IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. III (Mar – Apr. 2015), PP 27-29
www.iosrjournals.org

Class Diagram Extraction from Textual Requirements Using NLP Techniques

Vrushali Adhav[1], Darshana Ahire[2], Aarti Jadhav[3], Dipali Lokhande[4]
[1,2,3,4,] *(Computer Engineering, MET's BKC IOE/ Savitribai Phule Pune University, India)*

Abstract: *A new method for translating software requirements to object-oriented model is proposed. Software requirements specified in a natural language using scenario like format will be transformed into class diagrams of Unified Modeling Language using specification transformation rule. The transformation rule is based on a grammatical analysis, more specifically syntactical analysis, of requirements specification. This paper proposes a new method for translating software requirements specified using natural language to formal specification (in this context is executable and translatable Unified Modeling Language Class Diagram). Requirements specification written in a scenario like format will be transformed into class diagram's components the automation of class generation from natural language requirements is highly challenging. This paper proposes a method and a tool to facilitate requirements analysis process and class diagram extraction from textual requirements supporting natural language processing techniques. Requirements engineers analyze requirements manually to come out with analysis artifacts such as class diagram.*

Keywords: *NL (Natural Language), NLP (Natural Language processing), POS (Part Of Speech), RACE (Requirement Analysis and Class Diagram Extraction), UML (Unified Modeling Language)*

I. Introduction

Requirements are often poorly understood and articulated, even by domain experts. It is common to find requirements specifications that are incomplete, inconsistent, ambiguous, and unstable. This problem emerges since software requirements are inherently complex and involving various stakeholders. The fact that specifying requirements should involve various stakeholders, who normally have no technical knowledge on software analysis and design, imposes developer (software analyst and designer) to use specification media which can be easily and intuitively understood by novice. In most practices, therefore, practitioners choose to specify software requirements using natural languages. The reason why natural language is preferred as the specification medium is because most stakeholders are more familiar with natural language compared to using other types of media (such as formal notation or modeling language). Requirements specified in a natural language are, however, normally written in a free form style. They still therefore need to be further analyzed and transformed into more formal specification especially for development and documentation purposes. In a number of software development methodologies, such as translate approach, formal specification is important since software (source) code is obtained by translating it.

II. Related Work

a. Need of system [4]

One of the most challenging and costly functions performed by IT staff today is the deployment of various software's to new or existing client computers. Currently, organizations spend a great deal of time and expense planning, designing, and rolling out the latest version of the operating system throughout the organization. Often this process is done manually, requiring a help desk professional to physically visit each computer. This paper will assist analyst by providing an efficient and fast way to produce the class diagram for their natural language requirement.

b. Existing Tools [3]

There have been several efforts for the analysis of natural language requirements. However, few are focused on class diagram extraction from natural language requirements. Thus, few tools exist to assist analysts in the extraction of class diagram. In this section we survey the works that use NLP or domain ontology techniques to analyze NL requirements, and the works that aim to extract class diagram based on NLP or domain ontology techniques. Following are some tools:

i. CIRCE

Ambriola and Gervasi present a Web-based environment called Circe. Circe helps in the elicitation, selection, and validation of the software requirements. It can build semi-formal models, extract information from the NL requirements, and measure the consistency of these models. Circe gives the user a complete environment that integrates a number of tools. Cico is the main tool that is considered as a front-end for the other components; it recognizes the NL sentences and extracts some facts from them. These facts are handed to the remaining tools for graphical representation and analysis. Natural language sentence of requirements is parsed converted into a forest of parse trees. These parse trees closely correspond to the original requirements but many surface features of requirement document are not included in abstract UML model.

ii. Zhou and Zhou

It proposes a methodology that uses NLP and domain ontology. It is based on that the core classes are always semantically connected to each other's by one to one, one to many, or many to many relationships in the domain. This methodology Class Diagram from textual requirements using NLP techniques finds candidate classes using NLP through a part of speech (POS) tagger, a link grammar ,parser, linguistic patterns and parallel structure, and then the domain ontology is used to refine the result.

iii. LOLITA

Mich L. proposes a NLP system, LOLITA to generate an object model automatically from natural language. This approach considers nouns as objects and use links to find relationships amongst objects. LOLITA system is built on a large scale Semantic Network (SN) that does not distinguish between classes, attributes, and objects. This approach is limited to extract objects and cannot identify classes. Hence for to overcome this problem we are trying developing our system that mean we think how to solve this problem and we find solution on this problem and we can apply on this our paper.

