A NATURAL LANGUAGE METAMODEL FOR GENERATING CONTROLLED NATURAL LANGUAGE BASED REQUIREMENTS

Shabana Ramzan[1], Imran Sarwar Bajwa[2], Bushra Ramzan[1]
[1] Department of Computer Science & IT, Islamia University of Bahawalpur
[2] School of Computer Science, University of Birmingham, UK
shabanaramzan@hotmail.com, i.s.bajwa@cs.bham.ac.uk

ABSTRACT.: *In natural language processing, generation of formal representation such as Semantics of Business Vocabulary and Rules (SBVR) or First-Order-Logic (FOL) from Natural Language (NL) text is an important phase. In recent years, model transformation has been an efficient way of transforming a source to a target representation. For NL to SBVR model transformation, metamodels of both source (such as NL) and target (such as SBVR) representations are required but currently there is no NL metamodel available. In this paper, we present a primary version of NL metamodel that can be used to model transform NL requirements to SBVR from a restricted domain. The results of preliminary experiments are very encouraging. The results manifest that the output of our approach can be machine processed for automatic generation of precise and reliable software models from NL requirements.*

Keywords: SBVR, Requirement Specification, Model Transformation, Transformation Rules.

I. INTRODUCTION

Requirement specification document is the key constituent for software development. System analyst usually specifies software requirements in natural language and inherent ambiguity of natural language creates conflict between clients and software developers on the interpretation of a requirement. Our research will provide the solution to avoid these conflicts by generating controlled natural language based requirements. We presented an approach to transform NL based requirements into the SBVR (Semantics of Business Vocabulary and Business Rules) [1] standard of OMG (Object Management Group) by using model transformation. For NL to SBVR transformation, we require source and target metamodels. In this paper, firstly we present NL metamodel and then transform this metamodel into SBVR metamodel to provide controlled representation of requirements. SBVR based requirements are machine processable which resolve the issue of ambiguity. SBVR based requirements lessen the semantics changes and reduce disasters in software industry caused by wrong communication between user and development team.

The remaining paper has following sections, section II presents related work, section III describes basic concepts involved in model transformation; section IV states the model transformation from NL to SBVR, section V illustrates the case study and followed by experiments and results section. The last section covers conclusions.

2. Preliminaries

Our research work presented the model transformation of natural language metamodel to SBVR metamodel as well as proposed natural language model.

2.1 Semantics of Business vocabulary and Rules

SBVR (Semantics of Business Vocabulary and Rules) [1] is a standard established by object management group. If we transform natural language requirement specification into SBVR based requirements, we have an opportunity to easily process these requirements by machine due to formal logic of SBVR. Semantic models for business rules and vocabulary are developed by metamodel, which is defined by SBVR [1].

SBVR has two main constituents, SBVR rules and vocabulary. The detail description of SBVR constituents is given below.

SBVR Business Vocabulary: The main elements of SBVR vocabulary are concepts and fact types. Business people used SBVR vocabulary for their official writing. Concepts are of three types, Object Types and Individual Concepts and Verb Concepts. Common nouns of natural language text are represented as object types or general concepts and proper noun as individual concept. All helping verb and action verb are classified as verb concept. A proposition, a verb or a combination of verb and preposition is denoted as fact type [1].

SBVR Business Rules: These rules provide guidance about the actions taken for business and also define the structure of business. There are two types of rules

- Definitional rules or structural rules: The setup of organization is defined by these rules [1] e.g. It is possible that each customer can order for more than one meal per day.
- Behavioral rules or operative rules: The behavior of an entity is characterized by these rules [1] e.g. It is obligatory that each truck can transport the tools to factory.

Semantic Formulation of NL Text: SBVR rules are semantically formulated by using logical formulations. A set of logical formulations are defined in SBVR document [1]. The semantic formulations are used to control English statements such as atomic Formulation, instantiation Formulation, logical Operations, quantifications, and modal formulation.

