International Journal of Open Source Software and Processes
Volume 10 • Issue 2 • April-June 2019

Requirements to Class Model via SBVR: RECM via SBVR TOOL

Murali Mohanan, Cochin University of Science and Technology, Kochi, India
Imran Sarwar Bajwa, Department of Computer Science and IT, The Islamia University of Bahawalpur, Bahawalpur, Pakistan

ABSTRACT

A user's software requirements are represented in natural language or a speech such as English. Translating these requirements into the object-oriented models is a tough process for designers. This article proposes a neoteric approach to generate Unified Modeling Language (UML) class models instantly from software requirement specifications (SRS). Here the authors make use of the Open Natural language processing tool (OpenNLP) for lexical analysis and to generate the necessary parts of speech (POS) tags from these requirement specifications. Then, the Semantics of Business Vocabulary and Rules (SBVR) standard is used to extract the object-oriented elements from the natural language (NL) processed SRS. From this, the authors generate UML class models. The prototype tool can generate accurate models in less time. This automated system for designing object-oriented models from SRS reduces the cost and budget for both the designers and the users.

KEYWORDS

Natural Language Processing, Object Oriented Modeling, Requirement Analysis, Software Requirement Specification, UML Class Diagrams

1. INTRODUCTION

Modern methods in information technology enable software engineers to develop various software quickly and effectively with different styles. In this paper, the problem addressed is related to the requirement analysis in the software development phase. Recent trends of software engineering largely depend on object-oriented design paradigm that uses the Unified Modeling Languages (UML). UML is used for modeling the user software requirements, documenting the software assets, software development and redevelopment (Perez-Gonzalez, 2002). The UML class model is the core for object-oriented analysis and design. The existing tools such as ReBuilder (Oliveria et al., 2006), CM-Builder (Harmain & Gaizauskas, 2003), GOOAL (Perez-Gonzalez & Kalita, 2002), NL-OOML (Anandha & Uma, 2006) and UML-Generator (Bajwa et al., 2009) attempt to generate UML class diagrams automatically from the natural languages.

The problem with these tools is that they generate the class diagram with lower accuracy due to the informal nature of NL and its ambiguity (Li et al., 2005; Mich, 2001). This paper is focused on designing the perfect class models from the user requirements. The user specifies their requirements

DOI: 10.4018/IJOSSP.2019040104

in natural languages such as English. Since natural language processing is a difficult task this research work is divided into two phases. The initial phase is parsing and in this phase, we make use of OpenNLP (Mohanan et al., 2016). The second phase is a transformation phase and for transformation phase, Semantics of Business Vocabulary and Rules (SBVR) (Feuto et al., 2013) process is implemented.

The OpenNLP is used to produce the POS (parts of speech) (Toutanova & Manning, 2000) tags of SRS which is in the form of a natural language such as English. The POS tags contain the required details such as noun, verb, adverb, etc., of the sentences of SRS. We use Apache OpenNLP to implement the pre-processing phase and to carry out Sentence splitting, tokenization and POS tagging. In the second phase Semantics of Business Vocabulary and Rules process generates the vocabulary and rules. Since SBVR has natural language syntax it provides a suitable way to capture the Object-Oriented items in requirement specifications and also easy to understand by both the user and machine (Fernandez et al., 2000; Lane & Henderson, 2001; Bajwa et al., 2011). The SBVR vocabulary extraction, SBVR rules generation, Object Oriented Analysis of SBVR rules and UML class model generation are the phases in SBVR process. The remainder of this paper is organized as below.

Section 2 describes some related works which are available in the literature. Section 3 describes the class identification method, its preliminaries, explains the SRS to class identification using SBVR, object oriented analysis from the SBVR rule and class diagram generation. Section 4 explains the evaluation of our methodology followed in this paper and includes experimental results of various case studies of this work which are used to demonstrate precision and recall of the proposed concept. Section 5 concludes this paper.

2. RELATED WORK

The natural language processing is a research area in which various research methodologies are proposed and quite a few methods are for analyzing the software requirements. Some of the researchers focused on class diagram extraction from the NL requirements (Arora et al., 2015; Falessi et al., 2013). This section describes the survey of some methods that uses NL processing or domain ontology for NL requirement analysis and to generate the class models.

Nan Zhou and Xiaohua Zhou (Zhou & Zhou, 2004) proposed an automated system to generate the class diagram from the user requirement document. This approach used natural language processing to analyze the written requirements and domain-based ontology 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 find more classes and its attributes. The final output is filtered by the domain ontology. Ambriola and Gervasi (Ambriola & Gervasi, 2006) designed a framework to develop the models such as Entity Relationship diagram, Data Flow Diagram, even UML diagrams. The system takes the user requirement written in natural language as input and applied the CICO parser for parsing and information extraction. A basic model called CIRCE Native Meta-Model is derived from the requirements for generating the UML class diagram and Entity Relationship Diagram.

Priyanka More and Rashmi Phalnikar (More & Phalnikar, 2012) proposed an approach to design the UML diagrams from the informal natural language requirements. A tool named 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. The class is identified by the class extraction engine. The class diagram is generated from the RAPID concept management Deeptimahanti and Babar (Toutanova & Manning, 2000) 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. Then a generic

XML parser is applied for the visualization of the generated model. Model generated by the UMGAR can be viewed in any UML Modeling tool and this approach is used for very large application.

It was R. J. Abbot (1983) 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 recognized from nouns and the methods of the classes can be recognized from verbs in the sentence. Here, the Natural Language Processing System LOLITA (Large-scale Object-based Linguistic Interactor Translator Analyser), a system developed at Durham University (Long and Garigliano,1994) was used as a core system.

Imran Sarwar Bajwa and M. Abbas Choudhary (Bajwa et al., 2006) proposed a natural language processing system based on automatic generation of UML diagrams. This system provided the UML diagrams by analyzing the given business details in text format. Rule based systems are utilized for analyzing the natural languages and extracting required information from the given story line by the user. After compound analysis and extraction of associated information, the proposed system generates several UML diagrams like activity diagrams, sequence diagrams and class diagrams.

