# Development of Use Case Model from Software Requirement using in-between SBVR format at Analysis Phase

**Devendrasingh Thakore[1] & Akhilesh R. Upadhyay[2]**
[1]JJTU, India
[2]Dept.of EC Engineering, Sagar Institute of Research and Technology – Bhopal, 462041(M.P.), India
E-mail : deventhakur@yahoo.com[1], akhileshupadhyay@yahoo.com[2]

**Abstract – Software Analysis process is accomplished by constructing several models of system such as use case model, class model. To take out the basic building blocks such as actor, use cases, and relationships between them use case models from the unstructured textual requirement specification document expressed in English like natural language is not an easy task. There are plenty Nouns (Real Time Entities or actors), verbs or verbs phrases (Events or use cases), in the system requirement document. Also the size of unstructured source requirement document, writing style, present in Natural Language (NL) works as barriers to find out such analysis phase model elements.**

**Thus analyzing requirements and generating the software artifacts to build analysis phase use case model are huge and complex task which need automated support. In the last two decades, major tools that can automatically analyze the NL requirement specification and generate the analysis models are developed. Most of the attempts are concentrating on generation of incomplete class model. Also none of these tools cannot be used in real time software development as they provide with quite less coverage and accuracy (60% to 75%) in generating software artifacts. The key reason of lesser accuracy that has been identified by various researchers is ambiguous and informal nature of NL.**

**To beat some of these issues this paper proposes a techniques. Initially this technique converts the NL requirements in to some formal, controlled middle representation of software requirement such as Semantic Business Vocabulary and Rules (SBVR) Language (Standard introduced by OMG) to increase in accuracy of generated artifacts and model. Then it focuses on identifying the software artifacts such as actors, use cases, relationships between actor and use cases, to generate analysis phase use case models.**

***Keywords – POS Tagging, OOA, UML, Use case, actor, XMI.***

## I. INTRODUCTION

In analysis phases of software development natural language are used to describe the precise business problem need to be resolved. But the natural language are often complex, vague and ambiguous, sentences are vague when they contain generalization. Some time they are missing important information such as subject or object needed by the verb for completeness or contains pronouns. All these difficulty arise when any one discuss the business problem in using natural language. On other hand software requires more precision, correctness and cleanness that are not found in NL. The main Analysis is to capture a entire, definite and consistent picture of the requirement of system and what system must do to satisfy the user requirement and needs. This is accomplished by constructing several models of system. In this process user's needs and wants are transformed in to set off problem statement and requirements specification document called as Software Requirement Specification (SRS) in natural language (NL). After that this natural language (Such as English) SRS are translated to the formal specifications such as UML use case models. This translation consists of generation of structural model of the system such as identifying the actors and related use cases, and relationships among them [1].

However requirement (SRS) explained in NL can often uncertain, imperfect, and incoherent. It is usually job of requirement business analyst to detect and fix potential ambiguities, inconsistencies and incompleteness in such natural language SRS. But due to business analyst can overlook defects in Natural language which can lead to multiple interpretations and difficulties in recovering implicit requirements if analysts do not have enough domain knowledge. Also fault occurred in this stage of software development process can be quite expensive to fix on in later stages of development. Thus evaluating requirements and generating use case model is massive and difficult task which need some automated support.

In automated software development process the software requirements described in natural language are transformed in to some formal specification means that the business models or computation independent models are transformed in to some platform independent module which are very near to the any of the platform specific models or development environment. In last few years there are so many attempts has been made to transform the natural language business models in to platform independent models. But these tools are not used in real time software development process due there lesser accuracy and coverage in generating the formal models of system. Such tools produce 65 to 70 percentages of accuracy and coverage. The main reason behind failure of these tools is ambiguous and casual nature of natural languages. A better solution to this problem is to convert the natural language business model in to some formal representation which is very simple for computation or machine process and also easy to understand by human being as natural language. Semantic Business Vocabulary and rule (SBVR) specification is the standard developed by Object Management Group fulfills this need. This specification defines the vocabulary and rules for documenting the semantics of business vocabularies, business information, and business rules. This specification is applicable to the domain of business vocabularies and business rules of all kinds of business activities of all kinds of organizations.

