Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques

Article · May 2010

DOI: 10.1109/ICCRD.2010.71

Rodina Ahmad
University of Malaya
101 PUBLICATIONS **742** CITATIONS

Class diagram extraction from textual requirements using Natural language processing (NLP) techniques

Mohd Ibrahim Al-Qaoud, Rodina Ahmad
Department of Software Engineering,
Faculty of Computer Science and Information technology,
University of Malaya, Malaysia
E-mail: m_qaoud@yahoo.com
rodina@um.edu.my

Abstract

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 NLP and Domain Ontology techniques. Requirements engineers analyze requirements manually to come out with analysis artifacts such as class diagram. The time spent on the analysis and the low quality of human analysis proved the need of automated support. A "Requirements Analysis and Class Diagram Extraction (RACE)" is a desktop instrument to assist requirements analysts and SE students in analyzing textual requirements, finding core concepts and its relationships, and step by step extraction of the class diagram. The evaluation of RACE system is in the process and will be conducted using two forms of evaluation, experimental and expert evaluation.

.

Keywords: Natural language processing (NLP), Domain Ontology, UML Class Diagram.

1. Introduction

The common way to express requirements is with large volumes of text [1] which can be referred to as natural language (NL) requirements. NL requirements are typically coming from a pool of natural language statements which are gathered from interview excerpts, documents and notes. Due to the inherent ambiguity of natural language, it is often difficult to prove properties on natural language requirements [2]. For this reason, Informal natural language requirements are better to be expressed as formal representations. Object-Oriented Analysis and Design (OOAD) has become a popular approach for software development since the 1990's [1]. 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 [4]. 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. [5, 6]. 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. Concepts and relationships in a real world can be represented in ontology which can be then used as a resource of domain knowledge. Using ontology allows several kinds of semantic processing to be achieved in requirements analysis process [7].

The aim of this paper is to demonstrate the use of natural language processing (NLP) and domain ontology techniques for the extraction of UML class diagram from informal natural language requirements by implementing a prototype tool that uses the mentioned techniques. The proposed tool is referred to as **Requirements Analysis and Class Diagram Extraction (RACE).** The RACE 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-centred user interface.

2. Related Work

There have been several efforts for the analysis of natural language requirements [2, 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

Ambriola and Gervasi [2] 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 text of the requirements, and can then measure and check the consistency of these models. Circe gives the user a complete environment that integrates a number of tools. Cico [2] is the main tool that is considered as a front-end for the other components. It recognizes the natural language sentences and extracts some facts from them. These facts are handed to the remaining tools for graphical representation and analysis.

Zhou and Zhou [8] propose a methodology that uses natural language processing (NLP) and a domain ontology. Their methodology is based on that the core classes are always semantically connected to each others 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]. This approach is implemented using a java program.

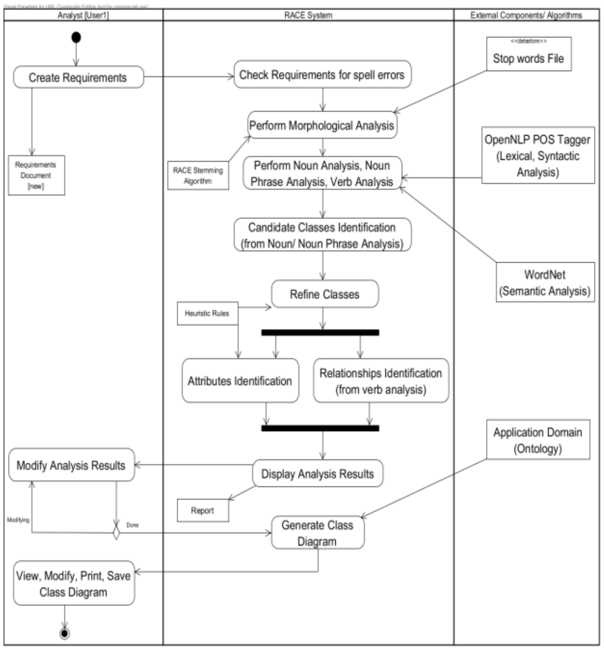
Mich L. [9] proposes a NLP system, LOLITA to generate an object model automatically from natural language. This approach considers nouns as objects and use links to find relationships amongst objects. LOLITA system is built on a large scale Semantic Network (SN) that does not distinguish between classes, attributes, and objects. This approach is limited to extract objects and can not identify classes [4].

