SBVR Business Rules Generation from Natural Language Specification

Imran S. Bajwa, Mark G. Lee, Behzad Bordbar

School of Computer Science
University of Birmingham
United Kingdom
+44 (0)121 414 8201
i.s.bajwa@cs.bham.ac.uk, b.bordbar@cs.bham.ac.uk, m.g.lee@cs.bham.ac.uk

Abstract

In this paper, we present a novel approach of translating natural languages specification to SBVR business rules. The business rules constraint business structure or control behaviour of a business process. In modern business modelling, one of the important phases is writing business rules. Typically, a business rule analyst has to manually write hundreds of business rules in a natural language (NL) and then manually translate NL specification of all the rules in a particular rule language such as SBVR, or OCL, as required. However, the manual translation of NL rule specification to formal representation as SBVR rule is not only difficult, complex and time consuming but also can result in erroneous business rules. In this paper, we propose an automated approach that automatically translates the NL (such as English) specification of business rules to SBVR (Semantic Business Vocabulary and Rules) rules. The major challenge in NL to SBVR translation was complex semantic analysis of English language. We have used a rule based algorithm for robust semantic analysis of English and generate SBVR rules. Automated generation of SBVR based Business rules can help in improved and efficient constrained business aspects in a typical business modelling.

1. Introduction

A robust business model is always based on a set of business rules. The actual role of the business rules in business modelling is to constraint business structure or control behaviour of a business process. In modern business modelling, the business rule analyst generates and manages business rules. Moreover, Business rule management (BRM) systems are employed to separate business logic from the application code. Separate business logic (represented in the form of business rules) helps in simplifying the process of making changes in business applications. In modelling of a typical business application, a business analyst and the business owners define the requirements for the new business application. Then a business rule analyst understands the requirements and defines business rules in plain English. However, the problem with plain English representation of business rules is that such rules can never be machine processed as these rules are syntactically inconsistent and semantically informal. To enable business rules for machine processing, the plain English business rules are represented in some formal representation such as SBVR (Semantic Business Vocabulary and Rules). SBVR [3] is OMG's recent standard that provides formal representation to business rules written in plain English or any other natural language.

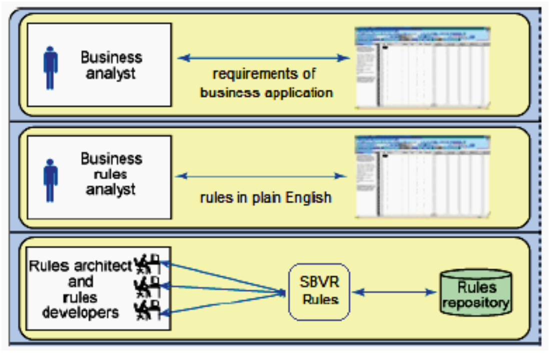


Figure 1. Typical process of business rule generation

A SBVR rule is the key constituent of SBVR standard. A SBVR rule can easily be machine processed to perform object rule modelling, perform rule consistency analysis, or generate formal representations such as OCL constraints [2], databases, business rules repositories, business blueprints, business object models, software components, etc. A business rules always belong to a particular business domain. In SBVR, the business domain is represented by SBVR vocabulary [3] that is the second major constituent of SBVR standard. In our approach, a UML class model represents a business domain and we extract the business vocabulary from the UML model. To create SBVR rules, a business rule analyst has to manually translate business rules (written in plain English) to SBVR syntax. But, the manual translation of plain English business rules to SBVR rules is not only complex but also difficult and time consuming. Moreover, the manual effort to generate SBVR rules from NL specification can result in erroneous rules. The scenario becomes more complex in the absence of any tool support for automated translation of plain English business rules to SBVR rules.

