Journal of Theoretical and Applied Information Technology
15th March 2017. Vol.95. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN AUTOMATED USE CASE DIAGRAMS GENERATOR FROM NATURAL LANGUAGE REQUIREMENTS

[1]MALEK ZAKARYA ALKSASBEH, [2]BASSAM A. Y. ALQARALLEH, [3]TAHSEEN A. ALRAMADIN, [4]KHALID ALI ALEMERIEN

[1&2&3]Faculty of Information Technology, Al-Hussein Bin Talal University, Ma'an, Jordan

[4]Department of Computer Information Systems, Tafila Technical University, Tafila, Jordan

E-mail: [1]malksasbeh@ahu.edu.jo, [2]alqaralleh@ahu.edu.jo, [3]tahsen@ahu.edu.jo, [4]khalid.alemrien@ttu.edu.jo

ABSTRACT

Use case modeling is an important requirements engineering technique which plays an important role in describing the systems specifications and facilitating systems development. The use of linguistic representations of system requirements as a source of information for generating use case models is a challenging task and can be considered relatively a new field. This paper has tackled the problem of extracting the required elements that are needed to automatically generate use case diagrams from specification documents which are written in common natural language. Therefore, we have developed 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. Our proposed approach was evaluated using both recall and precision performance measurements. The experiments revealed that our system has an average of 96% recall and 84% precision.

Keywords: *Use case diagrams, Natural Language Processing, User requirements analysis, Automatic Diagrams Generation, Information Extraction*.

1. INTRODUCTION

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 [1]. NLP is a field of both computer artificial intelligence and computational linguistics which is used to facilitate the interactions between computers and human (natural) languages. NLP has significantly contributed to the area of human-computer interaction [2]. Generally, NLP employed to automatically extract the information stored in natural language and convert it to a machine understandable format. However, the high ambiguity and complex grammar of unstructured data create significant challenges for the variety of NLP approaches [3]. NLP has become increasingly important with growing various applications which range from search, automated machine translation to general human-computer interaction [4].

Use case diagrams play a broader role in describing systems specifications, and facilitating systems analysis, design, and implementation [5]. However, building such diagrams is very important and time-consuming task which requires a complete understanding of the system requirements [6]. The use case models describe and represent the interaction between the actors and the system in order to achieve a goal [7]. An actor may be certain user type or a role played by a user, such as a person, an organization, another software system, a hardware device, a process [8, 9]. On the other hand, a use case represents some system's functionality, a specific way of using the system. However, generating use case models by extracting use case elements from linguistic textual representations using NLP is still a challenging task.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In this paper, we introduce a new automated structured approach to acquiring, analyze and then transform natural language descriptions into use case 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.

It is important to mention, that our proposed system extends the use of NLP to have semantic analysis besides the structured analysis to infer new linguistic features such as reference, comparing and additive cohesive devices such as pronouns, "equally", "similarly", "same as", "also", "plus", "moreover", "furthermore", "besides", "additionally" and "as well as". These cohesive devices are words and phrases that connect sentences and paragraphs together creating a smooth flow of ideas. Our methodology can be considered a step towards the analysis and the understanding of the English text as a semantic unit. The concept of cohesion is a semantic one which refers especially to non-structural text forming relation.

The rest of the paper is organized as follows: related work is presented in Section 2. Our proposed approach is described in Section 3, evaluation results are presented in Section 4, and conclusion in section 5. Finally, limitations and future work are presented in Section 6.

2. LITERATURE REVIEW

This section briefly reviews some leading research efforts which focused on applying natural language processing to extract knowledge from requirement specifications to generate use case diagrams, ER diagrams, and various UML diagrams.

Recently, some commercial products for representing use case models have been developed, including Visual UML, GD Pro, Smart Draw, Rational Rose, Microsoft's Visio, etc.[10]. Some advanced tools were proposed to automate some software engineering activities which are more complicated than just providing help in drawing various UML diagrams. In [11], the authors proposed an approach that can extract the basic elements for generating a class diagram such as classes, data members and member functions from user requirements written in a clear way. This approach was implemented as a software tool to generate the class diagrams. Two other approaches[12] and [13] take unstructured requirements in the form of plain text as input in order to derive activity diagrams.

