a verb, a subject and an object) then the actor is the main subject in the statement.
– When there are two statements combined with a connection then, there are three cases:

a) The subject is the actor.
b) If the subject is redundant in the second statement then the actor doesn't change.
c) If the subject changes in the second statement then this is another actor.

3) To identify the use cases from the user requirements, more heuristics that can be used to extract the use cases from the user requirements are presented.

– If the statement is simple (i.e. it contains only a verb, a subject and an object) then the use case is the main object in the statement.
– If the statement contains the connector (و) without any verb or actor in the second statement then the second statement is the use case.
– If the statements that contain the connector (أو) without any verb or actor in the statement then the first verb in the statement with the first noun after each connector is a use case.

4) The Use Case model is obtained by taking an actor with all its associated use cases to generate a Use Case diagram. The set of all Use Case diagrams represent the Use Case model.

B. Class Diagram

Class diagrams are meant to describe the structure of the system through classes, their attributes, methods and relationships among those classes. Classes need not necessarily match a semantic entity of the system. These can be abstraction of any concept in the system.

These can be abstraction of any concept in the system. For example: Let us consider the following requirements statement: RS1: The system initiates the allocation of courses to students based on their preferences. In RS1, system, courses, students are the semantic entities whereas; allocation is an abstract concept relating the entities – course and student. Therefore, allocation should also be modeled as a class along with other classes – system, students and courses. Software requirements are most commonly expressed in NL as reported in various survey reports too like [11], [12].

In [11] it is mentioned that Natural language is widely used in industry to state requirements for all types of systems, because it is flexible and universal. Even if a modelling technique such as the UML is used, the requirements are usually stated in natural language first. However, natural language has one major drawback, which is its inherent ambiguity. This report surveys the state of the practice and the state of the art in techniques that aim at making natural language more precise.

There have been several studies exploring the automated or semi-automated generation of class diagrams from NL requirements. Mich and Garigliano have proposed NL-based tool, LOLITA, to support object-oriented analysis [13]. Their tool is limited to the generation of object models only. CM- builder tool proposed by Harmain and Gaizauskas.

[14] aims at generating class model of the given scenario by performing lexical and syntactic analysis of the scenario. The tool makes use of Discourse Model after lexical and syntactic analysis to add information from predefined "world" model.

Linguistic Assistant for Domain Analysis, LIDA tool [15] helps analysts identify type elements in the object-oriented model like class, attribute, role etc. LIDA also supports hypertext descriptions of model to help validate a model. However, LIDA requires user-interaction to mark a

All rights reserved by *www.ijsrd.com*

word or phrase as candidate model element. Similar approaches of natural language processing to identify concepts in the NL requirements and generate class diagrams using identified concepts have been adopted in the works of other authors like [16], [17], [18] etc. However, it is difficult to assess the effectiveness of their approaches for they do not provide sufficient supporting examples.

Song etal. [18] Propose a taxonomic class modeling, TCM, approach for objectoriented analysis. The approach improves earlier proposed approaches by identifying the verb concepts in addition to noun concepts as candidate classes followed by dropping vague and irrelevant classes. But identification of vague and irrelevant classes in their approach is subjective.

Herchi and Abdessalem [19] have suggested rules for identifying concepts and then generating class diagrams from NL requirements. Some of these rules are very specific in Nature with limited application.

Static UML Model Generator from Analysis of Requirements (SUGAR) follows object-oriented analysis for object elicitation from NL requirements to generate static UML class model and Use Case models [20]. The authors suggest syntactic reconstruction rules for requirements statements as processing complex NL statements is challenging. The re-constructed requirements statements are then used to identify actors as noun phrases and Use Case as event flows in the system. Though their system identifies duplicate classes but their approach towards identifying and dropping irrelevant classes is not applicable in general.

Recently, Landhauber, Korner and Tichy [21] have proposed RECCA process to analyze requirements statements and derived class diagrams, statecharts and activity diagrams. The RECCA process utilizes Auto-annotator tool to encode the semantics of the requirements statement from nearly 70 thematic roles. The annotated statements are represented in an intermediate graphical representation from which class models are derived.

V. METHODOLOGY

The proposed system extracts the necessary information and draws the diagrams using the following modules.

A. Text Acquisition

This module provides the capability of acquiring the business scenario in the form of paragraphs of English Language text. This module allows the user to write the business scenario in the designated area.

Fig. 1:

1) Text Segmentation

This module analyzes the boundaries of each sentence and splits the text into sentences. Usually each sentence must be terminated with a period. Example: "User can send invitation to connect and admin can grant invitation. Also, they can send multimedia and receive multimedia." The above sentence can be segmented as

– User can send invitation to connect and admin can grant invitation.
– Also they can send multimedia and receive multimedia.

B. Text Tokenization

This module tokenizes the given text and splits them into lexicons or tokens.

