International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 8 (2016) pp. 293-300

Open NLP based Refinement of Software Requirements

Murali Mohanan [1], Philip Samuel [2]

[1]Department of Computer Science,
SOE,Cochin University of Science and Technology, Kochi,India.
[1]mohananmurali@gmail.com

[2] Division of Information Technology,
SOE,Cochin University of Science and Technology, Kochi,India.
[2]philips@cusat.ac.in

Abstract: . Software requirements are usually written in natural language (NL) or speech language which is asymmetric and irregular. This paper presents a suitable method for transforming user software requirement specifications (SRS) and business designs written in natural language into useful object oriented models. Here a neoteric approach is proposed to generate object oriented items from SRS. For NL processes like sentence detection, tokenization, parts of speech tagging and parsing of requirement specifications we incorporate an open natural language processing (OpenNLP) tool. It provides very relevant parts of speech (POS) tags. This parts of speech tagging of the SRS is quite useful for further identification of object oriented elements like classes, objects, attributes, relationships etc. After obtaining the required and relative information, Semantic Business Vocabulary and Rules (SBVR) are applied to identify and to extract the object oriented elements from the NL processed requirement specifications.

***Keywords*:Requirement Elicitation, Software requirement specification, OpenNLP, SBVR, class model generation.**

I. Introduction

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

The natural language processing is a research area in which many researchers proposed several methods for analyzing the natural language (NL) requirements [14],[15]. Nan Zhou and Xiaohua Zhou (2004) proposed a methodology to generate object oriented model from the user requirement document. This approach used natural language processing to analyze the written requirements and domain based ontology is used to improve the class identification performance. The author used a linguistic pattern to differentiate the class and attributes, numeric pattern to analyze the relationship and parallel structure pattern to found more classes and its attributes. But it was not good enough in automatically identifying object oriented elements.

In this paper we have addressed the problem related to the software analysis and development phase. Here we use open natural language processing (OpenNLP) to process software requirement specification(SRS).The OpenNLP [10],[11] is used to produce parts of speech (POS) tag from the SRS which contains natural language statements. The POS tag captures the required details such as noun, verb, adverb, etc. of the natural language statements. Sentence splitting, tokenization and Pos tagging [12] are the phases of OpenNLP. These phases help to process the requirement specifications in NL which are easy to understand by both the user and the machine [13].

The recent trends of the software engineering largely depends on object oriented paradigm that widely use unified models. Unified Modelling Language(UML) is commonly used for modelling the user software requirements, documents the software assets, development and redevelopment of software [1]. Our research work proposes a methodology which is used to extract object oriented elements from NL processed SRS. Object oriented analysis applies the object oriented paradigm to model software information systems by defining classes, objects and relationships between them.

The UML model is an important component for Object Oriented analysis and design.The existing tools such as ReBuilder [2], CM-Builder [3] , GOOAL [4], NL-OOML [5], UML-Generator [6] generates the UML class diagram automatically from the natural languages.The problem with these tools are they generate the object oriented models with lower accuracy due to the informal nature of NL and its ambiguity [7],[8].

This paper is focused on reducing the complexity in designing the object oriented models from the user requirements. User specifies their requirements in natural languages such as English. Using OpenNLP, user requirement statements are processed first. It tokenizes the input and then syntatically and semantically processes the input. Since the natural language processing is such a difficult task, our research work is divided into two phases to provide to efficient work. The initial phase is OpenNLP tool process and second phase is Semantic Business Vocabulary and Rules (SBVR) process [9].

II Related works

The natural language processing is a research area, researchers proposed several methods for analyzing the NL requirements [14],[15]. Some researchers focused on class diagram extraction from the NL requirements. This section describes the survey of some methods that uses the NLP or domain ontology for NL requirement analyst and generates the class diagram.

Nan Zhou and Xiaohua Zhou (2004) proposed an automated system to generate the class diagram from the user requirement document. This approach used a natural language processing to analyze the written requirements and domain based ontology is used to improve the class identification performance. The author used a linguistic pattern to differentiate the class and attributes, numeric pattern to analyze the relationship and parallel structure pattern to found more classes and its attributes. The final output is filtered by the domain ontology.