Also, DMG [14] provides a basis for the development of new heuristics applied in ER-Converter which extracts knowledge from requirements specifications. DMG is the only existing work that proposes a large number of both syntactic and semantic heuristics to be used in transforming natural language into ER models. However, the work has not been implemented. In[15], the authors give another good example of heuristics-based approaches for generating ER elements from natural language specifications. A number of syntactic and semantic heuristics were proposed in order to produce good results in identifying the relevant and correct results of the ER elements. Another ER generator [3] is a rule-based system that generates ER models from natural language specifications. The proposed methodology is based on a set of generic and specific rules that combine different concepts from others work.

Another paper [10] presents NLP based automated system for generating UML diagrams after analyzing the given system requirements in the form of text written in simple and correct English. Once the associated and needed information is analyzed and extracted, the designed system draws the various UML diagrams such as sequence, class and activity diagrams.

The authors in [16]formalize use cases as instances of a Meta model. However, the Meta model instance has to be manually provided by the user directly which require extensive user efforts. Another approach [17] introduced an approach to derive use-case and class diagrams from an event table. Their approach completely depends on the availability of a comprehensive event table which is built from the system requirements. The authors in[18] proposed an approach 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

technologies: Recursive Object Model (ROM) and Expert Comparable Contextual (ECC) Models. Another paper [19] proposed an approach called computer automated use case diagram generator (CAUse) which can generate the use case diagrams from a text written using a special language called ADD. Furthermore, [20]proposed a semi-automated approach which can generate use case diagram from textual user requirements written in the Arabic language. This approach relies on the use of an Arabic NLP tool called MADA+TOKAN to parse the Arabic statements of the textual user requirements in order to obtain the necessary information needed to determine the potential actors and use cases.

3. THE ARCHITECTURE OF PROPOSED APPROACH FOR USE CASE DIAGRAM SYSTEM

Generating use case models in less time and effort is an important requirement [21]. In order to satisfy such requirement, we have to provide some robust solutions. Therefore, we provide a helpful framework which has a sound ability to assist the software analysts and engineers via reading the given system requirements in plain text and then extract the actors, use cases, and relationships which are the basic elements of use case models. Our proposed approach incorporates the capability of automatically generating accurate and complete use case diagrams. We also provide an integrated development environment for user interaction and efficient input of system requirements and output of use case diagrams.

In the context of this research, the user is expected to input the business scenario in the form of paragraphs of text written in English which is related to the business domain. Once the input text is segmented into sentences, the lexical analysis of these sentences is performed in order to generate the lexicons [22] by concatenating the input characters. Then, syntax analysis is performed on word level to recognize the word category [23]. All available lexicons are categorized into nouns, adverbs, pronouns, adjectives, prepositions, articles, conjunctions, etc. Nouns in system requirements can be identified as actors, and verbs can be identified as use cases. A relationship is an association between the actor and its use cases. Relationships can be derived from sentence boundary or based on some exceptions.

Our proposed automated system for use case diagrams generation has the ability to draw use case diagrams after analyzing the text scenario provided by the user. As shown in Figure 1, our proposed system extracts the necessary information and draws the use case diagrams using the following modules: Text Acquisition, Text Segmentation, Text Tokenization, Part of Speech Tagging, Grammatical and spelling error Detection, Stemming, Knowledge Extraction, and finally, The Generation of Use Case Diagrams.