Example: "I like computer information systems department" can be tokenized as follows:

< I >< like >< computer >< information><systems><department>.

This module handles the complexity which may result from the use of compound words that contain comma or periods such as “Dr. Malek” will recognize it as one token rather than splitting it into Dr and Malek based on the period.

C. Part of Speech Tagging

This module categorizes the tokens into various classes based on its definition and context. The token scan is classified as verbs, helping nouns, pronouns, proper noun, noun phrase, modal verbs, verb phrase, adverbs, adjectives, prepositions, prepositional phrase and conjunctions, Article, etc based on predefined rules for categorization which was adopted by Word Net.

D. Stemming

Stemming is used in order to remove the suffixes from the word. Terms with common stem will usually have similar meanings. Consider the words connect, connected, connecting, connection. These words might be used based on the context but the stem of the word remains same that is connect. Stemming is a process of removing such morphological and in flexional endings and deriving the stem of the word. The algorithm being used for this purpose is called the Porter Stemmer which works on the principle of suffix stripping.

E. Knowledge Extraction

The required Use Case diagram features are extracted in this module according to the given rules. This module classifies noun phrases as actors and then adds it to the actors data set and verb phrases as use cases and adds it to users data sets. It then determines the relationship between the actors and their use cases. It Works based on three rules.

1) Rule 1: Identify Actors

1) A common noun may indicate an actor type.
2) A proper noun may indicate an actor.
3) If a consecutive noun exists and the last noun is not included within the following set of words: [number, no, code, date, type, volume, birth, id, address, name], then the consecutive noun may be an actor (e.g., Company staff).
4) Ignore every proper noun such as (Location name, Person name, etc.).

2) Rule 2: Identify use cases

1) The main verbs may indicate a use case type.
2) The transitive verbs may indicate a use case type.
3) If a noun follows a verb such as "verb+ noun" then may indicate a use case type (e.g., Generate diagram).
4) Ignore every verb included in the following list {include, involve, consist of, contain}

All rights reserved by *www.ijsrd.com*

3) Rule 3: Identify Relationships (Associations)

The Association between the actor and the use case can be derived from sentence boundary or based on some exceptions. Let A={A1,A2,.....An}be the set of all actors. Each actor is related with none, some, or, all use cases. The use case is denoted by v. Certain use cases can be added to or deleted from the last actor An . Let Tk be set of all such use cases k={1,2...n}.

The relationship between can be obtained by following the set theory rules as follows

1) Ai=Aj if and only if for all v(v Ai v Aj)
2) Ai UAj ={v:vAi or vAj}
3) Ai U Tk={v:vAi or vTk}
4) Uim=1 Am ={v : vAm for some m {1,2,....i}}
5) Unm=1 Am ={v : vAm for some m {1,2,....n}}

To this end, we can make a clear correspondence of some relationships between actors as follows:

a) Comparing cohesive device such as:
- Similarly, and equally (e.g. Ai and Aj are equally) can be formulated as Ai=Aj
- Ai is same as Aj can be formulated as Ai=Aj
- Ai is same as before, can be formulated as Ai=Ai-1. iv. Ak is same as Ai and Aj can be formulated as Ak = Ai U Aj
- Ai is same as all above actors can be formulated as Um=1i Am
- Ai is same as all below actors can be formulated as Um=1n Am

b) Backwards reference cohesive device, Pronoun, can be formulated as An U Tk c. Additive cohesive devices such as "also", "plus", "moreover", "furthermore", "Besides", "additionally" and "as well as" combined with backward reference Devices, Pronouns, can be formulated as An U Tk.

4) Rule 4

Identify Classes All proper nouns are identified as classes. Actors are considered to be actors and prompt the user to select the required classes.

5) Rule 5

Identify Attributes, Operations All the classes identified from proper nouns has the id and name as attributes. The related use cases would be identified as the functions. User is given the access to enter the attributes and functions for the checked classes.

F. Generation of Use Case Diagram

This module uses use case diagram symbols to draw use case diagrams according to the extracted information which are supplied by the previous modules.

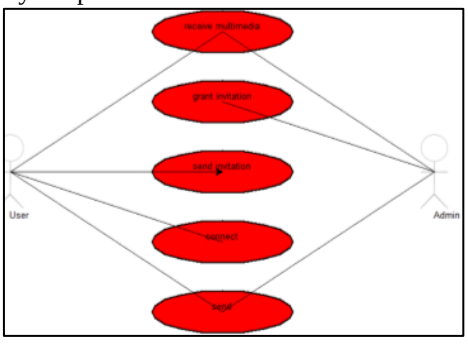

Fig. 2:

G. Generation of Class Diagram

This module uses class diagram symbols to draw the class diagrams from the information extracted by the previous modules.