In this paper, we present a novel approach NL2SBVR to translate natural language (such as English) specification of business rules to SBVR rules. A major challenge, in accurate translation of English text to SBVR rules, was to overcome the English's inherent syntactic ambiguities and semantic informalities. SBVR provides a set of logical formulations [3] that can help in robust semantic analysis of English language text. We have written an algorithm for semantic analysis of English text that is based on SBVR's semantic formulation. This algorithm is used in our approach to parse English text and extract SBVR syntactic elements. Our approach extracts SBVR vocabulary from a UML [1] class model. Afterwards, the SBVR elements are mapped to SBVR vocabulary to ensure that the output SBVR rules will be targeting a particular business domain. Finally, the SBVR rule is generated from UML mapped SBVR elements. Our approach NL2SBVR is implemented in a tool 'RuleGenerator' as a proof of concept.

The rest of the paper is structured as follows: section 2 describes preliminary concepts related to SBVR; section 3 describes in detail the NL2SBVR approach; section 4 presents overview of the tool 'RuleGenerator' and also shows the experimentation and results of the tool; section 5 discusses the evaluation methodology followed by a discussion on the related work. The paper ends with a conclusion section.

2. Semantic Business Vocabulary and Rules

Semantic Business Vocabulary and Rules (SBVR) [3] is an adopted standard, introduced by OMG. By using SBVR, informal specifications can be captured in natural languages and represented in the formal logic so that they can be machine-processed. SBVR allows the production of business vocabulary, business facts and business rules in a particular business domain. Brief description of the major elements in SBVR is given below.

I. Business Vocabulary

Business vocabulary defines a particular business domain. In SBVR 1.0, business vocabulary can have two major types of elements [3]: Concepts and Fact Types. A concept is a key term that represents a business entity in a particular domain. There basic types of concepts [3] are noun concepts, individual concept, and verb concepts. Typically, the common nouns are classified as noun concepts while the proper nouns or quantified nouns are denoted as individual concepts. A verb concept can be an auxiliary verb or action verb or both. However, a fact type is a combination of a verb concept and noun concepts. A Fact type specifies the relationship among different concepts in a business rules.

In our research we use a UML class model as a business domain. Hence, the business vocabulary comes from the target UML class model. With respect to UML class model: class names and their attributes names are represented as noun concepts, object names are represented as individual concepts, operation names are named as action verbs, and the associations and generalizations are represented as fact types.

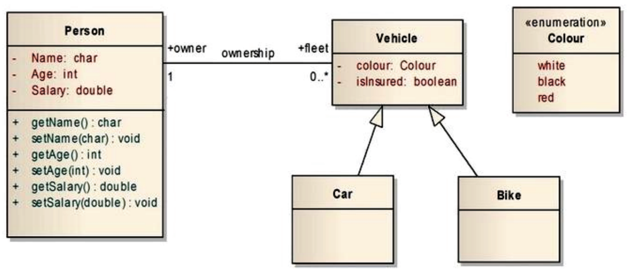


Figure 2. A UML class model defining business domain

In example of figure 2, person, vehicle, car are noun concepts. Similarly, the getName, getAge, getSalary are verb concepts and association ownership is fact type in SBVR.

II. Business Rules

In SBVR, the business rules represent particular business logic in a specific context. Each SBVR business rule is based on at least one fact type. Business rules The SBVR rules can be of two types [3]: definitional rules and behavioral rules:

- Definitional Rules or structural rules are used to define an organization's setup [3] e.g. **It is necessary that each** <u>customer</u> *has* **at least** one *bank account.*
- Behavioural Rules or operative rules express the conduct of an entity [3] e.g. **It is obligatory that each** <u>customer</u> *can withdraw* **at most** GBP 200 per day.

In perspective of UML class model, the definitional or structural rules usually involve the attributes of the classes and objects, while the Behavioural or operative rules involve the operations, associations, and generalizations of classes and objects.