Ambriola and Gervasi (2006) designed a framework to develop the models such as Entity Relationship diagram, Data Flow Diagram, even UML diagrams. The system take the user requirement written in natural language as input and applied the CICO parser for parsing and information extraction. Priyanka More and Rashmi Phalnikar (2012) proposed an approach to design the UML diagrams from the informal natural language requirements. Requirement analysis to provide Instant Diagrams (RAPID) is a novel tool used in this approach. An openNLP tool is applied for parsing and the RAPID performs the RACE stemming process to improve the efficiency by reducing the redundancy. A Domain Ontology in RAPID tool improves the performance of concept identification. Object oriented items like classes are identified by the class extraction engine.

Deeptimahanti and Babar (2009) developed a UML Model Generator from Analysis of Requirements (UMGAR) a domain independent tool for designing the UML models with proper relationship. A simple requirement is generated from the complex user requirement by the syntactic reconstruction rule.

A set of natural language analysis tools are available in Stanford CoreNLP [26]. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract open-class relations between mentions, etc.

Stanford CoreNLP is an integrated framework. Its goal is to make it very easy to apply a bunch of linguistic analysis tools to a piece of text. Starting from plain text, you can run all the tools on it with just two lines of code. It is designed to be highly flexible and extensible. With a single option you can change which tools should be enabled and which should be disabled. Stanford CoreNLP integrates many of Stanford's NLP tools [27], including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, sentiment analysis, and the bootstrapped pattern learning tools. Its analyses provide the foundational building blocks for higher-level and domain-specific text understanding applications. Stanford CoreNLP Provides:

1)An integrated toolkit with a good range of grammatical analysis tools
2)Fast, reliable analysis of arbitrary texts
3)The overall highest quality text analytics
4)Support for a number of major (human) languages)
5)Interfaces available for various major modern programming languages

Stanford CoreNLP is very effective to quickly and painlessly get linguistic annotations for a text.It supports to hide variations across components behind a common API and to have a minimal conceptual footprint, so the system is easy to learn.It also provides a lightweight framework, using plain Java objects (rather than something of heavier weight, such as XML or UIMA's [28] Common Analysis System (CAS) objects).

III Proposed Methodology

Aiming to give a suitable support for software developers as well as software engineers we have proposed a neoteric approach for natural language processing and object oriented modeling. This work is focused on natural language processing and then to extract useful object oriented elements. Software requirements are usally written in the natural language or speech language which is asymmetric and irregular. In our work the open natural language processing (NLP) analyzes the user requirements and provides the parts of speech (Pos) tagging. The SBVR process implemented here is used to extract object oriented elements like classes, objects, attributes, relationships etc from the NL processed SRS. The SBVR process includes SBVR vocabulary extraction and rule generation. This can be further refined to form Unified Models which depict the major functionalities of a software system.

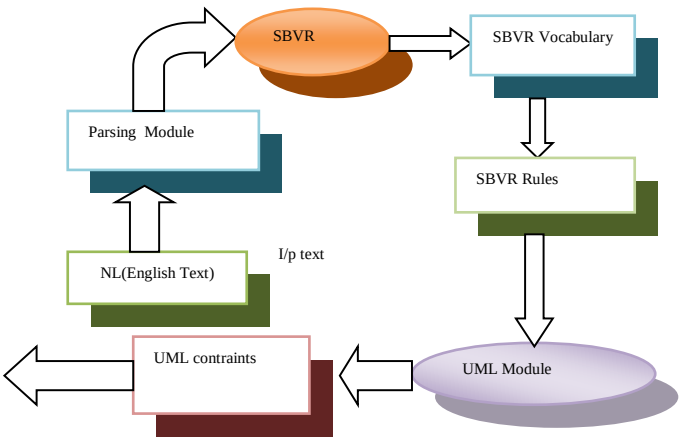

Figure 1. Proposed Approach Framework

IV Open Natural Language Processing

The OpenNLP is a research area that aims to obtain how computer understands and process the natural language.It is really fast to implement. The OpenNLP tool used in the proposed system understands the natural language as suggested by Deeptimahanti (2009). The OpenNLP starts to execute by extracting tokens from user requirement statements and then it proceeds to the syntax [10] and semantic analyzes [20] by parsing each and every sentence. The parser removes the stop words which has no information and also removes the function words such as *on, over, between, has, do* and generates the content word which has noun, adjectives, adverb and verb. Thus a Pos tagger [21] is generated.

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. The common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution are done using this tool. OpenNLP project is developed to create a mature toolkit for the above mentioned NLP tasks.

