Impact Factor (SJIF): 3.632

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 3, Issue 5, May-2016

Ontology Based Class Diagram Generation.

Sachin Kalamkar, Vicky Shinde, Pratiksha Mandge, Harshal Pawar

Computer Engineering Department, Dr.D.Y.Patil Collage of Engineering Ambi Talegaon.

Abstract *— UML (Unified Modeling Language) is not about creating diagrams but a way of creating models. Therefore, to fill out this we have purposed "Computer Automated Object Oriented Analysis and Design from Requirement Specification "a software system that accepts software requirement specification (SRS) as input to the system in the form of simple text language i.e. English plain text and does the OO analysis on requirement specification and generates class diagram and sequence diagram automatically. User is not required to have expert knowledge of domain. There are several tools to support draw these diagrams. But here our purpose is to make this process of extraction of diagrams more specific, user-friendly and automated manner .*

Keywords- *SRS, OO analysis and design, NLP, Domain Ontology.*

I. INTRODUCTION

In order to capture essential and relevant software requirements for constructing a software system, Natural Language descriptions often need to be analyzed, transformed and restructured into a form of design notation, eg Object-Oriented Analysis might produce Unified Modeling Language (UML) models. However, OOA exhibits some unique difficulties that are inherited form NL. Firstly, NL is highly informal in nature, with speakers or writers inventing new forms and combinations of language. Indeed, sentence patterns of NL can be complex and ambiguous, which may lead to multiple interpretations. Secondly, based on the ambiguous and complex NL descriptions of user requirements, implicit requirements can be difficult to recover; especially when requirements engineers are not familiar with the problem domain of the system under consideration. Finally, such informal requirements descriptions could contain large amounts of information that can be time consuming to analyze

Previous studies have shown that using Natural Language Processing in OOA can lead to automatic text analysis. Therefore, NLP can disambiguate natural language descriptions and assist model generation. The objective of this paper is to investigate how the problems related to NL can be addressed in OOA by NLP-based tools. We begin with a background of OOA and consider linguistic theories and techniques used in NLP systems. Next, we investigate some existing NLP systems for automatic OOA and evaluate possible gaps in their coverage. Finally, we suggest enhancements to existing OOA systems by including dialogue with the user.

II. LITERATURE REVIEW

2.1 Introduction

A literature survey is a discussion of the literature in a given area of the studies. It is concise overview of what has been studied, argued and established about a topic, and it is usually organized chronologically or thematically. It is not an annotated bibliography, because it groups related works together and discusses trends and developments rather than focusing on one item at a time. It is not a summary, rather it evaluates previous and current research in regard to how relevant and or useful it is.

2.2 Design Information

Domain Ontology:-

Domain Ontology serves as area knowledge to progress the presentation of abstract modeling.

Candidate Class Identification:-

Candidate Class Identification unit outputs beginning Candidates and polished Candidates, which serve as input of spider model. The part of speech tagger and sentence parser create beginning candidates. Further, word sense disambiguation system and semantic association are functioning to make polished candidates, which are semantically connected to the concepts defined in the ontology.

Sentence.

Stems:-

Stemming algorithm is used to find the stem (root) of each concept for further analysis. Once stop words are identified and removed, each inflected word is reduced back to its stem. We have used a Stemming technique that abbreviates word by removing affixes and suffixes.

POStagger:-

It parse output to extract Proper Nouns (NN), Noun phases (NP), verbs (VB). And save it in Concepts list. We have used OpenNLP POS Tagger to perform Noun Analysis, & Verb Analysis.

Relationship classification:-

Relationship classification section is the most significant component of the spider model. It uses relation space to identify all concept pairs with strong semantic relationship within a sentence. Then assigns dissimilar weight to each concept pair to indicate how strong the correlation is according to the constituents the concepts provide as in the sentence.

Attribute Identification:-

Attribute Identification module distinguishes attributes from classes. After two concepts are originate to be strong associated to each other, then whether the concepts stand for a class or an attribute of a class is recognized.

Naming Relationship:-

Naming Relationship module applies linguistic patterns to find aggregation association and simplification relationship.It uses numeric connection accepting technique to distinguish one-to-one organization p, one-to-many organization, and many-to-many organization.