SBVR Based Notation for NL Text: In order to formalize natural language text according to some standard, one of the possible notations is structure English, in annex C of SBVR 1.0 document [1]. We have used the following formatting rules of SBVR Structured English.

- Double underlined all individual concepts e.g. gold loan customer
- Underlined all noun concepts e.g. student
- SBVR keywords are bolded e.g. at least, it is possible, it is obligatory, some, each, etc.
- All verbs are italicized e.g. has , should be

In this paper, we have italicized the adjectives and possessive nouns like verb concepts but used different color to represent them.

2.2 Model Transformation

The components required for model transformation from NL metamodel to SBVR metamodel are source metamodel (NL), transformation engine, transformation rule, transformation description. We used Sitra [17] as a transformation engine.

Transformation engine transform the source metamodel to target metamodel by using transformation rules. The complete sketch of model transformation is depicted in Figure 1.

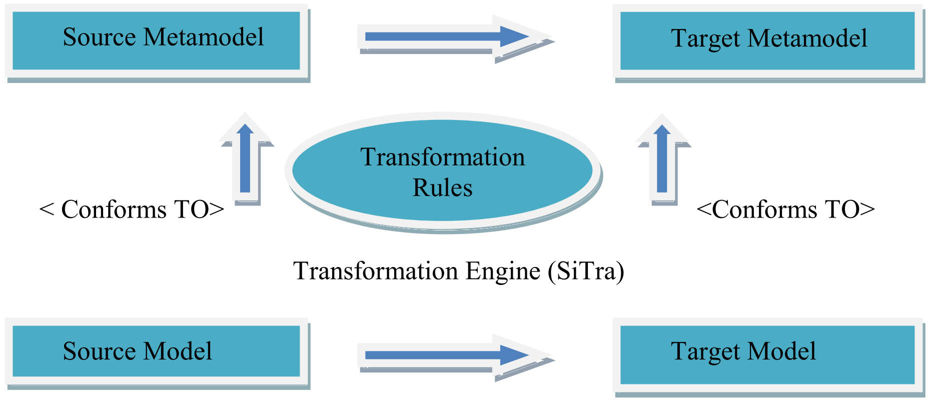

Fig. 1. Model Transformation from NL to SBVR

MDA (Model Driven Architecture) is model based approach to develop soft elements of source metamodel into related elements of target metamodel. This mapping is used to generate transformation rules. Transformation engine execute these rules to automatically generate target metamodel from source metamodel. . Model to model, model to text and text to model, are various types of model transformation. Model to model transformation transforms a model into another model. In our study, we employed the model to model transformation.

2.3 Natural Language Metamodel

There is no standard metamodel for natural languages such as English. ProjectIT-RSL metamodel is natural language based metamodel proposed by Videria and Silva [2]. But this metamodel has some deficiencies, all concepts of natural language (such as English) are not described e.g. actor, operation and entity are commonly used semantic role labels but not represented in the metamodel. However, in our research, a complete metamodel of English is required to transform natural language (such as English) to SBVR. We have complemented the ProjectIT-RSL natural language based metamodel (see Figure 2) to model transform natural language to SBVR. Figure 2 shows the newly added elements in ProjectIT-RSL metamodel and the metamodel is called English metamodel.