III. Semantic Formulation

In SBVR, logical formulations are used to semantically formulate the SBVR rules. The common logical formulations are [3]:

1. Atomic formulation specifies a fact type in a rule e.g. "customer *should be* old" is atomic formulation from the fact type "customer *is* old".
2. Instantiation formulation denotes an instance of a class e.g. "silver account" is an *Instantiation* of the noun concept "bank account".
3. Logical operations e.g. conjunction, disjunction, implication, negation, etc are also supported in SBVR. In natural languages, the logical operations allow combining NL phrases to create more complex logical expression.
4. Quantification states the enumeration of a noun concept or verb concept e.g. "at least one", "at most one", "exactly one", etc are used to quantify concepts.
5. Modal Formulation identifies the meanings of a logical formulation. e.g. "It is obligatory" or "It is necessary" are used to formulate modality.

IV. SBVR Notation

In SBVR 1.0 document [3], the Structured English is proposed, in Annex C, as a possible notation for the SBVR rules. The Structured English provides a standardized representation to formalize the syntax of natural language representation. In this paper, we have used the following Structured English specification: The noun concepts are underlined e.g. person; the verb concepts are italicized e.g. *should be*; the SBVR keywords are bolded e.g. **each**, **at least**, **at most**, **obligatory**, etc; the individual concepts are double underlined e.g. white car or black bike.
Another available notion in SBVR standard 1.0 is RuleSpeak. Our tool RuleGenerator supports both SBVR notations.

3. NL to SBVR Translation

The NL2SBVR is a modular NL-based approach that generates SBVR business rules from English text with respect to a target Business domain. It takes two inputs: a single English statement and a UML class model. Here English statement is English specification of a business rule and the UML class model provides a business domain. To process the input English text first it is linguistically analyzed. In linguistic analysis of the English text, the English text is Parts-Of-Speech (POS) tagged. Then a rule based parser [8] is used to further process the POS tagged information to extract basic SBVR elements e.g. noun concept, fact type, etc. Here, the SBVR vocabulary is mapped to a SBVR rule. Finally, to generate an SBVR business rules, the SBVR vocabulary is mapped to SBVR elements using the rule-based approach [11]. These steps can be summarized as follows.

1. Obtain a text document that is English description of a constraint and a target UML model.
2. Use a NLP module to syntactically and semantically analyse the informal constraint text and keep all the intermediate analyses result for further analysis.
3. Use the results produced by NLP module to map with the input UML model to make it sure that the output SBVR rules are related to the target UML model.
4. Use the UML model to extract the SBVR vocabulary. Use the results produced by NLP module to extract SBVR elements e.g. noun concept, object type, Individual concept, verb concept, etc.
5. Use the SBVR vocabulary & SBVR rule to generate SBVR business rule.

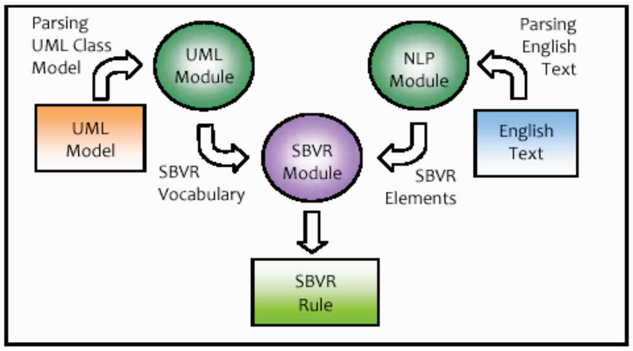


Figure 3. The NL2SBVR Approach

Figure 3 illustrates the main steps of the NL2SBVR approach. The working of the above defined steps has been described in detail in the following section:

I. The Input Documents

The NL2SBVR approach takes two input documents: an English text document (.txt file) and a UML class model (.ecore file). The English text is taken as a plain text file containing only English constraint. Current version of the RuleGenerator handles only one English constraint at a time. The given English text should be grammatically correct. UML model is taken as ECORE or XMI format. We used Eclipse UML2 Ecore Editor to create a UML model and export it in XMI format.