This paper proposes such approach to analyze, extract, transform and generate software artifacts from natural language business model to build the formal semantic models of the system. Proposed approach first convert the natural language requirement documents in to intermediate SBVR format for better accuracy. Then by doing some semantic and syntactic analysis on such SBVR intermediate result it focuses on identifying the software artifacts such as actors, use cases, relationships between actor and use cases, to generate use case.

## II. BASIC CONCEPTS

In this section, a brief introduction about the basic concepts of the UML Use Case SBVR is provided

### 2.1 UML Use Case Model:

"Use case model is nothing but a sequence of transition in a system whose task is to yield to result of measureable value to an individual actor of the system"

A use case model is graph or diagram of actors, a set of use cases and communication relationships between actor and use cases. Use case model defines the outside (Actor) and inside (Use Case) of system's behavior.

A use case is a special flow of events through the system. An actor is a user playing a role with respect to the system. Actor is a key to findings the correct use cases. Actor carries out the use cases. In use case model single actor can perform many use cases or a use case may have many actors performing it. A use case must help actor to perform a task that has some identifiable value.

### 2.2 UML Class Model:

The UML class model is the main static analysis model. This model shows the static structure of the system to be analyzed. A class model is nothing but the collection of the static modeling artifacts such as classes and their relationships, and multiplicity among them connected as graph to each other and to their contents. The key element of class model is he classes and relationships among them. The class can have sub artifacts as attributes, and methods. Such model represents the mapping of objects in the real world to actual objects to be used in computer program.

### 2.3 SBVR

SBVR is a short form of "Semantic Business Vocabulary and Rules" which has been introduced by Object Management Group (OMG) to reduce the gap between Business analyst and IT persons. This is an contemporary an better way of capturing the business requirements in natural language like structure which is very easy to understand for human beings and also very simple to machine process due to its higher order of logic foundation. One can generate a business model of the system using the SBVR with the same communicative influence of standard natural language. In SBVR all specific expressions and definition of facts and concepts used by an organization in course of business are considered as vocabulary. Also in SBVR a formal presentation under the business influence are considered as rules which are used to express the operation of particular business entity under certain conditions.

## III. BACKGROUND

Many approaches and techniques have been proposed up till now to automate the process of various model generations from natural language requirement specification. However theses approaches are not used in real world system development due to their limitations in coverage and accuracy generation. Also majority of models concentrates on the class model only and require the high order of human interaction to complete the generated models

CM-Builder aims at supporting the analysis stage of development in an Object-Oriented framework. CM-Builder uses robust Natural Language Processing techniques to analyze software requirements texts written in English and build an integrated discourse model of the processed text, represented in a Semantic Network. This Semantic Network is then used to automatically construct an initial UML Class Model. The initial model can be directly input to a graphical CASE tool for further refinements by a human analyst.

CM- Builder analyzes the requirements text and build initial class diagram only. This model can be visualized in graphical case tool by converting it into standard data interchange format where human analyst can make further refinements to generate final class model. Also CM-builder makes the extensive use of NLP techniques.

A Natural Language Object Oriented Production System (NL- OOPS) [3] generates object oriented analysis model from SemNet obtained by parsing NL SRS document. It considers noun as objects and identifies the relationships among objects using links. This approach lacks in accuracy in selecting the objects for large systems and cannot differentiate between class nouns and attribute nouns.

Linguistic assistant for Domain Analysis (LIDA), provide linguistic assistance in the model development process. It presents a methodology to conceptual modeling through linguistic analysis. Then gives overview of LIDA's functionality and present its technical design and the functionality of its components. Finally, it presents an example of how LIDA is used in a conceptual modeling task.

