ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 8, August 2014

Textual Requirement Analysis for UML Diagram Extraction

Prof. Dhanraj Deshpande, Kashmira Dikshit

Asst Professor, Dept. of Computer Engg, GES RH Sapat COE, Nashik, India

dhanraj.deshpande@gmail.com

B.E Student, Dept. of Computer, GES RH Sapat COE, Nashik, 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 requirements model. The automatic generation of UML diagram from natural language requirements is highly challenging. 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. 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).

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

I. INTRODUCTION

In object-oriented systems, the notion of class is carried over from analysis to design, implementation, and testing. Thus, finding a set of domain classes is the most important skill in developing an object-oriented system. Auto-generation of class diagram will without doubt save the time and effort of system analyst, especially for novice. However, the automation of class generation from a written source is highly challenging due to the following three reasons (Richter, 1999; Maciaszek, 2001).

- Natural language is ambiguous. Thus, rigorous and precise analysis is very difficult
- The same semantics could be represented in different ways.
- Concepts that were not explicitly expressed in a written source are often very difficult to model. Usually, expert domain knowledge is needed to identify the hidden classes.

As far as the best knowledge to us, the previous researchers have not offered any sound solutions to above-mentioned problems yet. Our approach proposed in this paper tries to address the problem mentioned but not intends to resolve all. The approach first applies natural language processing (NLP) techniques to understanding of the written requirements, and then uses domain knowledge represented by domain ontology to improve the performance of class identification. The basic notion of our approach is based upon the belief from heuristics that a class especially core class of the domain is always semantically connected with other classes and its attributes. The novice part of this approach is the use of a NLP-based RAUE model to identify core classes of the domain as starting point, for which we are usually highly confident, and further find classes that are related with the identified ones.

This methodology use NLP techniques to address the following challenging research questions: (1) How to use domain ontology but not limited to domain ontology? (2) How to find candidate concepts? (3) How to find concept pairs with strong semantic connection in the context? (4) How to distinguish attribute name from class name for each concept? (5) How to name each inter-class relationship, aggregation, generalization, or association? (6) Are there any missing classes or attributes?

In First and Second Section provides background information and describes related work. Section 4, 5 and 6 describes in detail our fully implemented NL-based RAUE tool which analyses software requirements written in

English and produces first-cut class models represented in the UML. In last section, it presents conclusions anddiscusses future work.

II. RELATED WORKS

There are several approaches for identifying classes, among which the noun analysis is the most popular one (Abbot, 1983; Chen, 1983; Rumbaugh, Blaha, &Premerlani, 1991; Booch, 1994; Richter,1999; Rosenberg, 1999). Other methods use of class categories as tips (Booch,1994; Rumbaugh, Jacobson, &Booch, 1999; Starr, 2001; Larman, 2001), use case descriptions (Jacobson, 1992; Richter, 1999), or CRC (Class-Responsibilities-Collaborators) cards (Wirfs-Brock, Wilkerson, & Wiener, 1990).

Also there have been several efforts for the analysis of natural language requirements [9,10, 11,16]. However, few are focused on class diagram extraction from natural language (NL)requirements. Thus, few tools exist to assist analysts in the extraction of class diagram. In this section we survey theworks that use NLP or domain ontology techniques to analyze NL requirements.

Ambriola and Gervasi[10] present a Web-based environment called Circe. Circe helps in the elicitation, selection, and validation of the software requirements. It canbuild semi-formal models, extract information from the NL requirements, and measure the consistency of these models.It integrates anumber of tools likeCico[10].It is the main tool that is considered as a front-end and it recognizes the NLsentences and extracts some facts from them.

Zhou and Zhou [17] propose a methodology that uses NLPand domain ontology. It is based on that the core classesare always semantically connected to each other's by one toone, one to many, or many to many relationships in thedomain. This methodology finds candidate classes usingNLP through a part of speech (POS) tagger, a link grammarparser, linguistic patterns and parallel structure, and then thedomain ontology is used to refine the result [17].

