

NATURAL LANGUAGE BASED OBJECT-ORIENTED SOFTWARE MODELLING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

TARANNUM

Roll: 108CS005

AVISIKTA SAHOO

Roll: 108CS006



Department of Computer Science and Engineering
National Institute of Technology
Rourkela-769008, Odisha, India

NATURAL LANGUAGE BASED OBJECT-ORIENTED SOFTWARE MODELLING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

TARANNUM

Roll: 108CS005

AVISIKTA SAHOO

Roll: 108CS006

Under the Guidance of

PROF. S.K. RATH



Department of Computer Science and Engineering
National Institute of Technology
Rourkela-769008, Odisha, India



Department of Computer Science and Engineering

National Institute of Technology

Rourkela-769008, India, www.nitrkl.ac.in

CERTIFICATE

This is to certify that the thesis entitled “**Natural Language Based Object-Oriented Software Modelling**” submitted by Tarannum and Avisikta Sahoo, in partial fulfilment of the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela is an authentic work carried out by them under my supervision and guidance. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

DATE:

Dr. S. K. Rath

ACKNOWLEDGEMENT

We express our sincere gratitude to *Prof. S. K. Rath* for his motivation during the course of the project which served as an impetus to keep the work on schedule. We convey our regards to all the other faculty members of Department of Computer Science and Engineering, NIT Rourkela for their valuable guidance and advices at appropriate times. Finally, we would like to thank our friends and family for their help and assistance all through this project.

Tarannum

Roll: 108CS005

Avisikta Sahoo

Roll: 108CS006

Abstract

Deriving useful information from natural language has been a task of much relevance for fields ranging from machine translation, software modelling, and artificial intelligence and so on. Sufficient literature is available on utilisation of grammatical inference in object oriented software modelling. The major advancements in this field along with the challenges faced by researchers as well as practitioners have been outlined. An amalgamation of ideas taken from existing theories and models along with proposed methodology has been worked out so as to utilise natural language text in the field of object oriented analysis and design. The very first step of Natural Language (NL) text processing is Parts-of-Speech (POS) tagging. Grammatical rules, some already existing and some deduced through careful observation of NL structures has been extensively discussed and implemented. After appropriate tagging the words to their respective parts of speech the objective is to recognise the classes among them. The classes along with their attributes and methods were listed out. These classes essentially are identified as part of the major functionalities in an information system. The information system consists of requirement specification given by clients for their target software. Comprehending client specification is a time consuming process. Therefore analysing classes from the specification provided becomes mandatory. Several ambiguities and redundancy in class identification were faced and were effectively resolved. Final classes from the given requirement specification were found out. Subsequently the knowledge acquired from the same is put to use while analysing these functionalities through various UML (Unified Modelling Language) diagrams. There are several UML tools that serve the purpose of drawing these diagrams. But the motive is to make the entire process of deriving the UML diagrams in a logical and automated manner.

ABBREVIATIONS

NLP: Natural Language Processing.

POS: Parts of Speech.

OOA: Object-Oriented Analysis.

UML: Unified Modelling Language.

CASE: Computer Added Software Engineering.

SRS: Software Requirements Specification.

OOAD: Object-Oriented Analysis and Design.

SVO: Subject-Verb-Object.

LIST OF FIGURES

1.Flow diagram for Parts of Speech Tagging	15
2.Object-Oriented Modeling.....	19
3.Class Diagram for ATM specification	28
4.Class Diagram drawn based on the proposed Object Model.....	30
5.Screenshot of result obtained after POS Tagging	31
6.Screenshot of the result obtained after generation of Class Diagram in UMLet	32

CONTENTS

1. INTRODUCTION	2
1.1 BRIEF REVIEW	2
1.2 NATURAL LANGUAGE PROCESSING	2
1.2.1 PARTS OF SPEECH TAGGING	3
1.3 OBJECT-ORIENTED MODELLING	3
1.3.1 UML DIAGRAMS	4
2. LITERATURE REVIEW	8
2.1 EARLIER WORK ON NATURAL LANGUAGE PROCESSING	8
2.2 EARLIER WORK ON OBJECT-ORIENTED SOFTWARE MODELLING	8
2.3 UML TOOLS BASED ON NLP	10
3. PARTS OF SPEECH TAGGING	12
3.1 INTRODUCTION	12
3.2 HIDDEN MARKOV MODEL	13
3.3 RULES USED	13
3.4 WORKFLOW	14
4. OBJECT-ORIENTED MODEL	17
4.1 PROPOSED SYSTEM	17
4.2 NEED FOR IMPROVEMENTS	17
4.3 FURTHER MODIFICATIONS	18
4.3.1 NL-TEXT PROCESSING	19

4.3.2 FINALISING CLASSES.....	20
4.3.3 GENERATION OF CLASS DIAGRAMS.....	20
5. DESIGN AND ALGORITHM.....	23
5.1 ALGORITHM	23
5.2 DESCRIPTION AND ANALYSIS OF THE ALGORITHM.....	24
5.3 IMPLEMENTATION AND DISCUSSION.....	26
6. CONCLUSION.....	33
7. BIBLIOGRAPHY	34

Chapter 1

INTRODUCTION

Chapter 1

1 INTRODUCTION

1.1 BRIEF REVIEW

For development of any software, proper outline of client requirements is highly essential. The motive is to analyse the major functionalities of proposed Information System. Any given Software Requirements Specification (SRS) can be comprehended by humans but this process becomes very time-consuming when big projects or software is taken into consideration. So Object-Oriented Modelling is taken up for dynamic generation of the object model using just the SRS as an input to the system. Researchers have categorised two main approaches for this purpose:

- 1) Traditional approach
- 2) Object oriented approach (also referred as OOA)

The traditional approach focuses mainly on the identification of major functions of the system, but the object-oriented approach takes into consideration the objects, the static entities of the system and the associations among them. OOA applies the object oriented paradigm to model information systems by defining classes, objects and the relationships between them.

1.2 NATURAL LANGUAGE PROCESSING

The major challenge in the design of software is the ability to comprehend tedious, long-drawn-out user requirements as outlined by the clients. The software analysis if done precisely saves a lot of time of the system analyst and design phase can be started right away. In the field of information technology, there has been innumerable change in the way this

problem has been tackled. Though there are many traditional approaches which aim at recognising the functionalities of the information system, the modern object-oriented approach based on Natural Language Processing has garnered maximum popularity because of its strong role in object oriented modelling [12].