Parallel Structure:-

Parallel Structure uses the characteristic of English sentence structure to help draw out extra attributes of a class, more child classes of a super class, and more aggregation part of a class. All over this method NLP techniques are applied. Part-of-speech tagger and sentence parser are used to create beginning class candidates even as word sense disambiguation (WSD) and semantic network are in work to purify beginning candidates. Association space created by Link Grammar Parser is used to find concept pairs within strong semantic association within a sentence. Dissimilar weights are assigned to all concept pair according to the linguistic patterns the concept pair belongs to .Linguistics patterns are used to verification class-attribute relationship, aggregation relationship, and generalization relationship, and distinguish three types of association relationships. Parallel structure serves as a clue to find more attributes of a class, more child classes of a super class, and more parts of a composite class.Secondly, this approach well integrates domain ontology with class production for the first time. The domain ontology contains main or core domain information. The ontology feeds the organization significant classes and their attributes .However this technique wishes a set of linguistic pattern as input[1].

III. SURVEY OF PROPOSED SYSTEM

In our paper, we focus on implementing class diagram and its java code generation. Software requirements are collected from customer as per their needs. Then this software requirements are forwarded to a NLP.Which will identify noun, verbs and phrases. These identified contents are again forwarded to OO analyzer. This will convert Noun words to Class and verbs to attributes. From this all data class diagram will be generated. From the class diagram the code will be generated. This will be a pure java code.

IV.DESIGN ALGORITHM

4.1 Class Identification Rules:

If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as class. If a concept is related to the design elements then ignore as class. Examples: “application, system, data, computer, etc…” If a concept is related to Location name, People name, then ignore as a class. Examples: “Seeta, Ram, Mumbai, etc…” If a concept is an attribute, then ignore as a class. Examples: “name, address, number” If a concept is noun phrase (Noun+Noun), if the second noun is an attribute then the first Noun is a class. The second noun is an attribute of that class. Examples: “Student Name” or “Book ISBN” If a concept is found in the high level of hyponyms tree, this indicates that the concept is general and can be replaced by a specific concept, then ignore as class. Examples: “user, object, etc.”

4.2 Attribute Identification Rules:

If a concept is noun phrase (Noun+Noun), then the second noun is an attribute. We have predefined a list of popular attributes, if concept matches with predefined list it is an attribute.

4.3 Relationship Identification Rules

If the concept is verb (VB), then by looking to its position in the document, if we can find a sentence having (CT1 - VB – CT2) where CT1 (concept 1) and CT2 (Concept 2) are classes, then (VB) is an Association relationship. If the concept is verb (VB) and satisfies Rule1,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.Given a sentence in the form CT1 + R1 + CT2 + "AND"+ CT3 where CT1, CT2, CT3 is a classes, and R1 is a relationship. Then the system will indicate that the relation R1 is between the classes (CT1, CT2) and between the classes (CT1, CT3) Given a sentence in the form CT1 + R1 + CT2 + "AND NOT"+ CT3 where CT1, CT2, CT3 are classes, and R1 is a relationship. Then the system will indicate that the relation R1 is only between the classes (CT1, CT2) and not between the classes (CT1, CT3).

V. MATHEMATICAL MODEL

Let S be the set of Whole system which consists:

S= {L_User,Stemming,POS Tagging,UML }

Where,

1. L_User will give SRS.

   L_User= {SRSList}.

   Where,
2. Ulist is set of noun and verbs.

   Ulist={noun,noun1,……..nounn} ..

3. Stemming={noub and verbs identification.}.

4. POS Tagging={store a non and verbs}.

5. Noun will be class name.

6. Verb will be attributes.

7. Relationships graphics will be used to draw UML digram.

8. NLP = (stemming,POS tagging).

9. UML = used to draw a the diagram

10. Result = gives the result of the java code generation.

.