Hakan Burden and Rogardt Heldal (2011) generated the textual descriptions from the class diagram. Initially the author transformed the class diagram into an intermediate linguistic model using Grammatical Framework. Then the linguistic model is further transformed into natural language text. A Platform-Independent Model (PIM) was improved in such a way that the generated texts can both paraphrase the original software models and includes the basic motivations behind the design decisions. The mappings from marked PIM to natural language Platform Specific Model (PSM) are generic which can be applied to any marked PIM. Brosch and Randak (2010) have designed a system to transform the class diagrams into natural language texts based on standard phrases in natural language. Their system differs from which marks all model elements along with the equivalent linguistic realization. This system allows the user (teacher) to automatically generate the textual specification from the sample solution of an exercise.

Imran Sarwar Bajwa (2007) and Irfan Hyder (2007) present a natural language processing-based approach (Language Engineering System for Semantic Analysis (LESSA). This LESSA is utilized to understand the natural language text and capture needed information automatically. This information is used to draw the Use Case diagrams. The User writes his interface in simple English in a few paragraphs and the proposed system analyzes the given script. After compound analysis and extraction of associated information, this approach draws the class diagrams (OMG, 2008) effectively.

Overmyer, Benoit, and Owen proposed a prototype tool, Linguistic assistant for Domain Analysis (LIDA), that gives linguistic assistance in the model development process. The authors initially present the new methodology to conceptual modeling through linguistic analysis. The overview of LIDA's functionality, technical design and their component functionality are described in their approach.

3. CLASS IDENTIFICATION FROM SRS USING SBVR

Aiming to give a suitable support for software developers as well as users, we have proposed a neoteric approach for automatic generation of Class models from user software requirements. Proposed framework provides a robust solution for object-oriented modelling which supports both the users and software engineers. The user software requirements are usually written in the natural language or speech language such as English which is asymmetric and irregular (Abbott, 1983). Natural Language Processing (NLP) tool implemented here processes the user requirements and provides the necessary parts of speech (POS) (Toutanova & Manning, 2000) tagging. By using the SBVR process, object-oriented elements like classes, objects, attributes, relationships, etc., are extracted from the NL processed SRS. The SBVR process includes SBVR vocabulary extraction and rule generation. This can be further refined to form a class diagram which depicts the structural aspects of a software system.

3.1. Preliminaries

In this subsection two popular state of the art technologies for natural language processing are introduced. They are as follows.

3.1.1. Open Natural Language Processing (OpenNLP)

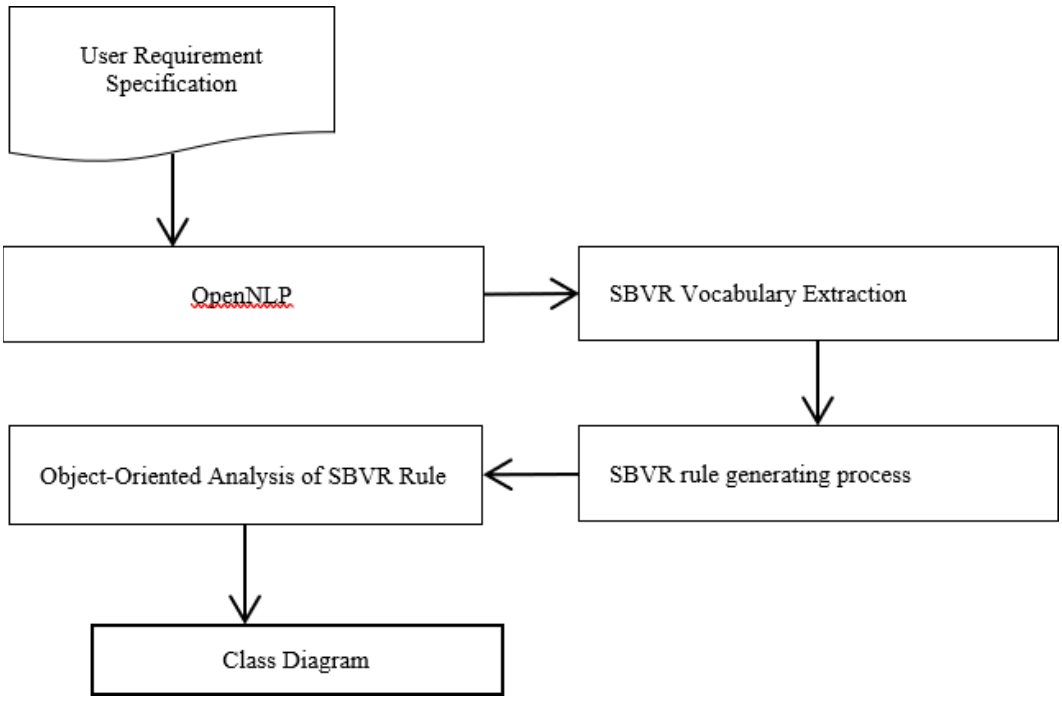
The OpenNLP (Bajwa & AsifNaeem, 2011) aims to obtain how computer understands and process the natural language. It is really fast to implement. The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. The common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolutions are being done using this tool. OpenNLP project is developed to create a mature toolkit for the above-mentioned NLP tasks.

3.1.2. Semantics of Business Vocabulary and Rules

The Object Management Group (OMG) introduced the Semantics of Business Vocabulary and Rules (ThiTouiPhan, 2015) 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 a rich capability of languages that are used by the people in various domains. The Object Management Group proposal called Semantics of Business Vocabulary and Business Rules (SBVR) (Mohanan et al., 2016) offers a vocabulary for describing the meaning of a sentence. A part of SBVR called Logical Formulation of Semantics focuses on the structure of meaning.

The proposed approach framework is as shown in the Figure 1. 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

Figure 1. Proposed approach framework



to the syntax and semantic analysis by parsing each and every sentence. The parser removes the stop words which have 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. Figure 2 shows the generated POS tagger (Toutanova & Manning, 2000).

