ISSN 2278-3091
Volume 10, No.1, January - February 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse041012021.pdf
https://doi.org/10.30534/ijatcse/2021/041012021

A Framework for the Generation of Class Diagram from Text Requirements using Natural language Processing

Fatma Alharbia[1],Shadi R .Masadeh[2], Faiz Alshrouf[3]
[1]Software Engineering Department .Isra University, Amman-Jordan,ab1577047@gmail.com
[2] Cyber security Department, Isra University, Amman-Jordan, shadi.almasadeh@iu.edu.jo
[3]Computer Science Department, Isra University, Amman-Jordan, faiz.shrouf@iu.edu.jo

ABSTRACT

The software development procedure begins with identifying the requirement analysis. The process levels of the requirements start from analysing the requirements to sketch the design of the program, which is very critical work for programmers and software engineers. Moreover, many errors will happen during the requirement analysis cycle transferring to other stages, which leads to the high cost of the process more than the initial specified process. The reason behind this is because of the specifications of software requirements created in the natural language. To minimize these errors, we can transfer the software requirements to the computerized form by the UML diagram. To overcome this, a device has been designed, which plans can provide semi-automatized aid for designers to provide UML class version from software program specifications using natural Language Processing techniques. The proposed technique outlines the class diagram in a well-known configuration and additionally facts out the relationship between instructions. In this research, we propose to enhance the procedure of producing the UML diagrams by utilizing the Natural Language, which will help the software development to analyze the software requirements with fewer errors and efficient way. The proposed approach will use the parser analyze and Part of Speech (POS) tagger to analyze the user requirements entered by the user in the English language. Then, extract the verbs and phrases, etc. in the user text. The obtained results showed that the proposed method got better results in comparison with other methods published in the literature. The proposed method gave a better analysis of the given requirements and better diagrams presentation, which can help the software engineers.

Key words: Part of Speech,UML, NLP, SRS and Class Diagram.

1. INTRODUCTION

The software development method is a lengthy and tiresome process. It works with getting the client demands (requirements) because of this, the backbone against which the complete software will be created [2] . This phase includes several agreements and meetings until the final plan of requirements specifications are provided. This documented representation of the specifications named Software Requirement Specification document or the SRS document [10].The developers use the Software Requirement Specification (SRS) for developing the software. It provides information about the classes that should be modern, the characteristics and techniques they should include, and so on. This document is human-universal. But large projects have various pages of the SRS and henceforth virtually infeasible for a human to investigate. Therefore, we need to find a powerful approach to systemize this process [1]

In general, these approaches have been used to find the class diagrams form the requirements specification are categorized into two main approaches: traditional approaches and object-oriented approaches. These approaches are assigned to getting out the purposes and functions of the system only whereas the latter method is involved with the object-oriented standard. It represents classes, characteristics (attributes), and processes (methods). It also determines the connection between classes if they are existent [9], [12].The first-rate manner to achieve basic factors of a category diagram from natural language (NL) is NLPC (natural language Processing for class). Those necessities are provided by the person in a simple statement of English and NLPC (herbal language processing for class) bids natural language processing (NLP) methods to examine enter said. In order to purchase training, member features and records participants, and herbal Language text is semantically examined. With accurate inputs, NLPC finds stages like Pre-processing, Tagging part of Speech (POS, identification of class, characteristic, and function recognition and after that plotting the classes [12].

As mentioned before, obtaining critical data from the requirement specification report can be tedious and occasionally unpractical. It is here; we need the assistant

of natural language processing to solve this problem. We aim to work on this analysis stage in a precise and more intelligent process so that we can preserve time. The main aim of Natural Language Processing (NLP) is to produce and generates software that can check, review, understand, and create languages that help humans apply directly. But, with experience and more complicated, the NLP methods began to produce different results with the extraordinarily unsolved and un-resolved variables constructed as tangible outputs. [12].

The proposed method is used to extract the class diagram from the requirements specification. The proposed method helps analysts by implementing an effective and speedy process to create a class diagram from the class requirements. It maintains excellent cooperation with users by providing a familiar human unified user interface.

2. RELATED WORK

This section shows the relevant work and discusses its main procedures.The central problem that arises in the SDLC is through the requirements analysis and designation. The issues faced through the first stage of the transfer process to other steps, resulting in a high-cost process in comparison with the original method. The human language style can assist developers in determining the software specifications by changing the elements in an electronic design using UML diagrams [12]. Here, this article focused on producing the sequence design and activity design diagrams by utilizing the requirements by presenting them in the natural language [16]. The parser and the PSO tagger methods are used to analyze the user input, provided in the English language, when selecting the procedures and expressions, and others from the text.