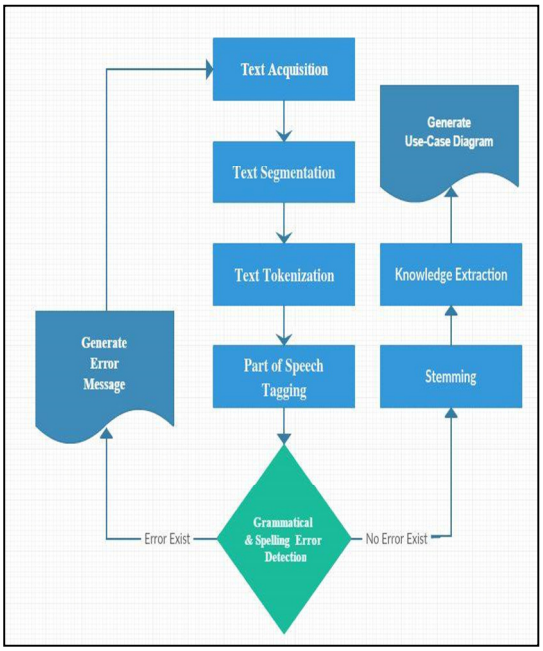

Figure 1: The Architecture of the Proposed Automated System

3.1 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 or import the scenario as a text file.

3.2 Text Segmentation

This module of the proposed framework performs morphological analysis on the natural language text to determine the boundaries of each sentence and split the text into sentences. Usually, each sentence must be terminated with a period. As an example, our segmentation module splits the following text: "User can send invitation to connect and admin can grant invitation. Also, they can send multimedia and receive multimedia." into a set of sentences as presented below.

1) User can send invitation to connect and admin can grant invitation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2) Also, they can send multimedia and receive multimedia

3.3 Text Tokenization

This module is the implementation of the tokenization (lexical) phase where the lexicons, tokens or symbols are generated. As an example, the proposed tokenization module can break up the sentence "I like computer information systems department" into a set of lexicons, tokens or symbols as follows:< I >< like >< computer >< information><systems><department>. This module uses String.Split() method to break up the given text into tokens. It important to mention, that our proposed system can handle the complexity which may result from using compound words that contain commas or periods such as "Dr. Malek". Our tokenization module has to recognize that the period in "Dr. Malek" does not terminate the sentence. Figure 2 shows an example of text tokenization process.

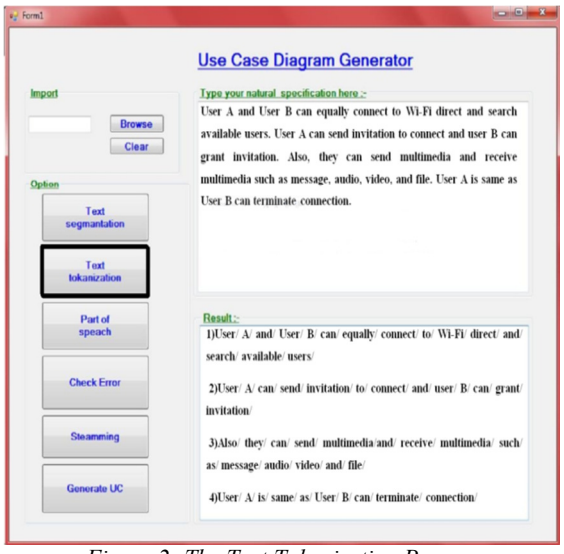

Figure 2: The Text Tokenization Process

3.4 Part of Speech (POS) Tagging

This module of the designed framework categorizes the tokens into various classes (Part Of Speech (POS) Tags) based on its definition and context [3]. 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 2.1[24, 25]. This set of rules is defined based on the Standard English grammatical rules which are called parts of speech conventions. For example, the POS analysis of the sentence "She ran to the school quickly" is as follows: {She/Pronoun, ran/Verb, to/preposition, the/Article, school/Noun, quickly/adverb}.

3.5 Grammatical and Spelling Error Detection

This module is responsible for detecting grammatical and spelling errors in the English text. It informs the user if there are any errors exist via showing error messages on the screen. Our proposed system provides the capability of detecting grammatical errors using probabilistic parsing [26]. Also, we use a dictionary lookup technique [27] to detect the spelling errors in every single token of the input text and to confirm its existence in the dictionary. As an example, the proposed module can detect the spelling error of the word “senid” in the following sentence: "John can **senid** and receive multimedia". As a result, the system will display a message as portrayed in Figure 3.