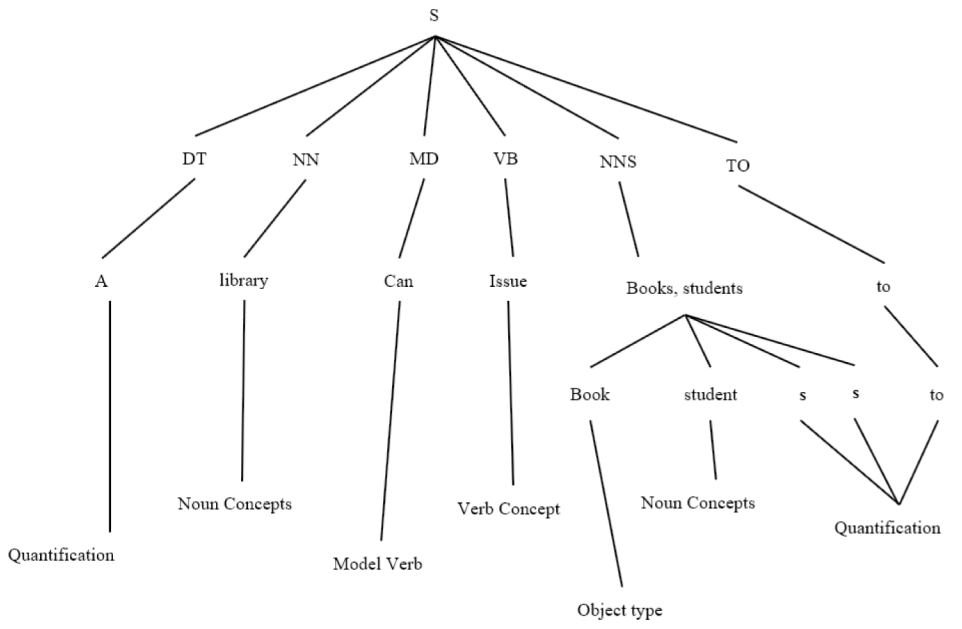
3.2. SBVR Vocabulary Extraction Model

Semantics of Business Vocabulary and Rules generates the vocabulary and rules for a business domain. In our research work, it helps in identifying the various object-oriented items from natural language processed requirements specification. Thus, SBVR provides a suitable way to capture the object-oriented items from the requirements specification (Bajwa & AsifNaeem, 2011). SBVR does not include quantifiers or logical operators, which are symbols, but has the concepts of quantification and conjunction. SBVR consists of all the specific terms and definitions of concepts used by an organization. In SBVR, a concept can be a noun concept or fact type. 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 concepts mentioned in SBVR 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 (Mohanan et al., 2016). Plural nouns (prefixed with s), articles (a, an, the) and cardinal numbers (2 or two) are considered as Quantification.

The Associative fact types are identified by the binary fact type in POS tagging. To understand the binary fact type the following sentence i.e. "*The belt conveys the parts"* is considered. An association is there in the above sentence between the words *belts* and *parts.* In the SBVR model, object relation association is mapped to associative fact types, aggregation is considered as partitive fact types and

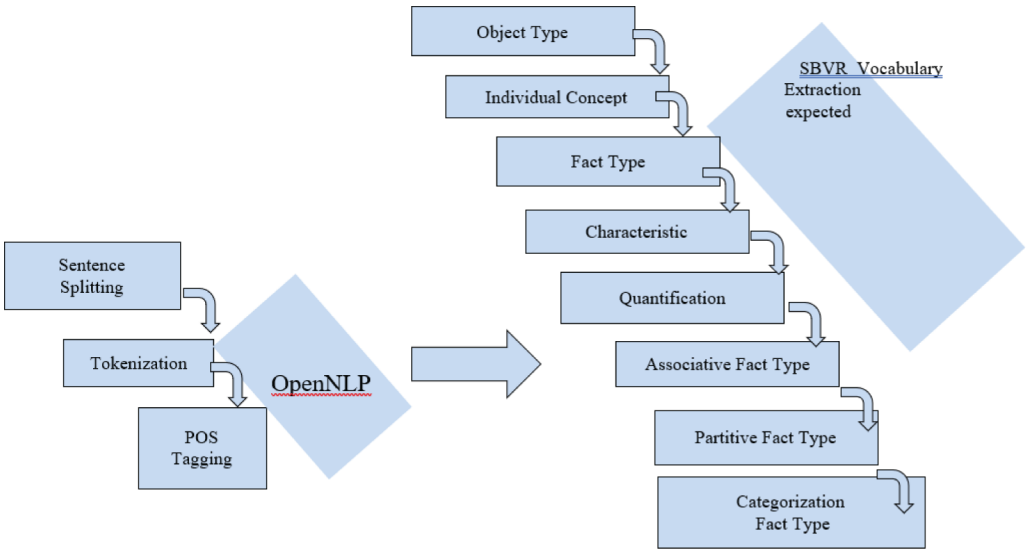
Figure 2. Parse tree representation for SBVR vocabulary extraction



the generalization is denoted as categorization fact types. In this phase, the basic SBVR elements e.g. noun concept, individual concept, object type, verb concepts, etc. are identified from the natural language text input that is preprocessed by Open Nlp process. As shown in the Figure 3 Sentence Spliting, Tokenization and POS tagging are the major steps used in this process. The extraction of various SBVR elements is as described below:

- **Object Types Extraction:** All common nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the object types and are mapped to classes;
- **Individual Concepts Extraction:** All proper nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the individual concepts;
- **Fact Types Extraction:** The auxiliary and action verbs are represented as verb concepts. The combination of an object type/individual concept + verb forms a unary fact type e.g. "vision system *senses*". Similarly, the combination of an object type/individual concept + verb + object type forms a binary fact type e.g. belt *conveys* part is a binary fact type;
- **Characteristics Extraction:** The characteristic or attributes are typically represented using *is-property-of* fact type e.g. "name *is-property-of* customer". Moreover, the use of possessed nouns (i.e. pre-fixed by *'s* or post-fixed by *of*) e.g. student's age or age of student is also characteristic;
- **Quantifications Extraction:** The key-words such as "Each" or "All" represent SBV Runiversal quantifications. All indefinite articles (*a* and *an*), plural nouns (prefixed with s) and cardinal numbers (2 or two) represent SBVR non-universal quantifications;
- **Associative Fact Types Extraction:** The associative fact types are identified by associative or pragmatic relations in English text. In English, the binary fact types are typical examples of associative fact types e.g. "The belt conveys the parts". In this example, there is a binary association in belt and parts concepts. This association is one-to-many as '*parts'* concept is plural and associative fact types are mapped to associations;
- **Partitive Fact Types Extraction:** The partitive fact types are identified by extracting structures such as "*is-part-of*", "*included-in*" or "*belong-to*" e.g. "The user puts two-kinds-of parts, dish and cup". Here 'parts' is generalized form of '*dish'* and '*cup'* and categorization fact types are mapped to aggregations;