To write program code in Java, one can use a variety of IDEs:Eclipse, IDEA and NetBeans or just use a text editor and for Java development, we suggest using Maven to build a new project using the plugin Maven Archetype Plugin-Simple.
Here is some example code to extract Noun Phrases from sentences using OpenNLP models. It involves tokenizing some (plain) text to sentences, sentences to words, extracting the POS tags of words, then chunking the token+POS into phrases, then filtering noun phrases out of that. So it covers quite a few basic text preprocessing tasks, so may be the code serves as an useful example. Its a single test, and we need to put it into a Java class (call it TestClass.java, say) under the directory src/test/java/com/ourcompany/ourapplication (where ourcompany and ourapplication are the parameters we gave to the mvn project create command). [Java] JUnit/OpenNLP code to extract Noun Phrases from text - Pastebin.com.
To run the code, we need the OpenNLP libraries and JUnit as well. We need to add this into our pom.xml that Maven generated for us.

* OpenNLP - org.apache.opennlp, opennlp-tools
* JUnit- junit

When we run "mvn clean compile" from command line it will download these libraries and compile the code against it. We can also run "mvn -Dtest=ourTestClass test" to run the JUnit test.

A. General Library Structure

The Apache OpenNLP library contains several components, enabling one to build a full natural language processing pipeline. These components contain parts which can be enabled to execute the respective natural language processing tasks and to train them as a model and also to evaluate the model. Each of these facilities is accessible via its application program interface (API) [7].

B. Methods used in our concept are:

1) Sentence Detection

The OpenNLP Sentence Detector can detect that a punctuation character marks the end of a sentence or not. In this sense a sentence is defined as the longest white space trimmed character sequence between two punctuation marks. The first and last sentence makes an exception to this rule. The first no whitespace character is assumed to be the begining of a sentence, and the last non whitespace character is assumed to be a sentence end.

2)Tokenization

The OpenNLP Tokenizer segments an input character sequence into tokens. Some of the tokens generated are punctuation, words, numbers, etc.

Input text

Eg.`Robert Clive,50yearsold,will join the Company as an executive chairman Jan.15. Mr. Harry is President of Ford B.V., the German automobile company.`

Output is shown as individual tokens in a whitespace separated representation.

```
Robert Clive, 50 years old , will join the
Company as an executive chairman Jan. 15.
Mr. Harry is President of Ford B.V., the
German automobile company.
```

3)Tagging

The Part of Speech Tagger marks tokens with their corresponding word type based on the token itself and the context of the token. A token might have multiple pos tags depending on the token and the context. The OpenNLP POS Tagger uses a probability model to predict the correct pos tag out of the tag set. To limit the possible tags for a token a tag dictionary can be used which increases the tagging and runtime performance of the tagger.

```
Robert Clive, 50 years old , will join the
company as an executive chairman Jan. 15 .
Mr. Harry is President of Ford B.V. , the
German automobile company.
```

POS Tagger generates the following:

```
Robert_NNPClive_NNP ,_, 50_CD
years_NNSold_JJ ,_,
will_MDjoin_VBthe_DTCompany_NNas_INan_DTe
xecutive_JJchairman_NNJan._NNP 15_CD ._.
```

```
Mr._NNP
Harry_NNPis_VBZPresident_NNof_INFord_NNPB
.V._NNP ,_,
the_DTGerman_NNPautomobile_VBGcompany_NN
```

4) Parsing

OpenNLP parsing can be done by training the API.

Input text Eg. `The slow white cat jumps over the quick rat .`

The parser output is.

```
(TOP (NP (NP (DT The) (JJ quick) (JJ white)
(NN cat) (NNS jumps)) (PP (IN over) (NP (DT
the)
(JJ slow) (NN rat))) (. .)))
```

V. SBVR

Semantic Business Vocabulary and Rules [23] is introduced by the Object Management Group (OMG) in 2008 for software and business people. It describes the desired vocabulary and rules for providing the semantic documentation of vocabulary, facts and rules of business. It provides a multilingual, unambiguous and rich capability of languages that are used by software designers and business people in various domains.

The Object Management Group proposal called Semantics of Business Vocabulary and Business Rules (SBVR) offers a vocabulary for describing meaning. A part of SBVR called Logical Formulation of Semantics focuses on the structure of meaning.