Figure 3:Spelling Errors Detection

Furthermore, this system ignores numbers, dates, Web and email addresses, and mixed alpha-numeric strings (e.g. “20MHz”).

3.6 Stemming

In our approach, we use the Porter stemming algorithm (or ‘Porter stemmer’) in order to remove the commoner morphological and inflexional endings from words written in the English language[28]. This algorithm utilizes suffix stripping in order to stem both verbs and nouns. This module can change some parts of the verb (phase to phase) to get the root (verb stem). On the other hand, we use this algorithm to eliminate all non-word tokens like punctuations, plural suffixes of nouns (e.g. s, es, and ies) in order to convert actor names from plural to singular. We have to mention, that despite the fact that there are many stemming algorithms exist in the literature, we decided to use this stemmer because it works well with our algorithm and offers excellent results. As an example, the proposed module can stem the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

words “uses” and "purchases" in the sentence "**Customer uses website to make purchases online**" to be "use" and "purchase", respectively.

3.7 Knowledge Extraction

The required use case diagram features are extracted in this module according to the given rules. Basically, this module classifies the noun phrases as actors, and then adds it to the actor's data set, and the verb phrases as use cases and add it to the use cases data set. Then, it determines the relations for each actor with its use cases. As mentioned earlier, out proposed system has the capability of performing a complete of analysis of the given scenarios which contain cohesive devices. Our system can deal with reference, comparing and additive cohesive devices such as: *pronouns, "equally", "similarly", "same as", "also", "plus", "moreover", "furthermore", "besides", "additionally" and "as well as".*

3.7.1 Rule 1: Identify actors

- A common noun may indicate an actor type.
- A proper noun may indicate an actor**.**
- 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).
- Ignore every proper noun such as (Location name, Person name, etc.).

3.7.2 Rule 2: Identify use cases

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

3.7.3 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 $A = \{A_1, A_2, \dots, A_n\}$ 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 A_n.

For convenience, we define T_k to be a set of those certain use-cases, where $k = \{1, \dots, n\}$.

To pave the way of such relationship we need to recall some mathematical structure of the set theory as follows.

- $A_i = A_j$ if and only if for all $v(v \in A_i \leftrightarrow v \in A_j)$.
- $A_i \cup A_j = \{v: v \in A_i \, or \, v \in A_j\}$.
- $A_i \cup T_k = \{v: v \in A_i \, or \, v \in T_k\}$.
- $\bigcup_{m=1}^{i} A_m = \{v: v \in A_m \, for \, some \, m \in \{1, \dots, i\}\}$.
- $\bigcup_{m=i}^{n} A_m = \{v: v \in A_m \, for \, some \, m \in \{1, \dots, n\}\}$.

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

a. Comparing cohesive device such as:

i. Similarly, and equally (e.g. A_i and $A_j$are equally) can be formulated as. $A_i = A_j$.

ii. Same as:

1. “$A_i$is same as A_j ” can be formulated as $A_i = A_j$.
2. “A_i is same as before”, can be formulated as $A_i = A_{i-1}$.
3. “A_i is same as all the above actors”, can formulated as $\bigcup_{m=1}^{i} A_m$.
4. “A_i is same as all the below actors”, can be formulated as: $\bigcup_{m=1}^{n} A_m$.
5. “A_k is same as A_i and A_j”, can be formulated as $A_k = A_i \cup A_j$.

b. Backwards reference cohesive device, Pronoun, can be formulated as $A_n \cup T_k$.

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 $A_n \cup T_k$.

Demonstrating this process, we provide the following simple scenario: " User can send invitation to connect and admin can grant invitation. Also, they can send multimedia and receive multimedia". The knowledge in this scenario can be extracted as follows.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step 1. According to Rule 1:

User and *Admin* indicate Actors. And we donate them by A_1 and A_2 respectively.