II. The NLP Module

The core of NL2SBVR approach is a NLP module that consists of a number of processing units organized in a pipelined architecture. This NLP module is highly robust and is able to process complex English statements. The NLP system is used to lexically and syntactically process the English text and then perform semantic analysis to identify basic SBVR elements. The core system processes a text into three main processing stages:

a. Lexical Processing
The lexical processor comprises for sub-modules: a tokenizer, a sentence splitter, POS tagger, and a morphological analyzer. The input to lexical analyzer is a plain text file containing English description of the target

SBVR business rule. The output is an array list that contains tokens with their associated lexical information.
In array list, each sentence is separated by '.'. The lexical processing steps are performed in the following sequence:

1. *Tokenization:* In first step, the input English text is read and tokenized to identify the tokens e.g. "A person should be 18 years old." is tokenized as [A] [person] [should] [be] [18] [years] [old] [.]
2. *Sentence Splitting:* The sentence splitter identifies the margins of a sentence and each sentence is separately stored.
3. *Parts-of-Speech (POS) Tagging:* In third step, partsof- speech tagging is performed of given text to identify the basic POS tags for input text. For POS tagging, the Stanford POS tagger v3.0 [9] has been used that can identify 44 POS tags.
4. *Morphological Analysis:* After performing the POS tagging, morphological analysis was performed for all nouns and verbs. The morphological analysis is performed to separate the suffixes possibly attached to the nouns and verbs [10]. For example, a verb "purchased" is analyzed as "purchase + ed" and similarly a noun "employees" is analyzed as "employee + s".

b. Syntactic Analysis
We have used an enhanced version of a rule-based parser for the syntactic analysis of the input text used in [11]. The text is syntactically analyzed and a parse tree is generated for further semantic analysis. Figure 3 shows the generated parse tree of the above example.

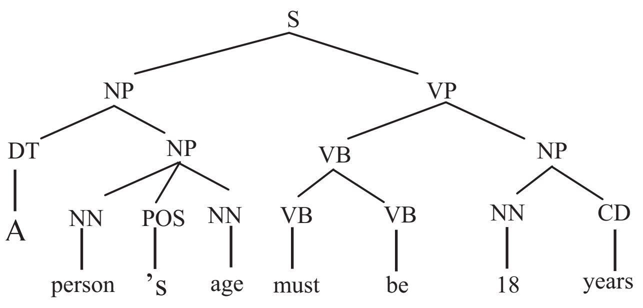


Figure 4. Parsing English text using dependency grammar

c. Semantic Analysis
In this semantic analysis phase, role labeling [12] is performed. The desired role labels are actor, co-actor, action, thematic object, and a beneficiary if exists. These roles will assist in identifying different SBVR elements in the next phase and also be used in constructing fact types from the extracted SBVR elements. In semantic analysis phase, after role labeling, the order is identified in which subject, verb, object, and adverb appears in the input English text. The output of the NLP module is an xml file that contains the parsed English text with all the extracted information.
Basic SBVR elements e.g. Noun concept, individual concept, object type, verb concepts, etc are identified from the English input that is preprocessed by the NLP module. Following mapping rules are used to identify the SBVR elements:

- All proper nouns are mapped to the individual concepts
- All common nouns appearing in subject part are mapped to the noun concepts or general concept.
- All common nouns appearing in object part are mapped to object type.
- All action verbs are mapped to verb concepts.
- All auxiliary verbs and noun concepts are mapped to the fact types.
- The adjectives and possessive nouns (i.e. ending in 's or coming after 'of') are mapped to the attributes.