In this paper [3], a novel technique is introduced to improve the overall processing of the NL specification, and the proposed approach detects defects in NL specifications. Moreover, they show from the papers of the previously published work, how well the proposed method support even non-software-engineers in editing texts for generating software engineering requirements. The obtained results in this research showed that the proposed method could speed up creating texts with more scattered defects significantly.

This paper represents the NLP mechanism, which endeavors to help the investigation step of software improvement in an object-oriented structure [4]. This NLP procedure is to investigate software demand texts reproduced in English and make a combined dialogue paradigm of the prepared text, described in a Grammatical System. This system is then applied by itself with little or no direct human control to build a UML class diagram such as class design illustrating the terms types specified in the given text and the relations between them and a sequence diagram of the electric model. The specification review defines the user needs for a specific purpose.

Software specifications are an essential action of the software process, as failures at this step necessarily direct to difficulties later on in system configuration and implementation. The demands are formulated in NL, with the potential for uncertainty, inconsistency or mistake, or naturally a failure of programmers to deal with a large volume of knowledge. This paper proposed a novel technique for the natural language processing of requirement descriptions of the universal natural language and their automatic changing to the object-oriented interpretation system [5].

To change the human language into use case and class diagrams, a new method is introduced to perform these operations; two states have been produced. Firstly, the recursive object design, which changes from human language to graphic style (writing). Also, convert the recursive object design to the UML style. As well, to change the UML designs to the human language, introduced a new system using a grammatical structure to convert the class design diagram into a common linguistic or human language. Then, the linguistic form converts into the human language (text).

The class diagram's automatic generator is introduced as an approach that aggregates the human language (NL) methods' analytical and design verification characteristics. Various models were determined to generate the class diagram. When the ideas are created, the XML metadata file was generated and transmitted with a computer-aided tool to generate the UML design diagram [6], [7].

A new algorithm is introduced to facilitate the extraction rulein generating UML design diagrams from the human language produced by various people [17]. In terms of improving the UML's regular syntax and identifies the legal issue in the initial stage will decrease the time as well as the cost. This work also give a symmetrical syntax model for UML diagram and use case diagram [6].

There is usually a means of existing domain-specific, NL data possible to improve lead developers of object-oriented methods. This information is frequently manageable to NLP in order to get valuable configuration information. However, from a review of the field, they contend that popular methods have not been capable of extracting all the semantic and design specification that is present in such data. For example, they notice that there is a lack of sufficient system descriptions; they discover doubt for scholars to use fusion solutions - where users establish and detailed automated reports, and they recommend that there is work required to define a comprehensive review of potential associations between arrangement elements [8]. In partial pursuance of such a declaration, this paper discusses the proposed algorithmic for treating NL into UML diagrams with supplemental user study quickly.

3. THE PROPOSED METHODOLOGY

In this section, we present the proposed method that is transforming the problem of the users' requirements into the UML class diagram to make these processes easy for

the users, which is illustrated in Figure 1. The requirement specification is an element created by the user, which describes the system employing simple natural language.

3.1 Main steps of the proposed method

The proposed method decreases the complexity by distributing applications in smaller modules as follows:

3.1.1 Tokenizer and Sentence selection

In the first step, the tokenizer will determine the different stop words in the sentence, such as a, the, in, you, me, and, no, etc. various approach has been proposed to identify the stop words. However, these approaches are not useful in obtaining knowledge. In this research, we will collect the most common stops words, and then a matching system will extract these words from the text [9]. Moreover, we will use the stemming algorithm to get the root for each word for more examination by removing the affixes and suffixes.

