A Step Towards Ambiguity Less Natural Language Software Requirements Specifications

Ashfa Umber[1], Imran Sarwar Bajwa[2]
[1] Department of Computer Science & IT
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
[2] School of Computer Science
University of Birmingham
Birmingham, UK
ashfaumber@yahoo.com, i.s.bajwa@cs.bham.ac.uk

ABSTRACT: *In modern software engineering practice, the ability to specify ambiguity less software requirements in a natural language (NL) in a seamless way is highly valuable and desirable. Though, the software requirements are typically captured in natural languages (NL) such as English, there is a very high probability that more than half NL requirements can be ambiguous. For example, Mich identified that approx. 72% of the NL requirements are potentially ambiguous. A primary reason of such ambiguous NL requirements is syntactic and semantic ambiguities in a natural language such as English. A problem with ambiguous NL requirements is that a software engineer can miss-interpret requirements and can generate an erroneous and absurd software model. In this paper, we aim to address this challenge by presenting a novel approach that is based a semantically controlled NL representation for software requirements. To generate a semantically controlled NL representation, we propose the use of Semantic of Business Vocabulary and Rules (SBVR) standard. We solve a case study to bear out that a SBVR based controlled representation can not only help in generating accurate and consistent software models but can also simplify the machine processing of requirements. The results show that our approach can be helpful in generating the accurate and consistent software models from NL software requirements. A Java implementation of the used approach is also presented a proof of concept that is also available as an Eclipse plugin.*

Keywords: Software Requirements Specifications, Ambiguity in NL Specifications, SBVR

Received: 1 September 2011, Revised 23 November 2011, Accepted 30 November 2011

1. Introduction

It is a typical practice that software requirements are specified in natural languages (NL). It is a common knowledge that 71.80% of the software requirements specifications are captured in NL [1]. However, the natural languages are intrinsically ambiguous. For automated software modeling, impervious and explicit software requirements are a primary necessity as computers cannot accurately process ambiguous requirements. A few scientists have proposed various approaches to identify and measure the typical ambiguities in NL based software requirements specifications (SRS) e.g. Kiyavitskaya et al. [3] presented a couple of tools to identify ambiguous sentence in a NL SRS document and find the reason of ambiguity. Similarly, Popescu et al. presented a tool *Dowser* [4] to identify ambiguous and inconsistent sentences in a NL SRS. However, a drawback of the used approach is that input should be in a constrained language, and this pitfall makes the approach impractical. According to our knowledge, there is no appropriate approach or tool that can provide an automatic procedure of minimizing or removing ambiguity in NL SRS.

In this paper, we aim to present an approach capable of automatically generating an unambiguous and semantically consistent representation of SRS specified in English language. To achieve a semantically controlled representation, we propose the use of Semantic of Business Vocabulary and Rules (SBVR) 1.0 [4]. SBVR is an OMG standard, initially presented to assist business requirements specifiers and analyzers. In [5] and [18] we presented that similar to business requirement, the software requirements can be captured and specified using SBVR syntax. In this paper, we propose the use of SBVR to overcome the typical ambiguities in a natural language. The SBVR incorporate not only ability of generating accurate and consistent software representation but also provides capability of machine processing as SBVR is based on mathematical or higher order logic [4]. The presented approach is also implemented in Java. The performance of the tool is evaluated by solving a case study, presented in section 4. The remaining paper is structured into the following sections: Section 2 states preliminaries of the presented research. Section 3 presents the framework for translation of English to SBVR representation. Section 4 presents a case study. The evaluation of our approach is presented in section 5. Finally, the paper is concluded to discuss the future work.

2. Semantic Business Vocabulary and Rules (SBVR)

In 2008, OMG presented a new standard *Semantic Business Vocabulary and Rules* (SBVR) [4]. SBVR supports capturing of requirement in a controlled natural language. There are various controlled natural languages such as Attempto but we have used SBVR due to following reasons:

• SBVR is a standard. Latest available version is 1.0.

