Generating UML Use Case and Activity Diagrams Using NLP Techniques and Heuristics Rules

Abdelsalam M. Maatuk
Faculty of Information Technology, Benghazi University, Libya
abdelsalam.maatuk@uob.edu.ly

Esra A. Abdelnabi
Faculty of Information Technology, Benghazi University, Libya
esra.ali@uob.edu.ly

ABSTRACT

The process of generating Unified Modeling Language (UML) Diagrams from Natural Language (NL) requirements is considered a complex and challenging task. Software requirements specification is often written in NL format, which causes potential problems. Requirement's analysts analyze and process natural language requirements manually to extract the UML elements. The manual analysis takes a lot of time and effort, which justifies the need for automated support. This paper proposes an approach to facilitate the NL requirements analysis process and UML diagrams extraction from NL textual requirements using natural language processing (NLP) techniques and heuristics rules. This approach focuses on generating use-case and activity diagrams. The approach has been applied to a case study and evaluated through an experimental. The evaluation of the approach will be conducted through a comparative study. The experimental results prove that the proposed approach is considerably improved as compared to the other approaches.

KEYWORDS

Requirement Analysis, NLP, UML diagrams, Software Requirement Specification

ACM Reference Format:
Abdelsalam M. Maatuk and Esra A. Abdelnabi. 2021. Generating UML Use Case and Activity Diagrams Using NLP Techniques and Heuristics Rules. In *International Conference on Data Science, E-learning and Information Systems 2021 (DATA'21), April 05–07, 2021, Ma'an, Jordan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3460620.3460768

1 INTRODUCTION

Requirements analysis and production of UML models used in the analysis and design phases is a difficult and complex task [18]. Natural Language (NL) is often used to specify the software requirements, which is considered as one of the main reasons for potential problems that occur during the requirement analysis and extraction of UML models. NL requirements can often be ambiguous, incomplete, and inconsistent, which may cause multiple interpretations as well as difficulties in understanding the requirements. However, in software development Software Requirement Specifications (SRS) is written in natural language, because it is the communication language between different stakeholders. The requirements analyst later converts it to formal representations such as UML, after he has detected and fixed the problems found in natural language requirements documents [1, 2, 19–21].

However, the manual analysis process is a critical task and takes a lot of time and effort. Moreover, this manual processing can cause vital errors if the requirements analysts do not have extensive domain knowledge may overlook some NL defects, which can lead to multiple interpretations and difficulties in recovering implicit requirements [3]. Natural Language Processing (NLP) techniques show some encouraging results to overcome such NL problems [23, 24]. Because of that, it has become imperative to use NLP techniques as assistance in analyzing and processing natural language requirements [1].

NLP provides automatic text processing, which can be applied to NL requirements documents to generate more controlled requirement specifications, and extract the important information to produce UML models. The NLP systems perform different linguistic analysis levels to analyze the NL text: Phonological level, Morphological level, Lexical level, Syntactic level, Semantic level, Discourse level, and Pragmatic level. Besides NLP techniques, Domain Ontology has been widely used to support the analysis process and improve the efficiency of concept identification. Domain ontology refers to domain knowledge that consists of structured concepts which are semantically related to each other [4, 5].

However, producing UML models from NL requirements is a challenging and time-consuming task. UML models such as use-cases, activity, and class diagrams are usually used to document requirements in a more structured way [25]. A lot of research is done by proposed many approaches and tools to bridge the gap between requirement analysis and design phases by developing UML models from NL requirements [6, 22].

Although there is noticeable research in this area, the existing studies usually require a few manual processing on textual requirements. Therefore, in this paper, we propose an approach to generate use case and activity diagrams from informal natural language requirements. This paper is an extension to our earlier work [7] describing an approach for developing both static and dynamic UML models from NL requirements and evaluated with the help of a case study. Our approach uses NLP techniques for parsing and analyzing sentences, and a set of heuristics rules to transform the NL concepts into UML concepts. Our previous work just focused on generating UML class models from NL requirements.