1.2.1 PARTS OF SPEECH TAGGING

Parts of Speech tagging, commonly known as POS Tagging or POST[24], is basically the process of tagging or marking all the words in the input to their corresponding parts Of Speech. It takes into consideration two major aspects. The first aspect is the definition or the meaning of the word which is being tagged. The second aspect is the context in which the word has been used. It refers to the relative positioning of the word i.e. it depends on the surrounding words and their parts of speech. The first aspect is simple and is dealt by having separate databases for all the parts of speech. For addressing the second aspect, several rules have been proposed and implied successfully. Combining both the approaches, we conclude a solution for POS tagging which addresses both the aspects discussed before.

1.3 OBJECT-ORIENTED MODELLING

The main focus of the object-oriented paradigm lies in recognising classes, objects and the relationships between them. Object-oriented analysis design (OOAD) methodology uses the tools of Unified Modelling Language (UML). OOAD based software modelling is also called Component Added Software Engineering (CASE) [1]. Software that draw UML diagrams are Rational Rose, Smart Draw, Visual UML, UMLet, UMLGraph, GraphViz and many more[13]. R.J. Abott was the first person to propose that natural language can be used for object-oriented software modelling two decades ago. After that, several rules and methods

have been proposed by researchers as well as practitioners to use the Natural Language (NL) text for object-oriented software analysis. The object oriented modelling of the natural language text is only possible once the grammatical parsing is done successfully. It uses the result of the grammatical parsing to recognize the major entities required to model the object-oriented model of the system. The model is facilitated by dynamic generation of UML diagrams that would be discussed later.

1.3.1 UML DIAGRAMS

UML is a notation based on the object-oriented principles. It is a standardized general-purpose modelling language in the field of OOA. It is considered as the standard for modelling software in industries.

UML generally outlines nine types of diagrams given as follows:

- Class (package) Diagram-It is considered the backbone of object-oriented system which depicts the static structure of a system. The functionalities of the classes are given by methods and the associations represent relationship between classes.
- Object Diagram-It has a close link to the class diagrams. It is believed to be an instantiation of a class diagram and represents the static structure of a system. It can be used to test whether the class diagrams are accurate.
- Use-case Diagram-It represents the functionalities of the system using actors and use-cases. Use cases are functions provided by the system to its users.
- Sequence Diagram-Sequence Diagram describes interactions among classes that happen through sending and receiving of messages.

- Collaboration Diagram-It represents a combination of information taken from diagrams such as class diagram, sequence diagram and use case diagram which describe the static structure and dynamic behaviour of a system.
- Statechart Diagram-It shows how the classes behave when acted by triggers from external source. It gives the information about how the control flows dynamically from one state to another within a system.
- Activity Diagram-It illustrates the dynamic nature of a system by modelling the flow of control from activity to activity.
- Component Diagram-It describes how components are included together to form large components in a system.
- Deployment Diagram-Nodes, components, and connections which are the physical resources of a system are depicted by deployment diagram.

The UML modelling can be categorized to three types:

- Structural Modelling
- Behavioural Modelling
- Architectural Modelling

Structural modelling captures the static features of a system. They consist of the following diagrams:

- Class diagrams
- Object diagrams
- Deployment diagrams
- Package diagrams

- Composite structure diagrams
- Component diagrams

Structural model depicts all the components that are present in a system. All the above mentioned diagrams represent these elements and describe the mechanism to assemble the components. However they never describe the dynamic behaviour of the system. Among all the structural diagrams, class diagram is the most popular diagram.

Behavioural model describes the interaction in the system. It shows how the structural diagrams interact with each other. The dynamic nature of the system is represented through this model. They consist of the following diagrams:

- Activity diagrams
- Interaction diagrams
- Use case diagrams

Architectural model comprises both the structural model and behavioural model. It is said to give the complete picture of the system. Under architectural modelling only package diagrams are considered.

Chapter 2

LITERATURE REVIEW

Chapter 2

2 LITERATURE REVIEW

2.1 EARLIER WORK ON NATURAL LANGUAGE PROCESSING

For last many decades natural language has been the area of interest for the researchers. In the late 1960s and 1970s, several researchers like Chomsky [5], Chow, C., & Liu, C [6] worked in the area of information retrieval from natural languages. They contributed in understanding the natural language text, but still there was lot of effort required for its better analysis. Some researchers who concentrated in this area in eighties and nineties are Krovetz, R., & Croft, W. B [7], Salton, G., & McGill, M [8], Maron, M. E. and Kuhns, J. L [9], Losee, R. M [10]. Pedro Domingos [11] presented Markov Logic which has ability to handle uncertainty and learn from the training data which can be used for information extraction and processing.

2.2 EARLIER WORK ON OBJECT-ORIENTED SOFTWARE MODELLING

It was R. J. Abbot who came up with an idea for the first time that natural language text can be used for object oriented software modelling. It was his suggestion that classes or objects can be recognised from nouns and the methods of the classes can be recognised from verbs [14] in the sentence. Again Buchholz inferred that nouns are not only good indicators of classes and objects but also properties [15] of those classes or objects. Kapur, Ravindra and

Brown [4] gave few propositions which are popularly called KRB method. It helps in finding classes and objects manually. The KRB method has the following steps:

1. Identify candidate classes (nouns in NL).
2. Define classes (look for instantiations of classes).
3. Establishing associations (capturing verbs to create association for each pair of classes in 1 and 2).
4. Expanding many-to-many associations.
5. Identify class attributes.
6. Normalise attributes so that they are associated with the class of objects that they truly describe.
7. Identify class operations.

The major objective of object-oriented analysis is to identify NL concepts that can be modelled in the form of object-oriented concepts. This can then be further used to form UML diagrams which depict the major functionalities in an information system. Subsequently S. Naduri came up with the proposition that ‘associations’ in different objects can be pointed out by verbs [12]. Again there are various kinds of associations that exist among the classes and objects namely binary association, identification association, n-ary association, etc. This detailed categorization of associations was done by N. Juristo [16]. Hector [17] developed the tool GOOAL which automatically generates object models from natural language text. K. Li also presented his work to solve problems related to natural language that can be addressed in object oriented analysis and design [18]. Different NLP based tools have been proposed for this purpose. Juristo and Moreno [16] categorized eight specific NL structures which describe about the various associations and aggregation that exist as given below:

1. “is type of” denotes bottom-up simple inheritance.
2. “can be” denotes top-down simple inheritance.

3. “is a... and a...” denote multiple inheritances.
4. “does ... and ...” denote binary association.
5. “is identified by” denotes identification association.
6. “does ... to ... on ...” denote n-ary association.
7. “contains ... and ...” and “are part of” denote aggregation.

However, incorporating these rules is a manual approach and no dynamic modelling is considered. Thus, the discussed methods and techniques are not automatic as they require the user to take many decisions during the object-oriented analysis and modelling.

