Textual Requirement Analysis for UML Diagram Extraction by using NLP

S.D.Joshi, PhD.
BVCOE, Pune,
Maharashtra, India.

Dhanraj Deshpande
BVCOE, Pune,
Maharashtra, India.

ABSTRACT

Requirement analysis is the preliminary step in software development process. The requirements stated by the clients are analyzed and an abstraction of it, is created which is termed as requirement model. The automatic generation of UML diagram from natural language requirements is highly challenging and demanding very efficient methodology. Unified Modeling Language (UML) models are helpful for understanding the problems, communicating with application experts and preparing documentation. The static design view of the system can be modeled using a UML class diagram. System requirements stated by the user are usually in natural language form. This is an imprecise and inconsistent form which is difficult to be used by the developer for design UML model. We present a new methodology for generating UML diagrams or models from natural language problem statement or requirement specification. We have named our methodology as Requirement analysis and UML diagram extraction (RAUE).

General Terms

Textual Requirement Analysis by using NLP for generation of UML Diagram.

Keywords

Keywords Natural language processing (NLP), Domain Ontology, Unified Modeling Language, Requirement engineering, Software Requirement Specification

1. INTRODUCTION

Requirements prescribed by customer for their software development are generally stated in informal language which may be insufficient and inadequate. So before starting actual development process of software, developer should analyze textual requirements & then developer should create its abstraction or concept known as Requirement model. One way to solve this problem is to use Unified modeling language models which are helpful for understanding the problems, communicating with application experts & preparing documentation. By using UML models information can be effectively represented and can be used easily in all stages of software development process.

Natural language having inherent ambiguity and uncertainty. Since it is very difficult to understand and minimize ambiguity. We are using NLP and domain ontology techniques for requirement analysis. So we can extract UML diagrams easily and with more accuracy. It also exhibit different relationship like generalization, Association, Composition, aggregation and dependency. It also exhibit one-to-one, one-to-many, many-to-many relationships of association.

There are number of existing tools like CIRCE [9], LIDA (Linguistic assistant for domain analysis) [14], UCDA(Use case driven development assistant tool for class model generation)[15] etc. Using these tools we can generate different UML diagram. But these tools having different limitations that we have discussed in related work at section 2. In this paper we have reviewed certain tools.

Here we are using domain ontology with NLP for generating UML diagrams. It represents knowledge about concepts which is present in different vocabulary list and their relationships with each other. Functionality of RAUE is as follows;

Textual requirement document is given to RAUE system on which RAUE filtering algorithm is applied, which removes all irrelevant information from textual document, Ex. It converts plurals into singular forms etc. Refined result of RAUE filtering algorithm is then forwarded to lexical & syntactical parser which minimizes sentences into phrases like noun and verb phrase. Semantic parser is then, used to confirm semantic correctness of the sentences generated at the syntactic analysis. It facilitates us to display all synonyms. We can use these elements for different relationship where a noun phrase is supposed to be a kind of another noun phrase.

This result is then given to domain ontology technique & filtering rules. These are applied to identify important concepts, their relationship for UML representation of textual requirement. RAUE also facilitate user to print, save, and modify representations of UML model.

User can also add, delete, and modify some concepts which are extracted by RAUE tool.

2. RELATED WORKS

There are several efforts made for analysis of natural language. In this section we examine some previous work. Initially CASE tool called CM-Builder (Libya & R. Gaizauskas) [12] is based on natural language processing. It is used for analysis of casually expressed natural language requirements which is in textual document. It is basically used for object oriented system which support analysis phase of software development. By using robust natural language processing techniques, CM-Builder analyzes natural language textual form of software requirements which can be used further for building discourse model represented in semantic network. This semantic network then can be used for UML class model generation along with relationships among them. Even though coverage of CM-Builders parser's has proved adequate. It can be improved by encoding more grammar rules for the sentences, e.g. most sentences are most frequently used in software requirement texts. But because of its limitation CM-Builder can't identify operations for candidate classes.