Figure 3. SBVR vocabulary extraction from OpenNLP output

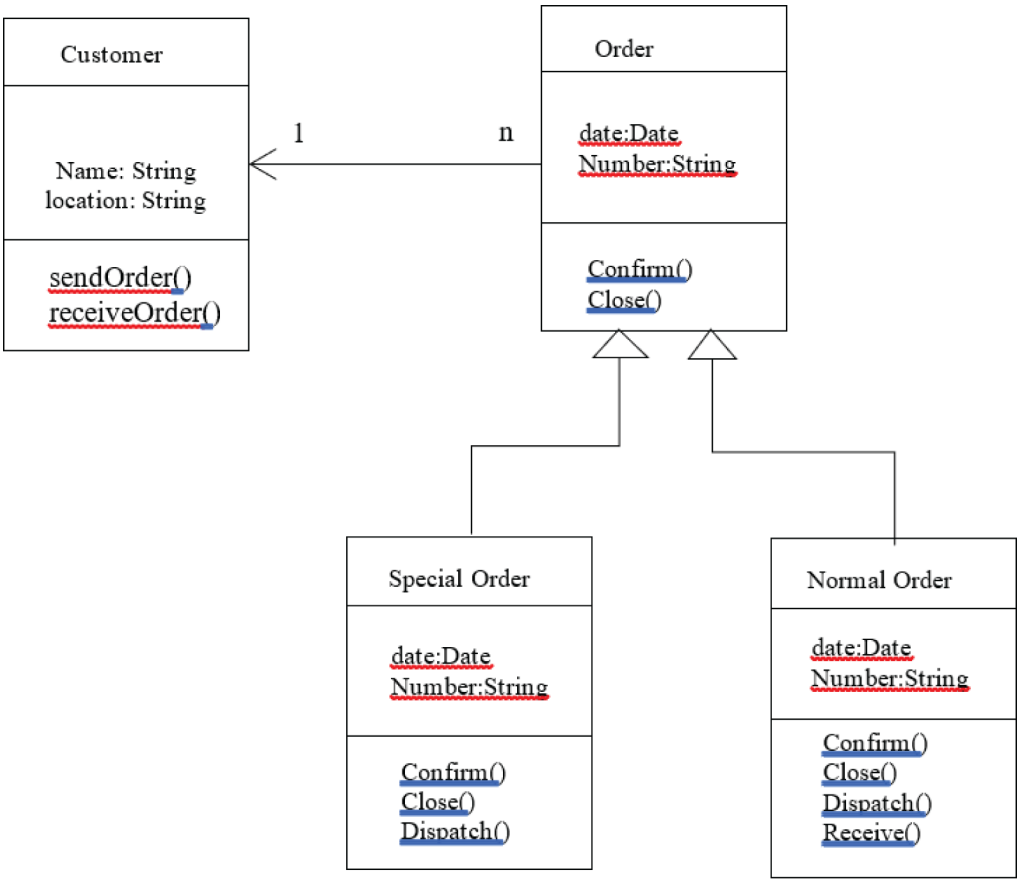


- **Categorization Fact Types Extraction:** The categorization fact types are identified by extracting structures such as "*is-category-of*" or "*is-type-of*", "*is-kind-of*" e.g. "The user puts two-kinds-of parts, dish and cup". Here 'parts' is generalized form of '*dish'* and '*cup'* and categorization fact types are mapped to generalizations. The above mentioned SBVR elements i.e. various fact types and object types are extracted from the output of the OpenNLP and it is shown in Figure 3.

In this paper, the UML class model is considered as the business domain. According to the UML class model the noun concepts are class names, their respective attribute names and the object names are denoted as individual concepts, their operation names are considered as action verbs and finally the fact types are referred to associations and generalizations. A simple UML class diagram example is shown in Figure 4. In the figure shown, customer, order, special order and normal order are noun concepts, date and number are attributes. Similarly confirm, close, dispatch are verb concepts, and the association ownership is fact type.

To generate the UML class model SBVR rule has to be extracted to analyze the specific software logic. The SBVR rule is based on any one of the fact type of SBVR vocabulary. Structural rule and behavioral rule are the two types (Bajwa & AsifNaeem, 2011; Mohanan et al., 2016) of SBVR rules. The structural rule defines the organizational setup whereas behavioral rule describes the conduct of an entity. Semantic formulations like logical formulation, quantification and modal formulation

Figure 4. UML class diagram of a purchasing system



are processes to be performed to generate the SBVR rules from the fact type. This process is shown in Figure 5. The SBVR rule is constructed by applying the semantic formulation to each fact type.

3.2.1. Logical Formulation

Using logical operators, the multiple fact type is composed for SBVR rule. 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 $(\urcorner)$. That, and are denoted as conjunction $(\wedge)$ similarly or is disjunction $(\vee)$ and tokens like infer, imply, indicate, suggest are considered as the logical operator implication $(\rightarrow)$.

3.2.2. Quantification

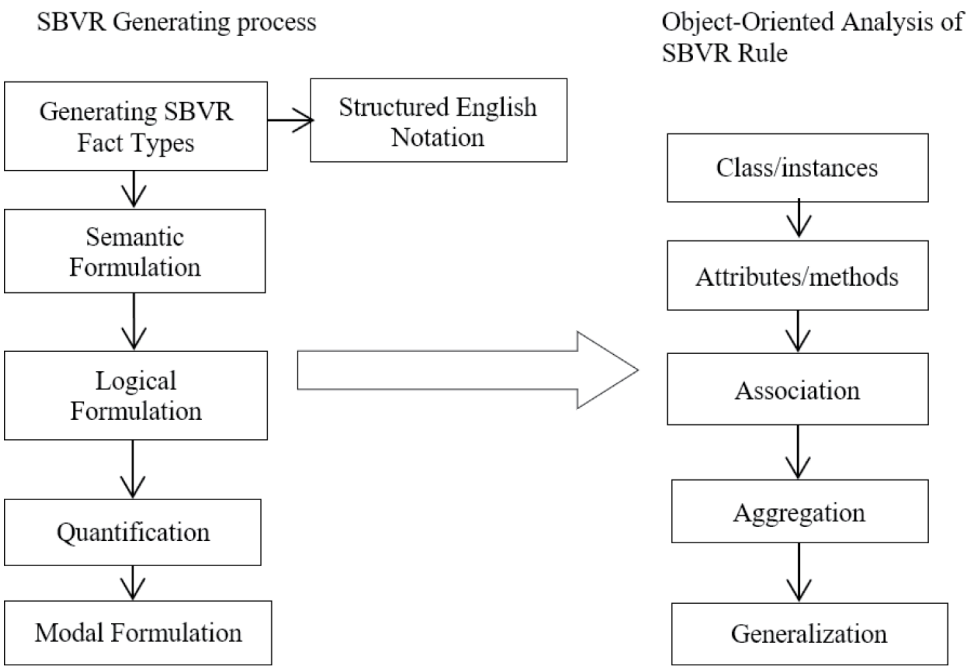
Quantification mentions the scope of the concept and it is applied in this work by mapping the tokens as below:

More than, greater than $\rightarrow$ at least n quantification
Less than $\rightarrow$ at most n quantification
Equal to, positive statement $\rightarrow$ n quantification

3.2.3. Modal Formulation

The modal formulation describes seriousness of a constraint and it formulates modality. In SBVR two modal formulations are there, one is possibility formulation (PF) and the other 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 is as shown below:

Figure 5. SBVR rule generation and object-oriented element extraction



Can, may → PF
Verb concept, should → OBF

3.2.4. Structured English Notation

In this phase the notations are provided for certain tokens. For examplem the noun concepts, verb concepts and SBVR keywords are underlined, italicized and bolded respectively.

3.3. Object Oriented Analysis of SBVR Rule

The final step of the proposed work is object-oriented analysis from the SBVR rule to extract the object-oriented elements such as classes and its attributes, objects, methods, generalization, aggregation and associations. This extraction procedure is shown in Figure 5. In the SBVR rule the generic entity is represented by the noun concept and on this basis the noun concept is mapped to classes. Similarly, the particular entity is obtained from the individual concept and 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. Object relation association is extracted by the unary relationship, binary relationship and multiplicity. So as per the SBVR rule we can map as shown below:

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 object-part is subclass 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.

The Final process is UML class model generation. The class model for the problem statement is as shown in Figure 9. From the considered case study 1 the identified classes are cup, dish, assembly, parts, and sensor. The attributes identified are consists, puts, enters, conveys, and informs. Having this information our tool generates the class diagram (OMG, 2008).

Aiming to develop a prototype tool that makes use of natural processing techniques for analysing software requirement specifications and to generate UML Class models directly from them we have implemented RECM (Requirements to Class Model) tool. Our tool uses Apache OpenNLP library for processing of SRS and SBVR standard for NL to object oriented transformation. RECM tool provides simple and easy to use GUI. Some of these user interfaces are shown as screen shots in the case study evaluation.

4. PERFORMANCE EVALUATION

A performance evaluation is carried out to evaluate that how accurately the English specification of the software requirements in the form of behavioural and structural statements have been translated into the SBVR based controlled representation by our tool. An evaluation methodology proposed by Hirschman and Thompson (2009) is used for the performance evaluation of our tool. The used performance evaluation is based on three aspects as mentioned below:

1. **Criterion:** It specifies the interest of evaluation. e.g. precision, error rate, etc.;
2. **Measure:** It specifies the particular property of system performance someone intends to get at the selected criterion. e.g. percent correct or incorrect;

3. **Evaluation method:** It determines the appropriate value for a given measure and a given system.

4.1. Evaluation Methodology

1. **Criterion:** A criterion was defined that how close $N_{correct}$ (named correct result) are the tool output to the opinion of the human expert N_{sample} (named sample results). Different human experts produce different representations and can be good or bad analysis. Here the human expert's opinion for the target input is considered as a sample result;
2. **Measure:** We have used two evaluation metrics which are commonly used for NLP applications i.e. recall and precision. These metrics are extensively employed to evaluate NL based knowledge extraction systems. These metrics are defined as follows:
   a. **Recall:** "The completeness of the results produced by the system is measured using recall". Recall can be calculated by using the following formula:

$$Recall = \frac{N_{correct}}{N_{sample}} \tag{1}$$

where $N_{correct}$ is the number of correct results generated by the tool and N_{sample} is the number of sample results (opinion of human expert).

   b. **Precision:** It expresses accuracy of the designed system where system accuracy means the correct number of results produced by the system. Precision is measured by comparing designed system's number of correct results by all (incorrect and correct) results produced by the system, and is calculated as:

$$Precision = \frac{N_{correct}}{N_{incorrect} + N_{correct}} \tag{2}$$

where $N_{incorrect}$ is the number of incorrect results and $N_{correct}$ is the number of correct results.

3. **Evaluation Method:** To evaluate the results of our tool, each outcome class names, attributes, method, associations, etc., of the RECM tool's output was matched with the expert's opinion(N_{Sample}) (sample solution). The outcome that accurately classified into respective category was declared correct($N_{correct}$)otherwise incorrect ($N_{incorrect}$). The information that was not extracted but it was given in the human expert's opinion (N_{sample}) was categorized as the missing information ($N_{missing}$).

In order to show the efficiency and accuracy of the proposed methodology various case studies have been considered. A total of three case studies and their performance evaluation are explained in the following sections.

4.2. Case Study I: Robot System

This case study is captured from the domain of a robot system. The following problem statement is fed as input to the OpenNLP which is further processed by the SBVR process to extract object oriented items and then proceed to generate the UML class diagram:

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 UML class diagram is generated from this problem statement using the proposed Requirements to Class Model (RECM) tool. The problem statement is given as the input to the tool. Initially the OpenNLP processes these statements and provides the necessary parts of speech. The output of the OpenNLP is shown in Figure 6.

This tagger is further processed by the SBVR to generate the vocabulary. Figure 7 shows the result of extracted vocabulary.

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 rules that are extracted by our tool.

The final step is UML class model generation. The class model for the problem statement is shown in Figure 9. From the problem statement the identified classes are cup, dish, assembly parts, and sensor. The attributes identified are consists, puts, enters, conveys, and informs. Having this information our tool generates the following class diagram.