• SBVR is easy to read and understand for human beings as SBVR uses syntax of natural languages e.g. English.

• SBVR is easy to machine translate as it is based on higher logic such as First Order Logic (FOL).

A typical SBVR representation is based on a set of SBVR business vocabulary and SBVR business rules in particular business domain.

2.1 SBVR Business Vocabulary

A business vocabulary [4] (section: 8.1) consists of all the specific terms and definitions of concepts used by an organization or community in course of business. In SBVR, A concept can be a noun concept or fact type. Noun concepts can be further categorized into object type, individual concept, and characteristic. Hence we have four key elements in SBVR:

• In SBVR, an object type is a general concept that exhibits a set of characteristics to distinguishes that object type from all other object types" [3] (section: 8.1) e.g. robot, user, etc.

• In SBVR, an individual noun is a qualified noun that corresponds to only one object [3] (section: 8.1) e.g. 'Robby', a famous robot.

• In SBVR, characteristic is an abstraction of a property of an object [4] (section: 8.1) e.g. name of robot is Robby, here name is characteristic.

• In SBVR, a fact type or a verb concept [4] (section: 8.1) specifies the relationships among noun concepts e.g. car has wheels. A fact type can be binary fact type e.g. "customer *places* orders".

2.2 SBVR Business Rules

A SBVR business rule is a formal representation under business jurisdiction 'Under business jurisdiction' [4]. Each SBVR business rule is based on at least one fact type.

• The SBVR rules can be a structural business rule [4] (section: 12.1) those are used to define an organization's setup.

• Another type of SBVR rules *is* a behavioural business rule [4] (section: 12.1) and they are used to express the conduct of a business entity.

2.3 SBVR based Controlled Representation

SBVR was originally presented to assist business people in creating clear and unambiguous business policies and rules in their native language [4]. The following characteristics of SBVR can help in generating a controlled representation of English:

2.3.1 Rule-based Conceptual Formalization

SBVR standard provides a rule-based conceptual formalization that can be employed to generate a syntactically formal

representation of English. SBVR contains a vocabulary for conceptual modeling and captures expressions based on this vocabulary as formal logic structures. The SBVR vocabulary can be used to formally specify representations of concepts, definitions, instances, and rules of any knowledge domain in natural language. These features make SBVR well suited for describing business domains and software requirements to implement software models.

2.3.2 Natural Language Semantic Formulation

SBVR is typically proposed for business modeling in NL. However, we are using the formal logic based nature of SBVR to semantically formulate the English software requirements statements. A set of logic structures called semantic formulations are provided in SBVR to make English statements controlled such as atomic formulation, instantiate formulation, logical formulation, quantification, and modal formulation.

2.3.3 SBVR Formal Notation

Structured English is one of the possible SBVR notations, given in SBVR 1.0 document, Annex C [4], is applied by prefixing rule keywords in a SBVR rules. The other possible SBVR notation is Rulespeak, given in SBVR 1.0 document, Annex F [4], uses mixfixing keywords in propositions. SBVR formal notations help in expressing propositions with equivalent semantics that can be captured and formally represented as logical formulations.

3. Translating NL to SBVR

This section briefly explains how English text is mapped to SBVR representation and object oriented information is extracted from SBVR representation. Figure 1 show the used approach that works in three phases:

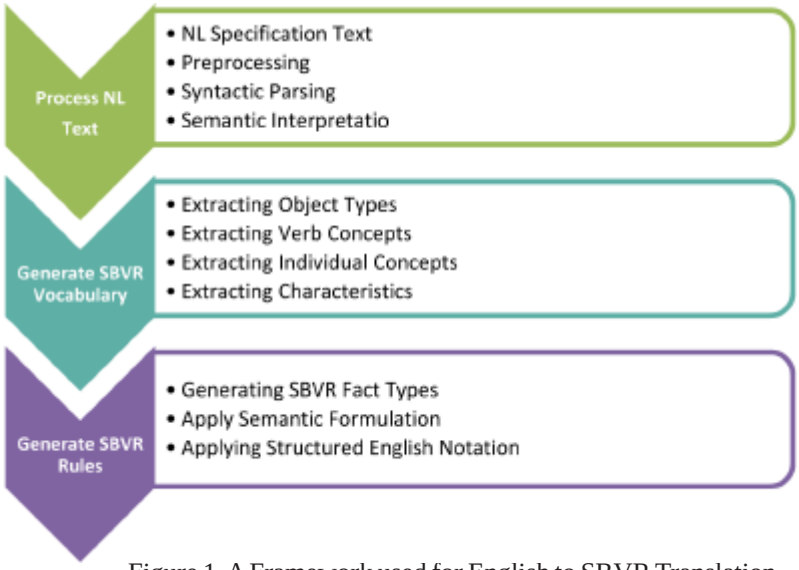

Figure 1. A Framework used for English to SBVR Translation

3.1 Parsing NL Software Requirement Text

The first phase of SR-Elicitor [5], [17] is NL parsing that involves a number of processing units (organized in a pipelined architecture) to process complex English statements. The NL parsing phase lexically, syntactically and semantically processes the English text as following:

3.1.1 Lexical Processing

The NL parsing starts with the lexical processing of a plain text file containing English software requirements specification. The lexical processing initiates with the tokenization of the input English text.The tokenized text is further passed to Stanford parts-of- speech (POS) [13] tagger v3.0 to identify the basic POS tags e.g.

3.1.2 Syntactic Processing

We have used an enhanced version of a rule-based bottom-up parser for the syntactic analyze of the input text used in [11]. English grammar rules are base of used parser. The text is syntactically analyzed and a parse tree is generated for further

semantic processing as shown in Figure 2.

A task is a component of the schedule with a start and end date.
[A/DT] [task/NN] [is/VBZ] [a/DT] [component/NN] [of/IN] [the/DT] [schedule/NN] [with/IN] [a/DT] [start/NN] [and/CC] [end/NN] [date/NN].

Figure 2. A Framework used for English to SBVR Translation

A task is a component of the schedule with a start and end date.
(ROOT (S (NP (DT A) (NN task)) (VP (VBZ is) (NP (NP (DT a) (NN component)) (PP (IN of) (NP (NP (DT the) (NN schedule)) (PP (IN with) (NP (DT a) (NN start) (CC and) (NN end) (NN date)))))))))

Figure 3. Parsing English text using Stanford Parser

3.1.3 Semantic Interpretation

In this semantic interpretation phase, role labeling [12] is performed. The desired role labels are actors (nouns used in subject part), co-actor (additional actors conjuncted with 'and'), action (action verb), thematic object (nouns used in object part), and a beneficiary (nouns used in adverb part) if exists. These roles assist in identifying SBVR vocabulary.

3.2 Extracting SBVR Vocabulary

In this phase, the basic SBVR elements e.g. noun concept, individual concept, object type, verb concepts, etc are identified from the English input that is preprocess in the previous phase. The extraction of various SBVR elements is described below:

3.2.1 Extracting Object Types

All common nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the object types or general concept (see figure 3) e.g. belt, user, cup, etc. In conceptual modelling, the object types are mapped to classes.

3.2.2 Extracting Individual Concepts

All proper nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the individual concepts.

3.2.3 Extracting Fact Types

The auxiliary and action verbs are represented as verb concepts. To constructing a fact types, 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.

3.2.4 Extracting Characteristics

In English, 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.

3.2.5 Extracting Quantifications
All indefinite articles (a and an), plural nouns (prefixed with s) and cardinal numbers (2 or two) represent quantifications.

3.2.6 Extracting Associative Fact Types
The associative fact types [4] (section 11.1.5.1) 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. In conceptual modeling of SBVR, associative fact types are mapped to associations.

3.2.7 Extracting Partitive Fact Type
The partitive fact types [4] (section 11.1.5.1) 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'. In conceptual modeling of SBVR, categorization fact types are mapped to aggregations.

