International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 1– No.8, April 2012 – www.ijais.org

Generating UML Diagrams from Natural Language Specifications

Priyanka More
Department of IT
MIT College of Engineering University of Pune,Pune -45

Rashmi Phalnikar
Department of IT
MIT College of EngineeringUniversity of Pune, Pune -45

ABSTRACT
The process of generating UML Diagrams from natural language specification is a highly challenging task. This paper proposes a method and tool to facilitate the requirements analysis process and extract UML diagrams from textual requirements using natural language processing (NLP) and Domain Ontology techniques. Requirements engineers analyze requirements manually to understand the scope of the system. The time spent on the analysis and the low quality of human analysis justifies - the need of a tool for better understanding of the system. "Requirement analysis to Provide Instant Diagrams (RAPID)" is a desktop tool to assist requirements analysts and Software Engineering students to analyze textual requirements, finding core concepts and its relationships, and extraction UML diagrams. The evaluation of RAPID system is in the process and will be conducted through two forms of evaluation, experimental and expert evaluation.

General Terms
Software Engineering

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

1. INTRODUCTION
Software requirements are often specified in natural language (NL). These NL requirements are typically coming from a pool of natural language statements which are gathered from interview excerpts, documents and notes [1]. However, requirements specified in NL can often be ambiguous, incomplete, and inconsistent. Moreover, the interpretation and understanding of anything described in NL has the potential of being influenced by geographical, psychological and sociological factors. For this reason, Informal natural language requirements are better to be expressed as formal representations [1]. It is the job of requirements analysts to detect and fix any potential ambiguities, inconsistencies, and incompleteness in the requirements specifications documents.

However, human reviewers can overlook some defects while reading complex NL descriptions which can lead to multiple interpretations and difficulties in recovering implicit requirements when the requirement analyst does not have extensive domain knowledge [2]. UML class diagrams are the main core of OO analysis and design systems where most other models are derived from [3]. Natural language processing (NLP) is recognized as a general assistance in analyzing requirements [5]. The NLP systems use different levels of linguistic analysis: Phonetic (phonological) level, Morphological level, Lexical level, Syntactic level, Semantic level, Discourse level and Pragmatic level [7, 9]. In addition to NLP techniques, Domain Ontology has been widely used to improve the performance of concept identification. Domain ontology refers to domain knowledge that consists of structured concepts which are semantically related to each other.

The aim of this paper is to demonstrate the use of NLP and domain ontology techniques for the extraction of UML diagrams from informal natural language requirements by implementing a prototype tool that uses the mentioned techniques. The proposed tool is referred to as **Requirement analysis to Provide Instant Diagrams (RAPID)**. The RAPID tool assists analysts by providing an efficient and fast way to produce the class diagram from their requirements. It supports a good interaction with users by providing a modern and human-centered user interface.

1.1 Related Work
There have been several efforts for the analysis of natural language requirements [4, 8, 9]. 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 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.

Deva Kumar [2 ,3] propose a domain independent tool, named, UML Model Generator from Analysis of Requirements (UMGAR), which generates UML models like the Use-case Diagram, Analysis class model, Collaboration diagram and Design class model from natural language requirements using efficient Natural Language Processing (NLP) tools. UMGAR implements a set of syntactic reconstruction rules to process complex requirements into simple requirements. UMGAR also provides a generic XMI parser to generate XMI files for visualizing the generated models in any UML modeling tool. With respect to the existing tools in this area, UMGAR provides more comprehensive support for generating models with proper relationships, which can be used for large requirement documents.

