Towards an Automatic Generation of UML Class Diagrams from Textual Requirements using Case-based Reasoning Approach

2022 4th International Conference on Applied Automation and Industrial Diagnostics (ICAAID) | 978-1-7281-8939-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAAID51067.2022.9799502

Omer Salih Dawood Omer
Department of Computer Science
College of Arts and Science
Prince Sattam Bin Abdulaziz University
Wadi Aldawasir, KSA
o.dawood@psau.edu.sa , omercomail@gmail.com

Safaa Eltyeb
College of Computer Science and Information Technology
Sudan University of Science and Technology
khartoum,Sudan
safaa-82@hotmail.com

Abstract*—Nowadays automatic information extraction plays a major roles in software development life cycle. It allows producing automated tools that help humans in routine tasks. The automatic extraction of Unified Modeling Language (UML) class diagrams from textual requirements can help in reducing the design time and cost. This paper proposes a novel case-based reasoning model to facilitate the process of generating the UML class diagrams from textual requirements. The result of this work can help on minimizing the manual work, and can be employed to extract other type of information from documents that we cope with during software development life cycle.

***Keywords*—component, formatting, style, styling, insert** *(key words)*

I. INTRODUCTION

The largest part of software specifications is the software requirements which are written in natural language (NL) [1]. In the design phase, the requirements are transformed into a system design document that precisely describes the design of the system and that can be used as an input to system development phase. Well design practices and automated tools have allowed industry to produce significant products [2].The automatic generation of Unified Modeling Language (UML) class diagrams from requirements can help in reducing design time and cost. Since requirements' documents are written in NL, information extraction and text mining techniques can be useful for this automaton.

This paper proposes a novel method for generating UML class diagram by extracting the candidate class diagram elements (i.e. classes, attributes, relationships, and methods) from NL requirements using a case-based reasoning (CBR) method in order to facilitate and speed-up the software design process. Our idea is to mine the requirements text by an automatic learnt patterns stored in a case base. To avoid the difficulty of building a large corpus to train models to apply a machine learning (ML) models and avoiding the cost and specificity of manually designed patterns, the proposed model will utilize the CBR approach to exploit the CBR ability of sustained learning. The rest of this paper is organized as following; Section 2 gives an overview of applying text mining in requirement engineering process. Section 3 introduces some studies support the automated transformation of textual requirements into analysis and design models using text mining techniques and highlights the motivations for the proposed method. Section 4 provides a brief idea about CBR approach since it is employed in this work. Section 5 presents a description for the proposed model and finally section 6 highlights the expected contribution of this work.

II. APPLYING TEXT MINING TO SOFTWARE REQUIREMENTS

There are various usages of text mining in software development life cycle (SDLC), specially in software requirement documents. For instance, [3] referred for using of mining for many activities such as tracing of requirements and retrieval of components from a repository, so the analyst was involved to assist in decision making when data mining conducted during software development. [4] applied the text mining to a large-scale of 2,061 policy documents in a purpose of supporting the requirements analysis process. These documents contain important information and used by requirements engineers as sources of requirements. [5] inferred the hierarchical structure of an attribute-based access control (ABAC) model from natural language access control policies (NLACPs) or authorization requirements. Access control policies, like other security and functional requirements, always are written in natural language in specifications documents. They extracted a set of authorization attributes based on the resulting structure using convolutional neural network ML techniques to classify the candidate instances.

Due to current evolution of ML algorithms, there are many researches utilized ML in mining the software documents. For example, in [6] a framework of machine-learned components was proposed to manage the uncertainty in requirements. [7] proposed a deep learning-based method to automate the process of filtering out quality requirements statements from software requirements specification and classifying them into quality attributes. [2] developed a ML-based framework to automate the task of extracting functional requirements from textual documents. [8] surveyed the ML techniques that used to automate the process of requirements prioritization in an efficient way with less human effort. As well [9] presented a survey in the automation of requirements classifications process and [10] in identification and classification of non-functional requirements in requirements documents.