CIRCE [9] is one of extraction tool that provides a simple but powerful framework for analyzing textual requirements expressed in natural language (NL) that is in English language. In this modeling & analysis activity is performed by a modular expert system. It assures easy extensibility & customizability of the environment. It depends on the concept of successive transformations that are applied to the requirements, in order to obtain concrete views of models extracted from the requirements. Here, firstly 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. Therefore it required subsequent analysis. CIRCE uses a domain-based parser called as "CICO". The parse tree obtained from the previous conversion are then encoded as tuples & stored in a shared tuple space. An embedded expert system is then called to refine the extensional knowledge in the tuple space and enriching it with more tuples. When a definite and accurate view on the textual requirements is desired, the needed information is find out from the shared tuple space by projectors. This conversion produces a conceptual description of desired view. In last conversion these abstract or conceptual views are converted into concrete views. Even though CIRCE is simple to use but it lacks in its efficiency in generating UML models from analyzed requirements.

Linguistic assistant for domain analysis (LIDA) [14] can also be used as an assistant in the model development process from software requirement. Here the analyst first imports the documents to be analyzed & POS (part-of-speech) tagging of the words. Then noun listing can be done for marking relevant candidate classes & iteratively removing those classes those are not relevant. Next task is identification of attributes of classes by adjective listing. Proceeding activity diagram the analyst then moves to Verb analysis for identifying methods & roles. Following this identification process, the analyst then proceeds to LIDA Modeler to graphically associate attributes, method, roles with appropriate classes. But problem with this approach is that LIDA needs more user interactions while generating diagrams. LIDA identifies just list of nouns, adjectives, verbs & developer and user has to decide which word goes in classes or attributes or operations. So LIDA depends on knowledge of problem domain to the Experts and those are familiar to these requirement document and LIDA system. If developer is having less knowledge then this tool may ignore some important functionality of system.

Use Case driven development assistant tool for class model generation (UCDA) [15] can also be used for requirement analysis purpose & model generation purpose. UCDA employs common requirement analysis technique to gather requirements & to document them in the textual requirement document. The software designer then analyzes requirement & identifies the objects by Object Model Creation Process (OMCP). Attributes, Associations & behavior are also established as a part of this model. Later object model is refined using generalization & object. Classes are identified based upon domain knowledge.

3. A MOTIVATING SCENARIO

- Provide the framework for auto-generation of class diagrams from free-text functional specification documents.
- Provide a quick and reliable way to generate UML diagrams to save the time and budget of both the user and system analyst.
- Allow user to visualize UML diagrams without need of installing any UML representation tool like Rational Rose.
- Automatically develop a Class model along with associated attributes & operations without much need of human interaction.
- Generate efficient class model with all possible relationships like Generalization, Association, Composition, aggregation and dependency that does not provided in existing tools.
- Provides a human-centered UI which makes user a part of the analysis process.

4. THE RAUE APPROACH.

RAUE follows the Object- Oriented Analysis and Design (OOAD) approach for object elicitation from requirements described in Natural Language to generate analysis and design UML models by following an approach based on NLP and domain ontology.

4.1. Theoretical Foundations

Developing the analysis of textual requirement and generating UML model, the different techniques of RAUE filtering model is used. This technique helps a requirements analyst to identify all possible objects or concepts from a given requirements document and generate analysis UML model by attaching attributes and methods along with their relationships.

However, the previously mentioned related work is a general framework for different UML diagrams as it doesn't clearly show the aspects NLP and having different limitations. We want to efficiently apply NLP and domain ontology techniques to our approach to achieve a fast and accurate analysis result.

4.2. OpenNLP Parser

As shown in figure 1, For natural language processing different tools are available like Sentence Detector, Tokenizer , Part-Of-Speech tagger(POS) ,Tree bank parser etc. Here we have used Open NLP Parser[7]. It is an open-source and re-usable parser. It provides lexical and syntactic parsers to the system. OpenNLP POS tagger (lexical) takes the English text as input and produces 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.

4.3. WordNet

WordNet [3] is used to confirm the semantic correctness of the textual document generated at the syntactic analysis. It gives all hypernyms for a selected noun to the user. It is used to validate Generalization relation where a noun phrase made-up of 'a kind of' another noun phrase.

WordNet can used to find semantically related and similar meaning terms. It is used to find out words which are semantically related to each other. We calculated the words occurrences in documents and find out its frequency in the

document. It is shown in figure 1 RAUE filtering Model by using NLP, how WordNet is used for semantically related word at verb, noun, and Phrase analysis phase.