3.2.8 Extracting Categorization Fact Types
The categorization fact types [4] (section 11.1.5.2) 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'. In conceptual modeling of SBVR, categorization fact types are mapped to generalizations. All the extracted information shown in figure 4 is stored in an arraylist for further analysis.

A task is a component of the schedule with a start and end date.
[A] [task/object_type] [is/verb_concept] [a] [component/characteristic] [of] [the] [schedule/object_type] [with] [a] [start/object_type] [and] [end_date/object_type].

Figure 4. Semantic interpretation of English text

3.3 Generating SBVR Rules
In this phase, a SBVR representation such as SBVR rule is generated from the SBVR vocabulary in previous phase. SBVR rule is generated in three phases as following:

3.3.1 Extracting SBVR Requirements
To generate a rule from an English statement, it is primarily analyzed that it is a structural requirement or a behavioural requirement. Following mapping rules are used to classify a constraint type.

3.3.2 Extracting Structural Requirements
The use of auxiliary verbs such as 'can', 'may', etc is identified to classify co requirement as a structural requirement. The sentences representing state e.g. "Robby is a robot" or possession e.g. "robot has two arms" can be categorized as structural requirements. Moreover, the general use of action verbs e.g. consists, composed, equipped, etc also represent a structural requirement.

3.3.3 Extracting Behavioural Requirements
The use of auxiliary verbs such as 'should', 'must' are identified to classify requirement as a behavioural rule. Moreover, the use of action verb can be categorized as a behavioural rule e.g. "robot picks up parts".

3.3.4 Applying Semantic Formulation
A set of semantic formulations are applied to each fact type to construct a SBVR rule. There are five basic semantic formulations proposed in SBVR version 1.0 [12] but we are using following three with respect to the context of the scope of proposed research:

a. Logical Formulation:
A SBVR rule can be composed of multiple fact types using logical operators e.g. AND, OR, NOT, implies, etc. For logical formulation, the tokens 'not' or 'no' are mapped to negation (# a). Similarly, the tokens 'that' and 'and' are mapped to conjunction ($a \rightarrow b$). The token 'or' is mapped to disjunction ($a \rightarrow b$) and the tokens 'imply', 'suggest', 'if, 'infer' are mapped to implication ($a \Rightarrow b$).

b. Quantification:
Quantification [13] is used to specify the scope of a concept. Quantifications are applied by mapping tokes like "more than" or "greater than" to at least n quantification; token "less than" is mapped to at most n quantification and token "equal to" or a positive statement is mapped to exactly n quantification.

c. Modal Formulation:
In SBVR, the modal formulation [13] specifies seriousness of a constraint. Modal verbs such as 'can' , '' or 'may' are mapped to possibility formulation to represent a structural requirement and the modal verbs 'should', 'must' or verb concept "have to" are mapped to obligation formulation to represent a behavioural requirement.

4. A Case Study

To demonstrate the potential of our tool SR-Elicitor, a small case study is discussed from the domain of office time management system. This case study is online available. Following is a part of the problem statement for the case study, solved in the thesis to test SR-Elicitor.

"*The two main functions of the time Monitor software system are to allow the developers to use a www browser to store timestamp records in a database, and to allow a manager to analyze these timestamp records. A timestamp record consists of the time duration consists of the duration of a specific activity with the unique identification. The unique identification is made of three components: the project, the user and the date when the activity is taken place. The description of an activity is divided into three components: a task name, an activity, and an artefact. For managerial purpose it is often useful to define the date in term of the current week. The current week is defined as the week starting on the Monday immediately preceding the current day of the week, and ending on the Sunday immediately following the current day of the week, inclusively. A task is unit of work defined by the manager and for which the developer is accountable. A task is a component of the schedule with a start and end date. Examples of task are Implement module A, Design library XYZ. Developers usually work with on assigned tasks. One developer may work on many tasks and a given task may involve many developers*."