III. RELATED WORK

There are some studies support the automated transformation of textual requirements into analysis and design models using text mining techniques in order to reduce the time and effort of creation these models. For instance, In [11] proposed a method to transform NL specification into requirements model. Text mining techniques (text analysis, information extraction, and ML) were used, and utilized object system models (OSMs). However, there is not enough details about the used algorithm and the expected results. ML can be used to automate grading of UML class diagrams, in [12] they developed approach that can be used to assist in diagram grading, firstly diagram manually graded by teachers or experts, then based on grading they built ML model that can be used to predict the correct result, however they obtained low accuracy for the model, so the model can't be used as a reference for class diagram grading. [13] generated UML use case diagrams form requirements using ML technique, their idea is concluded in; after the requirements are entered they are analyzed and converted to use cases and saved into a database, so by frequently using data can be trained. When entering new requirements, system will predict it if its stored in database, so this method is suitable for one type of system, because other systems use other different requirements. However, few studies have addressed the extraction of the UML class diagrams' elements (i.e., classes, attributes, methods, and relationships) owing to the lack of an open access gold standard corpora for evaluation purposes. Yet, most previous studies extracted these elements relied on ML models or relied on manually designed patterns. As it is known, machine learning models need a large corpus to train models. At the same time, extracting the elements of UML class diagrams using pattern-based methods only has many shortcomings. Specially the patterns which are built manually, whether are based on keywords or syntactic structure, because they require labour intensive effort and cost. Also, the designed patterns can be very general or very specific. So, using very specific pattern, achieves high precision and relatively low recall and quite the opposite is true. This in addition to, if the knowledge grows; modifying these patterns can become very complex. Another type of patterns are automatic learned which can use syntactic structure between prior known elements to learn the patterns; and then used the learned patterns to extract new potential elements. In this work we propose an automatic learned pattern-based model for extraction the UML class diagram elements. In our proposed method, we try to learn patterns automatically and build patterns from the tokens and syntactic structure information related to elements in order to balance between generalizability and specificity. Our proposed method aims to employ a large number of learnt patterns that are more flexible than specific patterns. Therefore, this combination has a tradeoff between the generalizability and strictness of patterns. At the same time, the proposed model will benefit from CBR ability of sustained learning. Thus, the model can accept new cases without the need to re-train the model like ML-based models. In addition, the CBR classifier has the potential to be further enhanced to gradually improve its performance.

IV. THE CASE-BASED REASONING (CBR) APPROACH

This section presents an overview of the semi-supervised learning CBR approach [14] as it will be used in our proposed work. CBR " *is the usual name given to problem solving methods which make use of specific past experiences. It is a form of problem solving by analogy in which a new problem is solved by recognizing its similarity to a specific known problem, then transferring the solution of the known problem to the new one* " [15]. Using its memory of past experiences, CBR could be applied to solve various real world problems such as course time tabling and solving legal cases.

The common process of CBR can be described by the CBR cycle; as shown in Fig.1. It consists of four major phases, namely Retrieve, Reuse, Revise, and Retain. They are linked to a central repository called the case base. The case base is like a storage place where past cases with known solutions are stored. As can be observed, CBR is based on the principle that similar cases will have the similar solution. For instance, when a new case is entered into the CBR cycle, the following steps will be followed to solve it:

a- Retrieve the nearly all similar cases from the case base.

b- Reuse the case or solutions from the retrieved cases.

c- Revise the solution for the new case if needed.

d- Retain by adapting the revised new cases into the case base.

When a new case (i.e., a problem) is received, the CBR model will get back the similar case(s) from the case base where solved cases are stored and the solution from the retrieved cases will be reused for the new case. If there are not similar cases in the case base, the solution for the new case will be revised and retained into the case base as a novel solved case.