The rest of this paper is organized as follows: Section 2 discusses some of the related works and addresses their limitations. Section

DATA'21, April 05–07, 2021, Ma'an, Jordan

ACM ISBN 978-1-4503-8838-2/21/04…$15.00
https://doi.org/10.1145/3460620.3460768

3 described in detail the proposed approach and the used techniques. Section 4 describes the use of the proposed approach with a case study. Section 5 discusses the evaluation methodology and experimental results, and finally, Section 6 presents conclusions and futures works.

2 RELATED WORKS

A semi-automated algorithm proposed by Nassar and Khamayseh [8] to generate an activity diagram from Arabic user requirements using NLP tools. MADA+TOKAN parser is used to splitting and tokenizing the Arabic text, in which the elements of the activity diagram have been extracted from user requirements. Also, a set of heuristics is presented to find the notations of an activity diagram (activity node, a decision node, control flow). The basic grammars for verbal and conditional sentences are presented. Common grammar rules of verbal and conditional sentences are used to obtain the activity and decision nodes. A high-level algorithm is presented to implement this approach. However, it generates only activity diagrams. Furthermore, it deals only with Arabic language requirements, and the requirements should be written in a simple and correct format following some general guidelines. The algorithm is not implemented with real cases written in Arabic language and it is not integrated with a graphical CASE tool.

A method proposed by Iqbal and Bajwa [9] to automatically maps SBVR rules into an activity diagram and applied in a tool named SBVR2ACTIVITY. First, the SBVR specification is pre-processed to find out the candidates for possible SBVR vocabulary. Stanford POS tagger is used to tokenize and POS tag the input. It is also used to syntactically analyze the POS tagged SBVR specifications, and generates the parse tree and type dependencies. Then, the extracted information is used for the detailed semantic analysis. SRL is performed to identify the SBVR vocabulary. The SBVR elements such as noun concept, individual concept, verb concepts, etc. are identified from the SBVR input. Then, a set of transformation rules are presented to transform SBVR into Activity Diagram. Finally, SBVR rules are further mapped to UML metamodel elements and a visual representation of the Activity Diagram is drawn. However, this approach generates only activity diagrams. Furthermore, the requirements must be written in the form of SBVR representation, since this method only takes the system requirements specified in SBVR syntax as input.

SUGAR is a tool presented to produce use case and class models from NL requirements [10]. It suggests syntactic reconstruction rules for processing complex NL statements. These statements are used to identify actors as noun phrases and use-cases as event flows in the system. NL text is analyzed using Stanford Parser to identify actors from the subjects of each sentence. The use cases associated with the actors are identified by extracting the predicates of each requirement. From this, the use case model is generated with the proper association among use cases. Then, noun phrases are identified to extract candidate objects using the use case model and the requirements. After that, the RUP approach is used to refine the identified class elements which are identified by recognizing the verb phrases and adjectives, and the relationship is identified. Finally, both analysis and design class models are generated. However, the accuracy of candidate class identification and relationship is limited. Although this system identifies duplicate classes, the identification and dropping of irrelevant classes are not applicable. It needs human interaction to eliminate irrelevant classes while deciding the candidate classes which may sometimes mislead if there is a lack of domain knowledge. The main gap is that it poses a constraint on specifying requirements like each sentence should be expressed in active voice and requires to rewritten most statements of the requirements in some restricted NL form.

An approach proposed by Kamarudin et al. [11] to generate use case and activity diagrams from user requirements. The developed tool is named RETRANS, which has been developed using JAVA. The approach receives the requirement and performs POS using Stanford POS Tagger, after that there are three phases. Requirement phase; which is requirements keywords tracking used to detect the actor and use cases for the requirements. Use Case Diagram generation phase; the use case diagram library is used to generate a use case diagram by connecting actors with the related use cases. Activity Diagram generation phase; which is after use case generation. The system specifies all use cases involved in the use case diagram, then it generates the activity diagram by linking the components inside the class library (activity). However, this approach concentrates only on some dynamic diagrams and some enhances are needed to generate more diagrams. Besides, it is unable to extract include, extend and generalize relationships.