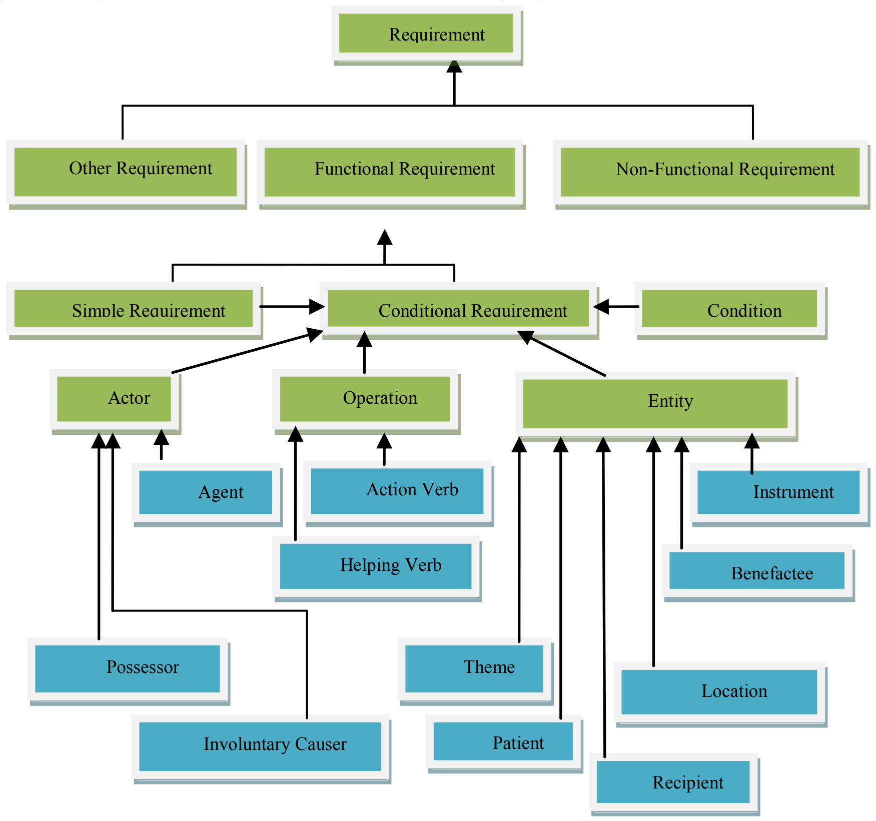

Fig. 2. Proposed NL Metamodel

3. Used Approach

To translate natural language into SBVR, the approach will take NL requirements as input and then these requirements are processed to obtain SBVR elements. This system transformed NL metamodel into SBVR metamodel in three steps, firstly requirements are POS (Parts–of–Speech) tagged, then parser is used to obtain basic SBVR elements (such as individual concept, verb concepts, noun concepts, objects type, etc. In second step, NL to SBVR mapping is performed through various sub steps. In last step, transformation rules are generated to transform NL metamodel into SBVR metamodel.