Mich L. [18] proposes a NLP system, LOLITA to generatean object model automatically from natural language requirement. It considers nouns as objects and it uses links to findrelationships between objects. LOLITA system does not distinguishbetween classes, attributes, and objects. This approach islimited to extract objects and cannot identify classes [4].

Taxonomic Class Modeling (TCM) methodology proposed by Song (Song et al., 2004) is the most recent and most complete work of identifying classes to our best knowledge. Starting with problem statement, it incorporates the noun analysis, class categories, English sentence structure rules, checklists, and other heuristic rules for modeling. TCM tries to identify three types of classes: noun classes, transformed classes (from verbs), and discovered classes.

Based on this survey, we recognize the TCM approach in [12] as the most successful model – for the extraction of class diagram from NL requirements.

III. 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.

IV. THE RAUE APPROACH.

RAUE follows the Object- Oriented Analysis and Design (OOAD) [10] 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.

A. Theoretical Foundations

The Noun-Phrase technique of RAUE is aimed at helping in developing the analysis class model. This technique helps a requirements analyst to identify all possible objects from a given requirements document and generate analysis class model by attaching attributes and methods with the associated object. This technique categorizes a list of nouns into three classes, namely relevant classes, fuzzy classes and irrelevant classes. Fuzzy classes are further sub-divided into adjective, attribute and redundant classes.

However, the TCM mentioned is a general framework as it doesn’t clearly show the aspects of using NLP. The aim of our approach is to efficiently apply NLP and domain ontology techniques to achieve a fast and accurate analysis result. Figure 1 illustrates the process model.

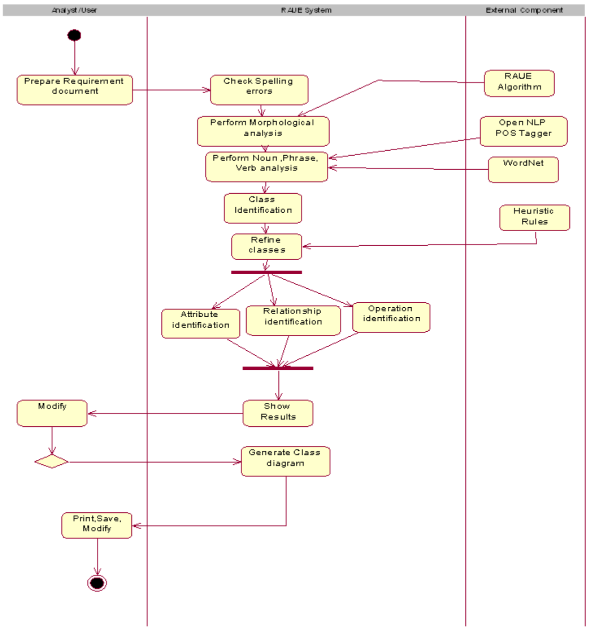

Figure 1.The Process Model of RACE Tool (Activity Diagram).

B. OpenNLP Parser

We choose OpenNLP[13] as a parser in our system. The OpenNLP Parser library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP is an open-source and re-usable algorithm. It provides our system with lexical and syntactic parsers. OpenNLP is an open-source and re-usable algorithm. It provides our system with lexical and syntactic parsers. 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.

C. WordNet

WordNet [13] 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 [11]. WordNet can used to find semantically similar terms, and for the acquisition of synonyms [14]. We used synonyms to extract words which are semanticallyrelated to each other. We calculated the words frequency to keep the synonyms with high frequency in the document.

V. RAUE FRAMEWORK AND DESIGN

A. NLP-based Spider Model:

In this section, we propose the algorithm for automatic generation of class diagrams. The input files include free-text functional specifications and structured domain ontology while the output is class diagram components including classes, attributes of each class, and inter-class relationships. The inter-class relationships can be classified into generalization, aggregation, and association. The association relationships can be further classified into one-to-one, one-to-many, and many-to-many. Our approach applies a NLP-based spider model to search classes of interest.
Figure 2 illustrates the framework model of RAUE:

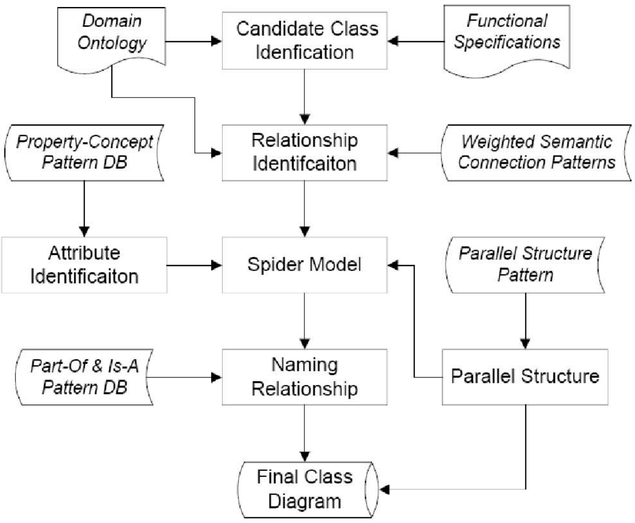
Domain Ontology
Candidate Class Idenfication
Functional Specifications
Property-Concept Pattern DB
Relationship Identifcaiton
Weighted Semantic Connection Patterns
Attribute Identificaiton
Spider Model
Parallel Structure Pattern
Part-Of & Is-A Pattern DB
Naming Relationship
Parallel Structure
Final Class Diagram

Figure 2: Framework of NLP-based Spider Model.

B. The RAUE stemming algorithm:

Stemming is a technique that abbreviates word by removing affixes and suffixes [14]. 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.
The **algorithm** is presented below.

Steps:

Step1: Use the requirements document as input.
Step2: Identify the stop words and save the result as {Stopwords_Found} list.
Step3: Calculate the total number of words in the documents without the stop words, the number of occurrences of each word, and then calculate the frequency Ƒ of each word, as in Ƒ
Step4: Use RACE stemming algorithm module to find the stemming for each word and save the result in a list.
Step5: Use OpenNLP parser in [13] to parse the whole document (including the stop words)
Step6: Use the parser output to extract Proper Nouns (NN), Noun phases (NP), verbs (VB). And save it in {Concepts-list} list.
Step7: Use Step2 and Step 6 to extract: {Noun phrases (NP)} – {Stopwords_Found} and
save results to {Concepts-list}
Step8: For each concept (CT) in {Concepts-list} if {synonyms_list} contains a concept (CT2) which have a synonym (SM) which lexically equal to CT, then CT and CT2 concepts are semantically related to each other.

Step9: For each concept (CT) in {Concepts-list} if {hypernyms_list} contains a concept (CT2) which have a hypernyms (HM) which lexically equal to CT, then CT2 "**is a kind of"** CT. Then save result as {Generalization-list}.

C.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 weorganize 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.

D.Candidate Classes Identification

Most previous works think that sources of candidate class name are major noun phrases and minor verbs. By using part of speech (POS) tagger, we obtain a list of verbs and nouns. However, nouns are not equal to noun phrases in many cases. We further use sentence parser to get noun phrases of interest. Here we only consider nouns with pre-noun modifiers like NN+NN and JJ+NN. After POS tagging and sentence parsing, we get a set of noun phrases and verbs, namely Preliminary Candidates. Most preliminary candidates are not good classes, and are even irrelevant to the domain at all. The domain ontology offers the chance to further refine the classes for the domain model.

E.Relationship Identification

Relationship identification refers to discover any semantic concept pairs within a sentence. The relationship of concept pair can be roughly classified into three categories: (1) relationship of two classes, (2) attribute of a class, and (3) value of an attribute. The relationship of two classes can be further classified into generalization, aggregation and association.

The association relationship can be either one-to-one, one-to-many, or many-to-many. However, in this step we don't have to specify the name of the relationship, but to determine whether two concepts have relationship and how strong the relationship is if any.Because it is difficult to directly test whether two concepts are semantically connected with each other in the context, we use linkage distance to measure the semantic connection of two concepts.Linkage is a term defined in the famous parser, Link OpenNLP Parser.

