NL-Based Automated Software Requirements Elicitation and Specification

Ashfa Umber[1], Imran Sarwar Bajwa[1], M. Asif Naeem[2]

[1]Department of Computer Science & IT, The Islamia University of Bahawlapur, Pakistan
[2]Department of Computer Science, University of Auckland, Auckland, New Zealand
ashfaumber@yahoo.com, imran.sarwar@iub.edu.pk, mnae006@aucklanduni.ac.nz

Abstract. This paper presents a novel approach to automate the process of software requirements elicitation and specification. The software requirements elicitation is perhaps the most important phase of software development as a small error at this stage can result in absurd software designs and implementations. The automation of the initial phase (such as requirement elicitation) phase can also contribute to a long standing challenge of automated software development. The presented approach is based on Semantic of Business Vocabulary and Rules (SBVR), an OMG's recent standard. We have also developed a prototype tool SR-Elicitor (an Eclipse plugin), which can be used by software engineers to record and automatically transform the natural language software requirements to SBVR software requirements specification. The major contribution of the presented research is to demonstrate the potential of SBVR based approach, implemented in a prototype tool, proposed to improve the process of requirements elicitation and specification.

Keywords: Requirements Elicitation, Requirement Engineering, Requirements Specification, Natural Language Processing

1 Introduction

Requirement engineering is a well-known software engineering discipline involving gathering, articulating and verifying the software requirements specifications (SRS). Requirement elicitation is the key phase of software requirement engineering as only the correct, complete, and unambiguous software requirements can result in correct, consistent and fault-tolerant software models [1]. A natural language (NL) is typically used to specify software requirements. However, the software requirements specified in English can be ambiguous and inconsistent due to inherent syntactic ambiguities and semantic inconsistencies [2]. The ambiguous SRS can not only result in conflicting and absurd software models but also complex to machine process.

In this paper, we report a novel approach to automatically translate the English SRS to SBVR (Semantic Business Vocabulary and Rules) [4] representation. The SBVR representation not only generate accurate and consistent software models but also machine process-able as SBVR has a pure mathematical foundation [4].

The presented approach works as the software engineer inputs a piece of English SRS and our approach transforms to SBVR based SRS. A multi-step procedure is adopted for NL to SBVR transition; firstly, the input English text is lexically, syntactically and semantically parsed and then SBVR vocabulary is extracted. Finally, the SBVR vocabulary is used to generate a SBVR rule representation of NL SRS.

The remaining paper is structured into the following sections: Section 2 states preliminaries of the presented research. Section 3 presents the framework of SR-Elicitor. Section 4 presents a case study and the results with performance evaluation are discussed in section 5. Finally, the paper is concluded to discuss the future work.

2. Preliminaries

2.1 Semantic Business Vocabulary and Rules (SBVR)

SBVR [4] is a recently introduced standard by OMG. Using SBVR, requirement can be captured in NL. The SBVR representation is simple to machine process as SBVR is based on formal logic. SBVR can produce SBVR business vocabulary, SBVR business rules and SBVR business facts in particular business domain.

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, there are four key elements:

- An object type is a general concept that exhibits a set of characteristics to distinguishes that object type from all other object types" [4] e.g. robot, user, etc.
- In SBVR, an individual noun is a qualified noun that corresponds to only one object [4] e.g. 'Robby', a famous robot.
- A characteristic is an abstraction of a property of an object [4] e.g. name of robot is Robby, here name is characteristic.
- A verb concept is a verb in English sentences e.g. *orders*.
- A fact type specifies relationships among noun concepts e.g. car has wheels.

SBVR Business Rules. A SBVR business rule is a formal representation 'Under business jurisdiction' [4]. Each SBVR business rule is based on at least one fact type. The SBVR rules can be a structural rule [4] used to define an organization's setup or a behavioural rule [4] used to express the conduct of a business entity.

2.2 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:

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 proposes the use of vocabulary (concepts, terms, etc.) for conceptual modeling. Furthermore, vocabulary can be employed to capture expressions in the form of formal logic structures. These features make SBVR well suited for describing software requirements to implement software models.

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. In SBVR 1.0, a collection of semantic formulations (such as atomic formulation, instantiate formulation, logical formulation, quantification, and modal formulation) are proposed to make English statements semantically controlled and restricted.

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 mix-fixing keywords in propositions. SBVR formal notations help in expressing propositions with equivalent semantics that can be captured and formally represented as logical formulations.