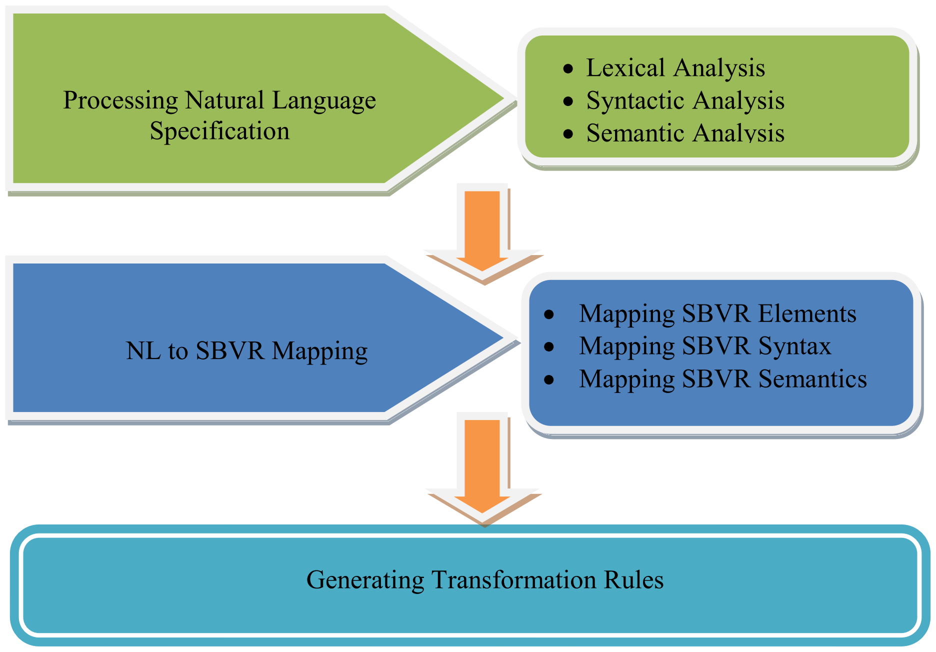

Fig. 3. Skeleton of Designed System

3.1 Processing Natural Language Specification

Lexical Analysis: Various tasks are performed to accomplished lexical analysis such as sentence splitting, tokenization, POS tagging, morphological analysis. The input for Lexical processor is NL requirements and output is the list of tokens with related lexical detail. The lexical analysis has following steps:

a) *Tokenization*: In first step, the natural language text is read and tokenized to produce tokens e.g. "Customer should have business account to get credit card" is tokenized as [Customer] [should] [have] [business] [account] [to] [get] [credit] [card] [.]

b) *Sentence Splitting*: In the next step the margins of a sentence is recognized by sentence splitter and then use array-list to store all sentences separately.

c) *Parts-of-Speech (POS) Tagging:* POS tagging marking up the tokens to respective part of speech, based on definition as well as its context such as noun, verb, adverb, adjective, helping verb, pronoun, prepositions etc. POS tagger has been identified by Stanford POS tagger v 3.0 in POS tagging [2].

d) *Morphological Analysis*: Morphological analysis is performed for structuring and analyzing complex problems. Morphological analysis performed on all verbs and nouns. By morphological analysis detaches suffixes normally attach to the nouns and verbs. For Example, a noun "students" is analyzed as "student + s" and a verb "selected" is analyzed as "select + ed

Syntactic Analysis: The structure of text is determined by syntactic analysis. The text is syntactically analyzed to generate parse tree. Figure 4 shows the parse tree of above example.

Parse Tree:
```
(ROOT
  (S
    (NP (NN Customer))
    (VP (MD should)
      (VP (VB have)
        (S
          (NP (NN business) (NN account))
          (VP (TO to)
            (VP (VB get)
              (NP (NN credit) (NN card)))))))
    (. .)))
```

Fig.4. Parse Tree of NL Text

Semantic Analysis: In semantic analysis phase firstly find out the meanings of all words and then unite them to uncover the meaning of sequence of words. Semantic analysis input is parse tree and output is the appropriate representation of text. Semantic role labeling is carried out in this phase [14]. Semantic role labeling is perform during natural language processing for identifying semantic roles used with the verb in a sentence. The meanings of the input sentence can easily understand by the identification of such information as given below.

AGENT: Agent is basically a participant who performs the action.

RECIPIENT: Endpoint of a transferred item is animated as a recipient.

PATIENT: If the action of a verb affects any participant, that participant is animated as a patient.

THEME: If predicate properties, location or involuntary movement of a participant, that participant animated as theme.

INSTRUMENT: If participant use an instrument which becomes the cause of some event or situation.

BENEFACTEE: when the action or situation is performed which provides benefits to participant, and then this participant is benefactee.

INVOLUNTARY CAUSER: The participant who is responsible for any event which is not performed for this purpose (intention).

LOCATION OR LOCATIVE PARTICIPANT: If the location of situation or an action, or the path, goal or source of a moving object is described by participant.

POSSESSOR: If some participant owned or temporarily controlled by some other participant, then participant who controlled is possessor.

These roles help out to semantically analyze the English text as shown in Figure 5:

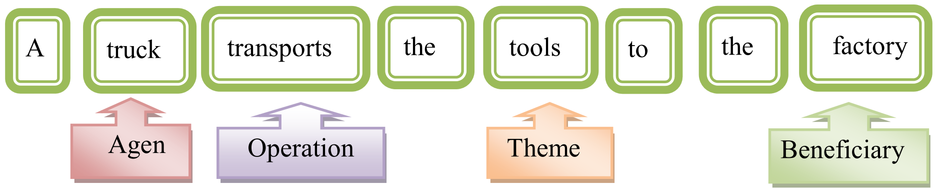

Fig. 5. Semantic Processing of English Text

According to the semantic role truck is identified as an agent, transports as an operation and tools as a theme and factory as a benefactee.