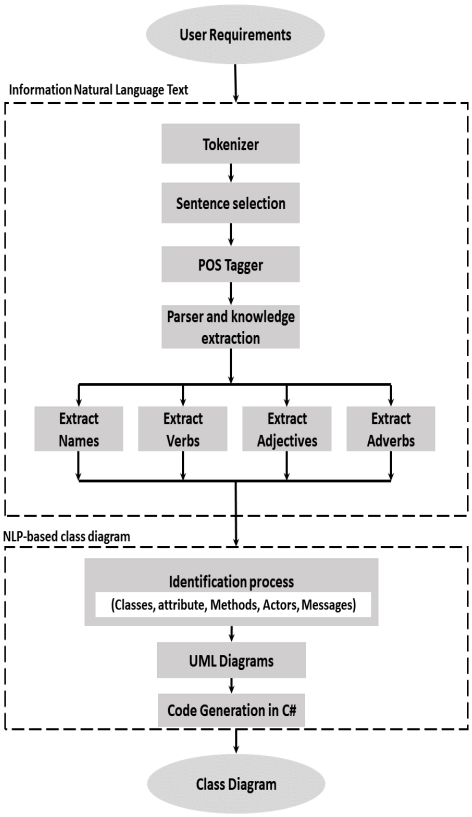

Figure 1: The proposed method

3.1.2 Part of Speech tagger and knowledge

Here, the method will parse the output to find out the nouns, verbs, adjectives, adverbs. We suggest using the Open NLP POS Tagger to do this analysis. "The other Library System is used by Informatics students."

3.1.3 NLP-based class diagram

In this part, we introduce a new approach to the computerized generation of class diagrams. The input of this process will be a regular text and structural text, while the output will provide the class diagram with its class and attributes, and relationship. The inter-class connections can arrange toward association, aggregation, generalization. The association connections can organize into three fundamental relations: one-to-one, one-to-many, and many-to-many [10]

The proposed approach implements the NPL to explore classes of case, that is, we distinguish main classes of the field as an origin point, for which we usually are highly positive, and besides find classes that are linked with the known ones. The strength of this process is that it defines types and connections in one step. Before going into this stage, we perform the following fundamental elements of the method and how they are ordered in this research, which will support in defining its rationale.

Class Identification

Both the POS tagger and the sentence parser provide fundamental candidates. On the other hand, the semantic network approach and word sense explicit implemented to get the original candidates.

Relationship Identification

To extract the relationship, it uses a linkage range to define all concept pairs with robust grammatical relations within the sentence. We allow several weights for all concept pair to show how robust the relationship is according to the elements the notions serve as in the sentence.

Attribute Identification

The attribute Identification system describes the class attributes. Then, two notions are obtained to be strongly correlated together; we require to decide if the assumptions are related to the class attribute or the class.

Naming Relationship

Naming Relationship implements the patterns semantic, which helps to obtain an aggregation association and an aggregation association. This model uses several relationship recognition approaches to classify the relationship type like one-to-one, one-to-many, and many-to-many.

The proposed approach uses the repetition procedure, to determine the classes and the attributes. In all steps, it picks one notion from the detailed set with the highest relationship score and associated with the defined classes. The relationship score is an indicator of how the new concept semantically joins with all other ideas of the set of nominee ideas. If the correspondence score is smaller

than the assemblage threshold, the flow ends. Differently, it will include the concept of the selection of classes or the set of attributes toward the number of features the idea owns.

Lastly, utilizing the previous outcome of the analysis, UML Class diagrams are created, and the code template will be generated by C#. The method manipulates the graphical design of the UML diagrams and permits the user to rename classes, add, delete, and relationships in the produced diagram. As a member of UI, concept management UI is a primary interface that allows the user to view, add, modify, and organize concepts and relationships. Users can add new ideas and change the concept type. The concept management method gives the user the resilience to drive the processing as the user wants.

4. EXPERIMENTS AND RESULTS

4.1 Case study

In this section, an implementation of the conceptual model production procedure is presented with moderate results. The given requirements specification and details are obtained from the ATM dilemma statement [12]. The first part is manually changed to eliminate kinds of references, pronouns, and wh-pronouns (who, whose, whom, whatever, whichever, etc.). It is further adjusted to assure regular usage of a term for a regular function, i.e., one function per conversation.