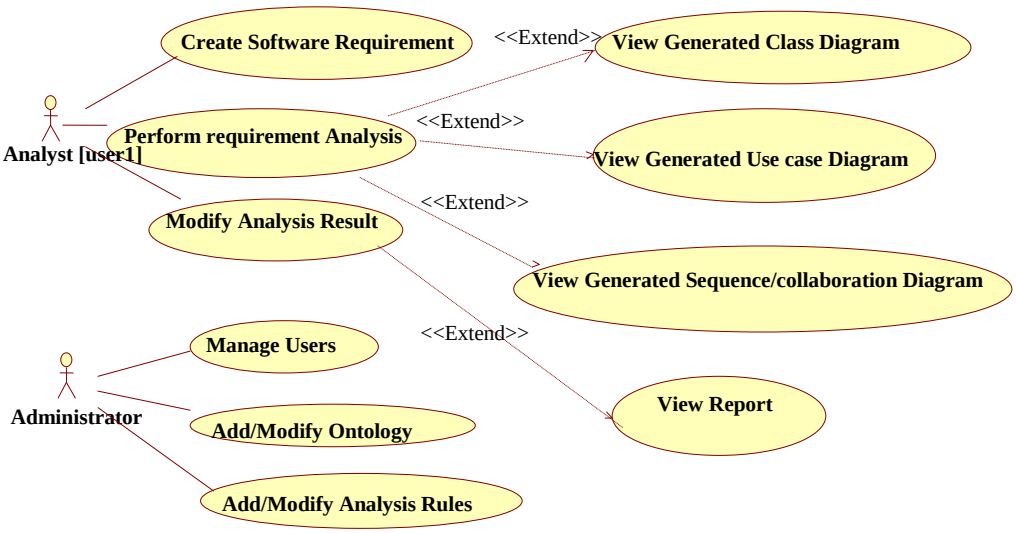

Fig 1 Use Case Diagram for RAPID System

Ambriola and Gervasi [4] 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 [4] 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.

Zhou and Zhou [10] propose 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 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 [8].

Mich L. [11] 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 [4].

Song et al. [12] propose a taxonomic class modeling (TCM) methodology for object-oriented analysis which incorporates several modeling rules such as noun analysis, English sentence structure rules, class categories, checklists and other heuristic rules. The TCM methodology works as follows. First, it finds candidate classes using noun analysis. Then the spurious classes are eliminated using class elimination rules. After elimination, the hidden classes are discovered using pre-defined class categories. Finally, the list is reviewed using domain knowledge. This survey reflects the current stage of using NLP techniques for analyzing NL requirements, the current stage of using domain ontology to express an application domain related to NL requirements, and the current stage of class diagram extraction from NL requirements.

1.2 Proposed approach for RAPID

In the previous section, we have reviewed the most recent works. Meanwhile, we recognized a typical class Identification framework and adapted the RACE of Mohd Ibrahim et al [1] to derive our process model.

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 use cases of our system. Further elaborations will be under sections 3.4 and 3.6.

2. RAPID ARCHITECTURE AND DESIGN

RAPID system is decomposed into internal and external components and sub-systems. Figure 2 illustrates the architecture model of RAPID

2.1 Normalizing requirements component (NLP Tool Layer)

This component aims at normalizing NL requirements to remove ambiguous requirements and identify incomplete requirements. This component consists of the following sub components:

2.1.1 Syntactic Reconstruction: The tool takes stakeholder's requests as input and performs syntactic reconstruction to split a complex sentence into simple sentences to extract all possible information from the requirements document. We have defined 8 syntactic reconstructing rules that have been implemented in UMGAR [2].