Business rules are generally expressed in natural language, although some rules are at times illustrated graphically. SBVR is not a logic language for restating business rules in some other language that business people don't use. Rather, SBVR provides a means for describing the structure of the meaning of rules expressed in the natural language that business people use. SBVR calls this "semantic formulation"[25]. Semantic formulations are not expressions or statements. They are structures that make up meaning. SBVR provides a vocabulary for describing them. Using SBVR, the meaning of a definition or statement is communicated as facts about the semantic formulation of the meaning, not as a restatement of the meaning in a formal language. Semantic formulations are described below with examples. Readers are referred to the SBVR document (currently available to OMG members) for the full Logical Formulation of Semantics Vocabulary.
Semantic formulations in SBVR [25] generally:

1) Involve concepts from other kinds of models
2)Provide basis in first order logic
3)Provide communication of semantics of vocabularies and rules using XML
4)Have extension to intentional logics

SBVR draws from ORM/NIAM and from ISO terminology work (particularly ISO 1087- 1). SBVR takes a fact-orientation from ORM/NIAM and incorporates the concept of fact type.

SBVR focuses on meaning independently of any possibilities for automating business rules completely or partially. However, semantics of business rules can be used as input to construct production rules for rule engines. Efforts at automating such transformations are already underway.

Example

Here is an example of a very simple business rule taken from rules for a car rental company. The rule is stated in different ways but is one rule having one meaning. Many other statements are also possible.
A barred driver must not be a driver of a rental.

It is prohibited that a barred driver be a driver of a rental.

It is obligatory that no barred driver is a driver of a rental.

Below is a description of the semantic formulation of the rule above expressed in terms of the SBVR Logical Formulations of Semantics Vocabulary. It is easily seen that SBVR is not meant to provide a concise formal language, but rather, to provide for detailed communication about meaning. The description is verbose, when separated into simple sentences. But it captures the full structure of the rule as a collection of facts about it.
The rule is meant by an obligation claim.
That obligation claim embeds a logical negation.
The negand of the logical negation is an existential quantification.
The existential quantification introduces a first variable.
The first variable ranges over the concept 'barred driver'.
The existential quantification scopes over a second existential quantification.
That second existential quantification introduces a second variable.
The second variable ranges over the concept 'rental'.
The second existential quantification scopes over an atomic formulation.
The atomic formulation is based on the fact type 'rental has driver'.
The atomic formulation has a role binding.
The role binding is of the fact type role 'rental' of the fact type.
The role binding binds to the second variable.
The atomic formulation has a second role binding.
The second role binding is of the fact type role 'driver' of the fact type.
The second role binding binds to the first variable.
Note that designations like 'rental' and 'driver' are used above to refer to concepts. The semantic formulations involve the concepts themselves, so identifying the concept 'driver' by another designation (such as from another language) does not change the formulation.

VI. SBVR Vocabulary and Rules

Semantic Business Vocabulary and Rules generates the vocabulary and rules for a particular business domain. In our research work it helps in identifying various object oriented elements from natural language processed requirements specification. Thus SBVR provides a suitable way to capture the object oriented items from the requirement specification in NL [22]. SBVR does not include quantifiers or logical operators, which are symbols, but has the concepts of quantification and conjunction. Variables are typed. Atomic formulations are based on fact types whose roles are bound by the atomic formulations to variables, constants or individual concepts. In a software model SBVR vocabulary describes the specific software domain and SBVR rules describe the specific logic. The elements of the SBVR vocabulary are concepts and fact types. The concepts represent an entity of a specific domain. Object types, individual concept, verb concept, etc., are the types of concepts. The common nouns are referred to the noun concepts, the proper nouns are considered as individual concepts, the auxiliary verbs and action verbs are the verb concepts.

The combination of noun concepts and verb concepts are the fact types in SBVR Vocabulary. The fact type which is represented in *is-property-of* relationship is considered as characteristics fact type which is extracted as suggested in [23]. Plural nouns (prefixed with s), articles (a, an, the) and cardinal numbers (2 or two) are considered as Quantification.