This tool identifies model elements through assisted text analysis and validates by refining the text descriptions of the developing model. LIDA needs extensive user interaction while generating models because it identifies only a list of candidate nouns, verbs and adjectives, which need to be categorized into classes, attributes or operations based on user’s domain knowledge.

"GOOAL" (Graphic Object Oriented Analysis Laboratory) [5] receives a natural language (NL) description of problem and produces the object models taking decisions sentence by sentence. The user realizes the consequences of the analysis of every sentence in real time. Unique features of this tool are the underlying methodology and the production of dynamic object models. GOOAL produces the class diagram by considering the validation threshold of 50% and its coverage accuracy (Precision matrices) is very minimum that is 78%

NL-OOML [6] presents an approach to extract the elements of the required system by subjecting its problem statement to object oriented analysis. This approach starts with assigning the parts of speech tags to each word in the given input document. The text thus tagged is restructured into a normalized subject-verb - object form. Further, to resolve the ambiguity posed by the pronouns, the pronoun resolutions are performed before normalizing the text. Finally the elements of the object-oriented system namely the classes, the attributes, methods and relationships between the classes, the use-cases and actors are identified by mapping the ‘parts of speech- tagged’ words of the natural language text onto the Object Oriented Modeling Language elements using mapping rules. But approximately 12.4 % of additional classes and 7.4 % of additional methods are identified in all the samples taken each of around 500 words. These additionally identified candidates are those that will usually be removed by human by intuition. Since the system lacks this knowledge, they were also listed as classes. Coverage accuracy is 82%

## IV. PROPOSED SYSTEM METHODOLOGY

This section describes the used methodology to identify the artifacts which are used to generate the models at analysis phase from natural language. This methodology consist of automatic conversion of natural language software requirement specification conversion to controlled intermediate SBVR format and secondly to identification of software artifacts and model generation, finally visualization of generated models. Used methodology works in different phases organized in pipelined fashion as follows.

1. Preprocess Analysis

This phase stars with the by reading the given English input and tokenizing the whole input in to individual tokens. To do so java tokenizer class is used. After tokenizing each token is stored in separate array list. While tokenizing the English input sentence splitter is used to identify the boundary of each sentence.

2. Tagging

This processed text is further given as input to Part Of Speech (POS) tagger to identify the basic POS tags. To do so Standard POS tagger is used which identifies the 44 basic POS tags.

3. Morphological Analysis

To remove the suffixes attached to noun phrases and verb phrases this type of analysis is performed on the tagged output from pervious phase. In this type of analysis WordNet is used to convert the plural into singular form also suffixes attached to verb phrases such as “ed” are also removed.

4. Pronoun Resolution

In this phase JavaRAP is used to replace all possible pronouns with correct noun form up to third person.

5. Parse Tree Generation

Stanford Parser is used to generate parse tree from pos tagged output for each requirements. This phase is very useful to find out artifacts such as actors, use cases to model the use case diagram.

6. Role Labeling and Element/Concept Identification

In this phase role labels are identified from preprocessed text such as performer, co actors, events, objects and receiver in the sentences. Also in this phase SBVR concept identification is done according to some identification constraints such as all proper nouns are identified to individual concepts, all common nouns are identified as noun concepts or object type, all action verbs are identified as verb concepts, all auxiliary verbs are identified as fact types, possessed nouns are identified as characteristics or attributes, indistinct articles, plural nouns and cardinal numbers are identified as quantification. Output of this phase is stored in an array list.

7. Rule Generation

To generate the SBVR rule we have to first produce fact types, in the form of sentences which represents some relationships between the concepts identified in the previous phase. For that purpose use the template such as noun-verb-noun to establish the relationship between two concepts. Thus a fact type is created by combining the noun concepts and verb concepts from pervious phase array list. Generated fact type is used to create the SBVR rule by applying various logical formulation such as use of logical expression AND, OR and NOT etc, Quantification token conversion rules, possibility and obligation formulation rules are used.