2.3 UML TOOLS BASED ON NLP

As mentioned in the initial sections the researchers developed various Computer Added Software Engineering (CASE) tools which used natural language text for object-oriented modelling. A. Oliveira used natural language constituents for object-oriented data modelling and made REBUILDER UML [19] which is an initial level CASE tool. It converts the given natural language text into an UML class diagram. Another tool that performs object oriented modelling from natural language text is LOLITA [20]. However, it can only identify objects from the sample text but it cannot differentiate between classes, methods and their respective attributes. Delisle [23] tried to enlist candidate objects, distinguishing between Subjects (S) and Objects (O), and Verbs (V), using the S-V-O sentence structure. Harmain and Gaizauskas made a noteworthy contribution as they developed the tool named CM-Builder [21]. However CM-Builder could only create a primary class model. The major drawback here is that these systems were not able to automatically identify all object-oriented constituents. The system analyst had to be prompted to help out while identifying the classes, objects and their respective methods and attributes.

Chapter 3

PARTS OF SPEECH TAGGING

Chapter 3

3 PARTS OF SPEECH TAGGING

3.1 INTRODUCTION

It deals with marking the words to the part of speech (POS) they belong to. The major challenges found here is that a particular word behaves as one part of speech at a time and as another part of speech with change in context. This changing behaviour becomes difficult to recognize. Some extra information is required in order to correctly define the part of speech. This invites the concepts of Artificial Intelligence to come into picture.

POS tagging takes into consideration of the following two aspects:

- The meaning of the individual word.
- Relative positioning of the word.

For addressing the first aspect, a long list of the words can be maintained for different categories of parts of speech. But the major challenge is dealing with the second aspect that demands knowledge of the context in which the word is used. The relative positioning of the words helps in deducing the probability of the word belonging to a specific POS.

Different ‘multiple taggers’ are available in literature that returns more than one tag for a particular word, but it is found that ‘single taggers’ are more adequate for this task. The POS of a particular word does not only depend on that particular word but also the surrounding word and the POS that those surrounding words belong to.

3.2 HIDDEN MARKOV MODEL

Hidden Markov Model is a statistical model which assumes the system to be modelled as a Markov process. Researchers in Europe thought of utilising this model for POS tagging. They made a table of probabilities for certain sequences of English NL structures. For instance a word following the word ‘the’ can never become a verb. As specified in the Hidden Markov model, chances of the word being a noun is 40%, that of being an adjective is 40% and a number is 20% [2]. Similarly, it can be proposed that words following an article can never be a verb. Thus, the ambiguity found with a particular word can be solved by looking at the previous or the next word and their associated POS. For example in the ATM specification [3] there is a sentence, ‘The shared system will be apportioned to the banks’. Clearly the word ‘the’ has to be an article. But the word ‘shared’, is present both in the database for verbs as well as in the database present for the adjectives. As the word ‘shared’ comes after an article so its chances of being a verb are ruled out and it is concluded that the word is an adjective. Similar rules for the different parts of speech to remove the ambiguity in determining the correct parts of speech are to be applied.

3.3 RULES USED

After analysing different sentences, it is found out that the words following the set of words {he, she, they, it, you, we etc.} generally belong to ‘verb’ category, whereas that following the set {his, her, their, its, yours, ours etc.} never belong to the category of ‘verb’ but can belong to ‘nouns’ or to the ‘adjectives’. Similarly in the previous case the words never belong to the category of ‘nouns’ or ‘adjectives’ or ‘prepositions’ etc. It is found that an auxiliary verb is generally followed by another auxiliary verb or a main verb. The concept of auxiliary verbs is again a relative concept. For example if the word ‘is’ is followed by a noun, then it

becomes a main verb. But when it is followed by another auxiliary verb or any main verb then it acts as an auxiliary verb. Also, nouns are never followed by pronouns. Thus all these rules and deductions are combined in a module to help in determining the most probable part of speech in the instances where ambiguities are faced.

The instances and the rules discussed above can be easily found to be related to two main parts of speech, i.e. the nouns and the verbs. The reason behind this is that the words for which POS generally changes with change in context are found to be nouns or verbs. The next step deals with correctly determining these two most important parts of speech, which will further help in building the object-oriented model. Thus, if any ambiguity is left unobserved regarding these parts of speech then it would result in an incorrect model. Further, the consistency of POS tagging is verified by checking the matching NP-VP rules [22], in order to improve accuracy.

3.4 WORK FLOW

As discussed in the flow chart specified below, first we tokenize the input. The input is parsed in such a way that all the individual words and symbols are separated. After tokenizing the input, the words are checked for in the database. Depending upon the presence of the words in the database, the words are tagged with their corresponding Parts of Speech or the user is prompted to specify the POS and thus enrich the database, such that in future, if the same word comes up, the user would need not specify the POS again, rather the previous knowledge would be used this time. Therefore, the efficiency of this model increases with increase in the use of this model for different input texts. The more the model is used, additions are made to its database and it becomes more efficient and robust. After that, more rules are used to finalize the tag. The Hidden Markov model is particularly used because it eliminates the confusion of a word either being a noun or a verb. NP-VP rules may also be

included to confirm the syntax of the sentence though it is not required for object-oriented modelling.