3.2 NL to SBVR Mapping

Mapping SBVR Elements: For NL to SBVR mapping, elements of natural language are mapped to corresponding SBVR elements. Table I shows the mapping of NL elements to SBVR elements.

Table I: Mapping Between NL and SBVR Metamodel Element

NL Metamodel Elements	*SBVR Metamodel Elements*
Nouns e.g., Agent | Involuntary causer | Theme | Patient | Benefactee	Object Type
Proper nouns e.g., Agent | Involuntary Causer| theme | Benefactee | Patient	Individual Concept
Helping Verb +Action Verb | Action Verb	Verb Concept
Object Type| Individual Concept + verb concept	Unary fact Type
Object Type| Individual Concept + verb concept+ Object Type	Binary Fact Type/ Associative Fact Type
Adjective Noun and Possessive Noun	Characteristics
Enumeration of Verb Concept or Noun Concept	Quantification
structures such as “is-part-of”, “included-in” or “belong-to”	Partitive Fact Type
structures such as “is-category-of” or “is-type- of”, “is-kind-of”	Categorization Fact Type

Mapping SBVR Syntax: Information required for the transformation of NL text to SBVR representation is extracted from syntax rules. Table II shows SBVR syntax model,

Table II: SBVR Syntax Model

Logical Formulations		SBVR
SBVR rule	⟶	Modal Formulation +Fact Type
Modal Formulation	⟶	Necessity Formulation/obligation Formulation/ Permissibility Formulation/ Possibility Formulation
Fact Type	⟶	Subject + Verb Concept + Object
Subject	⟶	Noun Concept
Object	⟶	Noun concept/Object Type + Characteristics
Verb Concept	⟶	Verb/Helping Verb + Action Verb
Noun Concept	⟶	Object Type/Individual Concept
Object Type	⟶	Object Type/Object Type + Logical Operators + Object Type/Quantification + Object Type
Individual Concept	⟶	Individual Concept/Individual Concept + Logical Operators + Individual Concept / Quantification + Individual Concept

Mapping SBVR Semantics: In SBVR version 1.0, there are five types of logical formulations but our approach used three logical formulations, quantification, logical operations and model operations.

Table III. SBVR Quantification

Logical Formulations	SBVR
at least one	existential quantification
exactly one	exactly-one quantification
each	universal quantification
at most one	at-most-one quantification
more than one	at-least-n quantification with n = 2
at least n	at-least-n quantification
exactly n	exactly-n quantification
at most n	at-most-n quantification
some	existential quantification

Table IV: SBVR Logical Operations

Logical Formulations	SBVR
it is not the case that p	logical negation
if p then q	implication
q if p	implication
p or q	disjunction
p if and only if q	equivalence
p and q	conjunction

Table V: SBVR Modal Operation

Logical Formulations	SBVR
always	necessity formulation
must	obligation formulation
never	necessity formulation
must not	obligation formulation
may	permissibility formulation

3.3 Generating Transformation Rules

The transformation of the source model into the target model is described by the transformation rules. There are two parts for each transformation rule, right hand side (RHS) for target pattern and left hand side (LHS) side for source pattern.