Step 2. According to Rule 2:

send invitation, connect, grant invitation, send multimedia and receive multimedia indicate use cases. Consider $v_1 =$ send invitation, $v_2 =$connect, $v_3 =$ grant invitation, $v_4 =$ send multimedia and $v_5 =$ receive multimedia.

Step 3. According Rule 3:

i. **User** (A_1) has relations with a set of use cases: {send invitation, connect}.

ii. **Admin** (A_2) has relation with only one use case: {grant invitation}

iii. T_1 is a set of ***additional*** use cases**:** {send multimedia, receive multimedia}.

In mathematical sense, $A_1 = \{v_1, v_2\}$, $A_2 = \{v_3\}$, and $T_1 = \{v_4, v_5\}$.

Step 4. Finally, according to Part b and c of Rule 3, there exist one additive cohesive device which is *Also*, and one backward cohesive device (pronoun) which is *they*. The latest cohesive device refers to both *User* and *Admin*, Therefore, sending multimedia and receiving multimedia are use cases of both A_1 and A_2 as well. In mathematical sense.

$$A_1 = \{v_1, v_2\} \cup T_1 = \{v_1, v_2, v_4, v_5\}$$

and

$$A_2 = \{v_3\} \cup T_1 = \{v_3, v_4, v_5\}.$$

Equivalently,

i. ***User*** has relations with the following use cases: {send invitation, connect, send multimedia, receive multimedia}.

ii. ***Admin*** has relations with the following use cases: {grant invitation, send multimedia, receive multimedia}.

3.8 Generate Use Case Diagrams

This is the last module which finally uses use case diagram symbols to draw use case diagrams according to the extracted information which are supplied by the previous modules. Figure 4 shows an example of use case scenario, and the corresponding use case diagram is shown in Figure 5. The example scenario is as follows:

Scenario: User **A** and User **B** can equally connect to Wi-Fi direct and search available users. User **A** can send invitation to connect and user **B** can grant invitation. Also, they can send multimedia and receive multimedia such as message, audio, video, and file. User **A** is same as User **B** can terminate connection.

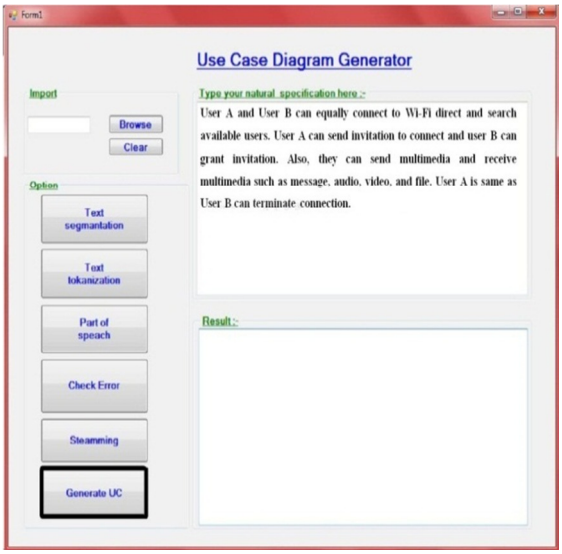

Figure 4: An example of use case scenario

As shown in the above figure, when the user clicks *generate UC* button, the system draws the use case diagram according to the extracted information which are supplied by the previous modules as portrayed in the Figure 5.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