5. RAUE FRAMEWORK AND DESIGN

5.1. RAUE filtering Model by using NLP

In this model, we have proposed requirement analysis and UML diagram extraction filtering model for UML diagram generation. It takes textual document as a input and produces class diagram or other UML diagram with their inter relationship between classes. It produces class diagram with attribute, methods and inter relationship between them. It extracts different relationship for concepts like aggregation, association, generalization etc. It also find out association's one-to-one, one-to-many, many-to-many relationship between concepts.

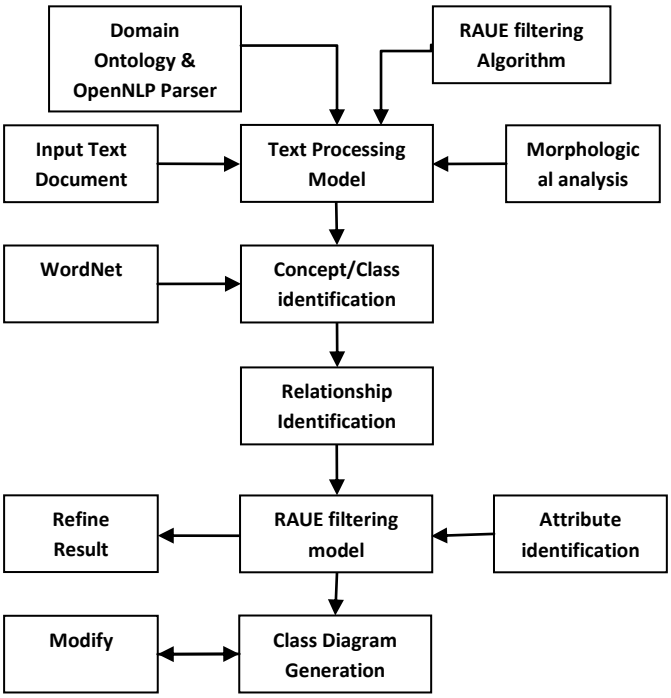

Figure 1: RAUE filtering Model by using NLP.

5.2. RAUE filtering algorithm:

RAUE filtering algorithm is used to remove suffix and affix. It is used for preparing list of "vocabulary-of-stopwords" and "vocabulary-of-concepts". It reduces redundancy of system. It is easy to use and very efficient for generation of UML diagram. It gives more accuracy than existing tool.

It gives result around 85% of Recall value and 85% of precision value as a performance parameter.

The **algorithm** is presented below.

Steps:

Step 1) Textual document is given as a input document to RAUE filtering model.

Step 2) Process the text document & tokenize the user requirement document. e.g. Library contains books and journals. This is input sentence which is afterword tokenized & produces output as,

<Library>, <Contains>,<Books>,<and>,<journals>,<.>

Step 3) searching of stop words & result of stop words is stored in "vocabulary-of-stopwords" list by using Morphological analysis.

Step 4) Defines total number of word in text document by removing stop word.

Step 5) Find out total occurrences of concept word in document by morphological & POS Tagging analysis.

Step 6) Use OpenNLP parser for segmentation, POS tagging, chunking of the textual document. It also does lexical & syntactical Parsing of document.

Step 7) In this step, result of open NLP parser is stored in "vocabulary-of-concepts" list. Result contain noun, verb, helping verb, adjective, article & noun phrase (NP).

e.g. Library contains books.

Step 8) Use of Step 2 to Step 7 for finding out "vocabulary-of- stopwords" and "vocabulary-of-concepts".

Step 9) In this step, different relationship between concepts words from "vocabulary-of-concepts" is recognized. It can extract different relationship like generalization, association, aggregation and dependency. It also find out one-to-one, one-to-many and many-to-many.

Step 10) At this step, UML diagram model is extracted by using previous steps information.

Step 11) Final step in which user can interact with diagram and make changes if required and save it.

5.3. Domain Ontology:

It is used in the phase of Concept Identification. It exhibit knowledge about concept. We can store these concepts as a "vocabulary-of-concepts" list. By using ontology, we can find out classes, attribute, operations and relationships more accurately. We have build RAUE tool and ontology by using java platform and XML.