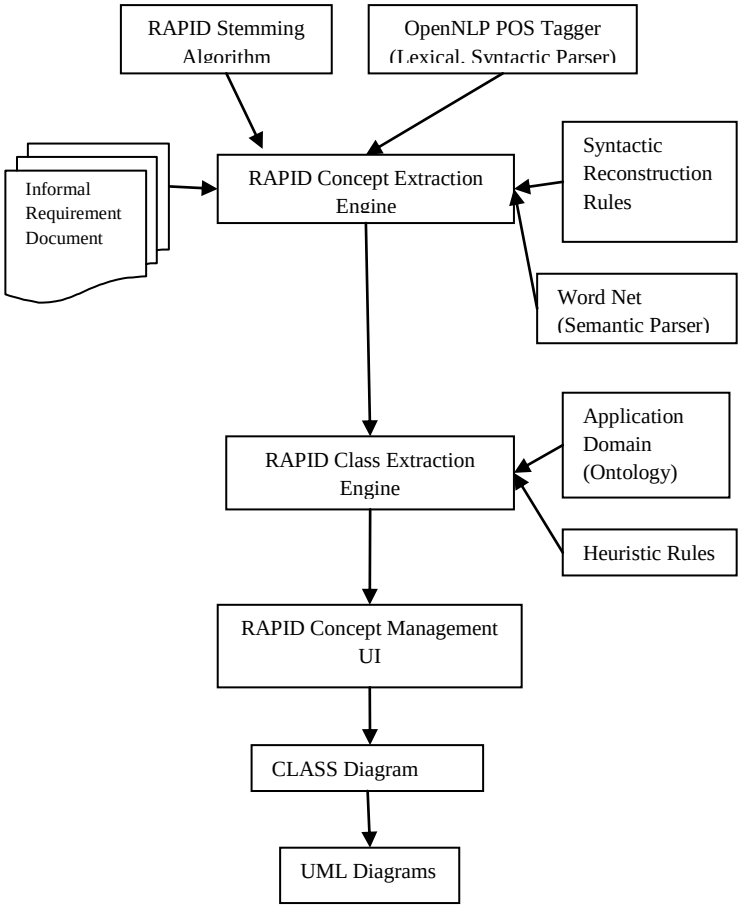

Fig 2 RAPID System Architecture

The tool scans each sentence to test whether that requirement satisfies the Statement sentence structure, which is of the form "Subject: Predicate" or "Subject: Predicate: Object", and applies rules accordingly. If a sentence does not satisfy the proposed rules, then it prompts a message to the user to change the sentence accordingly to the statement structure.
Some basic rules for syntactic reconstruction are as follows [3]:

1. Discard prepositional phrase (PP), adjective phrase (ADJP), determiner (DT) or adjective (JJ), if they precedes the subject of the sentence.
2. If NP and VP is preceded by "No", then convert it into "NP not VP".
3. Noun phrases (NP) which are separated by connectives like "and, or" are taken as individual sentences. If {{NP1}{VP1{ VBZ NP2,NP3 and NP4}}} then convert it into {{NP1}{VP1{ VBZ NP2 }}}, {{NP1}{VP1{ VBZ NP3}}}, {{NP1}{VP1{ VBZ NP4}}}.
4. Sentences which are connected by connectives like "and, or, but, yet" are spitted at their connectives and created at two individual sentences. If sentence1 and/or sentence2, then convert it into two sentences {sentence1} {sentence2}.
5. If a sentence has no verbs (VP) then discard that sentence.
6. If a sentence is of the form {{NP1} {VP1 {NP2} {VP2 {NP3}}}}, then convert it into two sentences like {{NP1} {VP1 {NP2}}} and {{NP2} {VP2 {NP3}}}.
7. In the Sentences which are having a semicolon, treat the sentence after the semicolon as extra information for the preceding sentence and so discard sentence after semicolon.
8. If a sentence is in passive voice, ask user to convert it into active voice. Normally passive voice sentences will contain word "be" which gives the sense as passive voice form. This needs some user interference to decide which sentence acts as passive voice.

2.2 NLP Technologies Used

The following are the NLP tools used for developing RAPID:

2.2.1 *OpenNLP Parser:* We chose OpenNLP [19] 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 Chunkier (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.

2.2.2 RAPID Stemming Algorithm: Stemming is a technique that abbreviates word by removing affixes and suffixes [14]. In RAPID 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 using C#. 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 RAPID system. Our stemming algorithm is simple and re-usable.

2.2.3 *Word Net:* Word Net [15] is used to validate the semantic correctness of the sentences generated at the syntactic analysis. It also enables users to display all hyponyms 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 [5]. Word Net 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.

2.2.4 *Concepts Extraction Engine:* The aim of this module is to extract concepts according to the requirements document. This module uses OpenNLP parser in [11], RACE stemming algorithm, and Word Net in [13], to extract concepts related to the given requirements. We illustrate the algorithm of this module by the following steps [1]:

- **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 Ol of each word, and then calculate the frequency F of each word, as in

  $F = Ol / \sum$
- **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 [11] 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 hyponyms (HM) which lexically equal to CT, then CT2 "is a kind of"CT. Then save result as {Generalization-list}.

2.2.5 *Domain Ontology:* As mentioned early in this paper, domain ontology is used to improve the performance of concepts identification. We used the XML to build the ontology.

2.2.6 *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 used as the following.

2.2.6.1 *Class Identification Rules [1]:* At the first step, concepts that extracted using the 'Concepts Extraction Engine" module will be used as the input and the following rules will be applied to extract classes [1].