Rule 1 [SBVR-Rule (modal-formulation, fact-Type)]
= modal-formulation + fact-type
Rule 1.1 [Modal-Formulation (modal-formulation]
= modal-formulation
Rule 1.1.1 [Modal-Formulation (necessity-formulation)]
= “It is necessary that”
Rule 1.1.2 [Modal-Formulation (obligatory-formulation)]
= “It is obligatory that”
Rule1.1.3 [Modal-Formulation (permissibility-formulation)]
= “It is permissible that”
Rule 1.1.4 [Modal-Formulation (possibility-formulation)]
= “It is possible that”
Rule 2 Fact-Type (subject, verb-concept, Object)
= subject + verb-Concept + Object

Rule 2.1 Subject-part (subject)
= subject
Rule 2.1.1 Subject (noun-concept)
= noun concept
Rule 2.2 Verb-Concept (verb-concept)
= verb-concept

Rule 2.2.1 Verb-Concept (action-verb)
=action-verb
Rule 2.2.2 Verb-Concept (helping-verb, action-verb)
= helping-verb + action-verb
Rule 2.3 Object-part (object)
=object
Rule 2.3.1 Object (noun-concept)
=noun-concept
Rule 2.3.2 Object (Object–Type, Characteristics)
= object–type + characteristics
Rule 3 Noun-Concept (noun-concept)
= noun-concept
Rule 3.1Noun-Concept (object-type)
= object-type
Rule 3.1.1 Object-Type (object-type)
= object-type
Rule 3.1.2 Object-Type (object-type, logical-operators,
Object-type)
= object-type + logical-operators+ object-type
Rule 3.1.3 Object-Type (quantification, object type)
= quantification + object type
Rule 3.2 Noun-Concept (individual-concept)
= individual-concept
Rule 3.2.1 Individual-Concept (individual-concept)
= individual-concept
Rule 3.2.2 Individual-Concept (individual-concept, logical- operators, individual-concept)
=individual-concept + logical-operator + individual- concept
Rule 3.2.3 Individual-Concept (quantification, individual- concept)
= quantification + individual-concept

4. Case study

We illustrate a case study of KeePass Password Safe [23] to show the performance of our approach in terms of accuracy and fulfillment of user need. The problem statement of case study is

KeePass consists of a database which contains data for one or more users. Each user's data are divided into groups and subgroups so that they are organized in a form that serves right the user. Every user has a unique Master Key which can be simple or composite and its combination opens uniquely the database. If lost there is no recovery. Groups and subgroups contain entries with usernames, passwords URLs etc that can be sent or copied to websites, application and accounts. There is also the ability for a onetime key creation to be used once in a transaction without the risk of reused by others for any reason.

NL (English) text of the problem statement of case study is parsed lexically, semantically and syntactically to extract SBVR vocabulary.

Table VI: SBVR Vocabulary Extracted from NL Text

Category	*Count*	*Details*
Object Types	15	keypass, user, database, data, masterkey, simple, composite, groups, subgroups, applications, webpage, accounts, onetimekey, form, transaction
Individual Concept	00	
Verb Concept	09	consists, contains, divided, organized, serves, sent, copied, reused, opens
Unary Fact Type	02	opens database, transactions reused
Associative Fact Types	06	database for user, form serves user, user has Masterkey, its open database, Group and Subgroup send or copy entries to websites, applications, and accounts, OneTimeKey used in a transaction
Characteristics	03	user name, password, url
Quantification	02	One, more
Categorization Fact Types	02	data is divided into groups and subgroups, MasterKey can be simple or composite base contains data
Partitive Fact Types	02	KeyPass consists database, database contains data

There are four requirements for the problem statement of the used case study, as shown in table VII:
According to SBVR structured English underlined the object types e.g. groups, subgroups, accounts etc. italicized the verb concepts e.g. has, can, contain etc. the SBVR keywords are as e.g. It is necessary, It is obligatory etc. The purple color used for characteristics which are also italicized e.g. usernames, passwords, etc

Table VII: SBVR Based Software Requirements

Category	*Count*	*Details*
Software Requirements	04	It is necessary that each user's data are *divided* into groups and subgroups so that they are *organized* in a form that *serves* right the user. It is obligatory that every user *has* a unique Master Key which *can* be simple or composite and its combination *opens* uniquely the database. If lost there *is* no recovery. It is necessary that Groups and subgroups *contain* entries with *usernames*, *passwords*, *URLs* etc that can be *sent* or *copied* to websites, application and accounts. It is possibility that there *is* also the ability for a onetimekey creation to be used once in a transaction without the risk of *reused* by others for any reason.