5.4. Class Identification

By using RAUE filtering model, we can identify classes form given text document. It uses OpenNLP parser [7] and domain ontology for identifying classes. It find out noun, verb, helping verb and adjective. Many nouns having different meaning which confuses to the system. Hence it uses OpenNLP parser to find out actual noun. It ignores design elements like 'server', 'computer', 'data' etc. If concept is in the form of noun phrase like 'noun+noun' and if second noun is attribute then first is class. If ontology contains information of concept like relationship, attribute, operation then it consider it as a class.

5.5. Morphological Analysis

At this phase, it extracts componential scenery of words, which are made by morphemes (different meaning of word). This is one of the processing levels of NLP. So NLP can distinguish the meaning expressed by each morpheme.

5.6. Relationship Identification

In this phase, it find out relationship of concept with other concept which is present in "vocabulary-of-concept" list. If ontology contains information about relationship with concept then we can find out different relationship. It find out aggregation, association, generalization, dependency as well as one-to-one, one-to-many, many-to-many relationships.

If two concepts are joined by verb like Concept-verb-Concept, then verb is a association relation.

If verb is like 'contain', 'comprise' etc. then it can say it is composition or a aggregation.

If verb like 'depends on', based on' etc. then relationship is dependency.

5.7. Attribute Identification

When a concept is extracted from given textual document, it may have semantic relationship with an existing class. At that time it is difficult to say that it is class or attribute.

If concept is noun phrase like 'Book_author', then Book is a class and author is an attribute if noun phrase is defined by underscore mark"_".

If concept/class having parameter like id, price, color etc. then it will consider as a attribute.

It also maintains 'vocabulary-of-attribute' list in which most occurring attributes are stored. When any input textual document is given then RAUE filtering model, initially model fire query on this list. If it is present in 'vocabulary-of-attribute' then it will consider as an attribute otherwise by using OpenNLP parser, it process on it and find out whether it is class or attribute.

6. RAUE IMPLEMENTATION

In RAUE implementation we have implemented RAUE filtering algorithm along with external application like OpenNLP parser, WordNet and some Java Native Interfaces. OpenNLP parser [7] is a open source and efficient for extracting information. It uses lexical and syntactic parser. We have selected POS (part-of-speech) tagging as a tagger (Eric Brill's rule-based, Brill, 1994), WordNet [3] is used to confirm the semantic correctness of the textual document generated at the syntactic analysis. It is written in java platform. We have used third party JNI for drawing UML diagrams. Our main application that is RAUE filtering model which is implemented in java language. Java is platform independent since our application can run Windows, Linux and UNIX also. We can expect, our system is able to run on different platforms. Initially We have developed and tested our application on Windows platform. The architecture of the RAUE system is presented in figure 2.

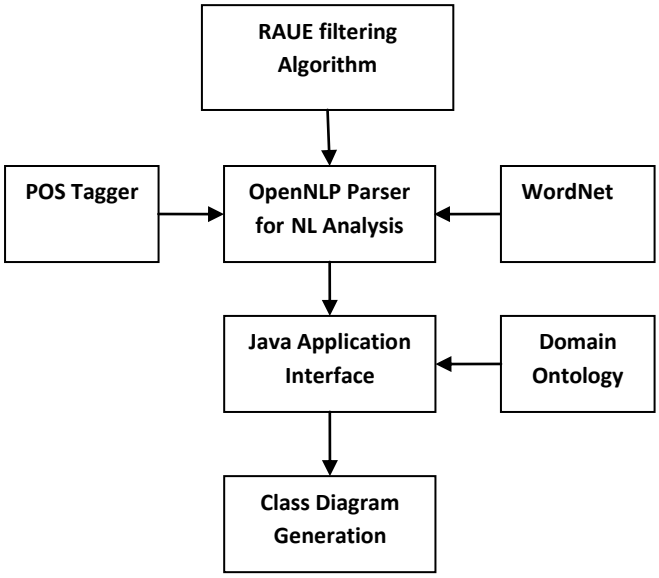

Figure 2: RAUE System Architecture

Our RAUE system can accept any free-text document describing functional and well-designed specifications.

RAUE filtering model can accept input textual requirements from different sources like words documents text files (TXT), (DOC) and hypertext document (HTML).