All articles and cardinal numbers are mapped to quantification. All these rules are applied to the English text and the output is stored in an array list. Following example highlights the proposition of basic SBVR elements in a typical SBVR rule.

$$\underbrace{\text{A}}_{\text{Quantification}}\quad \underbrace{\text{person}}_{\substack{\text{Noun}\\\text{Concept}}}\text{'s}\quad \underbrace{\text{age}}_{\text{Attribute}}\quad \underbrace{\text{must be}}_{\substack{\text{Verb}\\\text{Concept}}}\quad \underbrace{\text{18}}_{\text{Quantification}}\quad \underbrace{\text{years.}}_{\substack{\text{Noun}\\\text{Concept}}}$$

Figure 5. Semantic analysis of English text

III. The UML Module

The UML module reads both ECORE and XMI format of a UML class model generated from Eclipse. The UML module extracts all classes, objects, and their respective attributes, operations and associations and finally maps them to SBVR vocabulary. Following section also describes how the SBVR vocabulary is mapped to the SBVR elements generated by NLP module.
a. Generating SBVR Vocabulary
The SBVR vocabulary is generated from the input UML model. All the classes are mapped to noun concepts, attributes of the classes are named as the individual concepts, and all the class operations are named as verb concepts. The associations and the generalizations are mapped to the binary fact types. Binary fact types [14] are typically composed of two noun concepts and a verb concept. All these SBVR elements with their associated types are stored and exported as an array list. The list of SBVR vocabulary that consists of concepts and fact types are used as a reference element during construction of SBVR rules.
b. Mapping with UML Model
Before translation of English text to SBVR rules, the input English text is mapped with the input UML model to ensure that generated SBVR rules will be semantically related to the target business domain. The mapping is carried out in SBVR elements and SBVR vocabulary. The noun concepts in SBVR rules are mapped to the UML

classes. Individual nouns are mapped to the UML objects. Verb concepts are mapped to methods of a class.
Adjectives and possession nouns (with *of* and *'s*) are tagged as attributes. A fact types are mapped with the associations and generalizations. If the mapping is carried out then the output of the module is sent forward to the SBVR module otherwise the user is notified that business (English) rule is not related to the business domain (UML model). For example, if user gives following English text:

A person's nationality *should be* British.

In this example, nationality attribute is not available in person class, so SBVR rule for this English input will not be generated.

IV. The SBVR Module

The SBVR module is based on a rule based parser that contains set of rules to map SBVR elements with SBVR vocabulary and generate complete SBVR rules. In this phase detailed semantic analysis of the English text is performed. Following section describes how the SBVR rules are generated.

a. Generating SBVR Rule
SBVR rules are generated from the output of the NLP module. To generate SBVR rules, the first step is to create a fact type. A fact type is created by mapping the noun concepts and verb concepts to the fact types available in the SBVR vocabulary array list. Atomic formulization is used to map the input text to a suitable target fact type in SBVR vocabulary. The mapped fact type is used to generate a SBVR rule by applying a set of logical formulations. As For the different types of syntactic structures used in English language, respective types of logical formulations have been defined. Following are the details that how we have incorporated these logical formulations to map English language text into SBVR rule.

5. *Logical Operations*: The logical operations [3] are used to combine one or more expressions, known as logical operand to produce complex Boolean expressions. The logical expressions e.g. AND, OR, NOT, implies, etc are incorporated using logical formulations. To apply logical operations a set of rules were defined that help to identify the number and names of operands and the operator from the input English text. For example, the tokens "not" or "no" are mapped to negation. Similarly, the tokens like "and" are mapped to conjunction and "or" is mapped to disjunction, etc.
6. *Quantification*: Quantification [3] is a logical formulation that uses a variable to specify the scope of a concept. The basic types of quantifications were identified by using a set of rules i.e. the tokes like "more than" or "greater than" are mapped to *at least n quantification*. Similarly, the token "less than" is mapped to *at most n quantification* and the token "equal to" or a positive statement is mapped to *exactly n quantification*. These quantifications guide in identifying the cardinality of a noun concept.
7. *Modal Formulation*: Modal formulation [3] is used to specify meanings of the other logical formulations. A set of mapping rules based on modal verbs e.g. can, may, should, etc are defined to identify the four basic types of modal formulations. For example the modal verbs "can" or "could" are mapped to possibility formulation. Similarly, modal verb "should" or verb concept "have to" is mapped obligation formulation.

b. Applying SBVR Notation
The last step in SBVR rule generation is to apply a SBVR notation. RuleGenerator supports both SBVR notations: SBVR Structured English and RuleSpeak. To apply Structured English the noun concepts are underlined e.g. person; the *verb concepts* are italicized e.g. *can have*; the **keywords** are bolded i.e. SBVR keywords e.g. **each**, **at least**, **at most**, **obligatory**, etc; the individual concepts are double underlined e.g. black car.