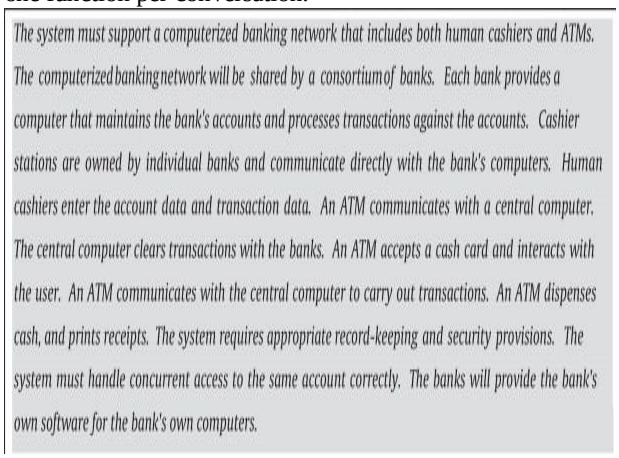
The system must support a computerized banking network that includes both human cashiers and ATMs. The computerized banking network will be shared by a consortium of banks. Each bank provides a computer that maintains the bank's accounts and processes transactions against the accounts. Cashier stations are owned by individual banks and communicate directly with the bank's computers. Human cashiers enter the account data and transaction data. An ATM communicates with a central computer. The central computer clears transactions with the banks. An ATM accepts a cash card and interacts with the user. An ATM communicates with the central computer to carry out transactions. An ATM dispenses cash, and prints receipts. The system requires appropriate record-keeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide the bank's own software for the bank's own computers.

Figure 2: Modified ATM statement

4.2 Performance Evaluation

The effectiveness assessment of this method was a difficult because there is no explanation of an 'accurate' conceptual model. The theoretical models that are typically performed were determined to include substantially added experience by the investigator. Specific knowledge in the conceptual model is similar to classes or\and relations, that are not specified in the text of the demand. For example, in the ATM case, Bank owns Bank Computer describes a particular opinion on a connection. At the same time, the class Remote Transaction is a constitutional theory that a specific class endures, although it is not declared in the requirements text.

4.3 Evaluation criteria

In this part, evaluation criteria are utilized to analyze the proposed automatically created conceptual model with a conventional published design or a human design model in the literature [11]. The used criteria are:

Recall percentage means the capacity of the computerization to create all classes, as shown in Equation (1).

$$Recall = \frac{Ncorrect}{Ncorrect+Nmissing} \quad (1)$$

Where N correct means the amount of true classes recognized; N missing means the number of classes selected by the human expert and ignored by the proposed conceptual method.

Precision means the accuracy or the relevance of the categories recognized in the proposed conceptual model, as shown in Equation (2).

$$Precision = \frac{Ncorrect}{Ncorrect+Nincorrect} \quad (2)$$

Where N incorrect means the amount of correct levels classified as wrong;

The over-specification rate (Over_SR) means the number of useless, but the right classes that the computerization process adds in the created conceptual model by the proposed, as shown in Equation (3).

$$Over\_SR = \frac{extra\ (valid)}{Ncorrect+Nmissing} \quad (3)$$

Where extra(valid) is the amount of correct additional classes regained.

Even though the model is not intended for assuming specific knowledge, we assess it to match it with human-created rules. To determine this, we propose another variable, N implicit, which means the number of classes that are calculated that are true and correct but not pronounced in the requirements text. The equations utilized to assign certain knowledge are as follows:

$$Recall\_implicit = \frac{Ncorrect}{Ncorrect+Nmissing+Nimplicit} \quad (4)$$

$$Precision\_implicit = \frac{Ncorrect}{Ncorrect+Nincorrect} \quad (5)$$

$$Over\_SR\_implicit = \frac{Nextra\ (valid)}{Ncorrect+Nmissing+Nimplicit} \quad (6)$$

While exceptional reference rates are possible for evaluating that do not constitute N implicit, published objective values do not survive for any of the criteria. Therefore, for the goal of the testing process, we set advantageous positions to assess the achievement. Recall and Precision should be as powerful as feasible (high) to correctly express the objective model. Over_SR should be low to evade attaching to various extra features.

4.4 Results of conceptual class modelling versus other comparative standard models

Table 1 presents the obtained results of the achievement measures versus various case studies. These cases were employed by several researchers to illustrate the

production of class diagrams, and similar results are obtained in the corresponding references. It is observed that these conventional designs were not developed for the requirement investigation, nor generated automatically, and therefore include a component of human knowledge, which our implementation requires

Table 1: Evaluation results

Case study / Methods		ATM (Rumbaugh et al., 1991)	Course registration (Kurt, 1995)	EFP (Swithinbank et al.)
Without implicit information or assumptions	Recall (%)	100	100	100
	Precision (%)	91.67	81.82	94.44
	Over_SR (%)	9.09	22.22	41.18
With implicit information/assumptions	Recall (%)	91.67	100	85
	Precision (%)	91.67	81.82	94.44
	Over_SR (%)	8.33	22.22	35
Proposed Method	Recall (%)	100	100	100
	Precision (%)	92.54	85.32	95
	Over_SR (%)	8.02	20.15	33

The drop-in precision illustrates the importance of human judgment to add related classes. Our approach combined some classes that may be acknowledged wrong. The recall measure is very high due to the all used classes are regularly recognized, candidate classes. Nevertheless, the over-specification weight should remain at a low rate, and it gives high values, although high over-specification costs cause visual clutter in the produced models.