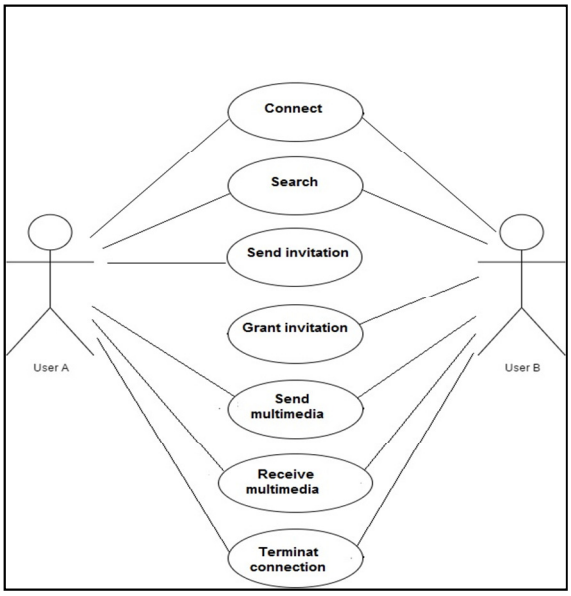

Figure 5: An example of generated Use case diagram

4. EXPERIMENTAL EVALUATION

Experiments were performed using 50 documents contain natural language requirements specification in English with ranges between 50 and 300 words in size. with the number of words per sentence range from 5 to 29 words. Table 1 shows the number of documents used per experiment and the size of the range of word count in these document.

Table 1: Details of case studies

Case Study	Number of Documents	Range Size (word)
1	10	50-100
2	10	101- 150
3	10	151-200
4	10	201-250
5	10	251-300

Two performance measurements, recall and precision, were selected to evaluate our system performance. Both measurements were originally developed for evaluating information retrieval systems [29]. It is important to mention, that these measurements have been recently widely used for evaluating information extraction systems [30].

In the context of this research, we rely on the definitions of recall and precision measurements which are stated in[31],where Recall (R) measurement refers to the amount of the correct information which is returned by our system. This correct information is then compared with the answer keys produced by human analysts. The following formula is used to calculate recall:

$$Recall = \frac{N_{correct}}{N_{key}}$$

Precision(P) shows the accuracy of the system in terms of the percentage of the correct information generated by our system. The following formula is used to calculate precision:

$$Pression = \frac{N_{correct}}{N_{key} + N_{incorrect}}$$

Table 2 reports the scores of the system on the fifty documents based on five categories. Scores for each category are given in one row and the last row shows the overall scores of the system. The overall performance of the system was 96% recall and 84% precision.

Table 2: Evaluation results

Case Study	Recall	Precision
1	100	94
2	96	85
3	96	83
4	95	80
5	94	80
Overall	96%	84%

5. CONCLUSION

The automation of structured IE from natural language text using NLP may still be considered a relatively new field. Few research efforts have attempted to generate use case diagrams automatically via applying NLP techniques to extract knowledge from user requirements written in English. Also, the capability of analyzing and understanding the user requirements text as a semantic unit to infer more linguistic features such as cohesive devices remains a major challenge.

In this research, we have developed a system which can read and perform a complete analysis of the user requirements given in the form of English language text. It can also generate use case diagrams automatically. Our system uses NLP techniques such as tokenization and POS tagging to parse the system specifications based on predefined set of syntactic heuristics rules. Furthermore, our proposed system incorporates the capability of analyzing and understanding the input scenario as a semantic unit to infer important linguistic features such as reference, comparing and additive cohesive

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

devices. Also, we provided an elegant GUI for entering the user requirements scenario and showing the generated diagrams.

6. LIMITATIONS AND FUTURE WORK

Our system can generate only self-contained concrete use cases which constitutes a complete flow of events. It does not have the capability to reuse other existing use cases via *include*, *extend* and *generalize* relationships. Therefore, we aim to provide this capability to reuse the existing use cases in order to reduce the efforts required to define the use cases in the system. On the other hand, we aim to extend our system to have more semantic analysis to infer more linguistic features and to improve the system performance.

REFERNCES:

[1] S. Geetha, and G. A. Mala, "Automatic Relational Schema Extraction from Natural Language Requirements Specification Text," *Middle-East Journal of Scientific Research,* vol. 21, no. 3, pp. 525-532, 2014.

[2] B. Manaris, "Natural language processing: A human-computer interaction perspective," *Advances in Computers,* vol. 47, pp. 1-66, 1998.