F.Attribute Identification

When a concept is found to have strong semantic association with an existing class, it is still needed to tell whether the concept is an attribute of the associated class or another independent class. Usually, if there are more than one property associated with the concept, it is a class otherwise an attribute. For example, if only the amount of the price is necessary to remember, the price serves as an attribute, but if it is necessary to remember the amount, the discount, and the effective date range, it might well be a class.Property and concept is a special subordinate-master relationship in terms of semantics. There are strongly indicative linguistics patterns for such relationship.

G. Naming Relationship

In the standard of Unified Modeling Language (UML), inter-class relationship can be classified into aggregation, generalization and association. Association relationship can be either one-to-one, one-to-many or many-to-many.Aggregation is a part-whole relationship while generalization is a general-special relationship. Both two types of relationships have strong linguistic patterns associated with them. Many previous researches on this topic are well done so that we simply follow their methods (Berlan&Charniak, 1999; Hearst, 1992). However, we adopt a conservative strategy for the identification of aggregation and generalization relationship in the paper. In addition to the criterion of linguistic patterns described in previous works, they must meet two additional criterions:

(1) The parts of the composite class and the child classes of the super class must be appeared in the parallel structure.

(2) At least one of the elements in the parallel structure has part-of or is-a relationship with the super class by looking up WordNet.

For examples:

A) The customer can create more than one account. The customer and account have a one-to-many relationship since "more than one" modifies "account".

B) Each customer can keep only one registered credit card in the profile. Here customer and credit card are one-to-one relationship because "only one" modifies "credit card".If the concept A and concept B are one-to-many relationship and vice versa, the two concepts have a many-to-many relationship.

H. Parallel Structure.

Our algorithm fully uses the domain knowledge represented by domain ontology to conceptually model the system. However, for the purpose of generalization, the ontology usually contains the general knowledge only. Ontology often covers core or important concepts in the domain, and important attributes of the concepts. Thus it doesn't exhaust all concepts and attributes for a specific system. Fortunately, parallel structure within an English sentence helps us find missed concepts or attributes. In detail, parallel structure can find (1) rest parts of a composite, (2) rest children of a super concept, and (3) rest attributes of a class.

VI. RAUE IMPLEMENTATION

The implementation of the system involves not only complex algorithm described in section 3, but also many external programs and resources. We choose Eric Brill's rule-based transformation tagger for part-of-speech tagging (Brill, 1994), and pick up Link Grammar as sentence parser because linkage information is very important to our approach. Meanwhile, WordNet serves as lexicon resources in our system. Though Brill's tagger, Link Grammar Parser and WordNet are all written in native code, we find third party Java Native Interfaces (JNI) for these programs. Therefore, we can still integrate them with our main application written in Java well. Since all external programs support Windows, Linux and UNIX, and Java is platform independent, our system suppose to be able to run in different platforms though we develop and test it on Windows platform only. The architecture of the system is presented in figure 3.

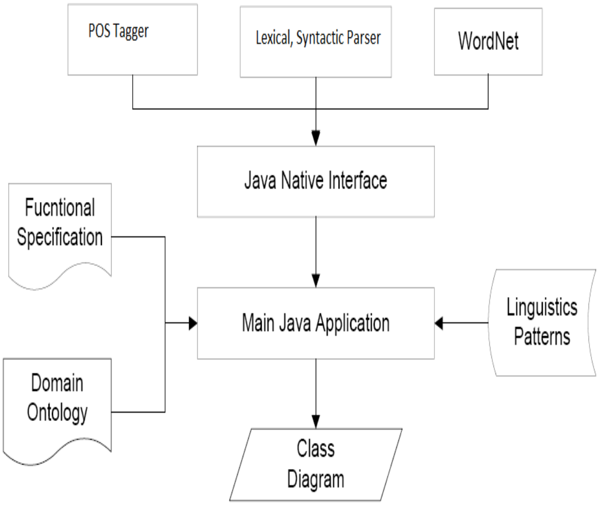

Figure 3: RAUE System Architecture

Theoretically the system can accept any free-text document describing functional specifications. The version implemented here would have best performance to deal with problem statement and use case description written in natural language.