The proposed method got better results in all studied cases according to the used evaluation measure, which means that the proposed method able to generate the classes accurately than other comparative methods. The recall values were similar for all comparative methods. However, the precision values were better obtained by the proposed method. As well as the Over_SR values were better achieved by the proposed method compared to other methods. We concluded from the mentioned results that the proposed method got better results overall.

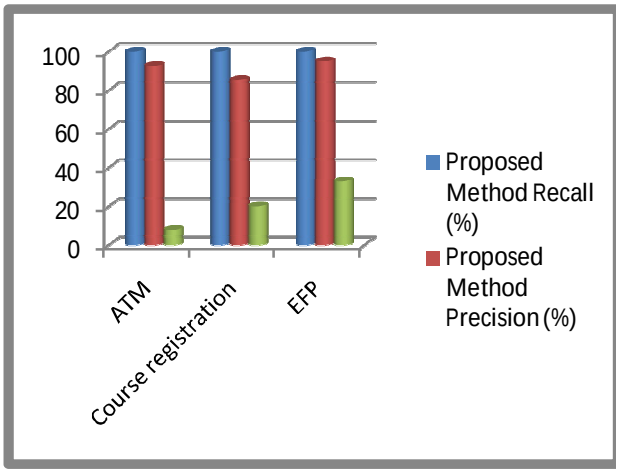

Figure 3: Statistical analysis results

Because of the need for a standard text and specialty models, the experiment was carried on individual subjects. The obtained results of the final CASE tools lab test were used for reference. Figure 3 presents the final statistical results for the results.

4.5 Performance evaluation

Recall, precision, and over-specification are utilized to assess the effectiveness of the proposed method for the connections between the classes, as presented in Table 6. The obtained results versus human subjects additionally give comparable outcomes for relations. The relationships that are associations show that the over-SR is very high when connected to standard results. Generally, the proposed method got better results.The results of the proposed method are comparable for relations. The connections that are associations confirm that the over-SR is very powerful when compared to standard outcomes.

Table 2: Performance evaluation

Case study / Methods		ATM (Rumbaugh et al., 1991)	Course registration (Kurt, 1995)	EFP (Swithinbank et al.)
Without implicit information or assumptions				
Overall	Recall (%)	83.33	100	93
	Precision (%)	100	80	88
	Over_SR (%)	100	100	93
Association	Recall (%)	80	100	86
	Precision (%)	100	100	75
	Over_SR (%)	120	0	171
Composition	Recall (%)	100	100	100
	Precision (%)	100	75	100
	Over_SR (%)	0	133	100
With implicit information/assumptions				
Overall	Recall (%)	50	57	61
	Precision (%)	100	80	88
	Over_SR (%)	60	57	61
Generalization	Recall (%)	44	100	100
	Precision (%)	100	100	100
	Over_SR (%)	67	0	100
Association	Recall (%)	100	50	50
	Precision (%)	100	75	75
	Over_SR (%)	0	67	100
Proposed Method				
Overall	Recall (%)	85	100	95
	Precision (%)	100	80	100
	Over_SR (%)	100	100	94
Generalization	Recall (%)	49	100	100
	Precision (%)	100	100	100
	Over_SR (%)	69	0	100
Association	Recall (%)	100	60	55
	Precision (%)	100	78	80
	Over_SR (%)	0	68	100

5. CONCLUSION AND FUTURE WORK

Most of the problems that happened in the software development stages occur during the requirements analysis and specification. Generating the UML diagram from these specifications from the natural language is profoundly challenging work. This research presents a technique to improve the procedure of constructing the UML diagrams by employing the Natural Language,

which will help the software development to analyze the software requirements with fewer errors and effective way. The proposed approach uses the parser analyses and Part of speech (POS) tagger to analyze the user requirements entered by the user in the English language.

