International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
*ISSN (Online)2349-6967, Volume **2**, Issue 1.*

OBJECT MODEL DESIGNING USING NATURAL LANGUAGE PROCESSING

Kashmira Dixit[1], Bhagyashree Wani[2], Prajakta Kshirsagar[3]
(Computer Department, GESRHSCOE/Savitribai Phule Pune University, India)

Abstract :-Software developmentis a complex process and begins with requirement gathering. Requirements are specified in their natural forms and are required to be converted into formal specifications such as object diagram for further use in any software development process. These diagrams are useful for clearly understanding the problem definition of software and other documentation purpose. This paper aims at developing specialized tool for generating class diagram and usecase diagram from requirements specified in natural language. For this purpose we present a new methodology to develop object diagram from natural language requirement specifications named Requirement Analysis and Object diagram Extraction Tool (RAOET).

Keywords :-Natural Language Processing (NLP), NLP Parser, Object Diagram, Ontology, POS Tagging, Semantic analysis, WORDNET

I. INTRODUCTION

The Object Diagram generation falls under the domain of NLP. NLP is processing on natural language. It is a branch of computer science which deals with computer and human interactions. There are many challenges in NLP such as understanding of the natural language and also making the computer system exactly derive same meaning which is required from natural or human language. Our tool RAOET also involves processing on input document and generating diagrams. For any software development the most complex and time consuming task is to develop object diagrams from the requirements given by the user. Users generally specify their requirements in their natural language. These requirements which are specified in natural language are usually difficult to understand and many times not acceptable as formal specification. They also create ambiguity in understanding the functional requirements of software. Hence it is necessary to convert this requirement specification in natural language to formal specification such as object diagrams. This conversion is possible manually but it will be a tedious and time consuming job for any software developer. The conversion of natural language specification into object diagram is based on certain transformation rules. These transformation rules depend on syntactic, semantic and grammatical analysis of the input document consisting of requirement specification. Thus we present a new automation tool for object diagram generation. It is a specialized tool which mainly concentrates on class diagram and usecase diagram generation. In order to draw class diagram our system will require classes, attributes and relationships among different classes. Class diagrams describe the responsibilities of a software and also they represent the static view of the software. Our system will generate class diagrams from the given input document which will be consisting of requirements in natural language by applying different class identification rules, attribute identification rules and different relationships identification rules. For usecase diagram our system requires actors, usecases and relationships between them. Usecase diagrams are one of the object diagrams that represent outside runtime view of the system . It also identifies various internal and external factors affecting the system.

II. SYSTEM ARCHITECTURE

The basic working of our tool for object diagram generation tool is shown in Fig 1. The RAOET algorithm is the algorithm specified in this paper according to which the proposed system will work. The input to our system is the text document on which the proposed algorithm will be applied and object diagrams will be generated. For this purpose our system takes help of OpenNLP Parser, WORDNET, POS Tagger and Domain Ontology,