[3] E. S. Btoush, and M. M. Hammad, "Generating ER Diagrams from Requirement Specifications Based On Natural Language Processing," *International Journal of Database Theory and Application,* vol. 8, no. 2, pp. 61-70, 2015.

[4] F. Hogenboom, F. Frasincar, and U. Kaymak, "An overview of approaches to extract information from natural language corpora," *Information Foraging Lab*, pp. 69-70, 2010.

[5] I. Jacobson, *Object-oriented software engineering: a use case driven approach*: Pearson Education India, 1993.

[6] M. I. Muhairat, and R. E. Al-Qutaish, "An approach to derive the use case diagrams from an event table," in Proceedings of the 8th WSEAS Int. Conference on Software Engineering, Parallel and Distributed Systems, Cambridge, United Kingdom, 2009.

[7] L. L. Constantine, and L. A. Lockwood, "Structure and style in use cases for user interface design," *Object modeling and user interface design*, M. v. Harmelan, ed., pp. 245-280, Boston: Addison-Wesley, 2001.

[8] W. Boggs, and M. Boggs, *Mastering UML with rational rose 2002*, 1 ed.: Sybex, 2002.

[9] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual*: The Pearson Higher Education, 2004.

[10] I. 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.

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

[12] 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.

[13] 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.

[14] A. 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.

[15] F. Gomez, C. Segami, and C. Delaune, "A system for the semiautomatic generation of ER models from natural language specifications," *Data & Knowledge Engineering,* vol. 29, no. 1, pp. 57-81, 1999.

[16] M. I. Muhairat, R. E. Qutaish, and A. A. Abdelqader, "UML diagrams generator: A new case tool to construct the use-case and class diagrams from an event table," *Journal of Computer Science,* vol. 6, no. 3, pp. 253, 2010.

[17] 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 Languages and Systems, Toulouse, France, 2008, pp. 83-96.

[18] 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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[19] C. Cayaba, J. A. Rodil, and N. R. Lim, "CAUse: Computer Automated Use Case Diagram Generator," pp. 1-4, 2006.

[20] N. Arman, "Using MADA+ TOKAN to Generate Use Case Models from Arabic User Requirements in a Semi-Automated Approach," in The 7th International Conference on Information Technology, Jordan, 2015.

[21] M. M. Kirmani, and A. Wahid, "Revised Use Case Point (Re-UCP) Model for Software Effort Estimation," *International Journal of Advanced Computer Science and Applications,* vol. 6, no. 3, pp. 65-71, 2015.

[22] L. R. Tang, and R. J. Mooney, "Using multiple clause constructors in inductive logic programming for semantic parsing," in European Conference on Machine Learning, Freiburg, Germany, 2001, pp. 466-477.

[23] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases–an introduction," *Natural language engineering,* vol. 1, no. 01, pp. 29-81, 1995.

[24] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An on-line lexical database," *International journal of lexicography,* vol. 3, no. 4, pp. 235-244, 1990.

[25] G. A. Miller, "WordNet2.1," http://wordnet.princeton.edu/, 2006.

[26] J. Wagner, J. Foster, and J. van Genabith, "Detecting grammatical errors using probabilistic parsing," in Workshop on Interfaces of Intelligent Computer-Assisted Language Learning, 2006, pp. 1-25.

[27] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM,* vol. 7, no. 3, pp. 171-176, 1964.

[28] F. Yamout, R. Demachkieh, G. Hamdan, and R. Sabra, "Further Enhancement to the Porter's Stemming Algorithm," *Ulm, September 21, 2004*, pp. 7, 2004.

[29] C. v. Rijsbergen, "Information Retrieval," Butterworths, London, 1979.

[30] R. Grishman, and B. Sundheim, "Message Understanding Conference-6: A Brief History," in International Conference on Computational Linguistics, 1996, pp. 466-471.

[31] H. Harmain, and R. Gaizauskas, "CM-Builder: A natural language-based CASE tool for object-oriented analysis," *Automated Software Engineering,* vol. 10, no. 2, pp. 157-181, 2003.