c. RACE System [2]

RACE system is decomposed into internal and external components and sub-systems. Figure 1 illustrates the architecture model of RACE:

i. OPEN NLP Parser

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.

ii. RACE Stemming Algorithm

Stemming is a technique that abbreviates word by removing affixes and suffixes. In RACE system, it is very important to return words back to its base form; this will reduce the redundancy and increase the efficiency of the system. To perform the stemming, we implemented a new stemming algorithm. Based on the stemming result, we find that our stemming algorithm is efficient and sufficient to be used in the morphological analysis of requirements in RACE system. Our stemming algorithm is simple and re-usable.

iii. 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.

iv. Concept Extraction Engine

The aim of this module is to extract concepts according to the requirements document. This module uses OpenNLP parser in, RACE stemming algorithm, and WordNet into extract concepts related to the given requirements.

i. Class Extraction Engine

This module uses the output of concept extraction engine module and applies different heuristic rules to extract the class diagram; However, We use domain ontology in this module to refine the extracted class diagram. We can summarize the heuristic rules.

i. Domain Ontology

As mentioned early in this paper, domain ontology is used to improve the performance of concepts identification. In RACE system we use a Library system Ontology as sample ontology. We gathered our ontology based on a survey we conducted on some library systems, and then we organize the ontology in such way to be easily maintained and re-used. The information in the ontology includes concepts related to classes for Library system, Attributes, and relationships. We used the XML to build the ontology.

III. Figures And Tables

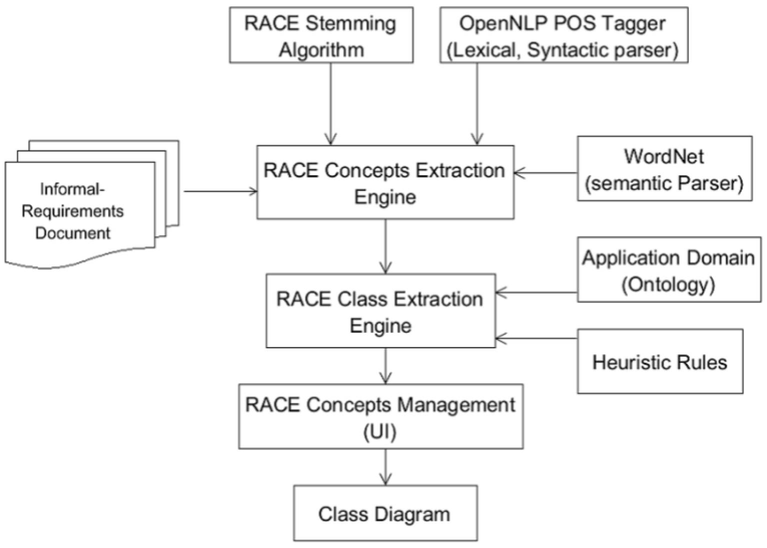
RACE Stemming Algorithm
OpenNLP POS Tagger (Lexical, Syntactic parser)
Informal-Requirements Document
RACE Concepts Extraction Engine
WordNet (semantic Parser)
Application Domain (Ontology)
RACE Class Extraction Engine
Heuristic Rules
RACE Concepts Management (UI)
Class Diagram

Fig 1: race system

IV. Conclusion

In this paper, we propose an enhanced approach that is based on NLP techniques to support the extraction of class diagram from NL requirements. We validate our approach by implementing a system called RACE referred to as “Requirements Analysis and class diagram Extraction”. RACE system efficiently demonstrates the using of NLP techniques in the extraction of class diagram from informal requirements.

Acknowledgements

We have taken efforts in this paper. However, it would not have been possible without the kind support and help of many individual. Our thanks and appreciation also goes to our beloved parents for their blessings, all the staff members, friends and colleagues for providing help and support in our work.

References

Journal Papers:

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

[2] Ambriola, V. and Gervasi, V. “Processing natural language requirements”, Proc. 12th IEEE Intl. Conf. on Automated Software Engineering, pp.

[3] Farid Meziane, Nikos Athanasakis, Sophia Ananiadou, 2007,Generating Natural Language specifications from UML class diagrams, Springer-Verlag London Limited .

[4] F Elizabeth D. Liddy & Jennifer H. Liddy, “An NLP Approach for Improving Access to Statistical Information for the Masses”.