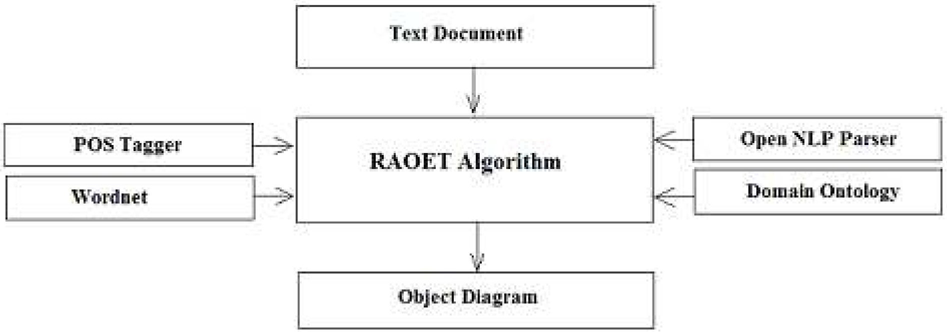

Fig.1: System Architecture

The detail architectural view of working of our system is as shown in Fig 2. The input to the system is the text document given by the user. It will have the requirements specified in natural language. Our system accepts input document in different formats such as word document, RTF document, text document etc. Tokenization is performed on the whole document. It is the process of breaking stream of text into word, phrases, symbols or other meaningful elements. From the whole document stop words are identified and frequency and occurrence of every word is calculated. Stemming is performed. In order to identify valid concepts that will be used further for classes, attributes etc our system will parse the whole document using openNLP Parser and will also perform POS Tagging. The output of this step will be all valid concepts . All these concepts are then compared with synonym and hyponym list and wordnet is also applied to it. Now classes, attributes, relationship identification algorithms will be applied and class diagram will be generated. By using same algorithmic strategies usecase diagrams will be generated.

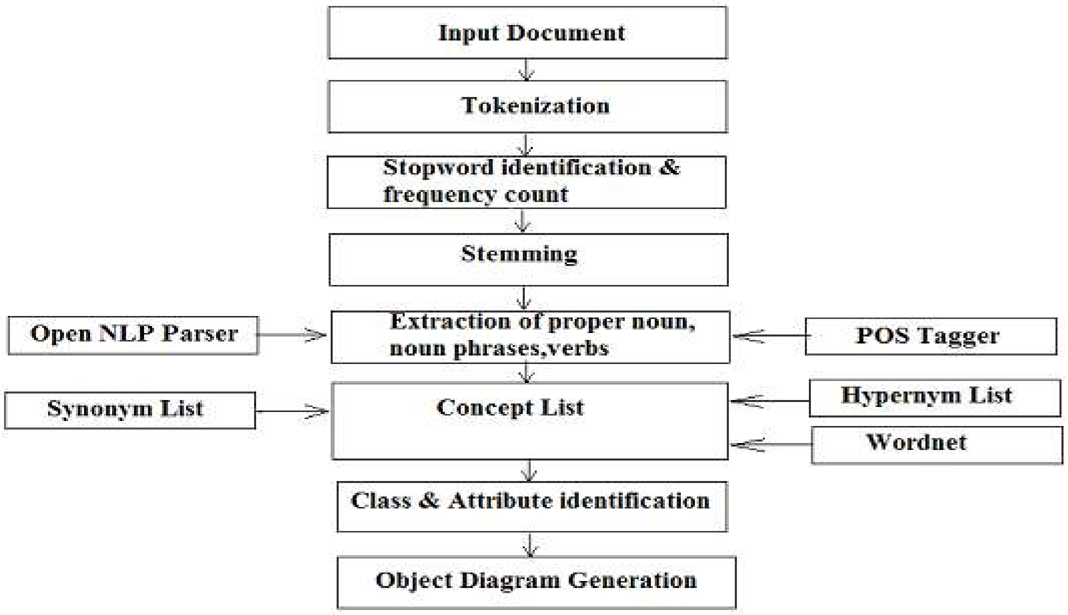
Input Document
Tokenization
Stopword identification & frequency count
Stemming
Open NLP Parser
Extraction of proper noun, noun phrases,verbs
POS Tagger
Synonym List
Concept List
Hypernym List
Wordnet
Class & Attribute identification
Object Diagram Generation

Fig.2:System Architecture

III. ALGORITHM AND STRATEGIES USED

Algorithm 1:

1. Take input :-
   The requirement document which is given by the user is the input document to the system on which the natural processing is done and the diagram will be generated

2.Stopwords Identification :-
   Read the input document and identify stopwords and save the result in a list say stopwords.

3. Frequency counting :-
   Count total no. of words in a document, count total no. of words without stopwords, count occurrence of each word and calculate the frequency of each word.