8. Applying Notations

In this phase SBVR notations are applied to generate rules such as noun concepts are underlined, verb concepts are italicized, keywords are bolded, individual concepts or attributes are double underlined.

9. Artifacts Extraction

In this phase produced SBVR vocabulary and rules are further processed to extract the basic building blocks or artifacts of use case. All SBVR noun concepts and object type are tends to be actor for use case model. All verb concepts associated to noun concepts are tend to be use cases of that actor for use case model. Association between actor and use cases are identified with the help of parse tree generated as well as from the SBVR unary fact types in the form of template noun-verb or binary fact types created in phase

Following figure 1 shows the process architecture of the proposed methodology.

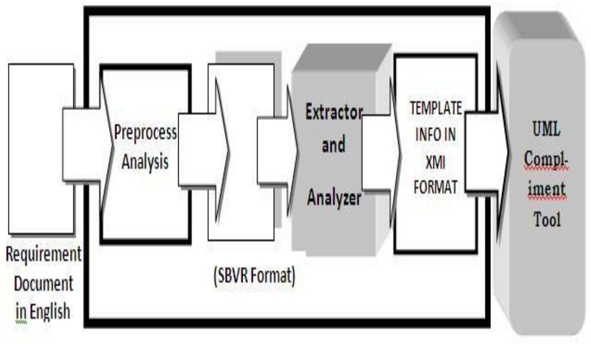
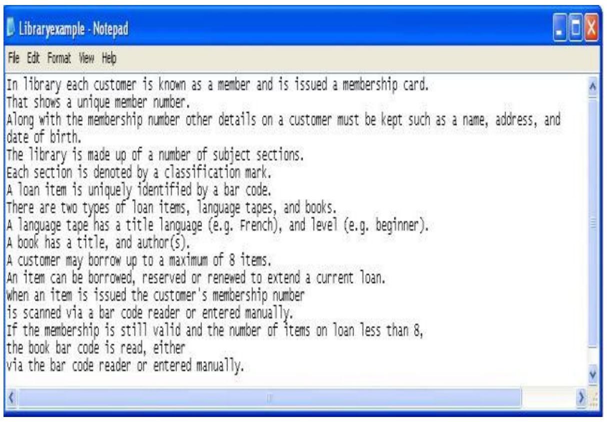


Fig. 1 : Process Architecture of System

## V. IMPLEMENTATION DETAILS

Steps are carried out during implementation of Generation of Software Artifacts at Analysis Phases is explained below along with example.

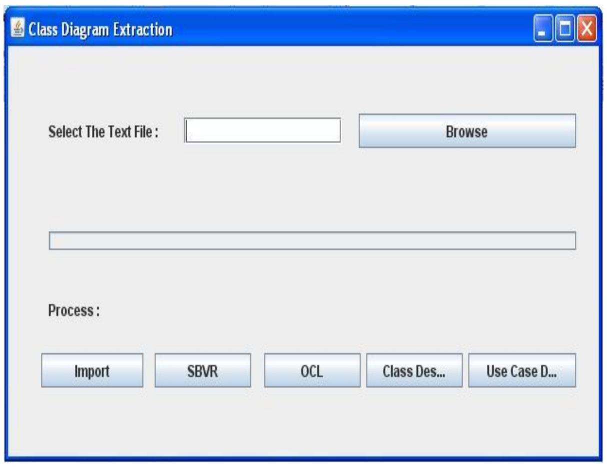


```
In library each customer is known as a member and is issued a membership card.
That shows a unique member number.
Along with the membership number other details on a customer must be kept such as a name, address, and
date of birth.
The library is made up of a number of subject sections.
Each section is denoted by a classification mark.
A loan item is uniquely identified by a bar code.
There are two types of loan items, language tapes, and books.
A language tape has a title language (e.g. French), and level (e.g. beginner).
A book has a title, and author(s).
A customer may borrow up to a maximum of 8 items.
An item can be borrowed, reserved or renewed to extend a current loan.
When an item is issued the customer's membership number
is scanned via a bar code reader or entered manually.
If the membership is still valid and the number of items on loan less than 8,
the book bar code is read, either
via the bar code reader or entered manually.
```

Fig. 2: Input File Contains Requirements in Natural Language

Step 1: The text file contains requirements in natural language is as shown in figure 2 and is given as input to the Generation of Software Artifacts at Analysis Phases system. The file contains n numbers of words can be given as input.