Figure 6. Generated POS tagger

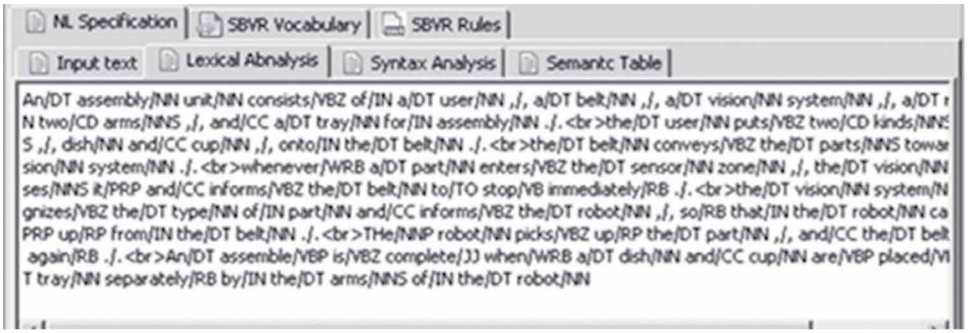


Figure 7. Extracted vocabulary for UML model

Figure 8. SBVR rules generated by the RECM tool

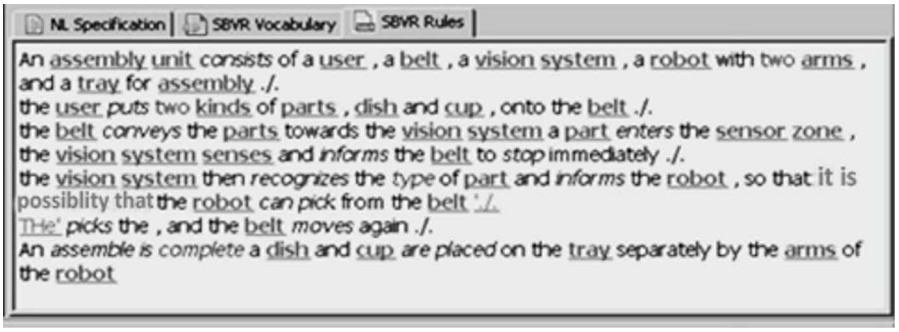
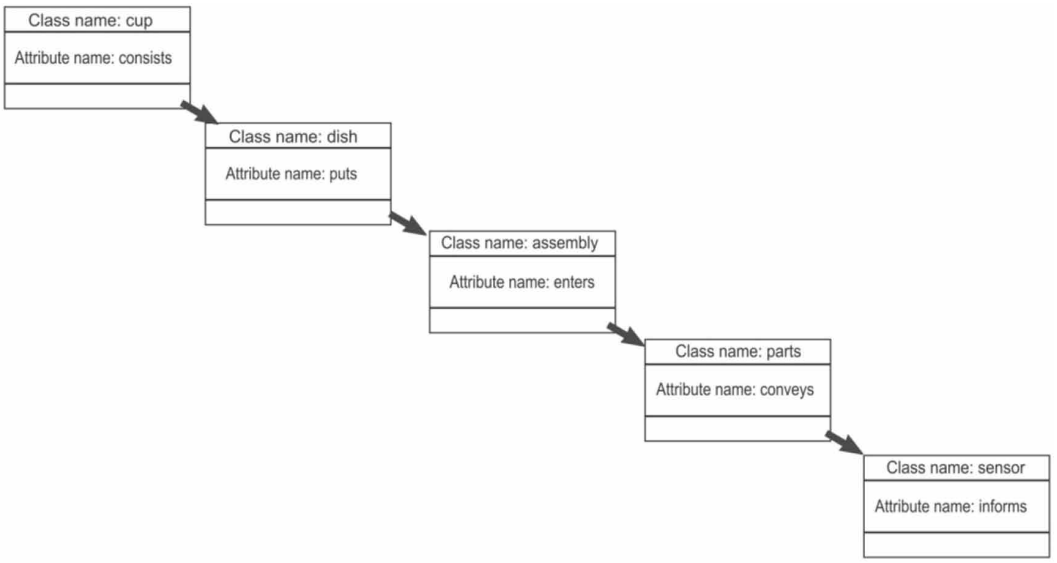


Figure 9. UML class model generation



Tables 1 and 2 shows the evaluation report of structural and behavioural statements of software requirement specifications extracted by SBVR in the form of classes, attributes, methods, associations and generalizations.

4.4. Case Study II: Retail Store Automation

This case study is taken from the domain of a retail store system and is as given below:

A retail store automation involves maintenance of customers, purchase of items by customer and billing of items. Customers can be regular visitors to the store in which case they are eligible for discounts based on the bill amount. The customers can be privileged ones, wherein they are given membership cards Platinum, Gold and Silver. Such customers are eligible for gifts based on the type of membership card. The billing staff does the billing and delivery of items to the customer. The bill calculation involves the logic of computation of the bill depending on customer type. The customer can pay the bill through credit card or cash. In the former case two percent processing charge is applicable. Sales tax is also applicable on the final bill amount.

Table 1. SBVR extracted requirement statements by RECM tool

Type/Metrics	Nsample	Ncorrect	Nincorrect	Nmissing
Object Types	05	4	0	1
Verb Concepts	10	10	0	0
Individual Concepts	01	0	1	0
Characteristics	06	5	1	0
Quantifications	08	8	0	0
Unary Fact Types	05	5	0	0
Associative Fact Types	08	7	1	0
Partitive fact Types	00	0	0	0
Categorization Fact Types	00	0	0	0
Total	43	39	3	1

Table 2. Recall and precision of RECM tool

Type/ Metrics	Nsample	Ncorrect	Nincorrect	Nmissing	Rec%	Prec%
SBVR extracted Requirements	43	39	3	1	90.69	92.85

The statements in the case study are processed by the implemented OpenNLP tool for tokenization and Lexical analysis and the necessary POS tags are generated. The OpenNLP processed output of the case study is as shown in Figure 10.

Figure 11 shows the screen shot of SBVR rule extracted result. Here SBVR rules for the case study retail store automation is generated.

The input statements of the case study are NL processed and then extracted for object-oriented elements by RECM via SBVR tool and the result is as shown in Table 3.

Precision and Recall values of the evaluated case study II is shown in the Table 4.

The overall performance of the RECM via SBVR tool is as shown in the graph below Figure12. The results obtained are very satisfactory.