4. Stemming :-
   Perform stemming of each word and save the result.

5. Use OpenNLP parser :-
   Use the openNLP parser and parse the whole input document. The output of parser is used to identify proper nouns, verbs and noun phrases.

6. Generate concept list :-
   Concept list consist of all nouns, verbs, stopwords.

7. Give synonym list as input and compare concept list and synonym list :-
For each concept (CT) in {Concepts-list} if {synonyms-list} contains a concept (CT2) which have a synonym which lexically equal to CT, then CT and CT2 concepts are semantically related to each other

8. Give hypernym list as input and compare concept list and hypernym list :-
For each concept (CT) in {Concepts-list} if {hypernyms-list} contains a concept (CT2) which have a hypernyms which is lexically equal to CT, then CT2 "**is a kind of**" CT.

9. Get generalization list :-
Generalization list consist of proper concepts on which various class, attributes and relationships identification rules can be applied.

10. Apply algorithm for class identification .

11. Apply algorithm for attributes identification.

12. Apply algorithm for relations identification.

13. Generate Object diagrams.

Algorithm 2 (Class Identification):

1. AS we know the frequency count of every concept if a particular concept is having a frequency less than 2% or even if the concept is occurred only one time ignore it as a class.

2. If a concept is related to design elements such as 'system', 'computer', 'data', 'application' etc then ignore it as a class.

3. If a concept is a name either of a person, location such as 'George', 'England' ,'Tommy' etc then ignore it as a class.

4. If a concept has a hpernym in the higher level of hypernym tree then it states that the concept is general and can be replaced by a specific concept then ignore it as a class.

5. If a concept is a attribute such as 'name', 'address', 'contact_no' etc then ignore it as a class.

6. If a concept does not satisfy any of the above rules then there is a possibility that a concept is a class.

7. If a concept is noun phrase (Noun+Noun), if the second noun is an attribute then the first is a class. The second noun is an attribute of that class such as 'customer name', 'book ID' etc.

Algorithm 3 (Attribute Identification):

1. 2. If a concept is noun phrase (Noun+Noun) including the underscore mark "_" between the two nouns, then the first noun is a class and the second is an attribute of that class Examples "customer_name", "departure_date".
2. If a concept can has one value, then it's an attribute. Examples:"name, date, ID, address".
3. We will also store some predefined attributes.

Algorithm 4 (Relationship Identification):
1. All the elements in the {generalization-list} will be transferred as **Generalization** (IS-A) relationship.

2. If the concept is verb, then by looking to its position in the document, if we can find a sentence having concept1-verb –concept2 where concept1 and concept2 are classes, then verb is an Associationrelationship.

3. If the concept is verb and satisfies R-Rule2, and the concept is equal to one of the following {"consists of", "contain", "hold, "include", "divided to", "has part", "comprise", "carry", "involve", "imply", "embrace"}, then the relationship that discovered by that concept is **Composition** or **Aggregation**. Example: "Library Contains Books" then the relationship between "Library" and "Book" is Composition relationship.

4. If the concept is verb and satisfies R-Rule2, and the concept is equal to one of the following {"require", "depends on", "rely on", "based on", "uses", "follows"}, Then the relationship that discovered by that concept is the **Dependency** relationship. Example: "Actuator uses sensors and schedulers to open the door", then the relationships between ("Actuator" and "sensor"), ("Actuator" and "Scheduler") are the Dependencies relationships.

5. Given a sentence in the form concept1 + relation1 + concept2 + "AND"+ concept where concept1, concept2, concept3 is a classes, and relation1 is a relationship. Then the system will indicate that the relation relation1 is between the classes (concept1, concept2) and between the classes (concept1, concept3).

6.Given a sentence in the form concept1 + relation1 + concept2 + "AND NOT"+ concept3 where concept1, concept2, concept3 are classes, and relation1 is a relationship. Then the system will indicate that the Relation R1 is only between the classes (concept1, concept2) and not between the classes (concept1, concept3).