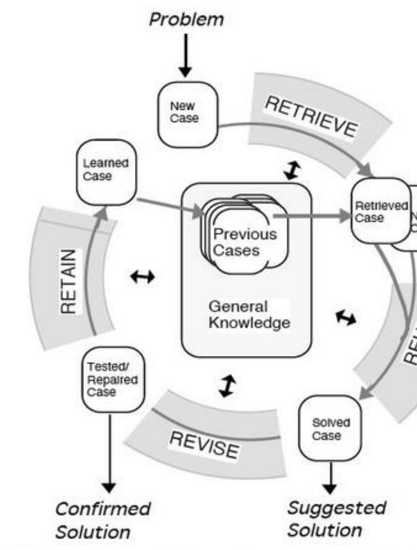

Fig. 1. The CBR Phases[14].

CBR generally requires much less knowledge gaining and it relies on the collection of past experiences [16]. An additional attractive feature of this method is that it is able to adapt new cases to its case base, without need to retrain the data which is essential for most supervised machine learning techniques [17].

V. THE PROPOSED MODEL

This section presents a description for the steps that will be followed to implement the proposed model. These steps are divided into two phases as displayed in Fig.2.

Phase 1: Text Pre-Processing: In this phase the textual requirements are processed to obtain linguistic and syntactic structure information of sentences that can be used in later steps. Text pre-processing phase includes the following steps:

Removing stop-words

Stop word is a word which has less significant such as: the, is, at, which, on, and etc. These words are removed to give more focus in the more significant words. In case of need to some of these words, this step can be can be ignored. For example in generalization relationship we should keep (is a) between classes, and don't remove it.

Text Tokenization

Aims to detect sentence and boundaries of meaningful elements called tokens like words, phrases, and etc.

Part-of-Speech (POS) Tagging

POS tagging concerns with marking up words in a text to a corresponding part of a speech tag, based on its context and definition. The POS tagging applied to tokens to markup noun, verb, etc. It is used for many purposes, here, after POS tagging step, a syntactic-based patterns are applied into text to extract the candidate class elements and used in the next phase.

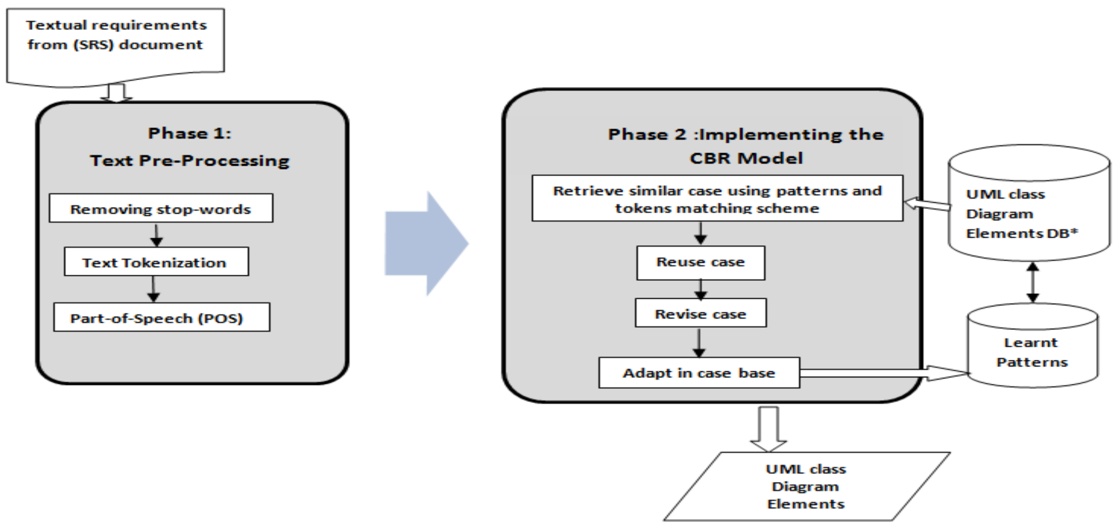
Textual requirements from (SRS) document
Phase 1: Text Pre-Processing
Removing stop-words
Text Tokenization
Part-of-Speech (POS)
Phase 2 :Implementing the CBR Model
Retrieve similar case using patterns and tokens matching scheme
Reuse case
Revise case
Adapt in case base
UML class Diagram Elements DB*
Learnt Patterns
UML class Diagram Elements

Fig. 2. The General Framework of the Proposed Model.