Step 2: To given above file as input to system press “Browse” button and then press “Import” button to start importing the natural language requirement inside the system.



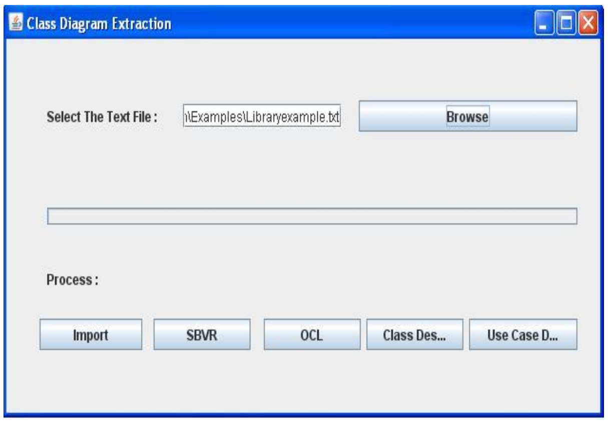


Fig. 3: Selecting the Text File for I/P to System

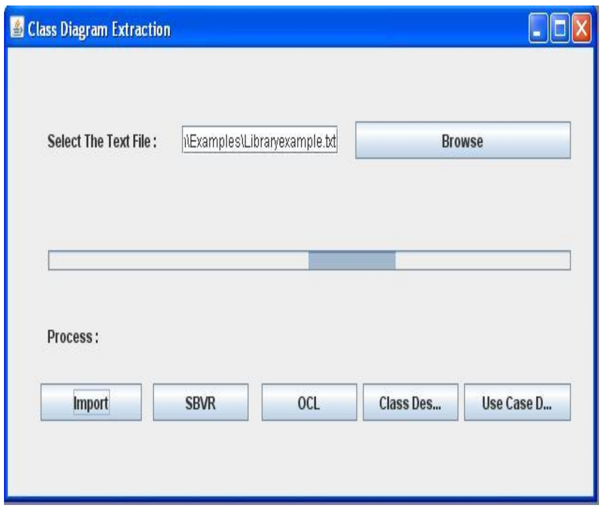
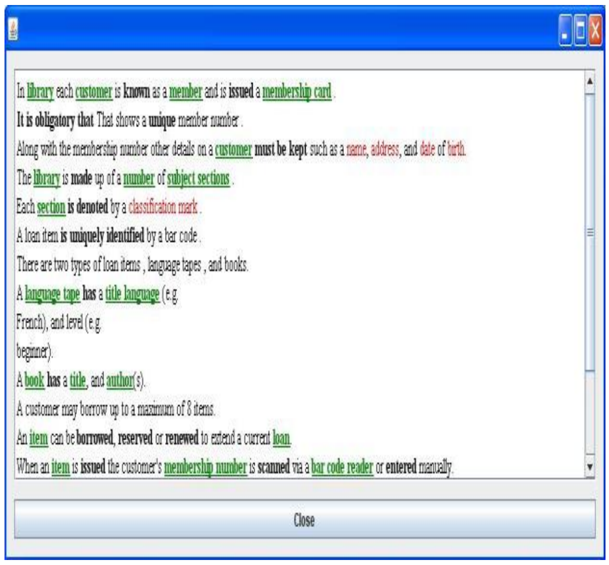


Fig. 4: Importing the Text File as I/P to System

Step 3: Once the input is accepted by the system, it converts the given input requirements in to SBVR format after pressing "SBVR" Button. It is shown in figure 5 as below.



