NLP based Knowledge Extraction for Software Modeling: A Review

Mrs. Rekhanjali Sahoo
Computer science & Engineering,
Einstein Academy of Technology & Management
Bhubaneswar, Odisha

Mr. Manjog Padhy
Computer science & Engineering,
Einstein Academy of Technology & Management
Bhubaneswar, Odisha

***Abstract*- This paper presents a natural language processing based automated system for natural language text to object oriented modeling. The user requirements and generating code in multi-languages. To find out remarkable and actionable sequence from natural language documents authored by software developers, software artifacts written in natural language are different from other textual documents due to the technical language used. A latest model is planned for analyzing and extracting the virtual and necessary information from the given software requirement notes by the user. User must have to write the necessities in simple English in a small number of modules and the model the System incorporates NLP methods to analyze the writing. Initially the natural language text is semantically analyzed to find out classes, objects and their respective attributes, methods and associations. Then we have to design UML diagram form the preceding sequence. The modeled system gives a fast and flexible way to design UML diagrams to accumulate the time and resources of both the user and system designer.**

Keywords-Natural language processing (NLP), Knowledge Extraction, Part of- Speech tagging, Software engineering.

I. INTRODUCTION

Software modeling is a crucial activity to make a certain degree of excellence in software systems. But, modeling is an effort-intensive activity. In its usual form, individual testers (test engineers) perform most (if not all) phases of software modeling physically. Single phase is test-case design in which the individual tester uses write (formal) necessities, written often in natural language (NL), to develop a set of modeling cases. Modeling-case design is also an effort-intensive activity, and practitioners are ready to find aid from any (partially) preset approach to remove experimental suites from necessities. Such a practice could keep away software companies extensive assets which are frequently tired to manually develop and text test cases from requirements. Also, as software requirements vary, analysis has to be maintain, an activity which incurs additional effort.

The quick increasing information technology field demands changes in predictable methods of software modeling and designing. Where, these changes has affected the use of present tools and techniques in the software engineering methodologies, at the same time, the latest tools and techniques also have been asked for to cop up with the rising demands of recent information technology needs. The problem addressed in this research is mainly related to the software analysis and design phase of the software development process. Modern software engineering is based on object oriented design (OOD) paradigm.OOD based software modeling is also called Component Added Software Engineering (CASE).

Keeping in view the requirements of current software designing, lots of methods and techniques
have been planned the use of NL text for automatic OO software modeling. No one from these tools is capable to remove the total information i.e. classes, objects and their respective, attributes, methods and associations. A few tools just remove names of classes and some of them just deal with objects. K. Li [7] suggested some enhancements to the existing OOA systems but still there was need of a NL-based CASE tool that may have ability of doing automatic analysis of the NL text and extract required information for OO modeling and also generate code from the generated model.

This article presents a rule-based system for NL-based OO Software Modeling. It starts by the
overview of the previously presented theories and designed systems for NL text based OO analysis and modeling. Next section points the methods of analyzing the natural language text and then a methodology has been presented to generate the software modeling diagrams. Then implementation details have been provided with the analysis of the performed experiments. In the end, Results and analysis have been presented with the advantages and disadvantages of the used approach.

II. LITERATURE REVIEW

Natural languages have been an area of interest for researchers for last many decades. In the late
Nineteen sixties and seventies, so many researchers as Noam Chomsky (1965) [5], Chow, C., & Liu, C (1968) [6] contributed in the area of information retrieval from natural languages. They contributed for analysis and understanding of the natural languages, but still there was lot of effort required for better understanding and analysis.