Song *et al.* [10] 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 image of 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. The survey covered three groups of studies:
Group 1: which includes the studies in [2, 9]. These studies use NLP techniques in analyzing NL requirements. The class diagram is not the goal in this group.
Group 2: which includes the studies in [8,10]. These studies use domain ontology techniques with or without using NLP.
Group 3: which includes the studies in [2, 8, 10]. Class diagram extraction from natural language requirements is the goal of these studies.
Based on this classification, we recognize the TCM approach in [10] as the most successful model – surveyed- for the extraction of class diagram from NL requirements.

3. Proposed Approach for RACE

In the previous section, we have reviewed the most recent works. Meanwhile, we recognized a typical class identification framework and adapted the TCM of Song et al. [10] to derive our process model. 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 and 2 illustrates the process model and the use cases of our system. Further elaborations will be under sections 4.4 and 4.6


Analyst [User1]
RACE System
External Components/ Algorithms
<<datastore>>
Stop words File
Create Requirements
Check Requirements for spell errors
Perform Morphological Analysis
OpenNLP POS Tagger (Lexical, Syntactic Analysis)
Requirements Document [new]
RACE Stemming Algorithm
Perform Noun Analysis, Noun Phrase Analysis, Verb Analysis
Candidate Classes Identification (from Noun/ Noun Phrase Analysis)
WordNet (Semantic Analysis)
Refine Classes
Heuristic Rules
Attributes Identification
Relationships Identification (from verb analysis)
Application Domain (Ontology)
Modify Analysis Results
Display Analysis Results
Report
Modifying
Done
Generate Class Diagram
View, Modify, Print, Save Class Diagram
Figure 1: The Process Model of RACE Tool (Activity Diagram)

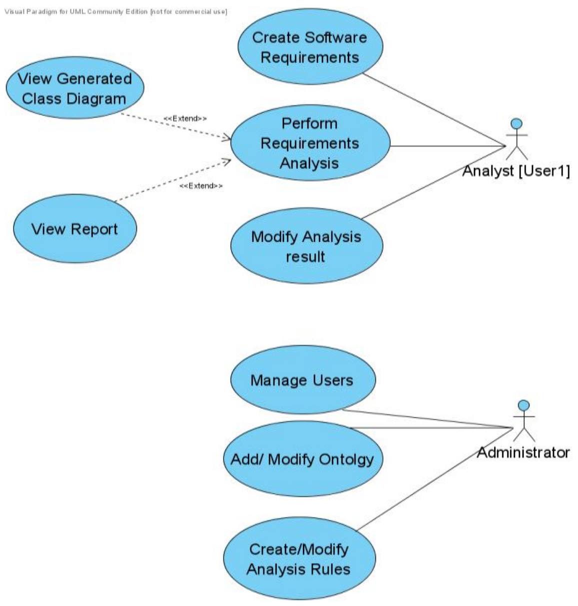

Visual Paradigm for UML Community Edition [not for commercial use]
Create Software Requirements
View Generated Class Diagram
<<Extend>>
Perform Requirements Analysis
<<Extend>>
Analyst [User1]
View Report
Modify Analysis result
Manage Users
Add/ Modify Ontolgy
Administrator
Create/Modify Analysis Rules