In library each customer is **known** as a member and is **issued** a membership card .
**It is obligatory that** That shows a **unique** member number .
Along with the membership number other details on a customer **must be kept** such as a name, address, and date of birth.
The library is **made** up of a number of subject sections .
Each section **is denoted** by a classification mark .
A loan item **is uniquely identified** by a bar code .
There are two types of loan items , language tapes , and books.
A language tape **has** a title language (e.g.
French), and level (e.g.
beginner).
A book **has** a title, and author(s).
A customer may borrow up to a maximum of 8 items.
An item can be **borrowed, reserved** or **renewed** to extend a current loan.
When an item is **issued** the customer's membership number is **scanned** via a bar code reader or **entered** manually.

Fig. 5: SBVR Representation of Inputted Requirements in NL.

Step 4: SBVR Representation is Scanned and Analyzed to generate the Use Case Model of Requirement in NL by pressing the Button "Use Case Diagram" and is shown in figure 6 as below.

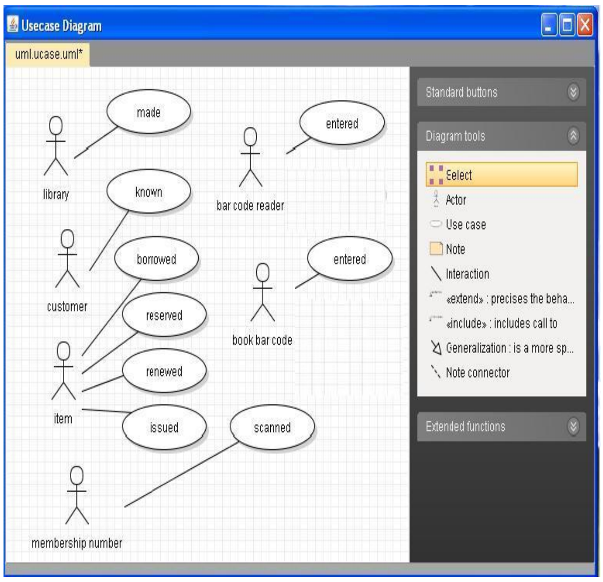


Fig. 6: Generated Use Case Model for Requirement in NL

## VI. RESULT ANALYSIS

An evaluation methodology proposed by Hirschman and Thompson [8] is used for the performance evaluation. According to this methodology the most enduring metrics of performance that have been applied to information extraction are termed as recall (Coverage of tool) and precision (Coverage Accuracy of tool).These metrics may be viewed as judging effectiveness from the application user's perspective.

Recall = no of Relevant-returned facts / actual relevant facts

Precision = no of relevant-returned facts / total no. of returned facts

Output generated by our system for different case studies is analyzed, in comparison with, output prepared by experts manually.

To evaluate the results of System, each outcome (Actor, use case, associations) of the systems output was matched with the experts opinion (Nsample) (manually generated sample solution). The outcome that accurately classified into respective category was declared correct (Ncorrect) otherwise incorrect (Nincorrect). Additionally, the information that was not extracted (or missed) by the system but it was given in the human experts opinion (Nsample) was categorized as the missing information (Nmissing). The calculated recall and precision values of the solved case study are shown in table 1.

Table 1: Result Analysis of System

| Case Study | NSample | NCorrect | NIncorrect | Nmissing | Recall % | Precision% |
|---|---|---|---|---|---|---|
| Library System Case Study | 76 | 72 | 6 | 4 | 94.73 | 92.30 |
| Keypass System Case Study | 44 | 42 | 3 | 2 | 95.45 | 93.33 |

The statistics of our system in Table 1 shows that this system is able to extract almost 90% accurate results that are matching with models generated manually. Almost 6 % - 8% are incomplete, which are extracted by this system and nearly about 5% are missing. Missing artifacts could be the result of differences in perspectives of every expert.