OpenNLP Parser:-

We chose OpenNLP as a parser in our system. OpenNLP is an open-source and re-usable algorithm. It provides our system with lexical and syntactic parsers. OpenNLP POS tagger (lexical) takes the English text as input and outputs the corresponding POS tags for each word; On the other hand, OpenNLP Chunker (syntactic) chunks the sentence into phrases (Noun phrase, verb phrase, etc.) according to English language grammar. The high accuracy and speed in OpenNLP encouraged us to choose it rather than other existing parsers. OpenNLP uses lexical and syntactic annotations to denote to the part of speech of the terms; for example, NN denotes to Proper Noun, VB denotes to Verb, and NP denotes to Noun Phrase. OpenNLP parser supports our system with an efficient way to find the terms' part of speech (POS) which we need in order to accomplish the noun and verb analysis.

Wordnet:-

Wordnet is used to validate the semantic correctness of the sentences generated at the syntactic analysis. It also enables users to display all hypernyms for a selected noun. We used this feature to verify Generalization relationship where a noun phrase is supposed to be 'a kind of' another noun phrase. WordNet can used to find semantically similar terms, and for the acquisition of synonyms. We used synonyms to extract words which are semantically related to each other. We calculated the words frequency to keep the synonyms with high frequency in the document.

Domain Ontology:-

It is used in the phase of Concept Identification. It exhibit knowledge about all the valid concepts. We can store these concepts as a concept list. By using ontology, we can find out classes, attribute, operations and relationships more accurately.

IV. CASE STUDY

Input Document:

The name of our college is GESCOE.The college id is 69.College has number of students, staff and different departments.Each student has name,id,and department.Each staff has name, salary, position in the department.Each staff member teaches different subject.Each department has its unique name and id.Each department has HOD.HOD is head of the department.HOD has name.HOD should have a good work experience.

Class Diagram:

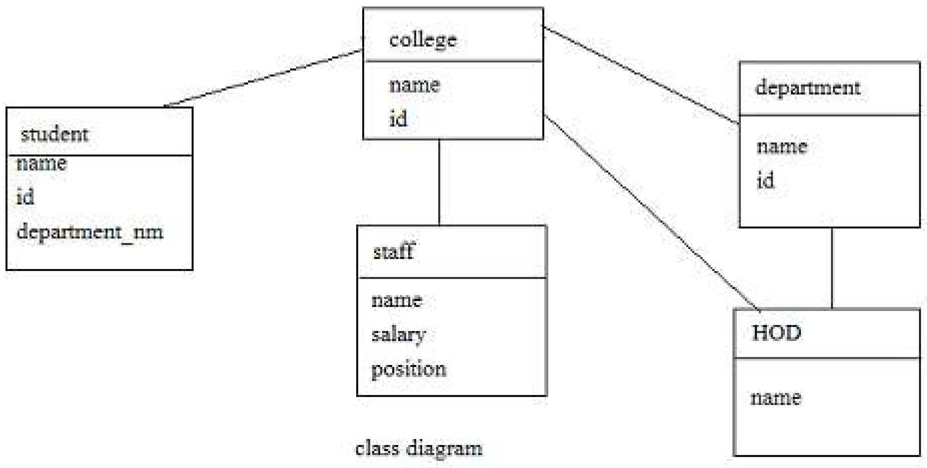

Fig.3.Class Diagram

V. CONCLUSION

Thus in this paper we bring up a new approach of generating object diagrams. Our approach is based on natural language processing techniques and domain ontology. The approach of this paper mainly concentrates on generation of class and usecase diagrams from the input given to the tool as document. It efficiently generates classes, attributes, relationships, usecases, actors. The tool is capable of identifying relationships between classes such as generalization, aggregation, association and dependency. We have named our system as Requirement Analysis and Object diagram Extraction Tool (RAOET). Our tool is restricted to identification of relationship mentioned above. It is difficult to efficiently identify the one to many and many to many relationships for this tool. In future it is required to concentrate on generation of other object diagrams and also increasing the efficiency by identifying other relationships.