It is obligatory that a person's age *should be* **at least** 18 years.

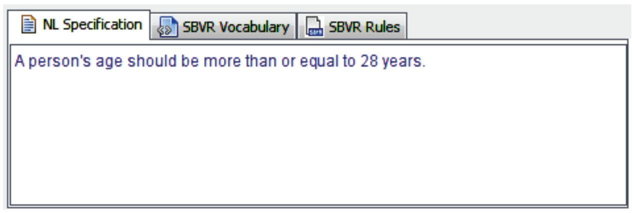
The SBVR produces a SBVR rule in the form of text string that is further formatted using the SBVR notation i.e. Structured English described in the section 2.4. The output SBVR module is saved and exported in two separate files: an xml file contains the SBVR vocabulary; a text file contains the formatted SBVR rule.

4. Experiments and Results

To find the bugs in the working of the tool experiments were performed to carry out the dynamic verification of the software tool. To test the accuracy of the SBVR rules generated by the designed system three classes were defined: invariants, preconditions and post-conditions. Various complexity levels of input i.e. simple, compound and complex SBVR rules were also defined to verify the consistency of tool's output. All rule types can be structural or behavioural. Both, the simple and complex rules contain only one fact type. Moreover, the simple rules do not involve association and generalizations but complex business rules involve associations and generalizations. Compound rules are complex rules with multiple fact types. Examples of defined complexity levels are following:

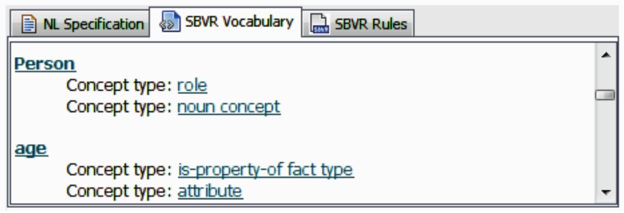
Simple: A person's age should be more than 18 years.
Complex: A vehicle's colour can be white or black.
Compound: If a person's salary is more than 10,000$ then the person can buy a car with white colour.

To test tools accuracy, 10 examples of each complexity level were used. Constraint types for each 10 examples were generated. Each generated SBVR business rule from each category was type-checked. To verify syntax of BVR rules, SBeVeaR tool was used that is an SBVR type checking tool. For the sake of type checking in SBeVeaR, the used class model and the generated SBVR business rule were given as input.



A person's age should be more than or equal to 28 years.

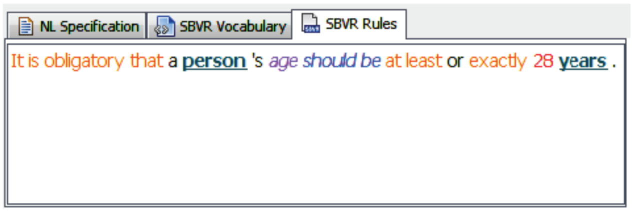
Figure 6. English input



Person
Concept type: role
Concept type: noun concept

age
Concept type: is-property-of fact type
Concept type: attribute

Figure 7. SBVR vocabulary generated from UML model



It is obligatory that a person 's age should be at least or exactly 28 years .

Figure 8. SBVR rule generated from English input

The above shown screenshots are from our tool RuleGenerator that gets a UML Model and English representation as input. The tool works as first of all, a XMI file is read as input for the target UML class model. All the classes and associations in given UML class model are used to generate the SBVR vocabulary. Then, the input English text is syntactically and semantically analyzed to create a SBVR rule after mapping with SBVR vocabulary.