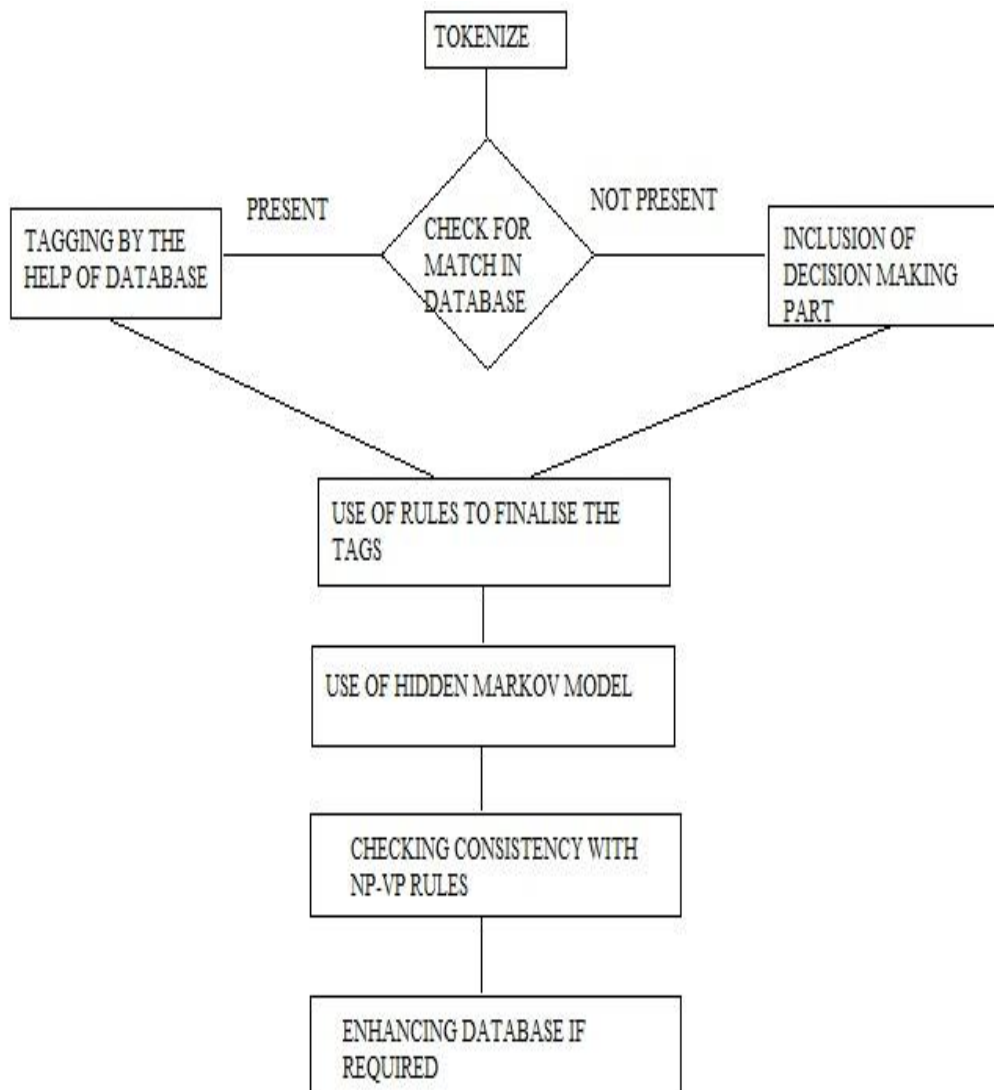


Figure 1: Flow Diagram for Parts of Speech Tagging

The proposed algorithm deals with construction of an object-oriented model based on this POS tagging. It has been found that the different parts of speeches are good indicators of different parameters of the object-oriented model.

Chapter 4

OBJECT-ORIENTED MODEL

Chapter 4

4 OBJECT-ORIENTED MODEL

4.1 PROPOSED SYSTEM

The object-oriented modelling done involves two steps:

- POS Tagging
- Classes Identification

The first step is the most essential step, which has already been discussed in the Chapter 3.

The second step mainly utilizes the output of the first step, using the concepts as given below:

- Nouns- name of the class
- Adjectives- attributes
- Verbs- methods/operations

4.2 NEED FOR IMPROVEMENTS

The relationships among these objects or classes need to be established. It can be of three forms, i.e., Inheritance, Association and Aggregation. This can be determined only if some NL-text processing concepts are introduced and the sequence and structures of the sentences used in the text are noted. The attributes need not be only the adjectives associated with the nouns. It may also be the first noun occurring after the word “if”. The constraints related to a particular class can lead to formation of attributes of the nouns associated with those constraints.

4.3 FURTHER MODIFICATIONS

To have a better model than that introduced in the initial attempt, some other modules are included and the previous model is expanded to comprise the following modules:

- POS Tagging
- NL-Text processing
- Classes Identification
- Finalizing the classes
- Generation of Class Diagrams

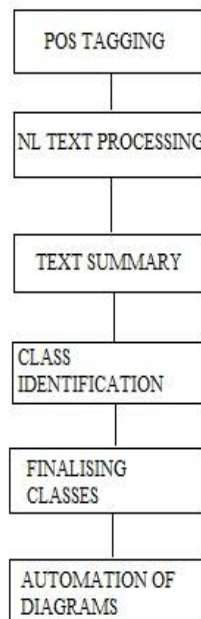


Figure 2: Object-Oriented Modelling

The new modules introduced to the initial attempt are discussed in the following sections.

4.3.1 NL-TEXT PROCESSING

While identifying classes and its associated parameters, some pre-processing needs to be done for the following reasons:

- The pronoun needs to be linked to the class it denotes. It can be done by examining the noun that comes before this pronoun, may be in the same sentence or the previous one.
- It helps in determining the NL-structures used to identify the type of relationships between the classes [18].
- It provides solution to one of the two problems faced in Object Identification.
 - Bad Classes
 - Redundant Classes

Two consecutive nouns occurring in a text generally refer to one particular entity and thus needs to be grouped to form one particular class by combining them with hyphen. For example let the input 'Human cashiers enter account data'. Here both human and cashiers are nouns and are potential of forming two different classes. This leads to the formation of bad classes or meaningless classes. Thus in, this step of NL-Text processing, whenever two consecutive words are found to be nouns, they are combined using hyphen and after this processing 'Human-cashiers' is identified as one single class.

Similarly, consecutive verbs can be combined to form one single verb. Generally, an auxiliary verb is combined with the main verb to form one single verb. This makes the text more compact and easy to deal with.

4.3.2 FINALIZING THE CLASSES

The classes identified in the module of ‘Classes Identification’ needs to be refined, as it may have redundant classes. The work of refinement of the classes is done in this module. It may be possible that two or more classes refer to the same entity. For example, in the ATM specification [3] there are two classes namely bank and banks. As both of them refer to the same entity, all the attributes and methods of both the classes are combined in one class and hence the other class is removed. This eliminates the redundant classes and thus decreases the total number of classes.

4.3.3 GENERATION OF CLASS DIAGRAMS

The class diagrams are generated using the tool UMLet. This tool helps in automation of class diagrams using a text editor.

For drawing the class diagrams we need to identify the objects, classes, associations between them and operations or methods. The result of POS-Tagging can help in identification of the above entities.

Nouns are generally good indicators of classes and objects, verbs serve to be good indicators of operations and methods, adjectives to be of attributes and for relationship identification. Juritso and Moreno [16] proposed few NL structures presence of which denoted the three types of relationships that exist among the classes. The classes may be associated through inheritance, association and aggregation. The NL structures and the corresponding relationship that they depict are as given below:

- a) ” is type of ” reflects bottom-up simple inheritance.
- b) ” can be ” reflects top-down simple inheritance.
- c) ” is a... and a...” reflects multiple inheritances.
- d) ” is identified by ” reflects identification association.

e) "are part of" reflects aggregation.

f) "contains ... and ..." also reflects aggregation.

g) "does ... to ... on ..." reflects n-ary association.