A semi-automated approach was proposed by Arman and Jabbarin [12] to generate a use case diagram from Arabic textual requirements. MADA+TOKAN is used to parse the statements to obtain the information needed to determine the potential actors and use cases. First, MADA+TOKAN is used to split and tokenize the text. Then, a set of heuristic rules is presented to obtain the diagrams and identify the actors and use cases. These heuristics use the tokens produced by a Stanford parser. Then, these tokens are used as the main components of the use case diagram. Finally, to generate the use case model, a matrix that consists of columns contain the potential use cases, and rows contain the potential actors is used. The matrix is filled with arrow symbols, which means that an actor is associated with one or more particular use cases. Once the matrix is created, the use case model is obtained by taking an actor with all its associated use cases to generate a use case diagram. However, it generates only use case diagrams and deals only with Arabic language requirements. It is assumed that the requirements are "good" in the sense implied by the IEEE good requirements assumptions. Moreover, it is unable to extract include, extend and generalize relationships automatically, which means that it needs human intervention to identify these relationships.

An automated structured approach was proposed by Alksasbeh et al. [13] to acquire, analyze, and transform NL descriptions into use case diagrams. To describe the system requirements the user writes a few paragraphs in simple English language. First, this approach uses NLP techniques such as tokenization and POS tagging to parse the specifications based on a predefined set of syntactic heuristics rules and to extract use case diagram elements through translating the specifications to words. Then, these words are categorized into several POS and then stemming algorithms and a set of syntactic heuristic rules are applied to identify the actors and use cases of the target software system. Finally, the system draws the desired use case diagram after the compound analysis

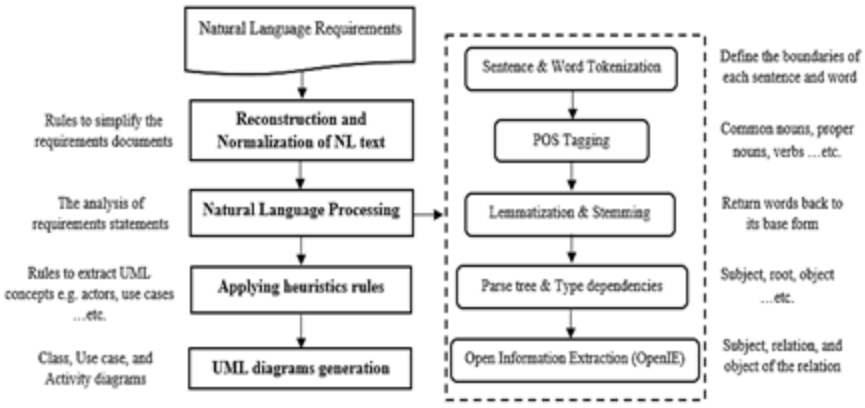

Figure 1: Generating Use Case and Activity diagrams from NL requirements

and extraction of associated information. This system can read and perform a complete analysis of the user requirements and generate use case diagrams automatically. However, it generates only a use case diagram. It can generate only self-contained concrete use cases which constitutes a complete flow of events. Moreover, it cannot reuse other existing use cases via include, extend, and generalize relationships.

A process proposed in [14] to generate use case diagrams from a set of user stories in the form of OWL ontologies stored in a text file. The process is based on NLP tools and techniques. TreeTagger parser applied POS tags to categorize terms into proper nouns, personal nouns, or verbs, etc. Given that the syntax of user stories simple and semi-structured, which allows classifying each term in the user story into a single POS easily. Then, the use case elements which are actors, associations, and use cases are identified. Finally, the generated use case diagram is visualized using the Visual Paradigm tool. This technique reducing ambiguity in NL requirements and generating automatically use case models which help designers to interpret the same user story in one way. However, some of the problems faced in the case study are derived from complicated sentences, which often require inclusion or exclusion relationships between use cases that are not supported by this plugin. It only generates use case diagrams and does not support sentences containing more than one compound noun. Moreover, it does not address other types of relationships, such as generalization and specialization between actors and use cases.

3 THE PROPOSED APPROACH

The proposed approach consists of five phases as shown in Figure 1. 3.1 Reconstruction and Normalization of NL text, 3.2 Natural Language Processing, 3.3 Applying mapping rules to extract UML elements and 3.4 UML diagrams generation.