Figure 2: Use case diagram for RACE System

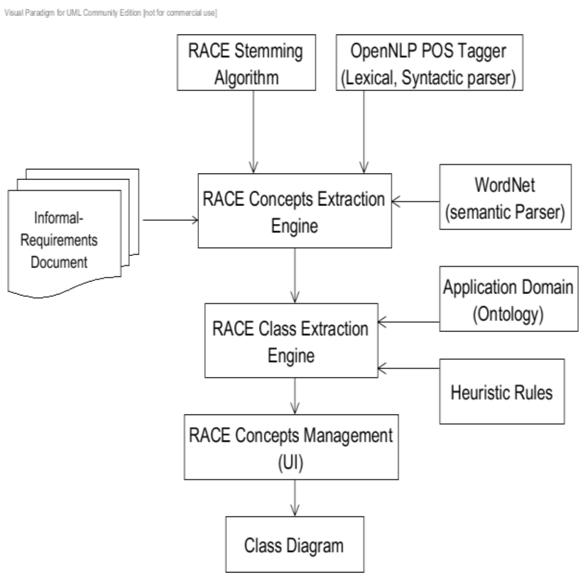

Visual Paradigm for UML Community Edition [not for commercial use]
RACE Stemming Algorithm
OpenNLP POS Tagger (Lexical, Syntactic parser)
Informal-Requirements Document
RACE Concepts Extraction Engine
WordNet (semantic Parser)
Application Domain (Ontology)
RACE Class Extraction Engine
Heuristic Rules
RACE Concepts Management (UI)
Class Diagram

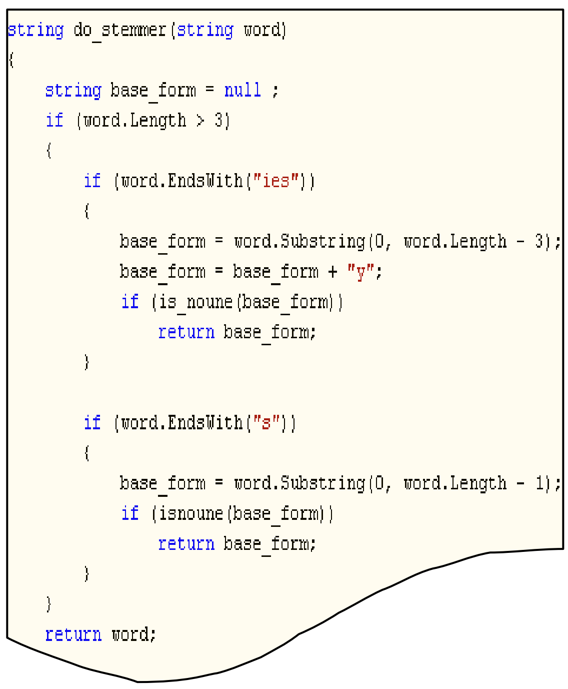
Figure 3: RACE System Architecture

4. RACE Architecture and Design

RACE system is composed into internal and external components and sub-systems. Figure 3 illustrates the architecture model of RACE:

4.1 OpenNLP Parser: We chose OpenNLP [11] 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 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.

4.2 RACE Stemming Algorithm: Stemming is a technique that abbreviates word by removing affixes and suffixes [12]. In RACE system, it is very important to return word 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 RACE system. Our stemming algorithm is simple and re-usable. Figure 4 shows a portion of code of RACE stemming algorithms:



```
string do_stemmer(string word)
{
    string base_form = null ;
    if (word.Length > 3)
    {
        if (word.EndsWith("ies"))
        {
            base_form = word.Substring(0, word.Length - 3);
            base_form = base_form + "y";
            if (is_noune(base_form))
                return base_form;
        }

        if (word.EndsWith("s"))
        {
            base_form = word.Substring(0, word.Length - 1);
            if (isnoune(base_form))
                return base_form;
        }
    }
    return word;
```

Figure 4: A portion of code of RACE stemming algorithms

4.3 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 [3]. WordNet can used to find semantically similar terms, and for the acquisition of synonyms [14]. 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.

4.4 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 WordNet in [13], to extract concepts related to the given requirements.. We illustrate the algorithm of this module by the following step:

Step1: Use the requirements document as input.
Step2: Identify the stop words and save the result as {Stopwords_Found}
Step3: Calculate # of words in the documents without the stop words. And Calculate the frequency of each word. Frequency (word) = # of occurrence (word) / Total # of words without stop words
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 hypernyms (HM) which lexically equal to CT, then CT2 "**is a kind of "** CT. Save result to {Generalization-list}

4.5 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.