Table 1: Evaluating results

Complexity level/ Constraint Type	Structural Rules	Behavioural Rules	Total
Simple	87.3%	91.5%	91.63%
Complex	86.2%	90.2%	89.73%
Compound	85.7%	84.8%	83.74%

Average accuracy: 87.33%

A matrix representing SBVR business rules accuracy test (%) for invariants, preconditions, and post conditions has been constructed. Overall accuracy for all types of SBVR business rules is determined by adding total accuracy of all categories and calculating its average that is 87.33%.
The following graph is showing the accuracy ratio of various SBVR rule types in terms of structural and behavioural rules. The accuracy of various SBVR rule types is analyzed against simple, complex and compound input English statements. The correctness of the software tool was verified though static verification by performing the described analysis.

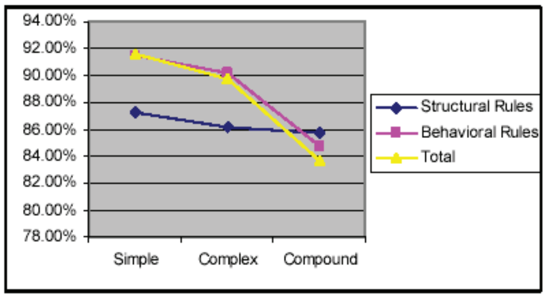


Figure 9: Evaluating results

5. RELATED WORK

Applications of NLP in the field of software and business modelling are really significant especially to improve accuracy, productivity, flexibility, multilinguality and robustness [13]. Hector presented a semi-natural language (4WL) to automatically generate object models from natural language text [14]. LOLITA is another Natural Language Processing System that was used to extract object oriented information from the NL text [15]. K. Li addressed NLP problems in object oriented analysis (OOA) [16]. NOESIS (Natural Language Oriented Engineering System for Interactive Specifications) is a WordNet based NL text analysis module [17]. A tool REBUILDER based on NOISES is introduced by Gomes (2004) to generate class diagrams from NL specification. A significant contribution by Harmain and Gaizauskas was their NL based CASE tool named CM-Builder [18]. All these tools motivate for design of a tool that can assist users in creating business rules from English.
Extraction of semantic knowledge from NL specification and its transformation to ER models was performed by Omar [19]. Semantic heuristics were used to extract the relevant ER elements such as entities, attributes, and relationships from the specifications. E-R Generator [20] is another rule-based designed tool that performs semi automatic generation of E-R Models from NL specifications. Similar to this work NLP has also been used for conceptual modeling [21]. Natural languages have also been mapped to java code [11].
A controlled natural language (CNL) is a subset of a natural language with restricted grammar and vocabulary [22]. Controlled grammar and vocabulary minimizes ambiguity and complexity involved in processing of natural languages. Controlled natural languages are useful in reliable automatic semantic analysis of the language [23]. A particular challenge for natural language processing is the translation of NL to formal semantics. In

recent times, SBVR has been presented as a language for the semantic formalization of natural languages [24].

6. CONCLUSION

In business modelling, business logic is easy to express in natural languages but the business logic expressed in natural langauge is usually not semantically formal and unambiguous. This research paper presents a novel approach that automatically translates natural language specification of business rules to SBVR business rules. The presented approach is also implemented in a tool 'RuleGenerator' as a proof of concept. RuleGenerator can find out the noun concepts, individual concepts, verbs and adjectives from the NL text. This extracted information is further incorporated to constitute a complete SBVR rule. The presented approach is fully automated. The presented approach not only assists the business rule analysts and architects by generating precise SBVR rules from NL specification in a simple and quick manner. As a next step, we are hoping to investigate usability aspects of the tool directly via empirical methods involving teams of developers.