3.1 Reconstruction and Normalization of NL Text

To reconstruct natural language text and split the complex requirements into simple statements, a set of syntactic rules is proposed. We have defined sixteen syntactic reconstructing rules in [7]. These rules should be applied while writing and normalizing the requirements documents to facilitate the UML elements extraction and increase the accuracy of the results.

3.2 Natural Language Processing (NLP)

NLP tools and techniques are used for analyzing and processing natural language requirements text to avoid manual processing. These tools receive the requirement text and perform different NLP techniques to the entered text. First, the tool splits and tokenizes user requirements texts into sentences. Then, the text is tokenized into words and normalizes these tokens through the stemming and lemmatization process. After that, part of speech (POS) tagging is performed to identify the role of each word on the sentence such as noun, verb, adjective, pronoun, preposition, conjunction, numeral, article, etc. Then, the noun and verb phrases are identified through the type dependencies tags or parse tree generation, and finally, the grammatical relations between words in a sentence are recognized through Open Information Extraction (OpenIE). "Stanford CoreNLP" [15] is used in this work to perform NLP because it offers excellent results. This phase comprises several sub-processing stages to process the natural language (English) statements. This phase includes the following five sub-phases:

- Text Tokenization
- Parts-of-Speech (POS) Tagging
- Stemming and Lemmatization
- Type Dependencies
- Open Information Extraction

3.3 Mapping Rules to Extract UML Concepts

In this phase, the UML use case and activity diagrams elements are extracted from the NL processing phase output after using the Stanford Corenlp NLP tool. A set of rules are proposed to apply the transformation process and extract the UML elements (actors, use cases, activity nodes, . . ., etc.). Finally, the resulted UML components can be drawn manually or using UML drawing tools.

3.3.1 Use Case Diagram Identification Rules. In this section, a set of rules to extract the use case diagram elements (actors, use cases, and associations between actors and use cases) are described. The noun

phrases (NP) are classified as actors and the verb phrases (VP) as use cases. The use-cases are mostly the predicates (verb phrases) in the sentence, which are associated with an actor (subject) (NP: VP). Then, it determines the relations for each actor with its use cases. The following rules are applied for use case element identification.

Actors Identification Rules. **AC-Rule 1:** Extract common noun that is the subject of the sentence from <NN> and <subj> tags, then they are mapped to candidate actors.

AC-Rule 2: Extract proper nouns that is the subject of the sentence from <NNP> and <subj> tags, then they are mapped to candidate actors. Proper nouns usually represent actors from the outside environment that interacted with the system.

AC-Rule 3: If a sentence in the form (Subject-Predicate) or (Subject-Predicate-Object), then the subject of the sentence <subj> is mapped to a candidate actor.

AC-Rule 4: If the subject of a sentence is a noun or noun phrase consisting only of nouns (Noun+ Noun), then this noun phrase (Noun+ Noun) is mapped to a candidate actor.

AC-Rule 5: 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 a candidate actor (e.g., Company staff).

AC-Rule 6: Proper nouns such as location name, person name, etc. ignore them as an actor.

AC-Rule 7: If the word "system" is the subject of the sentence, then ignore it as an actor.

AC-Rule 8: The actor should be the giver (Subject) of the action (Verb).

Use-cases Identification Rules. **UC-Rule 1:** Extract verb phrase that contains lexical or main verbs (e.g., want, act, pull, go, make) of a personal noun from <VB> tag and map them to a use case (or part of a use case) linked with an actor defined by this noun.

UC-Rule 2: Extract verb phrase that contains transitive verbs (e.g., take, send, buy) from <VB> tag and them to a candidate use case.

UC-Rule 3: Extract the phrasal verbs (verb <VB>+ preposition <IN> or verb <VB>+ adverb <RB>) and map them to a candidate use case, e.g., check-in, get into.

UC-Rule 4: If a noun follows a verb (VB+ NN) then map those to a candidate use case type (e.g., Generate diagram).

UC-Rule 5: If the subject of a sentence is an actor, then the following predicate (P) forms (P: V; P: V/NP; P: V/PP) are mapped to candidate use cases. V: Verb; NP: Noun Phrase; PP: Prepositional Phrase.

UC-Rule 6: If a sentence in the form (Subject-Predicate) or (Subject-Predicate-Object), which subject and object are mapped to actors, then predicates are mapped to use-cases.