Figure 10. Generated POS tagger

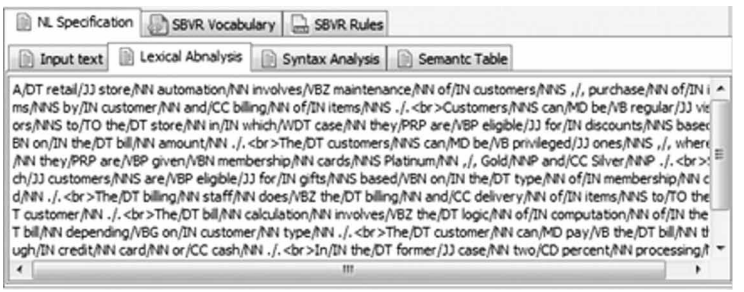


Figure 11. SBVR rules

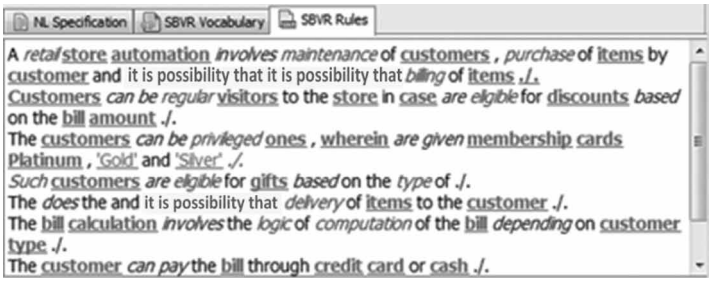


Table 3. Results of SBVR extracted requirement statements by RECM via SBVR tool

Type/Metrics	Nsample	Ncorrect	Nincorrect	Nmissing
Object Types	07	4	2	1
Verb Concepts	10	8	2	0
Individual Concepts	01	0	1	0
Characteristics	06	4	0	2
Quantifications	08	4	0	0
Unary Fact Types	05	5	0	0
Associative Fact Types	08	8	0	0
Partitive fact Types	00	0	0	0
Categorization Fact Types	00	0	0	0
Total	45	33	5	3

Table 4. Recall and precision of RECM via SBVR tool

Type/ Metrics	Nsample	Ncorrect	Nincorrect	Nmissing	Rec%	Prec%
SBVR extracted Requirements	45	33	5	3	73.33	86.84

The results of our tool is compared with other available tools that can perform automated analysis of the NL requirement specifications. Recall value was not available for some of the tools. The available recall and precision values of NL based knowledge extraction tools in this domain are as shown in Table 5.

Here, we can note that the accuracy of other NL tools used for information extraction and object-oriented analysis is well below than our proposed tool. Comparison of our tool with some of the available tools is as shown in Table 6. By using this approach, we could able to achieve good result for generation of Classes, Attributes, Methods, Associations, Multiplicity, Generalization and Instances.

Figure 12. Performance of RECM via SBVR tool

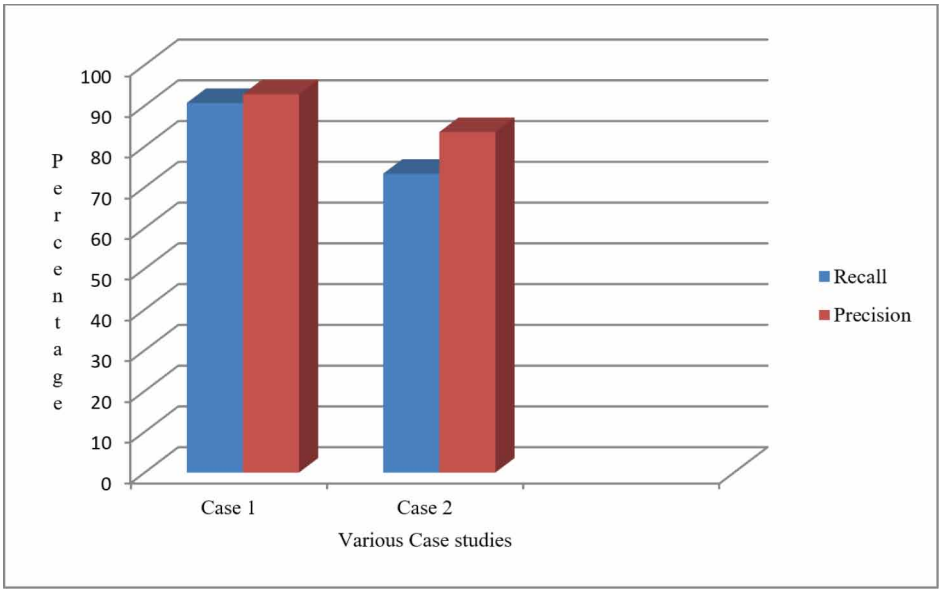


Table 5. A comparison of performance evaluation – RECM via SBVR tool vs other tools

NL Tools for Class Modelling	Recall	Precision
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%
RECM via SBVR Tool	82.01%	89.84%

Table 6. Comparison of RECM via SBVR tool with other NL based extraction tools

Support	CM-Builder	LIDA	GOOAL	NL- OOML	SUCM Tool
Classes	Yes	User	Yes	Yes	Yes
Attributes	Yes	User	Yes	Yes	Yes
Methods	No	User	Yes	Yes	Yes
Associations	Yes	User	Semi-NL	No	Yes
Multiplicity	Yes	User	No	No	Yes
Aggregation	No	No	No	No	Yes
Generalization	No	No	No	No	Yes
Instances	No	No	No	No	Yes

5. CONCLUSION

The aim of this research work is to provide a method to automatically generate UML class models from software requirement specifications (SRS) and we have developed a tool named RECM via SBVR tool (Requirement to Class Model via SBVR). One of the major advantages of this tool is quick and instant generation of object-oriented items from user requirements with the support of SBVR. Our tool makes use of an open natural language processing tool with a good range of grammatical analysis to analyze these requirements. This OpenNLP tool provides the necessary parts of speech tags (POS tags) of software requirements. After obtaining the required and relative information from SRS, the Semantics of Business Vocabulary and Rules process implemented in our RECM via SBVR tool generates the vocabulary and rules. The implemented SBVR process has the ability to extract object types, individual concepts, fact types, characteristics, associative fact types, partitive fact types and categorization fact types more accurately compared to other tools. We have performed many case studies and got average recall and precision values as 82.01% and 89.84% which are really encouraging. It is also noted that the RECM via SBVR tool is unique and have better recall and precision values than other NL tools used for information extraction and object-oriented analysis. OpenNLP together with SBVR provides an error-free method to capture the software requirements specification which is generally mentioned in NL such as English. This neoteric approach is quite fast to develop class models from SRS. This automatic generation of UML class models from the user software requirement specification, based on NL processing tool and SBVR process obtain good accuracy in less time.

REFERENCES

Abbott, R.J. (1983). Program Design by Informal English Descriptions. *Communications of the ACM*, *26*(11), 882-894.

Ambriola, V., & Gervasi, V. (2006). On the Systematic Analysis of Natural Language Requirements with CIRCE. *Automated Software Engineering*, *13*(1), 107–167. doi:10.1007/s10515-006-5468-2

Anandha, G.S. & Uma, G.V. (2006). Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification. In *Proceedings of the PRICAI 2006: Trends in Artificial Intelligence* (pp. 1155-1159). Springer.