REFERENCES

[1] OMG. 2007. Unified Modeling Language (UML), OMG Standard, v. 2.3.

[2] Bajwa I.S., Behzad Bordbar, Mark G. Lee, (2010), OCL Constraints Generation from NL Text, IEEE International EDOC conference 2010, Vitoria, Brazil

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

[4] Engels G., Heckel R., K¨uster J. 2001. Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model, LNCS Vol. 2185, pages 272-287

[5] Linehan M. 2008. Ontologies and rules in Business Models. 11th IEEE EDOC Conference Workshop, pp. 149-156,

[6] Linehan M. 2008. SBVR Use Cases. International Symposium on Rule Representation, Interchange and Reasoning on the web, RuleML, LNCS Vol.5321 pp.

182-196

[7] Cabot J., et al. 2009. UML/OCL to SBVR Specification: A challenging Transformation, Journal of Information systems doi:10.1016/j.is.2008.12.002

[8] Bajwa, I. S., Choudhary M.A. 2006. A Rule Based Paradigm for Speech Language Context Understanding. International Journal of Donghua University (English Edition). 23, 06 (June 2006), 39-42.

[9] Toutanova. K., Manning, C.D. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. 63-70. Hong Kong.

[10] Ilieva M., Olga O. 2005. Automatic Transition of Natural Language Software requirements Specification into Formal Presentation. Springer LNCS Vol. 3513, pp.392--397 (2005)

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

[12] Bryant B., et al. 2008. From Natural Language Requirements to Executable Models of Software Components. In Workshop on S. E. for Embedded Systems pp.51-58

[13] Leidner, J. L. (2003). Current issues in software engineering for Natural Language Processing. HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems – Vol. 8 (May 31 - 31, 2003), Morristown, NJ. p.45-50.

[14] Perez-Gonzalez, H. G., Kalita J. K. (2002). Automatically Generating Object Models from Natural Language Analysis.17th Annual ACM SIGPLAN conference, OOP, Systems, languages & applications. p.86-87.

[15] Mich, L. (1996). NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. Natural Language Engineering. 2(2) p.167-181.

[16] Li, K., Dewar R. G., Pooley R. J. (2004). Object-Oriented Analysis Using Natural Language Processing. Heriot-Watt University Technical Reports. www.macs.hw.ac.uk:8080/techreps/docs/files/HW MACS-TR-0033.pdf

[17] Seco, N., Gomes P., Pereira F.C. (2004). Using CBR

for Semantic Analysis of Software Specifications. Advances in Case-Based Reasoning. LNCS Vol. 3155/2004 p.41-43.

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

[19] Omar, N., Hanna P., Kevitt P. (2006). Semantic Analysis in the Automation of ER Modelling through Natural Language Processing. International Conference Computing & Informatics, June 2006. pp.1-5.

[20] Gomez, F., Segami C., Delaune C. (1999). A system for the semiautomatic generation of E-R Models from Natural Language Specifications. Data and Knowledge Engineering 29 (1) p.57-81

[21] Al-Safadi, L.A.E. (2009). Natural Language Processing for Conceptual Modeling. International Journal of Digital Content Technology and its Applications. 3 (3).

[22] Hart, G., Johnson M., Dolbear C. (2008). Rabbit: Developing a Control Natural Language for Authoring Ontologies. 5th European Semantic Web Conference (ESWC'08). 348-360.

[23] Spreeuwenburg, S., Healy K. A. (2009). SBVR's Approach to Controlled Natural Language. Workshop on Controlled Natural Language 2009, Marettimo Island, Italy Available: http://sunsite.informatik.rwthaachen.de/ publications/ CEUR-WS/Vol-448/paper26.pdf

[24] Kleiner, M., Albert P., Bézivin J. (2009). Parsing SBVR-Based Controlled Languages. Model Driven Engineering Languages and Systems 2009. LNCS Vol. 5795/2009 p.122-136.

[25] SBeaVer, http://sbeaver.sourceforge.net/