UC-Rule 7: Ignore every verb included in the following list {include, involve, consist of, contain}.

Relationships (Associations) Identification Rules. **UR-Rule 1:** If a sentence in the form (Subject-Predicate) or (Subject-Predicate-Object), which subject and object are mapped to actors, and predicates are mapped to use-cases, then an association exists between actors and use-cases.

UR-Rule 2: Identify prepositions from <IN> tag between predicates and objects. Sometimes, predicates internally contain NP which represents an actor (Verb+ Preposition+ Noun), prepositions like "from, to, about, with, in, etc." identified and associates use-case and actor together.

UR-Rule 3: If the subject of a sentence is an actor and the predicate is of the form (P: VP/NP1/"to"/NP2), then VP/NP1 is the use case and NP2 is an actor and an association exists from an actor to the use case.

UR-Rule 4: If the subject of a sentence is an actor and the predicate is of the form: (P: VP/NP1/ "from" /NP2), then VP/NP1 is the use case and NP2 is an actor and an association exists from the actor to the use case.

3.3.2 Activity Diagram Identification Rules. In this section, a set of rules to extract the activity diagram elements (i.e., activity node, a decision node, control flow) from requirements text are described. The verb phrases identify activities. Verbal sentences consist of (Subject+ Verb+ Object) identify the activity nodes and conditional statements identified the decision nodes. Finally, all elements in the activity diagram can be connected using control flows. The following rules are applied for activity diagram element identification.

Activities Identification Rules. **AS-Rule 1:** Generate an activity for each functional requirement.

AS-Rule 2: Extract the verbs <VB> tag from the text, then they are used to identify activities.

AS-Rule 3: Use each use case name as the activity name.

AS-Rule 4: Use the objectification; giving instances a type name, e.g., "*All books may be borrowed*" might be objectified as "*books borrowing*". An objectification allows to predicate things, like how, when and where they happened, etc.

AS-Rule 5: Common nouns <NN> and proper nouns <NNP> are mapped to an activity or action state, e.g., *bank, book,* etc.

Activity Node Identification Rules

AN-Rule1: If the sentence in the form (S+V+ O) and having transitive verbs *(e.g., take, send, buy)*, then it is mapped to an activity node.

Decision Node Identification Rules. **DN-Rule1:** Generate a decision node and additional actions for each exceptional step and conditional statements, e.g., "*The new system should produce reminders when a book is overdue".*

DN-Rule 2: If a sentence is in the form (IF+ Noun "Subject"+ Verb "Predicate" +Noun "Object") or (IF+ Noun "Subject"+ Verb "Predicate"), then map it into a decision node.

DN –Rule 2.1: Extract the adverb (If) from <RB> tag and mapped it to a conditional particle.

DN –Rule 2.2: Extract the verbal sentence after the conditional particle <If> and map it to a conditional sentence.

DN –Rule 2.3: Extract the verbal sentence after the particle (e.g., then) from <RP> tag that represents the conditional answer.

DN –Rule 2.4: If the next statement turns out to be without "else", then it would be a simple flow in the diagram without any decision box and separation.

DN –Rule 2.5: If the next statement turns out to be with conditional particle "else", then use an arrow to separate between two

verbal statements, which are the true and false branches of the decision node:

The verbal sentence after "then" represents the true branch of the activity node.

The verbal sentence after "else" represents the false branch of the activity node.

DN-Rule 3: The full stop <punct> tag after the end of each sentence, is converted to an arrow separating notations.

Action Name Identification Rules. **AN-Rule:** To name the actions which should be invoked during the transition from one activity to another merge the verb and last term (V+N) to create the name of the action, e.g., *if the user returns a book out of time, then the system will produce reminders,* action name "*produce reminders*".

Activity Group Identification Rules. **GR-Rule:** If a sentence has a transitive verb *(e.g., take, send, buy)*, then extract the giver of the action of the verb which is the subject <nsub> of the sentence, e.g., "*library member*" in "*library member may be borrow books*".

3.4 UML Diagrams Generation

The outcomes from Phase III produce a set of UML use case and activity diagrams elements. Finally, the resulted UML diagrams components are obtained to be drawn manually or using UML drawing tools. The algorithms to generate UML diagrams are described below.