Meanings of these phrases help in determination of the kind of relationship the classes share between them. This idea can be further explored by taking into consideration the meaning of further more words to help in analysis of the different features of the model to be constructed.

Chapter 5

DESIGN AND ALGORITHM

Chapter 5

5 DESIGN AND ALGORITHM

5.1 ALGORITHM

- 1) Do the following steps from a) to e) in the first scan.
 - a. Parse sentence into tokens (Using StringTokenizer in Java).
 - b. For each token, call find () function.
 - c. find() function (String b= find(a);) , finds the name of the parts of speech of the token.
 - d. If it is unable to identify the token then it returns null.
 - e. The value of the strings a and b are printed, i.e. the token- name of part of speech it belongs to.

- 2) In the second scan do the following:
 - a. If there is an article, then the possibility of the next word being a verb is ruled out.
 - b. If there is an auxiliary verb, then the next word can never be a noun.
 - c. If there is a pronoun, then the possibility of the next word being a verb is ruled out.

- 3) In the third scan, remove all the articles, the pronouns that are followed by some nouns, the auxiliary verbs that are followed by any auxiliary.

- 4) The modified input is scanned and the entities of the object model are identified.

- 5) The entities identified in step 4) are used to generate the object model.

5.2 DESCRIPTION AND ANALYSIS OF THE ALGORITHM

1) Instead of taking one sentence as an input, we take a paragraph or even more than that using text file as an input to the code. This way the input can also be increased to as much number of lines as possible making the code more general.

2) Databases are enriched using more and more words collected from different websites as stated in the reference page.

3) Another database for special symbols should also be made for symbols like full stop, comma, inverted comma, semi colon, colon etc.. Care should also be taken for numeric values which may be found in the text.

4) Database should be maintained such that all words are in same case (either upper case or lower case) so that when matching is done both the cases can be taken care of without having redundant values. While matching we search for the lower case of the token instead of searching the token directly as all the databases are in lower case. For example, the database for pronouns has the words as given below:

he

she

they

and the input sentence is “She is writing”. Here the word “She” is not present in the database but “she” is there.

So we first convert the word “She” to “she” using `toLowerCase()` function and then search for the word in the database. Else it would not find the right match returning null (incorrect result, despite of having the word in the database) or we would have to store both of them which would result in unnecessary data and make the database size double as required unnecessarily.

- 5) To increase the efficiency of the model, a decision making part is also included, that asks the user to specify the type of word when it is not in any of the database, i.e. when the find() function returns null. It also updates the database simultaneously so that if in future the same word comes up in any sentence then it would not ask the user to specify this time. Instead it would give the result based upon what the user had specified the last time.
- 6) Care should be taken during the decision making process to see to it that the user should specify the same name as that of the database else it would be of no use in the future as the new name specified by the user will result in creating a new database about which the model remains unaware and so would not access or search for that database in the future resulting in no benefit of the decision making process that we designed.
- 7) This decision making process helps in making the model more and more efficient with increase in the use of this model for different text inputs.
- 8) A particular word may act as more than one “parts of speech”. To resolve this ambiguity, we check the relative positioning of the words and determine the correct parts of speech it should refer to as in that context.
- 9) Number of scans is increased to achieve better accuracy in determining the parts of speech as well as in modifying the original text to eliminate redundant data.
- 10) The modified text eases the process of identification of the object oriented entities.
- 11) Solution to the bad classes and redundant classes is already discussed in Chapter 4.

5.3 IMPLEMENTATION AND DISCUSSION

Case I

The ATM specification [3] given as input is “Design a software to support a computerized banking network including both human cashiers and ATMs to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank’s computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate recordkeeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide their own software for their computers; you are to design the software for the ATMs and the network. The cost of the shared system will be apportioned to the banks according to the number of customers with cash cards.” The classes identified are free of bad classes and are as follows:

- software
- bank
- cashier-stations
- human-cashiers
- teller-machines
- teller-machine
- system
- banks

These classes are further refined as discussed in Chapter 4 and the final classes obtained after the removal of redundant classes are as follows:

- software
- bank
- cashier-stations
- human-cashiers
- teller-machine
- system

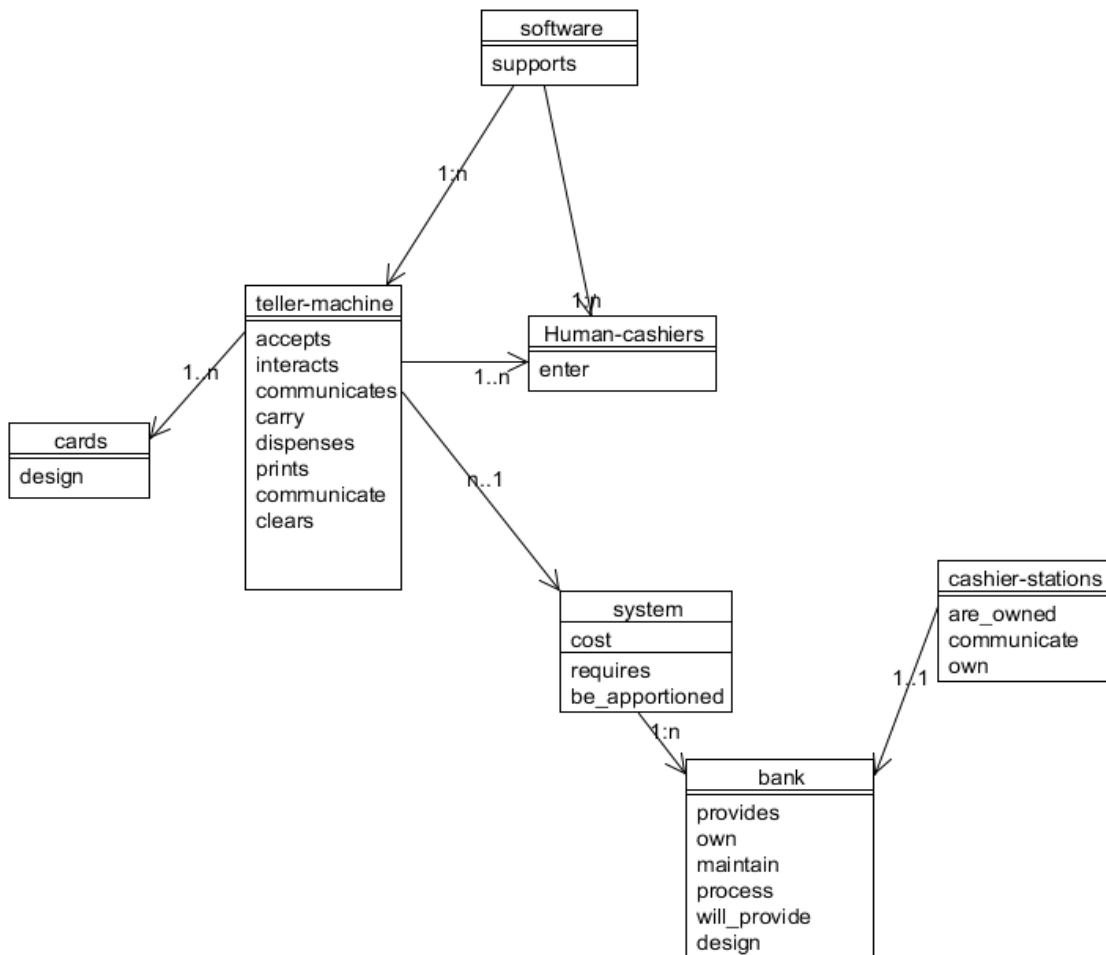


Figure 3: Class Diagram for the ATM specification

This class diagram is generated after the POS tagging of the whole specification. The class diagram for the ATM specification depicts the following:

- software supports teller-machine and human cashiers.
- teller-machine accepts cards, interacts with human cashiers and communicates with system.
- cost of system is apportioned to the banks.
- cashier stations are owned by banks.

As seen above, the class diagram describes all the important features of the ATM specification and hence can be used to as an efficient means of representation of any information system.

Case II

If the input to the model is, “The institute wants to develop an integrated student information system for monitoring admission, enrollment, payment of fees, academic activities in the department, examination procedures and result publication. The admission is granted only if previous CGPA is above 6.5 .The enrollment is possible only if the faculty advisor approves the subjects desired by the students. The fees are accepted if it is paid before the due date. The student is allowed to continue if he has a good background of discipline. The student is allowed to write the exam if he has the attendance above 85 %. A student is given pass grade P if his score is above 35 and overall CGPA is above 6.5.”

Then, the classes identified by this implementation are as follows:

- admission
- institute
- enrollment

- fees
- student

Class diagram drawn using the techniques described in the previous sections is given below.

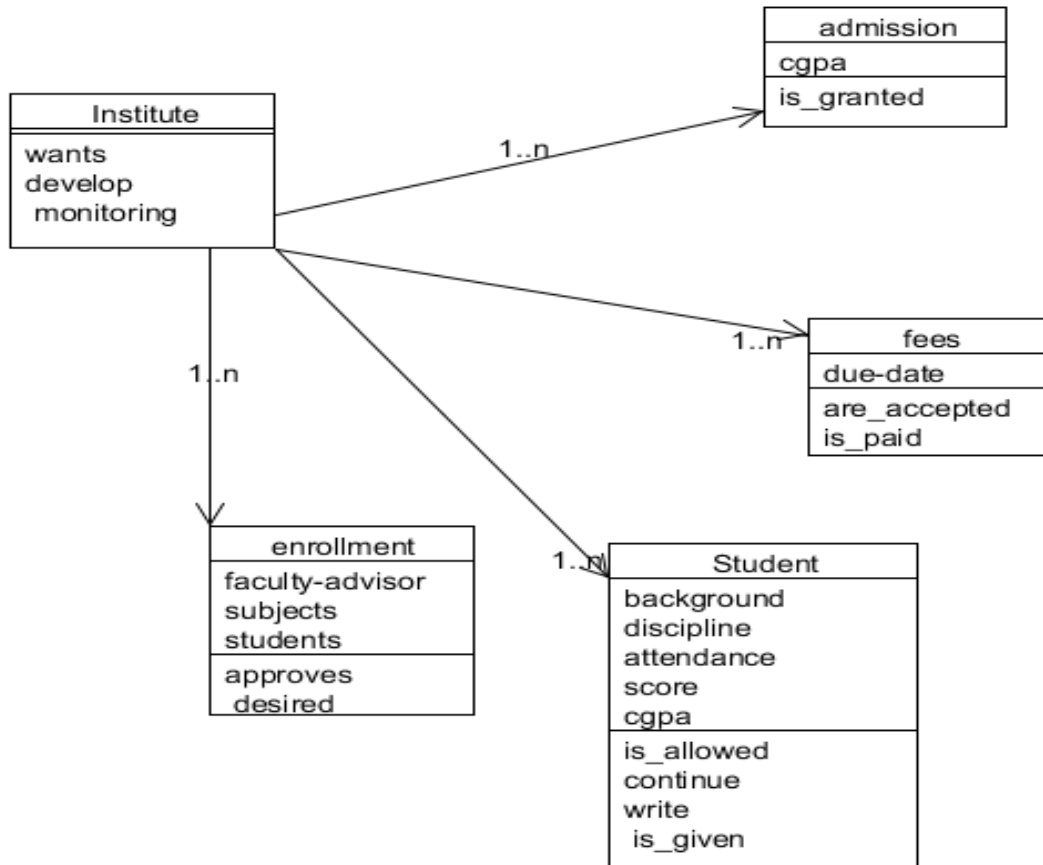
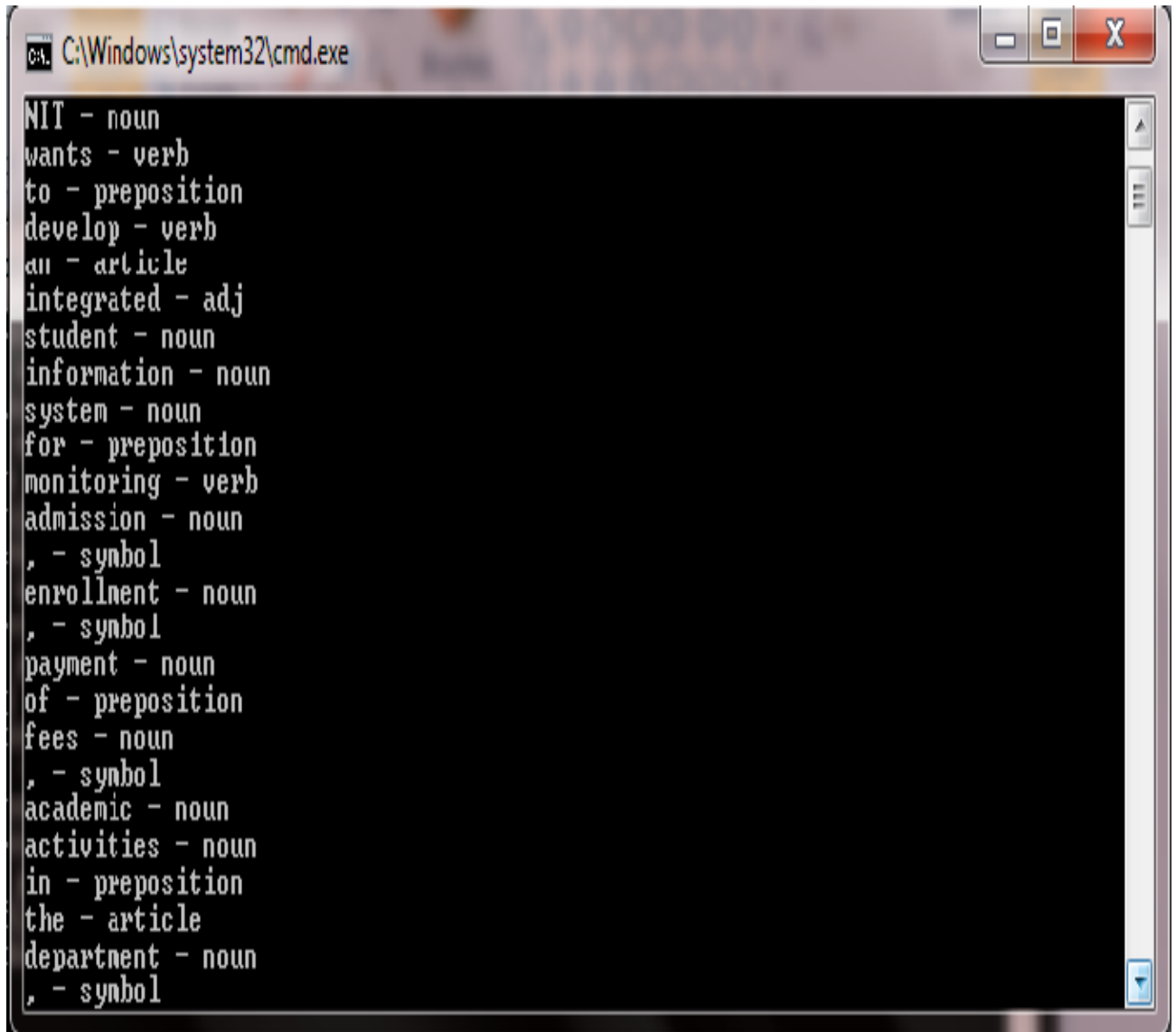