4.6 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:

4.6.1 Class Identification Rules:

At the first step, concepts that extracted using the 'Concepts Extraction Engine" module in section (4.4) will be used as the input and the following rules will be applied to extract classes:

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 hypernyms tree, this indicate 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.

4.6.2 Attribute Identification Rules:

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.

4.6.3 Relationship Identification Rules:

Using verb analysis as input, 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).

4.7 RACE Concept Management (UI)

User interaction is a vital in RACE system; RACE 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 RACE 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. RACE Concept management system gives user the flexibility to lead the processing in the way he/she wants.

5. RACE Implementation

RACE 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. RACE 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, 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 RACE, we use MsAccess to manage RACE databases. RACE supports two interfaces languages which are English and Malay language.

6. Case Study

The goal of this case study is to test and validate our approach and to perform a general assessment of RACE tool accuracy and efficiency.

6.1 Library System Requirements

The library System is used by the Informatics students and Faculty. The Library contains Books and Journals. Books can be issued to both the Students and Faculty. Journals can only be issued to the Faculty. Books and Journals can only be issued by the Librarian. The deputy-Librarian is in-charge of receiving the Returned Books and Journals. The Accountant is responsible for receiving the fine for over-due books. Fine is charged only to students, and not to the Faculty.

6.2 Requirements Analysis Results

As appeared in the figure 5, the concept management UI displays the analysis results according to our approach in section 3 and 4. The system needs not more than 2 seconds to analyze up to 100 words requirements similar to library system requirements given in this test case. According to the library domain, the concepts founds during the analysis such as “Library”, “Book”, “Librarian”, etc... (As shown in figure 5) are all valid classes, which proves that our approach efficiently extract class diagram.

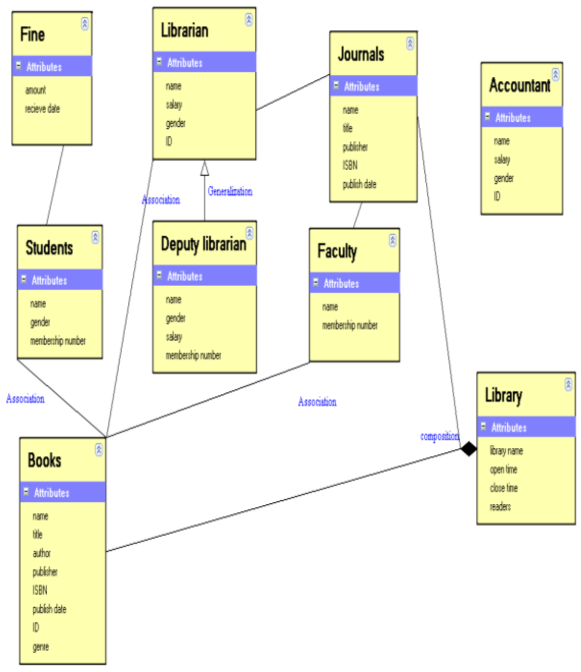


Figure 5: The Class diagram extracted by RACE

7. Conclusion and Future Works

In this paper, we propose an enhanced approach that is based on NLP and domain ontology 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 and domain ontology techniques in the extraction of class diagram from informal requirements. The concepts extracted by our system are completely valid and recognized in the application domain. RACE 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 graphically represented and modified. RACE provides a human-centred UI which makes user a part of the analysis process. At present, RACE does not support the following :

1) RACE couldn't identify one to one, one to many, and many to many relationships
2) RACE is restricted to windows platform and is not able to run in UNIX or other platforms.

In the future works, we will try to solve the mentioned challenges. Furthermore, we will add more ontolgies related to the information systems domains.

8. References

[1] Booch, G. (1994). 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. 36-45,1997*

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

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

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

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

[7] 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*

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

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

[10] 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.*

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

[12] 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.

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

[14] Jawad Makki, Anne-Marie Alquier, and Violaine Prince, 2008 Ontology Population via NLP techniques in Risk Management, ICSWE: Fifth International Conference on Semantic Web Engineering, Heidelberg, Germany , v.1