RAUE can open textual requirements from different sources including words documents (DOC), text files (TXT), rich text files (RTF), and hypertext document (HTML). The Class diagrams are visually represented. In addition, system can highlight nouns and verbs, in the document. For a good consistency, we use Threads to run different process at the same time. In the current version of RAUE, we use MsAccess to manage RAUE databases. RAUE supports two interfaces languages which are English. Our approach also needs external linguistic patterns including weighted semantic connection pattern, property-concept pattern, parallel structure pattern, part-of pattern, and is-a pattern.

VII. CONCLUSION

In this paper, we propose an enhanced approach that isbased on NLP and domain ontology techniques to support the extraction of class diagram from NL requirements. Wevalidate our approach by implementing a system called RAUE referred to as "Requirements Analysis and UMLdiagram Extraction". RAUE system efficiently demonstrates the using of NLP and domain ontology techniques in theextraction of class diagram from informal requirements. The concepts extracted by our system are completely valid andrecognized in the application domain. RAUE system able to finds concepts based on nouns, Noun phrases and verbs analysis. It also able to find four types of relationships:Generalization, Association, Composition, aggregation and dependency. The class diagram can be graphicallyrepresented and modified.RAUE provides a human-centered UI which makes user a part of the analysis process.

REFERENCES

[1]. Booch, G. (1994). Object-Oriented Analysis and Design with Applications, 2nd Ed., Benjamin Cummings.
[2]. Dori, D., et al, (2004). SMART: System Model Acquisition from Requirements Text. BPM 2004, LNCS 3080, pp. 179-194.
[3]. Berlan, M. &Charniak E., (1999). Finding Parts in Very Large Corpora.In Proceedings of the the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99).
[4]. Brill E., Some Advances in Transformation-Based Part of Speech Tagging. Proceedings of the Twelth national conference on Artificial intelligence, Pages: 722 – 727, 1994.
[5]. Requirements Validation via Automated Natural Language Parsing (Nanduri&Rugaber, 1995).
[6]. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., Object-oriented Modeling and Design, Prentice Hall, 1991.
[7]. Goldin, L. and Berry, D.M., (1997). A prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation, Automated Software Engineering Journal 4, (4) 375-412.
[8].WordNet(2.1)http://www.cogsci.princeton.eu/~wn/. Haruhiko Kaiya, Motoshi Saeki, 2005, "Ontology Based
[9]. "Static UML Model Generator from Analysis of Requirements (SUGAR)" 2008 IEEE by Deva Kumar, RatnaSanyal.
[10]. Ambriola, V. and Gervasi, V. "Processing natural languagerequirements", Proc. 12th IEEE Intl. Conf. on Automated SoftwareEngineering, pp. 36-45,1997.
[11].FaridMeziane, Nikos Athanasakis, Sophia Ananiadou, 2007,Generating Natural Language specifications from UML classdiagrams, Springer-Verlag London Limited 2007
[12]. Song, Il-Yeol, et al, (2004). "A Taxonomic Class ModelingMethodology for Object-Oriented Analysis", In InformationModeling Methods and Methodologies, Advanced Topics.InDatabases Series, Ed, pp. 216-240. Idea Publishing Group.http://www.ischool.drexel.edu/faculty/song/publications/p_TCM-ISM-2004.pdf.
[13].OpenNLP: http://opennlp.sourceforge.net/
[14]. Tobias Karlsson, 2004, "Managing large amounts of natural languagerequirements through natural language processing and informationretrieval support" ,Master's Thesis, Department of CommunicationSystems, Lund Institute of Technology, Lund University.
[15].WordNet(2.1)http://www.cogsci.princeton.eu/~wn/.
[16]. "On the Systematic Analysis of Natural LanguageRequirements with **CIRCE**" in 2006 Springer Science + Business Media, Inc. by VINCENZO AMBRIOLA,VINCENZO GERVASI (DipartimentodiInformaticaUniversit`adi Pisa, Italy).
[17]. Xiaohua Zhou and Nan Zhou, 2004, Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology.
[18]. L. Mich, NL-OOPs: "From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA.", *Natural Language Engineering,1996,pp.161-187.*