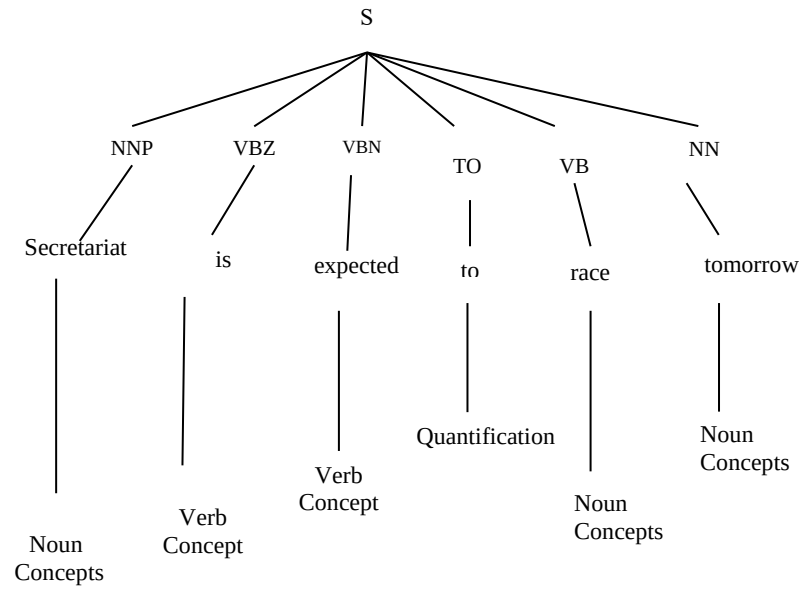

Figure 2. Parse tree representation for SBVR Vocabulary Extraction

The associative fact types are identified by the binary fact types in parts of speech (POS) tagging. "*The belt conveys the parts"* is an example sentence taken to understand the binary fact type. An association is there in the mentioned sentence between the words *belts* and *parts.* In SBVR model, association is mapped to associative fact types, aggregation is considered as partitive fact types and generalization is denoted as categorization fact types. SBVR elements are extracted from the output of the Open NLP.

SBVR includes model formulations for the following modalities from Deontic and Alethic logics:

1)Obligation
2)Permission
3)Logical necessity
4)Logical possibility

Distinguishing between guidance (rules that people break) and structural rules (rules about meaning) is very important in understanding business rules. Consider the following two rules.

It is obligatory that each person on a bus has a ticket.
It is logically necessary that each person on a bus has a ticket.

Based on the first rule, a person on a bus either has a ticket or is breaking the rule. Based on the second rule, being on a bus implies that there is a ticket.

VII. Model Design

In this paper the UML model is considered as the business domain. In the UML class model, the noun concepts are class names and their respective attribute names and object names are denoted as individual concepts. Also operation names are considered as action verbs and the fact types are referred to as associations &generalizations.

To generate the UML model SBVR rule has to be extracted to analyze the specific software logic. The SBVR rule is based on any one of the fact types of SBVR vocabulary. Definitional rule and behavioral rule are the two types [24] of SBVR Rules. The definitional rule defines the organizational setup whereas behavioral rule describes the conduct of an entity. Semantic formulation, logical formulation, quantification and model formulation are processes to be performed to generate the SBVR rules from the fact type. The SBVR rule is constructed by applying the semantic formulation to each fact type.

VIII. SBVR -Rules Generation

With regard to the scope of our project the SBVR rules are generated by the basic semantic formulations proposed in SBVR version 1.0 (OMG, 2008).The semantic formulations considered here are logical formulation, quantification and modal formulation and are explained as follows. Figure 3 represents logical formulation.

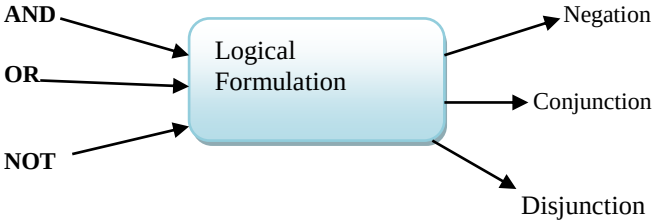

Figure 3. Logical Formulation

From the extracted vocabulary the required tokens are identified to map the logical operators. Tokens such as *not, no*

are considered as the logical operator negation (¬). That, and is denoted as conjunction(∧) similarly or is disjunction(∨)and tokens like *infer, imply, indicate, suggest* are considered as the logical operator implication(→).
Quantification mentions the scope of the concept and it is applied in this work by mapping the tokens as given below:-

More than, greater than→atleast n quantification
Less than→atmost n quantification and
Equal to, positive statement→n quantification

Figure 4.shows a sample quantification

The modal formulation describes seriousness of a constraint. In SBVR two modal formulations are there, one is possibility formulation (PF) and the second is obligation formulation (OBF). The structural requirement is represented by the PF and the behavioral requirement is represented by the OBF. The model verbs mapping to these formulations are as shown below:-