3.4.1 Algorithm for Extraction of the Use Case Diagram. The following algorithmic steps represent the generation of Use Case Diagrams from English User Requirements. The algorithm is presented below:

Input: English User Requirements.

Output: Use Case Diagram elements.

Step 1: Use the natural language requirements document as input.

Step 2: Use the same steps in the class diagram algorithm from Step 2 to Step10.

Step 3: Use the output of "Stanford POS tagger and parser" as the input to apply the heuristic rules to extract the use case diagram elements.

3.1 Extract the subject from the statement. It is a name that issues a command or asks for a service identified by the word "subj".

3.2 Extract the predicates (verb phrases (VP)) from the statement which are associated with an actor (subject).

3.3 If the subject of a sentence is an actor and the following predicate (P) a use case, then an association exists between actor and use-case.

Step 4: Integrate data obtained from steps 3.1 to 3.3 to draw the Use case diagram.

Step 5: Use the refinement rules to refine the extracted use case diagram**.**

3.4.2 Algorithm for Extraction of the Activity Diagram. The following algorithmic steps represent the generation of Activity Diagrams from English User Requirements. The algorithm is presented below:

Input: English User Requirements.

Output: Activity Diagram elements.

Step 1: Parse User Requirements using Stanford Parser.

Step 2: Use the same steps in the class diagram algorithm from Step2 to Step10.

Step 3: Use the output of "Stanford POS tagger and parser" as an input to apply the heuristic rules for the extraction of activity diagram elements.

3.1 Construct one initial node to show the start of an activity diagram and give it the name "start".

3.2 If the requirements statement is in the form of (S+V, S+V+O), then generate a corresponding activity node.

3.3 If there is a conditional particle (IF) <RB> tag in the requirements statement in Step 3.2, then we generate the decision node:

3.3.1 The verbal sentence after the conditional particle <If> is the Conditional sentence.

3.3.2 Check the next statement after the conditional sentence, If there is a particle (e.g., then), then the verbal sentence after it represents the Conditional answer.

3.3.3 If the next statement turns out to be without "else", then it would be a simple flow in the diagram without any decision box and separation.

3.3.4. If the next statement turns out to be with conditional particle "else", then use an arrow to separate between two verbal statements which are the true and false branches of the activity node:

a. the verbal sentence after "then" represents the true branch of the activity node.

b. the verbal sentence after "else" represents the false branch of the activity node.

3.4 The full stop (**.** / punct) after the end of each sentence, is converted to an arrow separating notations.

3.5 Construct one final node to show the end of the activity diagram and give it the name "End".

Step 4: Integrate data obtained from steps 3.1 to 3.5 to draw the activity diagrams.

4 EXPERIMENTS AND RESULTS

The study aimed to improve the process of generating UML use case and activity diagrams from informal natural language requirements, many related works are reviewed firstly to specify the limitation of each tool and what is the problem in this area, and to know the most efficient techniques that have been used to analyze informal natural language requirements and generate UML diagrams.

We describe the application of the approach on the Natural language text on a case system called "Qualification Verification System (QVS)". The used scenario case system has been considered as a case study in an existing approach named UMGAR [16]. This is to demonstrate the advantages of using our approach when compared with other approaches. For the validation of the proposed approach, we reported the outcome of a controlled experiment that we have conducted to compare the UML elements and diagrams generated by our approach with those generated by UMGAR. The results of the experiment clearly showed that the diagrams generated by our approach were significantly better than those generated by the other approach.

Moreover, the proposed approach has been compared with some of the existing approaches that have been presented in the literature. We presented in section II several UML diagram generation

Table 1: Comparison Table

Diagram	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[16]	**The proposed approach**
Class	N	N	Y	N	N	N	N	Y	Y
Use case	N	N	Y	Y	Y	Y	Y	Y	Y
Activity	Y	Y	N	Y	N	N	N	N	Y

approaches from NL software requirements using NLP techniques. We made a comparison between these approaches and the proposed approach because they also use similar techniques to extract the UML elements. We have compared the results and functionalities of our approach with these approaches. The summary of the comparison is given in Table 1 The evaluation shows that none of these tools can extract class, use case, and activity diagrams from NL requirements. Thus, the results of this evaluation are very encouraging and support both the potential of this technology and the approach adopted in this paper.