Arora, C., Sabetzadeh, M., Briand, L., & Zimmer, F. (2015). Automated checking of conformance to requirements templates using natural language processing. *IEEE Transactions on Software Engineering*, *99*.

Bajwa, I. S., & Abbas Choudhary, M. (2006). Natural language processing based automated system for UML diagrams generation. In *Proceedings of the 18th Saudi National Computer Conf. on computer science (NCC18)*. Riyadh, Saudi Arabia: The Saudi Computer Society (SCS).

Bajwa, I. S., & Hyder, I. (2007) UCD-generator-A LESSA application for use case design." In *Proceedings of the International Conference on Information and Emerging Technologies ICIET 2007* (pp 1-5). IEEE.

Bajwa, I. S., Lee, M. G., & Bordbar, B. (2011, March). SBVR business rules generation from natural language specification. In *2011 AAAI Spring Symposium Series*. AAAI.

Bajwa, I. S., Samad, A., & Mumtaz, S. (2009). Object Oriented Software modeling Using NLP based Knowledge Extraction. *European Journal of Scientific Research*, *35*(1), 22–33.

Bethard, S., Ogren, P., & Becker, L. (2014, May). ClearTK 2.0: Design patterns for machine learning in UIMA. In *LREC... International Conference on Language Resources & Evaluation:[proceedings]. International Conference on Language Resources and Evaluation* (Vol. 2014, p. 3289). NIH Public Access.

Brosch, P., & Randak, A. (2011). Position paper: m2n—A tool for translating models to natural language descriptions. *Electronic Communications of the EASST, 34*.

Burden, H., & Heldal, R. (2011). Natural language generation from class diagrams. In *Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation*. ACM. doi:10.1145/2095654.2095665

Callan, R. E. (1994). Building Object-Oriented Systems: An introduction from concepts to implementation in C++. *Computational Mechanics*.

Deeptimahanti, D. K., & Babar, M. A. (2009). An Automated Tool for Generating UML Models from Natural Language Requirements. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering ASE '09* (pp 680-682). IEEE. doi:10.1109/ASE.2009.48

Falessi, D., Cantone, G., & Canfora, G. (2013). Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *IEEE Transactions on* Software Engineering, *39*(1), 18–44.

Fernandez, P. M., & Garcia-Serrano, A. M. (2000). The role of knowledge-based technology in language applications development. *Expert Systems with Applications*, *19*(1), 31–44. doi:10.1016/S0957-4174(00)00018-X

Feuto, P. B., Cardey, S., & Greenfield, P. (2013). Domain Specific Language Based on the SBVR Standard for Expressing Business Rules Enterprise. In Proceedings of the 17th IEEE International Distributed Object Computing Conference Workshops (EDOCW) (pp. 31-38). Academic Press.

Harmain, H. M., & Gaizauskas, R. (2003). CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. *Automated Software Engineering*, *10*(2), 157–181. doi:10.1023/A:1022916028950

Hirschman, L. & Thompson, H.S. (1996). Overview of evaluation in speech and natural language processing. *Survey of the State of the Art in Human Language Technology, 13*(1).

Imran Sarwar Bajwa, M. AsifNaeem (2011) On Specifying Requirements using a Semantically Controlled Representation In: 16th International Conference on Applications of Natural Languages to Information Systems (NLDB 2011) Alicante, Spain: Springer Verlag. Pp. 217-220.

Kleiner, M., Albert, P., & Bézivin, J. (2009). Parsing SBVR-based Controlled Languages. In *Proceedings of the International Conference on Model Driven Engineering Languages and Systems* (pp. 122-136). Springer.

Krishnan, H., & Samuel, P. (2010). Relative Extraction Methodology for class diagram generation using dependency graph. In *Proceedings of the 2010 IEEE International Conference on Communication Control and Computing Technologies (ICCCCT)* (pp 815-820). IEEE.

Lane, P. C. R., & Henderson, J. B. (2001). Incremental syntactic parsing of natural language corpora with simple synchrony networks. *IEEE Transactions on Knowledge and Data Engineering*, *13*(2), 219–231. doi:10.1109/69.917562

Li, K., Dewar, R. G., & Pooley, R. J. (2005). Object-Oriented Analysis Using Natural Language Processing. *Linguistic Analysis*.

Mich, L. (2001). 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. IEEE.

Mohanan, M., & Samuel, P. (2016). Software Requirement Elicitation Using Natural Language Processing. In *Innovations in Bio-Inspired Computing and Applications* (pp. 197–208). Springer International Publishing.

More, P., & Phalnikar, R. (2012, April). Article: Generating UML Diagrams from Natural Language Specifications. *International Journal of Applied Information Systems*, *1*(8), 19–23. doi:10.5120/ijais12-450222

Oliveira, A., Seco, N., & Gomes, P. (2006). A CBR Approach to Text to Class Diagram Translation. In *Proceedings of the TCBR Workshop at the 8th European Conference on Case-Based Reasoning*, Turkey.

OMG. (2008). Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0. Object Management Group. Retrieved from http://www.omg.org/spec/SBVR/1.0/

Perez-Gonzalez, H. G., & Kalita, J. K. (2002, November). Automatically generating object models from natural language analysis. In *Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 86-87). ACM.

Perez-Gonzalez, H. G., & Kalita, J. K. (2002) GOOAL: A Graphic Object Oriented Analysis Laboratory. In *Proceedings of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages and applications (OOPSLA '02)* (pp. 38-39). Academic Press.

Thi, T. P. (2015, January). Identifying semantic and syntactic relations from text documents. In *Proceedings of the 2015 IEEE RIVF International Conference on Computing & Communication Technologies-Research, Innovation, and Vision for Future (RIVF)* (pp. 127-131). IEEE.

Toutanova, K., & Manning, C. D. (2000). 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). Academic Press. doi:10.3115/1117794.1117802

Zhou, N., & Zhou, X. (2004). Automatic Acquisition of Linguistic Patterns for Conceptual Modeling. *Info*, 629.

Murali Mohanan is an Associate Professor in Department of Computer Science at College of Engineering, Cherthala. Alleppy District. Kerala. India and at present Research Scholar in Department of Computer Science, CUSAT, Koch. M.Tech degree holder in Computer and Information Sciences from Cochin University of Science and Technology. His research interests include artificial intelligence, uml modelling and design, and software engineering.