Can, may - PF
Verb concept, should - OBF

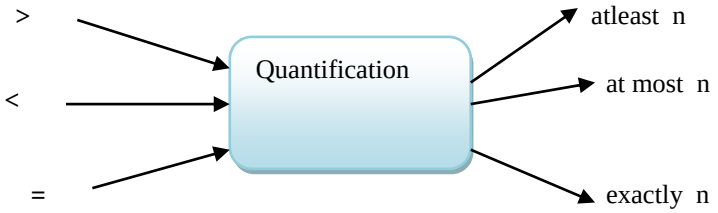

Figure 4. Quantification

Figure 5.shows modal formulation.

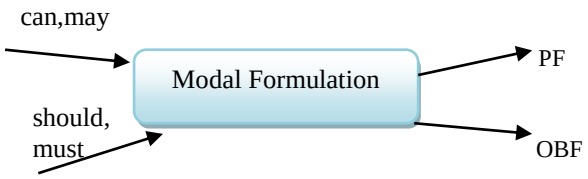

Figure 5. Modal formulation

Structured English Notation is the final step in rule generation and it is performed as in SBVR 1.0 document, Annex C (OMG, 2008). In this phase, notations are provided for generated tokens. For example the noun concepts are underlined e.g. Employee; the verb concepts are italicized e.g. *could be*; the SBVR keywords are bolded e.g. **at most**; the individual concepts are double underlined . Attributes are also italicized but with different colour.

IX. Object Oriented Analysis of SBVR

The final step of the proposed work is the object oriented analysis from the SBVR rule to extract the object oriented elements such as classes, its attributes, objects, methods, generalization, aggregation and associations. This extraction procedure is as naratted in the following sentences. In the SBVR rule the generic entity is represented by the noun concept. On this basis the noun concept is mapped to classes. Similarly the particular entity is obtained from the individual concept so it is mapped to objects. The attributes of a class are obtained by all the characteristics of the noun concepts without action verb. The verb concepts (issue(), order()) are mapped with methods. Association is extracted by the unary relationship, binary relationship and multiplicity. In the SBVR rule following are some relations:

Unary fact type → Unary relationship

Associative fact type→binary relationship

Quantification (noun concept)→ multiplicity.

In extracting the generalization the partitive fact type is divided into subject-part and object part, where the subject-part is main class and the object-part is sub class in generalization. The categorization fact type in SBVR rule is considered as aggregation. Similar to the generalization extraction the categorization fact types are divided as subject part and object part and are maintained as main class and sub class respectively. SBVR rules describe the specific logic in the software domain.

X. Experiment and result:

A simple case study is taken for the proposed work and it is captured from the domain of a robot system. The following problem statement is first processed with the openNLP which is further processed by the SBVR process to identify object oriented items.

"An assembly unit consists of a user, a belt, a vision system, a robot with two arms, and a tray for assembly. The user puts two kinds of parts, dish and cup, onto the belt. The belt conveys the parts towards the vision system. Whenever a part enters the sensor zone, the vision system senses it and informs the belt to stop immediately. The vision system then recognizes the type of part and informs the robot, so that the robot can pick it up from the belt. The robot picks up the part, and the belt moves again. An assembly is complete when a dish and cup are placed on the tray separately by the arms of the robot".

The problem statement is given as the input to our tool. Initially the openNLP processes these statements and provides the parts of speech. This output of the openNLP is shown in figure 6.

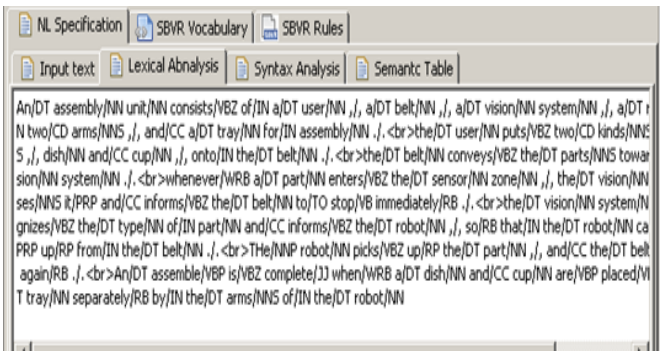

Figure 6. Generated Pos Tagger

This tagger is further processed by the SBVR to identify object oriented items from the generated vocabulary.