VI. SYSTEM ARCHITECTURE

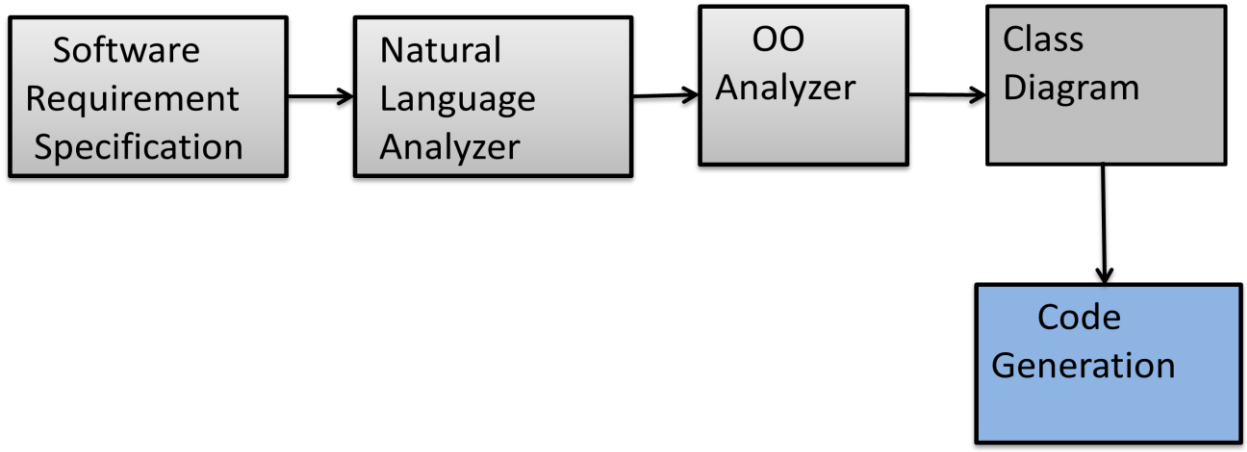

Fig 1:-SYSTEM ARCHITECTURE

VII. CONCLUSION AND FUTURE WORK

Various techniques developed earlier aim at making the analysis of user requirements easier. But these techniques expect user interference, in terms of providing the client information from time to time. The information available from the client in the form of natural language text is analyzed in such a manner that object-oriented modeling can be done by the system analyst precisely. The methodologies for simplifying natural language need to be straightforward, so that people can learn and apply them easily. It should be kept in mind that the models generated must be easily comprehensible and the natural language from which they have been derived can be traced back. The implementation of the proposed object oriented model gives way to easier understanding of lengthy NL text and subsequent validation of classes. Hence we can conclude that the aim and objective of the system is achieve by this proposed system .

VIII. ACKNOWLEDGMENTS

We might want to thank the analysts and also distributors for making their assets accessible. We additionally appreciative to commentator for their significant recommendations furthermore thank the school powers for giving the obliged base and backing.

XI. REFERENCES

[1]. Xiaohua Zhou and Nan Zhou, Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology, *Artificial Intelligence*, 2004.

[2]. Mohd Ibrahim, Rodina Ahmad, Class diagram extraction from textual requirements using Natural language processing (NLP) techniques, *second international conference on computer research and development*, 2012

[3]. L. Mich, NL-OOPs: From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA., *Natural Language Engineering*, 2(2, pp.161-187), 1996.

[4]. Song, Il-Yeol, et al. A Taxonomic Class Modeling Methodology for Object-Oriented Analysis , *In Information Modeling Methods and Methodologies:Advanced Topics in Databases Series*, *Ed*, pp. 216-240,2004.

[5]. Booch. G., Object-Oriented Analysis and Design with Applications, *2nd Ed., Benjamin Cummings*, 1994.

[6]. Gruber, Thomas R., A translation approach to portable ontology specifications", *Knowledge Acquisition* 5 (2):199–220, (June 1993).

[7]. Daniel Jurasky, Daniel H. Martin Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[8]. Soraya Setti Ahmed and Sidi Mohamed Benslimane, Reverse Engineering Process for Extracting Views from Domain Ontology.

[9]. Deborah L. McGuinness and Frank van Harmelen, WL Web Ontology Language Overview, Editors,*W3C Recommendation*, 2004

[10] M. Raya, J-P. Hubaux and I. Aad. DOMINO A System to Detect Greedy Behavior in IEEE 802.11 Hotspots. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, Boston, MA, June 2004.

[10]. Matthew Horridge, Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal2(1), Special Issue on Semantic Web Tools and Systems*, pp. 11-21, 2011.

[11]. R. Collobert, J. Weston, L. Bottou, M. Karlen, K.Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch, *Journal of Machine Learning Research(JMLR)*, 2011.