R. J. Abbot an idea that natural language text can be used for object oriented software modeling. He suggested that the state of a class or an object can be identified by nouns and the behavior or functionality of a class or object can be identified by verbs [4] in a sentence. On the other hand, H. Buchholz proposed that nouns not only specify classes and objects but also properties [3] of an object or a class. Different NLP based tools have been proposed for this

purpose. Nan Zhou proposed another conceptual modeling system based on linguistic patterns [2].
A. Oliveira used natural language constituents for OO data modeling and presented a CASE tool named REBUILDER UML [2]. This tool integrates a module for translation of natural language text into an UML class diagram. This module uses an approach based on Case-Based Reasoning and Natural Language Processing. But, this case tool needs continuous up gradation of case-base and only deals with class diagrams. There was another significant contribution by Harmain and Gaizauskas as they presented another NL based CASE tool named CM-Builder [1]. This CASE tool was restricted to create a primary class model. There was no appropriate mechanism for confining objects from NL text. Borstler and Overmyer presented another NL based system to generate class diagrams from user requirements given in NL text

III. METHODOLOGY

Typically, design is related to manage and control the complexity by splitting a big task into small ones. Object oriented architecture in NL text, distinct classes have been defined of the various constraints of natural language text. The class of a particular word or phrase is identified by analyzing the type of prepositions used with that particular word or phrase. The defined classes are as following:

1. Object Class
2. Method Class
3. Attribute Class

i. Object Class:

This class contains the objects and classes in a particular scenario. Objects can be defined in many terms in the natural language text e.g. a person/ thing who is performing some task or a person/ thing for whom some action is being performed or a person/ thing which is under some action.

ii. Method Class:

In method class, all the tokens related to some action are characterized. The action performed in the sentence represents the method or function of an object. For example, “Customer pays the bill”. In this example, ‘pays’ is action or method of object customer, which is the main actor object. Methods can be identified in two different ways:

- Methods or actions in a sentence can appear after the noun or pronoun and there is no helping verb and the action has‘s’ or ‘es‘suffix. For example, “Customer buys a pen”. Here ‘buy’ is a method of object ‘customer’.
- Methods or actions in a sentence can appear after a helping-verb. For example, “Customer is
  - buying a pen”. Here ‘buy’ is a method and has appeared after ‘is’ helping-verb

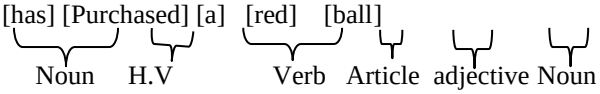
• Methods or actions in a sentence can appear after a preposition; to, of, etc. For example,
“Customer is demanding to book his order”. Here “book” is a method that has appeared after
‘to’ preposition.

iii. Attribute Class:

In this class, the attributes of an object are identified. In a sentence, adjectives are attributes of an object. All the adjectives related to an object are nominated as the attributes of that object. For
Example, “Customer buys the red ball”. Here ‘ball’ is the object and ‘red’ is its attribute. All the
Adjectives related to the ‘ball’ object are characterized its attributes.

Designed Algorithm:

A rule based algorithm was written to analyze NL text and then extract various OO modeling elements. The major steps of the algorithm are as following.
1. In first step, UMLG reads and tokenizes the text containing software requirements by the user
e.g. the output of a sentence “The Chair has four legs.” is
[Customer] [has] [purchased] [a] [red]
[ball].
2. In second step, morphological analysis is performed of given text to define the structuring and
transformation of the words. POS Tagging is also performed to identify different parts of speech. [Customer] [has] [Purchased] [a] [red] [ball]



Parse tree generated for the example

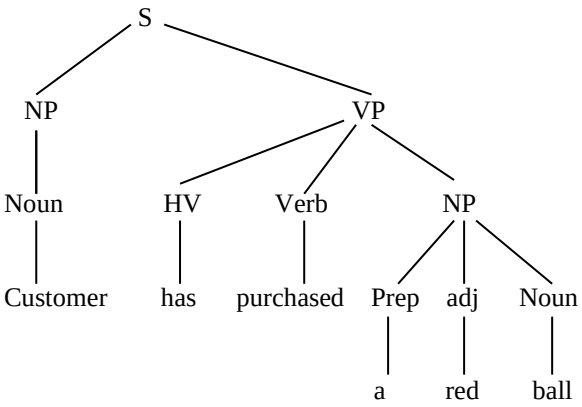


Figure. 2

3. Syntactic Analysis carried out to validate phrases and sentence according to grammatical rules defined by the English language. This step also helps in identifying the main parts of a sentence; object, subject, actions, attributes.
4. In this step, associations are identified by doing semantic analysis. It is determined in this specified that which actions have been performed by which object and a set of attributes belong to which object.
5. Then a rule based module specifies subject nouns as objects, verbs as methods of the objects, and adjectives as attributes of the object. Object nouns are sometimes specified objects and Sometimes as attributes.
6. In this step associations and relationships among extracted classes and objects are performed.
Prepositions are major tool for identifying relationships and associations.