5. EXPERIMENTS AND RESULTS

In order to evaluate the performance of our presented approach, we have solved five

Case studies including the case studies discussed in section V. Table VIII shows the computed average recall, precision and F-value by using the results of all these case studies.

Average recall for these case studies is 83.22% and average precision is 87.13%, calculated average F-value is 85.14 % that is very supportive for future development. For all solved case studies, the Figure 6 shows the Recall, precision and F-Value.

Table VIII: Evaluation Results for NL to SBVR Model Transformation

Input	N_{test}	$N_{approved}$	N_{wrong}	$N_{omitted}$	*Rec%*	*Prec%*	*F-Value*
C1	63	51	09	03	80.95	85.00	82.93
C2	48	39	07	02	81.25	84.78	82.98
C3	39	30	08	01	76.92	78.95	77.92
C4	51	43	06	02	84.31	87.76	86.00
C5	50	44	03	03	88.00	93.61	90.72
C6	58	51	04	03	87.93	92.73	90.27
				Average	83.22	87.13	85.14

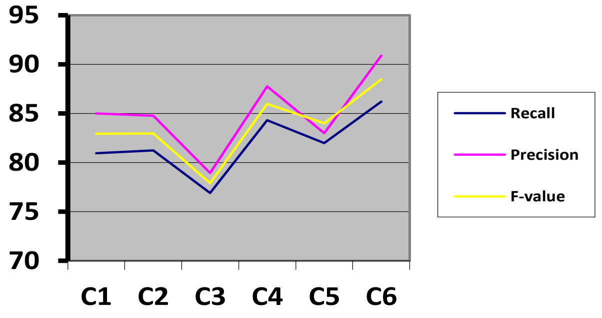

Fig.6. Recall, Precision and F-Value for set of case studies

The presented approach is able to solve the examples that have simple vocabulary. However, the presented approach is not able to handle the examples having textual entailments and discourse connections n the NL examples.

6. RELATED WORK

Natural languages requirements are processed by different tools to control ambiguity problems of NL. CM builder is a case tool proposed by Harmain to create conceptual model from NL requirements using UML [4]. LOLITA proposed a case tool NL-OOPS which gives object model to improve the process of software development [3]. Different solutions are presented [10,11,12,13,14] to generate UML models from natural language requirements

In last few decades various controlled natural languages are used instead of natural language to communicate requirements, collected by system analyst for development team. Controlled natural languages are of two types, human oriented [8] and machine oriented [9]. The human oriented controlled natural language is PENG (Processable English) [5], ACE (Attempto Controlled English) [6], CPL (Computer Processable Language) [7], etc. Brillant used SBVR to represent natural language requirements into models that can be executed [15]. Umber presented a SR-elicitor tool to generate ambiguity less requirement document by using SBVR [16]. Firstly this tool analyzed NL text lexically, semantically and syntactically and then extract SBVR elements to generate SBVR based rules.

Bajwa presented SBVR2OCL prototype tool to automatically generate OCL constraints [17]. This approach facilitates the process of software development. Firstly transform NL text into SBVR and then generate OCL constraints from SBVR. For SBVR to OCL constraints transformation involved different steps, firstly objects oriented information is extracted from SBVR rule, and then generate OCL expression from this extracted information and finally mapped OCL syntax and semantics. Raj presented transformation technique to generate different UML diagrams from SBVR rules and vocabulary [18]. In the domain of model transformation presented work [19,20,21] to transform different models into SBVR and as a reverse engineering generate SBVR model from other models.

The related work shows that the approach of model transformation has already been used to transform various models into other models, but no one tried to transform NL metamodel into SBVR metamodel. Our research presented the transformation of NL metamodel to SBVR metamodel.