5 CONCLUSION

In this paper, we have described an approach that is based on NLP techniques and heuristics rules to generate initial use case and activity diagrams from informal NL requirements. This approach uses NLP techniques to analyze NL software requirements documents and extract the relevant information. NLP techniques such as tokenization, POS tagging, and type dependency are used to avoid the manual processing of NL textual requirements and facilitate the transformation of these requirements into UML elements. The approach can extract the relevant concepts based on noun and verbs phrases. Then, apply a set of heuristics rules to transform this information into UML elements and find out concepts such as use case names, actors, etc. to finally produce UML diagrams.

The algorithms to generate UML has been explained in this paper. The approach is validated through a benchmark case study. The results produced by the proposed approach for a case study (Qualification Verification System (QVS)) were analyzed in detail [17], and the evaluation is based on comparing the outputs with other approaches and UML diagrams which are created by other tool using the same case study. Obtained results were encouraging and have demonstrated the benefits of the approach. Experimental results show that functionalities of the proposed approach are significantly improved as compared to the other approaches. Furthermore, the approach does not require any manual processing on textual requirements. This approach can help the analyst with the requirements elicitation and modeling and reduce the cost and time of software development.

Currently, the proposed approach does not deal with the generation of some use case relationships such as include and extend. In future works, we intend to solve the mentioned challenges and include such missing features. Furthermore, we intend to add more rules related to the information systems domains to improve the system's performance and functionalities and use many NL documents and case studies.

REFERENCES

[1] Vincenzo Ambriola and Vincenzo Gervasi. 1997. Processing natural language requirements. In Proceedings 12th IEEE International Conference Automated Software Engineering (ASE). IEEE, 36-45 https://doi.org/10.1109/ase.1997.632822

[2] Mohd Ibrahim and Rodina Ahmad. 2010. Class diagram extraction from textual requirements using natural language processing (NLP) techniques. In 2010 Second International Conference on Computer Research and Development. IEEE, 200-204. .

[3] Deva K. Deeptimahanti and Muhammad A. Babar. 2009. An automated tool for generating UML models from natural language requirements. In 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE, 680-682.

[4] Hatem Herchi and Wahiba Ben Abdessalem. 2012. From user requirements to UML class diagram. In International Conference on Computer Related Knowledge (ICCRK' 2012). Sousse, Tunisia.

[5] Priyanka More and Rashmi Phalnikar. 2012. Generating UML Diagrams from Natural Language Specifications. International Journal of Applied Information Systems. 1, 8 (April 2012), 19-23. https://doi.org/10.5120/ijais12-450222

[6] Deva K. Deeptimahanti and Ratna Sanyal. 2011. Semi-automatic Generation of UML Models from Natural Language Requirements. In Proceedings of the 4th India Software Engineering Conference. 165-174.

[7] Esra A. Abdelnabi, Abdelsalam M. Maatuk, Tawfig M. Abdelaziz, and Salwa M. Elakeili. 2020. Generating UML Class Diagram using NLP Techniques and Heuristic Rules. In 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA). IEEE, Monastir, Tunisia, 277-282. https://doi.org/10.1109/STA50679.2020.9329301

[8] Ibrahim N. Nassar and Faisal T. Khamayseh. 2015. Constructing activity diagrams from Arabic user requirements using Natural Language Processing tool. In 2015 6th International Conference on Information and Communication Systems (ICICS). IEEE, 50-54.

[9] Usama Iqbal and Imran S. Bajwa. 2016. Generating UML activity diagram from SBVR rules. In 2016 Sixth International Conference on Innovative Computing Technology (INTECH). IEEE, Dublin, Ireland, 216-219.

[10] Deva K. Deeptimahanti and Ratna Sanyal. 2008. Static UML Model Generator from Analysis of Requirements (SUGAR). In 2008 Advanced Software Engineering and Its Applications (ASEA). IEEE, 77-84.

[11] Nuri J. Kamarudin, Nor F. M. Sani, and Rodziah Atan. 2015. Automated transformation approach from user requirement to behavior design. Journal of Theoretical and Applied Information Technology. 81, 1 (November 2015), 73-83.