Figure 4: Class Diagram drawn based on the proposed Object Model.

The representation of any information system in the form of class diagrams helps the user in understanding the major functionalities of the information system. The class diagram that has been obtained is done after the two major steps that include POS Tagging and Object-Oriented Software Modelling. The outcome of each step is shown in the figures below.

The figure 5 gives the result of parsing the input text or the SRS and tags the POS to which the words belong to. The classes and its different parameters are identified and stored in text files.



```
C:\Windows\system32\cmd.exe
NIT - noun
wants - verb
to - preposition
develop - verb
an - article
integrated - adj
student - noun
information - noun
system - noun
for - preposition
monitoring - verb
admission - noun
, - symbol
enrollment - noun
, - symbol
payment - noun
of - preposition
fees - noun
, - symbol
academic - noun
activities - noun
in - preposition
the - article
department - noun
, - symbol
```

Figure 5: Screenshot of result obtained after POS Tagging

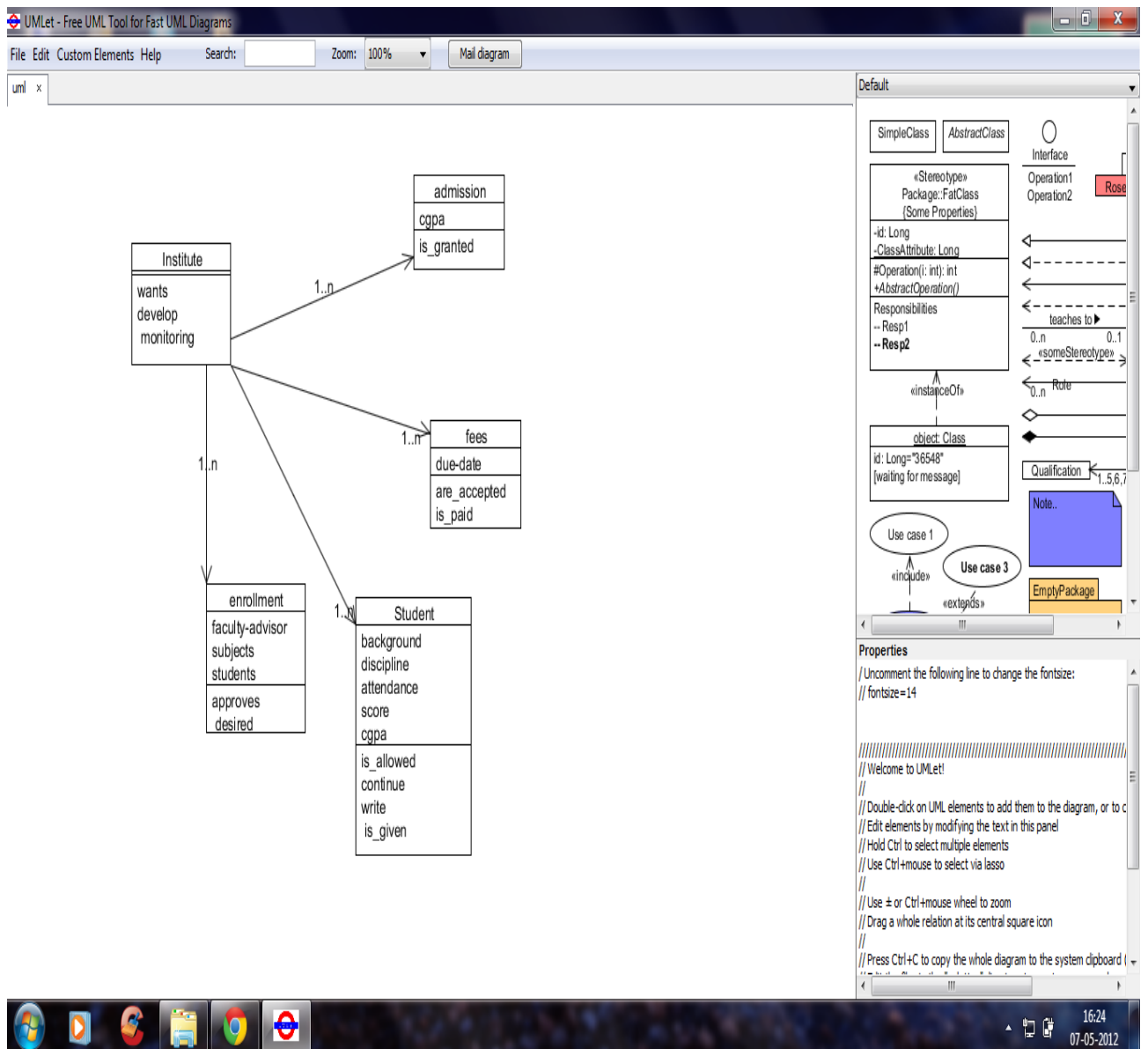


Figure 6: Screenshot of the result obtained after generation of Class Diagram in UMLet.