7 CONCLUSION AND FUTURE WORK

The most important benefit of this research was to gain software requirements specification with minimum ambiguity by generating SBVR based requirement specification using model transformation approach. For model transformation required source metamodel (NL) and target metamodel (SBVR).Target metamodel is standard and available in SBVR 1.0 document.NL metamodel is not available, our research also proposed NL metamodel. This research transforms NL metamodel to SBVR metamodel by using Sitra transformation engine.

REFERENCES

[1] OMG. 2013. Semantics of Business vocabulary and Rules (SBVR), OMG Standard, v. 1.2.

[2] Silveira, N., & Manning, C. (2015). Does universal dependencies need a parsing representation? an investigation of english. *Depling 2015*, 310.

[3] 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.

[4] Harmain, H. M., Gaizauskas, R. (2003). CM-Builder: A Natural Language-based CASE Tool For object Oriented Analysis. Journal of Automated Software Engineering, 10(2), 157-181.

[5] Nojiri, S., & Manning, C. D. (2015, October). Software Document Terminology Recognition. In *2015 AAAI Spring Symposium Series*.

[6] Fuchs, N.E., Kaljurand, K., and Kuhn, T. (2008). Attempto Controlled English for Knowledge Representation. In: Reasoning Web, LNCS, vol. 5224/2008:104–124.

[7] Mott, D. H., Shemanski, D. R., Giammanco, C., & Braines, D. (2015, May). Collaborative human-machine analysis using a controlled natural language. In *SPIE Sensing Technology+ Applications* (pp. 94990J-94990J). International Society for Optics and Photonics.

[8] Schwitter, R. 2010. Controlled Natural Languages for Knowledge Representation, Coling 2010: Poster Volume, Beijing, August 2010: 1113–1121

[9] Huijsen, W.O. 1998. Controlled Language –An introduction. In: Proceedings of CLAW 98:1–15.

[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] Ben Abdessalem Karaa, W., Ben Azzouz, Z., Singh, A., Dey, N., S Ashour, A., & Ben Ghazala, H. (2015). Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements. *Software: Practice and Experience*.

[12] Oliveira A., Seco N., Gomes P (2004). : A CBR Approach to Text to Class Diagram Translation, In TCBR Workshop at the 8th European Conference on Case-Based Reasoning. (2004).

[13] 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.

[14] 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.

[15] Kouamou., G. E., Feuto Njonko, P. B. (2010). Cohérence devues dans la spécification des architectures logicielles.In proceeding of 10th African Conference on Research in Computer Science and Applied Mathematics (CARI'2010), Ivory Coast, October 18–21, 2010.

[16] Umber, A., Bajwa, I. S. (2012). A Step towards Ambiguity less Natural Language Software Requirements Specifications. IJWA, 4(1), 12-21.

[17] Bajwa I.S., Behzad Bordbar, Mark G. Lee, (2010), OCL Constraints Generation from NL Text, IEEE International EDOC conference 2010, Vitoria,Brazil,pp.204-213.

[18] Raj A., Prabharkar T., Hendryx S., (2008). Transformation of SBVR Business Design to UML Models., In ACM Conference on India software engineering, pp.29-38.

[19] Malik, S., Bajwa, I. S. (2013). Back to Origin: Transformation of Business Process Models to Business Rules. In Business Process Management Workshops (pp. 611-622). Springer Berlin Heidelberg.

[20] freen, H., Bajwa, I.S., Bordbar, B. (2011) SBVR2UML: A Challenging Transformation, Frontiers of Information Technology (FIT), 2011 9th International Conference, pp:33-38

[21] Kouzari, E. (2008). Software Requirements Specifications for KeePass Password Safe. Software Engineering, Aristotle University Thessaloniki, Available at: http://keepass.info/extensions/base/docs/SoftwareRequirementSpecification-KeePass-1.10.pdf