The problem statement of the second case study was given as input (NL specification) to the SR-Elicitor tool. The SR-Elicitort parses the text first, which includes lexical processing (Tokenization, Sentence splitting, POS tagging and Morphological

Category	Count	Details
Object Types	18	time_monitor_software, developer, user, task, www_browser, component, date, timestamp_record, activity, identification, project, name, artifact, week, manager, schedule, start_date, end_date
Verb Concepts	15	allow, use, store, analyse, consist, made, taken_place, define, decided, preceding, starting, ending, assign, work, involve
Individual Concepts	04	Monday ,Sunday, Implement Module A, Design library XYZ,
Characteristics	06	function, duration, description, term, day, unit, component,
Quantifications	04	Two, three, one, many
Unary Fact Types	03	activity *take place*, *define* date, current week *defined*
Associative Fact Types	07	time monitor software *allow* developer, developer *use* www browser, www browser store timestamp record, manager *analyze* timestamp record, week *start on* Monday, identification *made of* components, work *defined by* manager, developer *work with* task,
Partitive fact Types	02	timestamp record *consists of* time duration, time duration *consists of* activity,
Categorization Fact Types	03	activity *divided in* components, task *is unit of* work, *task is component of* schedule

Table 1. SBVR vocabulary generated from English text

analysis) syntactic analysis and semantic analysis. It extracts the SBVR vocabulary from the case study during syntactic analysis of text which includes noun concept, individual concept, object type, verb concept, characteristics, quantifications, associative fact types and partitive fact types etc. as shown in Table 1:

The Table I show the extracted SBVR elements such as 18 object types, 15 verb concepts, 4 characteristics, 3 unary fact types, 7 associative fact types, 2 partitive fact type and 3 categorization fact type. In the used case study's problem statement, there were 06 requirements as shown in table II:

#	SBVR Rules
1.	It is permissible that the two main *function* of the **time monitor software** *are* to *allow* the **developer** to *use* a **www browser** to *store* **timestamp record** in a **database**, and to *allow* a **manager** to *analyze* these **timestamp record**. /.
2.	It is necessity that a **timestamp record** *consists* of the **time duration** *consists* of the *duration* of a specific **activity** with the unique **identification**./.
3.	It is necessity that the unique **identification** *is made* of three **component**: the **project**, the **user** and the **date** when the **activity** *is taken place*./.
4.	It is necessity that the *description* of an **activity** *is divided* into three **component** a **task name**, an **activity**, and an **artefact**./.
5.	It is necessity that for managerial purpose, it *is* often useful to *define* the **date** in *term* of the current **week**./.
6.	It is necessity that the **current week** *is defined* as the **week** *starting* on the 'Monday' immediately *preceding* the current *day* of the **week**, and *ending* on the 'Sunday' immediately *following* the current *day* of the **week**, inclusively ./.
7.	It is necessity that a **task** *is unit* of **work** *defined* by the **manager** and for which the **developer** *is accountable*./.
8.	It is necessity that a **task** *is* a *component* of the **schedule** with a **start** and **end date**./.
9.	It is permissible that the example of **task** *are* 'Implement module A', 'Design library XYZ' ./.
10.	It is possibility that the **developer** usually *work* with one *assigned* **task** ./.
11.	It is possibility that One *may work* on many **task** and a *given* **task** *may involve* many **developer**./.

Table 2. SBVR Rule representation of software requirements

There are 11 requirements processed by the SR-Elicitor (see Table 2). According to SBVR structured English the object types are underlined e.g. **system**, **developer**, **manager**, **time** etc. the verb concepts are italicized e.g. *allow*, *analyze* etc. the SBVR keywords are bolded e.g. **at least**, **It is possibility** etc. the individual concepts are double underlined e.g. Monitor, Design library, etc. The characteristics are also italicized but with different colour: e.g. consist of, made of etc.