- **C-Rule1:** If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as class.
- **C-Rule2:** If a concept is related to the design elements then ignore as class. Examples: "application, system, data, computer, etc…"
- **C-Rule3:** If a concept is related to Location name, People name, then ignore as a class. Examples: "John, Ali, London, etc…"
- **C-Rule4:** 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."
- **C-Rule5:** If a concept is an attribute, then ignore as a class. Examples: "name, address, number"
- **C-Rule6:** If a concept does not satisfy any of the previous rules, then it's most likely a class.
- **C-Rule7:** 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: "Customer Name" or "Book ISBN"
- **C-Rule8:** if the ontology (if-used) contains information about the concept such as relationships, attributes, then that concept is a class.

2.2.6.2 *Attribute Identification Rules [1]:* We use the following rules for attributes identification.

- **A-Rule1:** 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".
- **A-Rule2:** If a concept can has one value, then it's an attribute. Examples:"name, date, ID, address". Based on A-Rule2, we collected and stored a predefined list including the most popular attributes to be used as a reference in RACE system.

2.2.6.3 Relationship Identification Rules [1]:
Using verb analysis as input, y we can apply the following rules:

- **R-Rule1:** using step10 in the concept extraction engine (section 4.4), all the elements in the {generalization-list} will be transferred as Generalization (IS-A) relationship.

- **R-Rule2:** 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 and CT2 are classes, then (VB) is an Association relationship.
- **R-Rule3:** If the concept is verb (VB) 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.
- **R-Rule4**: If the concept is verb (VB) 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.
- **R-Rule5:** 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).
- **R-Rule6:** 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).

2.2.7 RAPID Concept Management (UI)[1]:

User interaction is a vital in RAPID system; RAPID includes an interactive user interface (UI) that manages the tasks such as creating, printing, saving and analyzing requirements. It also handles the graphical representation of the class diagram and let User add, delete, rename classes and relationships in the class diagram. As a part of RAPID UI, concept management UI is a very important interface which let user add, modify, view, and organize concepts and relationships. User can simply add new concept, change the concept type, and add new relationship. RAPID Concept management system gives user the flexibility to lead the processing in the way he/she wants.

3. RAPID IMPLEMENTATION

RAPID system interfaces and algorithms are implemented using C#. The External components are then added to the system and checked for consistency. The inconsistent and the incompatible components are re-implemented in C# to be conformed to our system. RAPID can open textual requirements from different sources including words documents (DOC), text files (TXT), rich text files (RTF), and hypertext document (HTML). The UML diagrams are visually represented. In addition, system can highlight nouns and verbs, in the document. For a good consistency, we use C# Threads to run different process at the same time. In the current version of RAPID, we use SQL SERVER to manage RAPID databases. RAPID supports one interface language which is English language.

4. ACKNOWLEDGMENTS

I thank my guide Prof. Rashmi Phalnikar, Assistant Professor, MITCOE, for her proper guidance, and valuable suggestions.

5. REFERENCES

[1] Mohd Ibrahim, Rodina Ahmad "Class diagram extraction from textual requirements using Natural language processing (NLP) techniques",IEEE journal 2010

[2] Deva Kumar Deeptimahanti, Muhammad Ali Babar "An Automated Tool for Generating UML Models from Natural Language Requirements" IEEE journal 2009

[3] Deeptimahanti Deva Kumar, Ratna Sanyal "Static UML Model Generator from Analysis of Requirements (SUGAR)" IEEE journal 2008

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

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

[6] Ke Li, R.G.Dewar, R.J.Pooley, a, 2003, "Requirements capture in natural language problem Statements"

[7] Elizabeth D. Liddy & Jennifer H. Liddy, 2001, "An NLP Approach for Improving Access to Statistical Information for the Masses".

[8] Gobinda G. Chowdhury , 2001, Natural Language Processing.

[9] Haruhiko Kaiya, Motoshi Saeki, 2005, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach", Proceedings of the Fifth International Conference on Quality Software (QSIC'05), 2005 IEEE

[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, 2(2), 1996, 87.

[12] Song, Il-Yeol, et al, (2004). "A Taxonomic Class Modeling Methodology for Object-Oriented Analysis", In Information Methods and Methodologies, Advanced Topics in Databases Series, Ed, pp. 216-240. Idea Publishing Group.

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

[14] 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,

[15] Word Net (2.1)http://www.cogsci.princeton.eu/~wn/.

[16] Jawad Makki, Anne-Marie Alquier, and Violaine Prince, 2008 Ontology Population via NLP techniques in Risk Management

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

[18] Ahmad Alsaadi ,"UML-Based Representation for Textual Objects".2008 IEEE.