[12] Nabil Arman and Sari Jabbarin. 2015. Generating Use Case Models from Arabic User Requirements in a Semiautomated Approach Using a Natural Language Processing Tool. Journal of Intelligent Systems. 24, 2 (June 2015), 277-286. https://doi.org/10.1515/jisys-2014-0092

[13] Malek Z. Alksasbeh, Bassam A. Alqaralleh, Tahseen A. Alramadin, and Khalid A. Alemerien. 2017. AN AUTOMATED USE CASE DIAGRAMS GENERATOR FROM NATURAL LANGUAGE REQUIREMENTS. Journal of Theoretical & Applied Information Technology. 95, 5 (March 2017), 1182-1190.

[14] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. 2018. Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques. Procedia Computer Science. 130 (January 2018), 42 – 49. https://doi.org/10.1016/j.procs.2018.04.010

[15] The Stanford Natural Language Processing Group. CoreNLP. v. 4.0.0. Retrieved May 7, 2020, from https://stanfordnlp.github.io/CoreNLP/

[16] Deva K. Deeptimahanti and Muhammad A. Babar. 2009. An Automated Tool for Generating UML Models from Natural Language Requirements (UMGAR). In 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE, University of Limerick, Ireland, 680-682.

[17] Esra A. Abdelnabi. 2020. Generating UML Diagrams using NLP Processing Techniques and Heuristics Rules. MSc Thesis, Software Engineering Department, University of Benghazi, Libya.

[18] A. O Mohammed, Z. A Abdelnabi, A. M. Maatuk, and A. S Abdalla. "An Experimental Study on Detecting Semantic Defects in Object-Oriented Programs using Software Reading Techniques". In *Pro. of ACM Int. Conf. on Engineering & MIS* (ICEMIS '15), 2015, 6 pp. DOI=http://dx.doi.org/10.1145/2832987.2833025

[19] A. M. Maatuk, M. A. Ali and S. Aljawarneh. "Translating Relational Database Schemas into Object-based Schemas: University Case Study". In *Recent Patents on Computer Science*. Innovations in Educational Technology and E-learning Social Networking. Vol. 8, No 2, pages 122-132, 2015. **DOI:** 10.2174/2213275908666150710174102.

[20] S. F. Alshareef, A. M Maatuk, T. M., Abdelaziz, and M. Hagal. "Validation Framework for Aspectual Requirements Engineering (ValFAR)". *In Proc. of the 6th Int.*

Conf. on Eng. & MIS, 2020, pp. 1–7. DOI:https://doi.org/10.1145/3410352.3410777

[21] S. F. Alshareef, A. M Maatuk, T. M., Abdelaziz. "Aspect-Oriented Requirements Engineering: Approaches and Techniques". In *DATA '18*, October 1–2, 2018, ACM. https://doi.org/10.1145/3279996.3280009

[22] T. M., Abdelaziz, A. M Maatuk and F. Rajab. "An Approach to Improvement the Usability in Software Products". In *Int. Journal of Software Engineering & Applications (IJSEA)*, Vol.7, No.2, March 2016. DOI : 10.5121/ijsea.2016.7202 11

[23] Maatuk, A. M., Elberkawi, E. K., Aljawarneh, S., Rashaideh, H. and Alharbi, H, The COVID-19 Pandemic and E-learning: Challenges and Opportunities from the Perspective of Students and Instructors. Journal of Computing in Higher Education, 2020.

[24] Aljawarneh, S., Jaradat, R., Maatuk, A. M. and Alhaj, A. Gene Profile Classification: A Proposed Solution for Predicting Possible Diseases and Initial Results. In *Proceedings of IEEE International Conference on Engineering & MIS (ICEMIS)*, Agadir, Morocco, 2016, pp. 1-7. doi: 10.1109/ICEMIS.2016.7745342

[25] Aljawarneh, S., Al-Rousan, T. and Maatuk, A. M.. Security Issues Influencing the Usage of Online Banking in the Arab World: Data Validation, *Security Journal (ISI Thomson)*, pages 10, doi:10.1057/sj.2012.10 (2012).