## VII. COMPARATIVE ANALYSIS

Many approaches and techniques have been proposed up till now to automate the process of various model generations from natural language requirement specification. However theses approaches are not used in real world system development due to their limitations in coverage and accuracy generation. Also majority of models concentrates on the class model only and require the high order of human interaction to complete the generated models

Following table 2 shows the comparison of results of available existing system with proposed system that can perform automated or semi-automated analysis of the Natural Language Requirement Specifications in to model creation. Recall value was not available for some of the tools.

Table 2: A Comparison of Performance Evaluation with Existing System

| Tools | Recall Value | Precision Value |
|---|---|---|
| CM-Builder (Harmain, 2003) | 73.00% | 66.00% |
| GOOAL (Perez-Gonzalez, 2002) | - | 78.00% |
| NL-OOML (Anandha, 2006) | - | 82.00% |
| LIDA (Overmyer, 2001) | 71.32% | 63.17% |
| Extract (only Event Extraction) (2009) | 92.00% | 85.00% |
| Implemented System | 95.09% | 92.82% |

As the existing system uses natural languages as direct input to tool so that their recall and precision values are very less. Such results are there due to problems associated with Natural Languages such as

- Ambiguous and informal nature
- Inherent semantic inconsistencies
- Complex to machine process.
- Informal sentence structure

## VIII. CONCLUSION

The primary objective of the dissertation was to deal with the dispute of addressing confusing nature of NL (such as English) and generate a restricted representation of English so that the accurateness of software processing can be improved. To tackle this challenge this system presented a NL based automated approach to tag and parse English software requirements specifications and generated a controlled demonstration. Automated object oriented analysis of SBVR specifications of software requirements using this system provides a superior accuracy.

This approach describes a computerized way to take out the software element at analysis phase. It uses Natural Language Processing techniques to consider business level software requirements and builds an incorporated analysis level model.

This approach can be used for the identification of software elements such as event list, use cases, classes, their attributes, and the static relationships among them with increase in accuracy due to use of intermediate format SBVR.

This Approach brings us to a very important outcome which allows the business people work independently of IT to analyze, design and build up their system.

This approach can be used for the identification of software elements such as event list, use cases, classes, their attributes, and the static relationships among them with increase in accuracy due to use of intermediate format SBVR. The outcome achieved have shown the utility of this approach

- Across all domain over unlimited requirement size expressed in Natural Language to generate analysis phase software elements and models.
- in understanding functional requirements because it gives list of events that represents the behavior of system
- For summarization and for extraction of important software elements in Objects Oriented Analysis Process.

## IX. REFERENCES

[1] Ali Bahrami, Chapter 6, Object Oriented Analysis Process, in Object Oriented System Development.

[2] H. M. Harmain and R. Gaizauskas, CM-Builder: An Automated NL Based CASE tool, in IEEE International Conference on automated software engineering (2000)

[3] Mich L., NL-OOPS: From natural language to object oriented requirement using natural language processing system (1996)

[4] Overmyer, S. P., Benoit, L. and Owen R., Conceptual modeling through linguistic analysis using LIDA. International Conference of Software Engineering (ICSE), (2001)

[5] Hector G perez-Gonzalez and Jugal K. Kalita, GOOAL : A Graphical Object Oriented Analysis laboratory, ACM 1-58113-626-9/02/0011 (2002)

[6] G.S. Anandha Mala, J. Jayaradika, and G. V. Uma, Restructuring Natrual Language Text to Elicit Software Requirements, in proceeding of the International Conference on Cognition and Recognition (2006)

[7] Sanddep K. Singh, Reetesh Gupta, Sangeeta Sabharwal, and J.P. Gupta, E-xtract : A tool for extraction, Analysis and Classification of Events from Textual Requirements, in IEEE 2009 international Conference on Advances in Recent technologies in communication and Computing.

[8] Hirschman L., and Thompson, H.S. 1995. Chapter 13 Evaluation: Overview of evaluation in speech and natural language processing. In Survey of the State of the Art in Human Language Technology.

[9] OMG. 2008. Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0.

◈◈◈