5. Evaluation

We have done performance evaluation to evaluate that how accurately the English specification of the software requirements has been translated into the SBVR based controlled representation by our tool ER-Elicitor. An evaluation methodology, for the performance evaluation of NLP tools, is used that was originally proposed by Hirschman and Thompson [14]. The used performance evaluation is typically based on three aspects:

There were seven sentences in the used case study problem. The largest sentence was composed of 39 words and the smallest sentence contained 10 words. The average length of all sentences is 24. The major reason to select this case study was to test our tool with the complex examples. The correct, incorrect, and missing SBVR elements are shown in Table 3.

Results of each SBVR element describe in above table separately. According to our evaluation methodology, table shows

sample elements are 68 in which 59 are correct, 7 are incorrect and 4 are missing SBVR elements.

The following table describes the Recall and precision of SR-elicitor for NL- software requirements. In Table 4, the average recall for SBVR software requirement specification is calculated 86.76% while average precision is calculated 89.39%. Considering the lengthy input English sentences including complex linguistic structures, the results of this initial performance evaluation are very encouraging and support both the approach adopted in this paper and the potential of this technology in general.

#	**Type/Metrics**	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$
1	Object Types	18	16	1	1
2	Verb Concepts	16	14	2	0
3	Individual Concepts	05	04	1	0
4	Characteristics	07	06	2	1
5	Quantifications	06	04	0	2
6	Unary Fact Types	03	03	0	0
7	Associative Fact Types	08	07	1	0
8	Partitive fact Types	02	02	0	0
9	Categorization Fact Types	03	03	0	0
	Total	68	59	7	4

Table 3. Results of NL to SBVR Translation by SR-Elicitor

Type/Metrics	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$	***Rec%***	***Prec%***	***F-Value***
Requirements	68	59	7	4	86.76	89.39	88.05

Table 4. Recall and Precision of SR-Elicitor for NL software requirements for case study

Four other case studies were solved in addition to the case study presented in section 4. All the case studies were unseen. The solved case studies were of different lengths. The largest case study was composed of 143 words and 13 sentences. The smallest case study was composed of 97 words and 8 sentences. Calculated recall, precision and f-values of the solved case studies are shown in Table 5.

Input	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$	**Rec%**	**Prec%**	**F-Value**
C1	48	37	8	3	77.08	82.22	79.65
C2	43	33	8	2	76.74	80.48	78.61
C3	39	31	5	3	79.48	86.11	82.79
C4	36	29	3	4	80.55	90.62	85.58
C5	68	59	7	4	86.76	89.39	88.05
				Average	80.12	85.76	82.94

Table 5. Evaluatin results of SR-Elicitor

The average F-value is calculated 82.94% that is encouraging for initial experiments. We cannot compare our results to any other tool as no other tool is available that can generate SBVR-based SRS from NL specification. However, we can note that other language processing technologies, such as information extraction systems, and machine translation systems, have found commercial applications with precision and recall figure well below this level. Thus, the results of this initial performance

evaluation are very encouraging and support both ER-Elicitor approach and the potential of this technology in general. Figure 5 shows the evaluation results of SR-Elicitor.

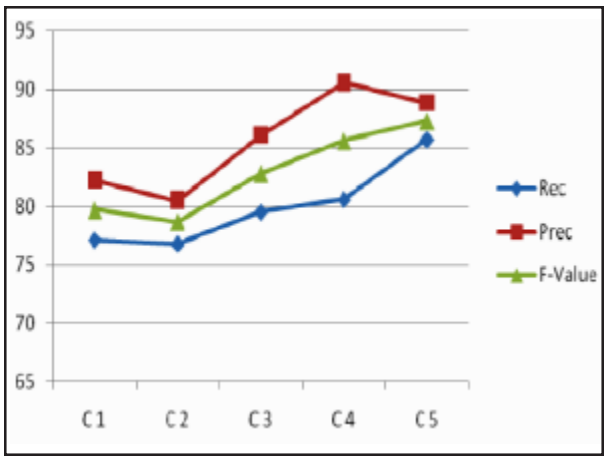

Figure 5. Semantic interpretation of English text

Figure 5 materializes the results of Recall, precision and F-Value that is obtained by five different case studies. According to our results C5 has high Recall, precision and F-Value. Moreover, C1 has lowest Recall, precision and F-Value.