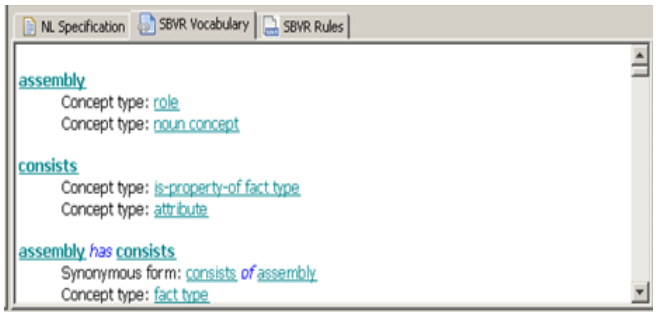

Figure 7. Extracted Vocabulary for UML model

The SBVR rules describe the specific logic in software domain. This rule is generated from the output of the previous phase. Figure 8 shows the rule extracted result.

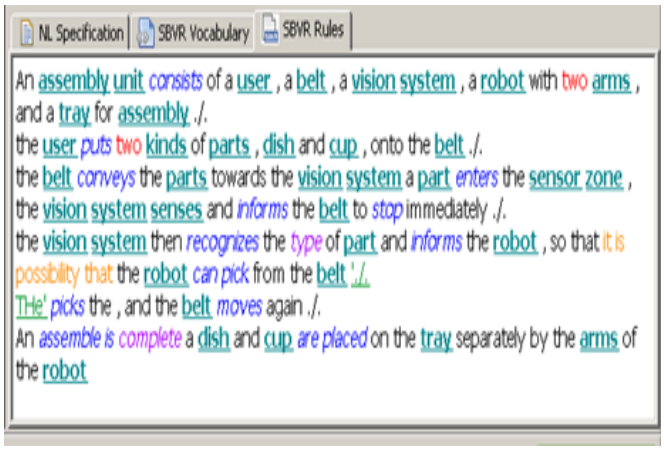

Figure 8. SBVR Extraction

XI. Conclusion

This research work is carried out for providing a robust solution to reduce the ambiguity in natural language and to extract object oriented information from the user requirements. This is a neoteric approach which largely supports machine processing. An open natural language processing tool is implemented to analyze and to parse the SRS which provides the essential parts of speech (POS). After obtaining the required and relative information in the form of POS tags. With the support of Semantic Business Vocabulary and Rules relevant vocabulary and rules are formed. This provides an automatic method to capture the object oriented elements in requirements specification. The automatic extraction of object oriented items from the natural language processed user requirements is a novel concept. Software designers can further refine the gathered information and can develop solid object oriented models.

References

[1] Hector G. Perez-Gonzalez, "Automatically Generating Object Models from Natural Language Analysis", 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, ACM New York, USA, pp.86 – 87,2002

[2] Oliveira, A., Seco N. and Gomes P.A CBR Approach to Text to Class Diagram Translation. TCBR Workshop at the 8th European Conference on Case-Based Reasoning, Turkey.[12],2006.

[3] Harmain, H. M., Gaizauskas R. CM-Builder: A Natural Language-Based CASE Tool for Object- Oriented Analysis. Automated Software Engineering. 10(2), pp.157-181, 2003.

[4] Perez-Gonzalez, H. G., Kalita, J.K..GOOAL: A Graphic Object Oriented Analysis Laboratory. 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '02), NY,USA, pp.38-39, 2002.

[5] Anandha G.S., Uma G.V.Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification. PRICAI 2006: Trends in Artificial Intelligence, LNCS 4099, pp.1155-1159, 2006.

[6] Bajwa I.S., Samad A., MumtazS.Object Oriented Software modeling Using NLP based Knowledge Extraction. European Journal of Scientific Research, 35(01), pp. 22-33, 2009.

[7] Li, K., Dewar, R.G., Pooley, R.J. Object-Oriented Analysis Using Natural Language Processing, Linguistic Analysis, pp. 75-76, 2005.

[8] Mich, L. "Ambiguity Identification and Resolution in Software Development: a Linguistic Approach to improve the Quality of Systems" in *Proc. Of 17th IEEE Workshop on Empirical Studies of Software Maintenance*, Florence, Italy. pp. 75-76, 2001.