Phase 2: Implementing the CBR Model : There are two major processes for implementing the proposed CBR model; that are:

a. Build the case base or carry out the case base representation; which based on syntactic information (i.e., automatic learned patterns) and lexical information (i.e., saved UML class diagram elements as shown in table 1) for extraction process.

b. Make use of the CBR model which comprise the main sub processes of each CBR model: retrieve reuse, revise, and retain.

After applying all previous phases the candidate elements of class diagram can be extracted. Fig.3 displays an example of extraction schemes. The obtained results will evaluated using the standard information extraction evaluation measures Recall, Precision and F-measure.

VI. CONCLUSION

Besides a review on the application of text mining techniques on software requirements, we give a description for our proposed CBR-based method for extracting the elements of UML class diagram. . In this work we propose an automatic learned pattern-based model for extraction the UML class diagram elements. The proposed model build patterns from the tokens and syntactic structure information of text, then store it in a case-base to be learned automatically and used in extracting the UML class diagrams. The result of this work can help in reducing design time and cost. Furthermore, the mining method developed in this work can be employed to infer another type of information from many documents cope with during the SDLC.

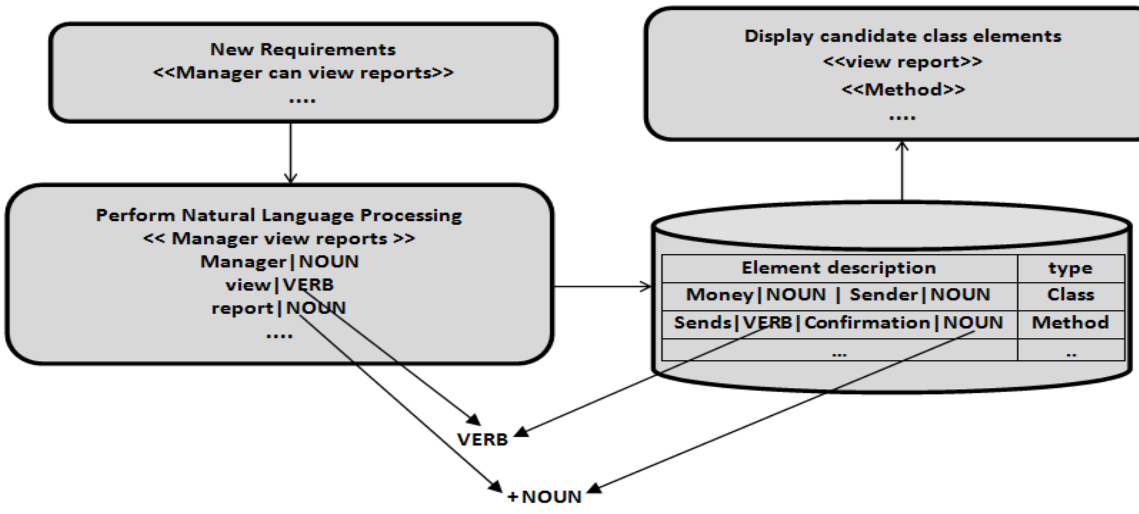

Fig. 3. An example of extraction scheme for UML class diagram's elements from requirements

TABLE I. *SAMPLE OF SAVED DATA CONCERNS THE UML CLASS DIAGRAM .

Element ID	Element Description	Element Type
1.	login system	Method
2.	Mobile operator	Class
3.	mobile number	Attribute
4.	Money Receiver	Class
5.	Bank	Class
6.	money amount	Attribute
7.	send SMS	Method
8.	Receive SMS message	Method
9.	providing mobile number	Method
.....	....	.....

REFERENCES

[1] B. Sateli, E. Angius, and S. S. Rajivelu, "Can Text Mining Assistants Help to Improve Requirements Specifications ," 2012.