REFERENCES

[1] Agung Fatwanto, Software Requirements Translation from Natural Language to Object Oriented Model,IEEE,2012.

[2] S.D.Joshi, Dhanraj Deshpande, Textual Requirement Analysis for UML Diagram Extraction by using NLP, International Journal of Computer Applications (0975 – 8887) Volume 50 – No.8, July 2012.

[3] Subhash K.Shinde,Varunakshi Bhojane,Pranita Mahajan, NLP based Object Oriented Analysis and Design from Requirement Specification**,** International Journal of Computer Applications (0975 – 8887) Volume 47– No.21, June 2012.

[4] Soumaya Amdouni, Wahiba Ben Abdessalem Karaa, Sondes Bouabid, Semantic annotation of requirements for automatic UML class diagram generation, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011.

[5] Imran Sarwar Bajwa, Irfan Hyder, UCD-Generator – A LESSA Application for Use Case Design,IEEE- International Conference on Information and Emerging Technologies- IEE-ICIET, Karachi, Pakistan.

[6] Graciela Brusa, Ma. Laura Caliusco, Omar Chiotti, A Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology, Australian Computer Society, Inc. (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology,Vol. 72. 2006.

[7] Prof. Dr.S.D.Joshi, Dhanraj Deshpande, Textual Requirement Analysis for UML Diagram Extraction, ISSN 2249-6343 International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 3, June 2012.

[8]Prof. D.M.Thakore, Ravi P.Patki, Generation of Software Artifacts and Models at Analysis Phase, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 , Vol. 2, Issue 5, September-October 2012, pp.1624-1630.

[9] Bhagat S. B., Kapadni P. R., Kapadnis N., Patil D. S., Baheti M. J, Class Diagramn Extraction Using NLP 1st International Conference on Recent Trends in Engineeringand Technology,ISSN,Mar-2012.

[10] Subhash K.Shinde, Varunakshi Bhojane, Pranita Mahajan, NLP based Object Oriented Analysis and Design from requirement Specification, International Journal of Computer Applications (0975 – 8887) Volume 47– No.21, June 2012.

[11] Luiz Marcio Cysneiros,Julio Cesar Sampaio do Prado Leite, Nonfunctional Requirements :From Elicitation to Conceptual Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,VOL. 30, NO. 5, MAY 2004

[12] Bhagat S. B., Kapadni P. R., Kapadnis N., Patil D. S., Baheti M. J., Class Diagram Extraction Using NLP 1st International Conference on Recent Trends in Engineering and Technology, 1st International Conference on Recent Trends in Engineering & Technology, ISSN: 2277-9477, Mar-2012.
[13] M.G. Ilieva and Olga Ormandjieva, Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation, A. Montoyo et al. (Eds.): NLDB 2005, LNCS 3513, pp. 392–397, 2005.© Springer-Verlag Berlin Heidelberg 2005.

[14] Imran Sarwar Bajwab, Ali Samad, Shahzad Mumtaz, Object Oriented Software Modeling Using NLP based Knowledge Extraction, European Journal of Scientific Research ISSN 1450-216X Vol.35 No.1(2009),EuroJournals Publishing, 2009.

[15] Priyanka More, Rashmi Phalnikar, *Generating UML Diagrams from Natural Language Specifications, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA,Volume 1– No.8, April 2012.*

[16] G.S.Anandha Mala, G.V. Uma,*Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirement Specification, Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 1155 – 1159, 2006.© Springer-Verlag Berlin Heidelberg 2006.*

[17] Devendrasingh Thakore , Akhilesh R. Upadhyay, *Development of Use Case Model from Software Requirement using in-between SBVR format at Analysis phase, ISSN (Print) : 2319 2526, Volume-2, Issue-2, 2013.*