[9] Feuto, P.B.; Cardey, S.; Greenfield, P; "Domain Specific Language Based on the SBVR Standard for Expressing Business Rules" Enterprise Distributed Object Computing Conference Workshops (EDOCW), 17th IEEE International, pp. 31 - 38, 2013.

[10] Fernandez, P.M. & Garcia-Serrano, A.M.The role of knowledge-based technology in language applications development. Expert Systems with Applications 19, pp. 31-44, 2000.

[11] S. Kok and P. Domingos, "Learning the structure of Markov logic networks', In Proc. Of ICML-05, Bonn, Germany, ACM Pres, pp. 441–448, 2005.

[12] P. C. R. Lane and J. B. Henderson, "Incremental syntactic parsing of natural language corpora with simple synchrony networks," IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 2, pp. 219-231, 2001.

[13] Imran S. Bajwa, Mark G. Lee, BehzadBordbar, "SBVR Business Rules Generation from Natural Language Specification", Artificial Intelligence for Business Agility -Spring Symposium (SS-11-03), pp. 2-8, 2011.

[14] Arora C, Sabetzadeh M, Briand L, Zimmer F,"Automated Checking of Conformance to Requirements Templates using Natural Language Processing", IEEE Transactions on Software Engineering.DOI 10.1109/TSE.2015.2428709

[15] Falessi D, Cantone G, CanforaG,"Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques", Software Engineering, IEEE Transactions on Vol:39, No:1, pp. 18 - 44, 2013.

[16] Nan Zhou and XiaohuaZhou,"Automatic Acquisition of Linguistic Patterns for Conceptual Modeling", Course INFO 629:Conceptsin Artificial Intelligence,DrexelUniversity.Fall 2004 .
[17] Ambriola V and GervasiV,"On the Systematic Analysis of Natural Language Requirements with CIRCE" Automated Software Engineering, Vol. 13, No.1, pp. 107-167, 2006.
[18] Priyanka More and RashmiPhalnikar. Article:Published by Foundation of Computer Science, New York, USA. Generating UML Diagrams from Natural Language Specifications. International Journal of Applied Information Systems 1(8), pp. 19-23, 2012.
[19] Deeptimahanti D K, Babar M A, "An Automated Tool for Generating UML Models from Natural Language Requirements", 24th IEEE/ACM International Conference on Automated Software Engineering ASE '09. pp. 680 – 682, 2009.
[20] Dinarelli M, Moschitti A, Riccardi G,"Discriminative Reranking for Spoken Language Understanding", Audio, Speech, and Language Processing, IEEE Transactions on Vol: 20, No:2,pp. 526 – 539, 2012.
[21] K. Toutanova and C. D. Manning,"Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger", In Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 63-70, 2000.
[22] Imran SarwarBajwa, M AsifNaeem, On Specifying Requirements using a Semantically Controlled Representation In: 16th International Conference on Applications of Natural Languages to Information Systems. Alicante, Spain: Springer Verlag (NLDB 2011) pp.217-220.
[23] OMG, "Semantics of Business vocabulary and Rules",(SBVR) Standard v.1.0. Object Management Group, Available: http://www.omg.org/spec/SBVR/1.0.(2008).
[24] Kleiner, M., Albert P., BézivinJ.Parsing SBVRBased Controlled Languages. Model Driven Engineering Languages and Systems, pp. 122-136, 2009.
[25] Semantic Formulations in SBVR.Don Baisley,Unisys Corporation, 2005.
[26] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O'Reilly Media, 2009.
[27] James Clarke, Vivek Srikumar, Mark Sammons, and Dan Roth. 2012. An NLP Curator (or: How I learned to stop worrying and love NLP pipelines). In LREC 2012.
[28] David Ferrucci and Adam Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering, 10,pp. 327–348, 2004.

Author Biographies

Mr.Murali Mohanan is an Associate Professor in Department of Computer Science at Model Engineering College Thrikkakara.Kochi. India and at present Research Scholar in Department of Computer Science, CUSAT,Kochi. M.Tech degree holder in Computer and Information Sciences from Cochin University of Science and Technology.

Dr. Philip Samuel is an Associate Professor in Information Technology at Cochin University of Science and Technology, Kochi.He holds a Ph.D degree in Computer Science & Engineering from Indian Institute of Technology, Kharagpur and M.Tech degree in Computer and Information Sciences from Cochin University of Science and Technology. He has several research publications in international conferences and journals. His research interests includes Artificial Intelligence,Distributed Computing, UML Modelling and Design and Software Engineering.