6. Conclusion and Future Work

The primary objective of the paper was to address the ambiguous nature of natural languages (such as English) and generate a controlled representation of English so that the accuracy of machine processing can be improved. To address this challenge we have presented a NL based automated approach to parse English software requirement specifications and generated a controlled representation using SBVR. Automated object oriented analysis of SBVR based software requirements specifications using SR-Elicitor provides a higher accuracy as compared to other available NL-based tools.

The future work is to extract the object-oriented information from SBVR specification of software requirements such as classes, instances and their respective attributes, operations, associations, aggregations, and generalizations. Automated extraction of such information can be helpful in automated conceptual modelling of natural language software requirement specification.

References

[1] Mich Luisa , Franch Mariangela , Inverardi Pierluigi, (2004). Market research for requirements analysis using linguistic tools, Requirements Engineering, 9 (1) 40-56, February.
[2] Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D. M. (2008). Requirements for tools for ambiguity identification and measurement in natural language requirements specifications, Requirements Engineering, 13 (3) 207-239, August.
[3] Popescu, D., Rugaber, S., Medvidovic, N.et al. (2007). Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models, Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs: 14th Monterey Workshop 2007, Monterey, CA, USA, September, p.10-13.
[4] OMG, (2008). Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0. Object Management Group, Available: http://www.omg.org/spec/SBVR/1.0/
[5] Ashfa Umber, Imran Sarwar Bajwa, Asif Naeem, M. (2011). NL-Based Automated Software Requirements Elicitation and Specification. 1st International Conference on Advances in Computing and Communications (ACC-2011), Kerala, India, p.30-39
[6] Spreeuwenberg, S., Healy, K. A. (2010). SBVR's Approach to Controlled Natural Language. CNL 2009 Workshop, LNCS Volume 5972, p.155-169
[7] Ilieva, M. G., Ormandjieva, O. (2005). Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation. *In:* NLDB, p.392-397
[8] 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.

[9] Fuchs, N. E., Kaljurand, K., Kuhn, T. (2008). Attempto Controlled English for Knowledge Representation. *In:* Reasoning Web, LNCS, V. 5224, 104–124.
[10] White, Colin, Rolf Schwitter. (2009). An Update on PENG Light. *In:* Proceedings of ALTA, p. 80–88.
[11] Clark, P., Harrison, P., Murray, W. R, Thompson, J. (2009). Naturalness vs. Predictability: A Key Debate in Controlled Languages. *In:* Proceedings 2009 Workshop on Controlled Natural Languages (CNL'09).
[12] Martin, P. (2002). Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English. *In:* Proceedings of ICCS LNAI, V. 2393, p. 77–91.
[13] Schwitter, R. (2010). Controlled Natural Languages for Knowledge Representation, Coling, Poster Volume, Beijing, 1113–1121
[14] Huijsen, W. O. (1998). Controlled Language –An Introduction. *In:* Proceedings of CLAW 98:1–15.
[15] Hirschman, L., Thompson, H. S. (1995). Chapter 13 evaluation: Overview of evaluation in speech and natural language processing. *In:* Survey of the State of the Art in Human Language Technology.
[16] Umber, Ashfa., Bajwa, Imran Sarwar. (2011). Minimizing Ambiguity in Natural Language Software Requirements Specification, *IEEE Sixth International Conference on Digital Information Management* (ICDIM 2011) Melbourne, Australia
[17] Bajwa, Imran Sarwar., Mark G. Lee, Behzad Bordbar, (2011). SBVR Business Rules Generation from Natural Language Specification *In:* AAAI Spring Symposium – Artificial Intelligence 4 Business Agility, San Francisco, USA, p.541-545