7. A logical model of the class diagrams is generates on the basis of previously extracted information.
8 .A drawing module converts the logical model into the class diagrams by connecting small pieces of images already stored in database.
9. In next step, associations among generated class diagrams will be also produces.
10. After generating class diagrams, diagrams are labeled with appropriate labels.
11. The final step is conversion of logical model to VB.NET and Java Coding.

IV. POS TAGGING

Parts of Speech (POS) tagging are the first phase of text processing. Each token is analyzed and classified into its respective POS classification: Noun, verb, pronoun, adverb, Helping-verb, adjective, prepositions, etc.

Architecture of the designed system

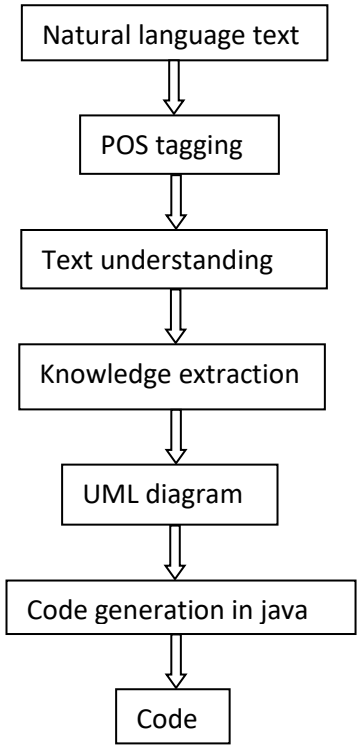


Figure.1

V. CONCLUSION AND FUTURE WORK

The designed system can find out the classes and objects and their attributes and operations using an artificial intelligence technique such as natural language processing. Then the UML diagrams such as Activity dig., Sequence dig., Component dig., Use Case dig., etc would be drawn. The designed system for generating UML diagrams and their respective code in multiple languages typically Java was started with the aims that there should be a software which can read the scenario given in English language and can draw the all types of the UML diagrams such as Class diagram, activity diagram, sequence diagram, use case diagram. But other types except class diagram are still under working.

REFERENCES

[1] L. Mich, R. Garigliano, (1996) "A linguistic approach to the development of object-oriented
system using the NL system LOLITA", Object Oriented Methodologies and Systems,
(ISOOMS), LNCS 858, pp. 371-386.
[2] Nan Zhou, Xiaohua Zhou, (2004) "Automatic Acquisition of Linguistic Patterns for Conceptual Modeling", *INFO 629: Artificial Intelligence*, Fall 2004
[3] H. Buchholz, A. Dusterhoft, B. Thalheim, (1996), "Capturing Information on Behavior with the RADD_NLI: A Linguistic and Knowledge Base Approach", Proc. Second Workshop
Application of Natural Language to Information System, IOS Press pp. 185-196
[4] R.J. Abbott, (1983) "Program Design by Informal English Descriptions", Communications of
the ACM, Nov. 26(11), pp. 882-894.
[5] Chomsky, N. (1965) "Aspects of the Theory of Syntax. *MIT Press, Cambridge*, Mass, 1965.
[6] Chow, C., & Liu, C. (1968) "Approximating discrete probability distributions with dependence trees". IEEE Transactions on Information Theory, 1968, IT-14(3), 462–467.
[7] K. Li, R.G.Dewar, R.J.Pooley, [2003] "Object-Oriented Analysis Using Natural Language
Processing",
www.macs.hw.ac.uk:8080/techreps/docs/files/HWMACS-TR-0033.pdf