[2] H. Akay and S. G. Kim, "Extracting functional requirements from design documentation using machine learning," Procedia CIRP, vol. 100, pp. 31–36, 2021, doi: 10.1016/j.procir.2021.05.005.

[3] J. H. Hayes, A. Dekhtyar, and S. Sundaram, "Text mining for software engineering: How analyst feedback impacts final results," in Proceedings of the 2005 International Workshop on Mining Software Repositories, MSR 2005, 2005, pp. 1–5.

[4] H. Alrumaih, A. Mirza, and H. Alsalamah, "Toward Automated Software Requirements Classification," in 21st Saudi Computer Society National Computer Conference, NCC 2018, 2018, pp. 4–13, doi: 10.1109/NCG.2018.8593012.

[5] M. Alohaly, H. Takabi, and E. Blanco, "Automated extraction of attributes from natural language attribute-based access control (ABAC) Policies," Cybersecurity, vol. 2, no. 1, 2019, doi: 10.1186/s42400-018-0019-2.

[6] M. Chechik, "Uncertain Requirements , Assurance and," 2019 IEEE 27th Int. Requir. Eng. Conf., pp. 2–3, 2019, doi: 10.1109/RE.2019.00010.

[7] T. Tamai and T. Anzai, "Quality Requirements Analysis with Machine Learning REQUIREMENTS DESCRIBED," no. Enase 2018, pp. 241–248, 2019, doi: 10.5220/0006694502410248.

[8] H. Alrumaih, A. Mirza, and H. Alsalamah, "Toward Automated Software Requirements Classification," in 21st Saudi Computer Society National Computer Conference, NCC 2018, 2018, pp. 51–55, doi: 10.1109/NCG.2018.8593012.

[9] H. Alrumaih, A. Mirza, and H. Alsalamah, "Toward Automated Software Requirements Classification," 2018 21st Saudi Comput. Soc. Natl. Comput. Conf., pp. 1–6, 2018.

[10] M. Binkhonain and L. Zhao, "Expert Systems with Applications : X A review of machine learning algorithms for identification and classification of non-functional requirements," vol. 1, 2019, doi: 10.1016/j.eswax.2019.100001.

[11] E. V. Chioaşcă, "Using machine learning to enhance automated requirements model transformation," in Proceedings - International Conference on Software Engineering, 2012, pp. 1487–1490, doi: 10.1109/ICSE.2012.6227055.

[12] D. R. Stikkolorum, P. Van Der Putten, C. Sperandio, and M. R. V. Chaudron, "Towards automated grading of UML class diagrams with machine learning," in CEUR Workshop Proceedings, 2019, vol. 2491, pp. 1–13.

[13] M. S. Osman, N. Z. Alabwaini, T. B. Jaber, and T. Alrawashdeh, "Generate use case from the requirements written in a natural language using machine learning," 2019 IEEE Jordan Int. Jt. Conf. Electr. Eng. Inf. Technol. JEEIT 2019 - Proc., pp. 748–751, 2019, doi: 10.1109/JEEIT.2019.8717428.

[14] A. Agnar and E. Plaza, "Case-Based reasoning: Foundational issues, methodological variations, and system approaches," AI Commun., vol. 7, no. 1, pp. 39–59, 1994, doi: 10.3233/AIC-1994-7104.

[15] R. Bareiss, "Exemplar based knowledge acquisition: a unified approach to concept representati on, classification, and learning." Academic Press Professional, Inc., 1989.

[16] S. Riedel, L. Yao, and A. McCallum, "Modeling Relations and Their Mentions without Labeled Text," in Machine Learning and Knowledge Discovery in Databases, 2010, pp. 148–163.

[17] R. Malhotra and A. Jain, "Software Effort Prediction using Statistical and Machine Learning Methods," Int. J. Adv. Comput. Sci. Appl., vol. 2, no. 1, pp. 145–152, 2011, doi: 10.14569/ijacsa.2011.020122.