Fig. 3:

VI. CONCLUSION & FUTURE WORK

The main objective of this project is to automate generation of Use Case and Class Diagrams from the natural language specifications given by the user. With a friendly user interface, task of generating Use Case and Class Diagrams has been implemented. A jpeg image of the specifications is also stored in the directory for future reference.

The future work includes the automation of acquiring the attributes and operations for the class diagram.

REFERENCE

[1] S. Bajwa, and M. A. Choudhary, "Natural language processing based automated system for uml diagrams generation," in The 18th Saudi National Computer Conf. on computer science (NCC18). Riyadh, Saudi Arabia: The Saudi Computer Society (SCS), Riyadh, Saudi Arabia, 2006, pp. 1-6.

[2] S. K. Shinde, V. Bhojane, and P. Mahajan, "Nlp based object oriented analysis and design from requirement specification," International Journal of Computer Applications (IJCA), vol. 47, no. 21, 2012.

[3] M. Ilieva, and O. Ormandjieva, "Models derived from automatically analyzed textual user requirements," in Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), 2006, pp. 13-21.

[4] G. Fliedl, C. Kop, H. C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, and C. Winkler, "Deriving static and dynamic concepts from software requirements using sophisticated tagging," Data & Knowledge Engineering, vol. 61, no. 3, pp. 433-448, 2007.

[5] M. Tjoa, and L. Berger, "Transformation of requirement specifications expressed in natural language into an EER model," in International Conference on Conceptual Modeling, Texas, USA, 1993, pp. 206-217.

[6] J. J. Gutiérrez, C. Nebut, M. J. Escalona, M.Mejías, and I. M. Ramos, "Visualization of use cases through automatically generated activity diagrams," in International Conference on Model Driven Engineering

All rights reserved by *www.ijsrd.com*

Languages and Systems Systems, Toulouse, France, 2008, pp. 83-96.
[7] S. M. Seresht, and O. Ormandjieva, "Automated assistance for use cases elicitation from user requirements text," in Proceedings of the 11th Workshop on Requirements Engineering (WER 2008), 2008, pp. 128-139.
[8] Cayaba, J. A. Rodil, and N. R. Lim, "CAUse: Computer Automated Use Case Diagram Generator," pp. 1-4, 2006.
[9] N. Arman, "Using MADA+ TOKAN to Generate Use Case Models from Arabic Use Requirements in a Semi-Automated Approach," in The 7th International Conference on Information Technology, Jordan, 2015.
[10] Denger, J. Doerr and E. Kamsties, "A survey on approaches for writing precise natural language requirements," IESE Research Report, 2001, Fraunhoffer-IESE.
[11] M. Luisa, F. Mariangela and N. I. Pierluigi, "Market Research on requirements analysis using linguistic tools," in Requirements Engineering, vol. 9, no. 1, 2004, pp. 40-56.
[12] Sommerville, Software Engineering, 9th ed., Pearson India, 2011, vol. 85, 1994, pp. 367– 376
[13] L. Mich and R. Garigliano, "A linguistic approach to the development of object-oriented systems using the NL system LOLITA," in international Symposium on Object-Oriented Methodologies and Systems (ISOOMS‟94), LNCS, vol. 858, 1994, pp. 371–386.
[14] H.M. Harmain and R. Gaizauskas, "CM-Builder: An automated NL- based CASE Tool," in Proceedings of 15th IEEE International Conference on Automated Software Engineering (ASE‟00), 2000, pp. 45 – 53.
[15] S. Overmeyer, B. Lavoie and O. Rambow, "Conceptual Modeling through Linguistic Analysis using LIDA," in Proceedings of 23rd International Conference on Software Engineering (ICSE‟01), Canada, 2001, pp. 401-410.
[16] M. Ibrahim and R. Ahmad, "Class Diagram extraction from textual requirements using natural language processing (NLP) techniques," in 2nd International Conference on Computer Research and Development, 2010, pp. 200-204.
[17] P. More and R. Phalnikar, "Generating UML Diagrams from natural language specifications," in International Journal of Applied Information Systems, vol. 1(8), 2012, pp. 1923.
[18] S.D. Joshi and D. Deshpande, "Textual requirement analysis for UML diagram extraction by using NLP," in International Journal of Computer Applications, vol. 50(8), 2012, pp. 42-46.interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[19] H. Herchi and W.B. Abdessalem, "From user requirements to UML class diagram," in CoRR abs/1211.0713, 2012.
[20] D.K. Deeptimahanti and R. Sanyal, "Static UML model generator from analysis of requirements (SUGAR)," in Proceedings of International Conference on Advanced Software Engineering and Its Applications (ASEA‟08), China, 2008, pp.
[21] M. Landhauber, S.J. Korner and W.F. Tichy, "From requirements to UML models and back: how automatic processing of text can support requirements engineering," in Software Quality Journal, Springer US, vol. 22, no. 1, 2014, pp. 121-149.

All rights reserved by *www.ijsrd.com*