The figure 6 draws the Class diagram in UMLet using the text editor specified at the right most bottom of the UMLet.

The implementation was carried out using the parsing mechanism followed by the tagging of words, with subsequent generation of class diagram through UMLet. This showed that analysis of lengthy SRS could be done via this procedure in much lesser time. All the static entities, the classes and the complete information of the methods and attributes could be derived from it. The dynamic generation of class diagrams was yet another advantage for system analysts to understand client requirement specification.

Chapter 6

CONCLUSION

Chapter 6

6 CONCLUSION

Various techniques developed earlier aim at making the analysis of user requirements easier. But these techniques expect user interference, in terms of providing the client information from time to time. The information available from the client in the form of natural language text is analysed in such a manner that object-oriented modelling can be done by the system analyst precisely. The methodologies for simplifying natural language need to be straightforward, so that people can learn and apply them easily. It should be kept in mind that the models generated must be easily comprehensible and the natural language from which they have been derived can be traced back. The implementation of the proposed object oriented model gives way to easier understanding of lengthy NL text and subsequent validation of classes.

This work can be further extended by

- Segmentation of the NL text in such a manner that the different modules and packages in the requirement specification can be identified.
- Improvement of parts of speech tagging by incorporating more rules in order to identify the system elements more effectively.
- Removal of user assistance at any point of time for extracting information regarding the functionalities of the system.

All in all, object-oriented software modelling when done using the concepts of natural language processing helps software practitioners to perform faster analysis of the SRS, given that the tagging is done logically.

7 BIBLIOGRAPHY

- [1] Joseph Schmuller. *Sams Teach Yourself UML*. TechMedia, 1999.
- [2] A. S. Fatima, A. Guessoum. *A Hidden Markov model-based POS tagger for Arabic*. Proc. of 8th International Conference on the Statistical Analysis of Textual Data, France, pp. 31-42, 2006.
- [3] James Rumbaugh, Michael Blaha, William Premerlani, Frederic Eddy, and William Lorenson. *Object Oriented Modelling and Design Methodology*. Prentice Hall, 1991.
- [4] D.W Brown. *An Introduction to Object-Oriented Analysis Objects and UML in Plain English*. John Wiley, Inc. New York, 2002.
- [5] N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass, 1965.
- [6] C. Chow, C. Liu. *Approximating discrete probability distributions with dependence trees*. IEEE Transactions on Information Theory, 1968, IT-14(3), 462–467, 1968.
- [7] R. Krovetz, W. B. Croft. *Lexical ambiguity and information retrieval*. ACM Transactions on Information Systems, 10, 1992, pp. 115–141, 1992.
- [8] G. Salton, M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1995.
- [9] M. E. Maron, J. L. Kuhns. *On relevance, probabilistic indexing, and information Retrieval*. Journal of the ACM, pp. 216–244, 1997.
- [10] R. M. Losee. *Parameter estimation for probabilistic document retrieval models*. Journal of the American Society for Information Science, 39(1), pp. 8–16, 1988.
- [11] S. Kok, P. Domingos. *Learning the structure of Markov logic networks*. Proc. Of ICML-05, Bonn, Germany, ACM Pres, pp 441–448, 2005.

- [12] S. Naduri, S. Rugaser. *Requirements Validation via Automated Natural Language Parsing*. Proc. 28th Hawaii International Conference, Systems Science: Collaboration Tech., Organizational Systems, and Technology, IEEE Computer Society, pp. 362-367, 1994.
- [13] I. Bajwa, A. Samad, S. Mumtaz . *Object Oriented Software Modelling Using NLP Based Knowledge Extraction*. European Journal of Scientific Research, ISSN 1450-216X, Vol.35, No.1, pp 22-33, 2009.
- [14] R.J. Abbott. *Program Design by Informal English Descriptions*. Communications of the ACM, Nov. 26(11), pp. 882-894, 1983.
- [15] H. Buchholz, A. Dusterhoft, B. Thalheim. *Capturing Information on Behavior with the RADD_NLI: A Linguistic and Knowledge Base Approach*. Proc. 2nd Workshop Application of Natural Language to Information System, IOS Press pp. 185-196, 1996.
- [16] N. Juristo, A.M. Moreno. *How to Use Linguistic Instruments for Object-Oriented Analysis*. IEEE Software, pp. 80-89, May/June 2000.
- [17] Hector G. Perez-Gonzalez. *Automatically Generating Object Models from Natural Language Analysis*. Proc. 17th annual ACM SIGPLAN conference on Object-oriented Programming, systems, languages, and applications, ACM New York, USA, pp. 86 – 87, 2002.
- [18] K. Li, R.G.Dewar, R.J.Pooley (2003) *Object-Oriented Analysis Using Natural Language Processing*, www.macs.hw.ac.uk:8080/techreps/docs/files/HWMACS-TR-0033.pdf.
- [19] António Oliveira, Nuno Seco, Paulo Gomes. *A CBR Approach to Text to Class Diagram Translation*. TCBR Workshop at the 8th European Conference on Case-Based Reasoning, Turkey, September 2006.
- [20] L. Mich, R. Garigliano, *A linguistic approach to the development of object-oriented system using the NL system LOLITA*. Object Oriented Methodologies and Systems, (ISOOMS), LNCS 858, pp. 371-386, 1996.

- [21] H.M. Harmain, R.Gaizauskas. *CM-Builder: An Automated NL-based CASE Tool*. Proc. of 15th IEEE International Conference on Automated Software Engineering, pp. 45-53, 2000.
- [22] M. Steedman, *Natural Language Processing*. <http://repository.upenn.edu/cisreports/323> .
- [23] S. Delisle, K. Barker, I. Biskri. *Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools*. Proc. 4th International Conference on Applications of Natural Language to Information Systems, Klagenfurt, Austria, pp. 167-171, 1999.
- [24] Daniel Jurafsky, James H. Martin. *Word Classes and Part-of-Speech Tagging*. www1.cs.columbia.edu/~julia/jmchapters/ch5.pdf.