The RAUE filtering model generate class diagram and other UML diagrams along with attribute, methods and their inter relationship. It is able to run multiple use concurrently.

We can extend our application with any database. We have tested our RAUE application with MS-Access. Right now RAUE supports to English language only.

7. CONCLUSION

In this paper, we proposed a better approach that is based on NLP and domain ontology techniques to support the extraction of class diagram along with other UML diagrams from input textual NL requirements. In this paper, we have mainly focused on class diagram extraction but RAUE tool can extract other UML diagram efficiently. We certify our approach by implementing a system called RAUE referred to as "Requirements Analysis and UML diagram Extraction". The concepts extracted by RAUE system are completely valid and more accurate based on NLP. It also able to find all types of relationships: Generalization, Association, Composition, aggregation and dependency. It is also able to find out one-to-one, one-to-many and may-to-many relationships. The class diagram can be generated and modified if user feels to make changes by interacting with generated diagrams. RAUE able to run on different platform. It gives result of 85% in terms of recall value and precision value. The concept and diagram extracted by RAUE is completely suitable and recognized by different user.

8. ACKNOWLEDGEMENT

I am very thankful to the people those who have provided me continuous encouragement and support to all the stages and ideas visualize. I am very much grateful to the entire BVDU group for giving me all facilities and work environment which enable me to complete my task. I express my sincere thanks to **Prof.Dr.S.D.Joshi**, **Prof.Dr.S.H.Patil**, Head of the Computer Department, BVDU College of Engineering, Pune who gave me their valuable and rich guidance and help in presentation of this research paper.

9. REFERENCES

[1] Brill E., Some Advances in Transformation-Based Part of Speech Tagging. Proceedings of the Twelth national conference on Artificial intelligence, Pages: 722 – 727, 1994.

[2] Requirements Validation via Automated Natural Language Parsing (Nanduri & Rugaber, 1995).

[3] WordNet(2.1)http://www.cogsci.princeton.eu/~wn/.

[4] "Static UML Model Generator from Analysis of Requirements (SUGAR) " 2008 IEEE by Deva Kumar, Ratna Sanyal.

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

[6] Song, Il-Yeol, et al, (2004). "A Taxonomic Class Modeling Methodology for Object-Oriented Analysis", In Information Modeling Methods and Methodologies, Advanced Topics. In Databases Series, Ed, pp. 216-240. Idea Publishing Group. http://www.ischool.drexel.edu/faculty/song/ publications/p_TCM-ISM-2004.pdf.

[7] OpenNLP: http://opennlp.sourceforge.net/

[8] Tobias Karlsson, 2004, "Managing large amounts of natural language requirements through natural language processing and information retrieval support" ,Master's Thesis, Department of Communication Systems, Lund Institute of Technology, Lund University.

[9] "On the Systematic Analysis of Natural Language Requirements with **CIRCE**" in 2006 Springer Science + Business Media, Inc. by VINCENZO AMBRIOLA,VINCENZO GERVASI (Dipartimento di Informatica Universit`a di Pisa, Italy).

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

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

[12] "CM-Builder An Automated NL-based CASE Tool", H.M. Harmain Dept. of Computer Science University of Sebha , Libya & R. Gaizauskas Dept. of Computer Science University of Sheffield, UK.

[13] Ambriola, V. and Gervasi, V. "Processing natural language requirements", Proc. 12th IEEE Intl. Conf. on Automated Software Engineering, pp. 36-45,1997.

[14] "Conceptual Modeling through Linguistic Analysis Using LIDA", Overmyer1, Scott P & Lavoie, Benoit, Rambow2, Owen.

[15] Kalaivani Subramaniam, Dong Liu, Behrouz H. Far and Armin Eberlein, "UCDA: Use Case Driven Development Assistant Tool for Class Model Generation", Department of Electrical and Computer Engineering, University of Calgary.

[16] European Journal of Scientific Research "Object Oriented Software Modeling Using NLP Based Knowledge Extraction" published by Imran Sarwar Bajwa, Ali Samad and Shahzad Mumtaz.

[17] "Relative Extraction Methodology for Class Diagram Generation using Dependency Graph" published by Hema Krishnan and Philip Samuel.

[18] "Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology" Published by Xiaohua Zhou, Nan Zhou.