The obtained results showed that the proposed method got better results in comparison with other methods published in the literature. The proposed method gave a better analysis of the given requirements and better diagrams presentation, which can help the software engineers. As result, the recall measure is very high due to the all used classes are regularly recognized, candidate classes. Nevertheless, the over-specification weight should remain at a low rate, and it gives high values, although high over-specification costs cause visual clutter in the produced models. The proposed method got better results in all studied cases according to the used evaluation measure, which means that the proposed method able to generate the classes accurately than other comparative methods. The recall values were similar for all comparative methods. However, the precision values were better obtained by the proposed method. As well as the Over_SR values were better achieved by the proposed method compared to other methods. We concluded from the mentioned results that the proposed method got better results overall.

From this research, we have classified two groups of future work directions that we find interesting. The first group is to create other views to catch the requirements specification. The second group is to get more use of the syntactic structure. So far, we have seen the inactive construction of the class diagram. A different aspect deserves more investigation, which is the dynamic action of the software requirements. This can be accomplished by changing the language code into textual observations, using the results from Translatable Unified Modeling Language and Model-Driven Architecture. This will later be mixed with NL details of the difficulty machines because they perform a vital function in the rule of objects. There are various ways to gain more value from the syntactic framework. Improve their created texts with the LATEX tool, something that can be utilized to encourage the motives for interest in the structure of the document. We also need to take more advantage of the syntactic framework's potential for some particular languages to give the same theoretical syntax. Being capable of creating a difference of methods from internal system designations would suggest that the rules can be obtained and assessed by those stakeholders that are not self-sufficient in utilizing English. One of the different languages can be the NL for reporting any system requirements, and then the syntactic framework can be utilized to create NL descriptions, legal requirements, and change among the two. Both groups of activity will, in the end, need an also critical evaluation, both to achieve the wanted construction and content of the books but additionally to examine in which area they can follow the original Computational Independent Model.

ACKNOWLEDGEMENT

The authors owe thanks to Scientific Research Deanship at Isra University for facilitating procedures of conducting this research and its financial support for this research.

REFERENCES

[1] Amdouni, S., Karaa, W. B. A., & Bouabid, S. J. a. p. a. (2011). Semantic annotation of requirements for automatic UML class diagram generation.

[2] Bajwa, I. S., & Choudhary, M. A. (2006). Naturallanguage processing based automated system for uml diagrams generation. Paper presented at the The 18th Saudi National Computer Conf. on computer science (NCC18). Riyadh, Saudi Arabia: The Saudi Computer Society (SCS).

[3] Ben Abdessalem Karaa, W., Ben Azzouz, Z., Singh, A., Dey, N., S. Ashour, A., & Ben Ghazala, H. (2016). Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements. Software: Practice and Experience, 46(11), 1443-1458.

[4] Bhagat, S., Kapadni, P., Kapadnis, N., Patil, D., Baheti, M. J. I. J. o. E., Communication, Science, S. C., & Engineering. (2012). Class Diagram Extraction Using NLP. 2, 125.

[5] Burden, H., & Heldal, R. (2011). Natural language generation from class diagrams. Paper presented at the Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation.

[6] Chanda, J., Kanjilal, A., Sengupta, S., & Bhattacharya, S. (2009). Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach. Paper presented at the 2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS).

[7] Friedrich, F., Mendling, J., & Puhlmann, F. (2011). Process model generation from natural language text. Paper presented at the International Conference on Advanced Information Systems Engineering.

[8] Hamza, Z. A., & Hammad, M. (2019). Generating UML Use Case Models from Software Requirements Using Natural Language Processing. Paper presented at the 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO).

[9] Jaafar, Y., & Bouzoubaa, K. (2018). A survey and comparative study of Arabic NLP architectures Intelligent Natural Language Processing: Trends and Applications (pp. 585-610): Springer.

[10] Kar, S. K. (2014). Generation of UML class diagram from software requirement specification using natural language processing.

[12] Landhäußer, M., Körner, S. J., & Tichy, W. F. (2014a). From requirements to UML models and back: how automatic processing of text can support requirements engineering. Software Quality Journal, 22(1), 121-149.

[13] More, P., & Phalnikar, R. J. I. J. o. A. I. S., Foundation of Computer Science. (2012). Generating UML diagrams from natural language specifications. 1(8), 19-23.

[14] Platt, R., & Thompson, N. (2019). The Past, Present, and Future of UML Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics (pp. 1452-1460): IGI Global.

[16] Shinde, S. K., Bhojane, V., & Mahajan, P. (2012a). Nlp based object oriented analysis and design from requirement specification. International Journal of Computer Applications, 47(21).

[17] Tazin, A. (2017). UML Class Diagram Composition Using Software Requirements Specifications. Paper presented at the MODELS (Satellite Events).