3. The SR-Elicitor

This section briefly explains the used approach in ER-Elicitor for transforming English text to SBVR representation. The Figure 1 highlights the used approach:

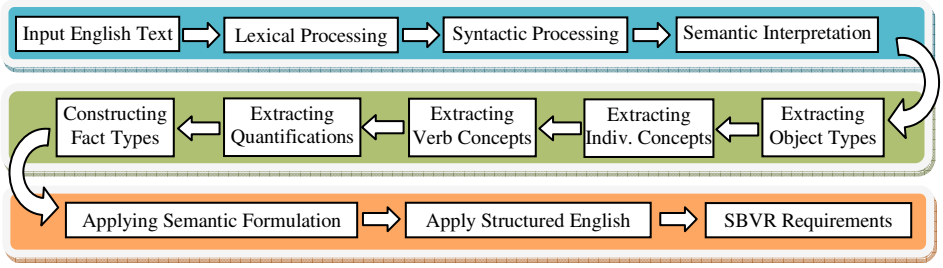

Figure 1. A Framework for automated transition of English to SBVR requirements

3.1 Parsing NL Software Requirement Text

The first phase of SR-Elicitor is NL parsing that involves a number of processing units (organized in a pipelined architecture) to process complex English statements. The NL parsing phase processes the English text as following:

Lexical Processing. The NL parsing starts with the lexical processing of a plain text file containing English SRS. The lexical phase comprises following four sub-phases:

Tokenization. The lexical processing initiates with the tokenization of the input English text e.g. "A library can issue books to students." is tokenized as [The] [belt] [conveys] [the] [parts] [towards] [the] [vision] [system] [.]

Sentence Splitting. The tokenized text is further processed to identify the margins of a sentence and each sentence is separately stored in an arraylist.

Parts-of-Speech (POS) Tagging. The tokenized text is further passed to Stanford parts-of- speech (POS) [7] tagger v3.0 to identify the basic POS tags e.g. [The/DT] [belt/NN] [conveys/VBZ] [the/DT] [parts/NNS] [towards/IN] [the/DT] [vision/NN] [system/NN]. The Stanford POS tagger v3.0 can identify 44 POS tags.

Morphological Analysis: After POS tagging, the input text is morphologically processed to separate the suffixes possibly attached to the nouns and verbs [10] e.g. a verb "applies" is analyzed as "convey+s" and a noun "parts" is analyzed as "part+s".

Syntactic and Semantic Interpretation. 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. In 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, shown in figure 3. These roles assist in identifying SBVR vocabulary and exported as an xml file.

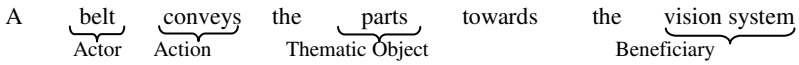

Figure 3. Semantic interpretation of English text

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:

Extracting Object Types: All common nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the object types/ general concept [4] (see figure 3) e.g. belt, user, cup, etc. In conceptual modelling, the object types are mapped to classes.

Extracting Individual Concepts: All proper nouns (actors, co-actors, thematic objects, or beneficiaries) are represented as the individual concepts [4] (see figure 3).

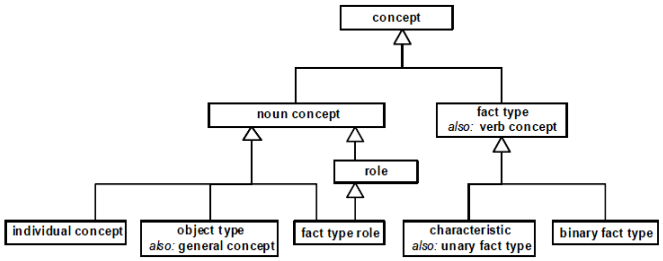

Figure 3. An extract of the SBVR metamodel: concepts ([4] figure 8.1)

Extracting Fact Types: The auxiliary and action verbs are represented as verb concepts [4] (section: 8.1.1) (see figure 3). To constructing a fact types [4] (section: 8.1.1), 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.

Extracting Characteristics: In English, the characteristic [4] (section: 11.1.2) 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.

Extracting Quantifications: The key-words such as "Each" or "All" represent SBVR universal quantifications [4] (section: 9.2.6). All indefinite articles (*a* and *an*), plural nouns (prefixed with s) and cardinal numbers (2 or two) represent SBVR non-universal quantifications [4] (section: 9.2.6).

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.

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.

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.

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:

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.

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.

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".

Applying Semantic Formulation. A set of semantic formulations are applied to each fact type to construct a SBVR rule:

Logical Formulation: A SBVR rule can be composed of multiple fact types using logical operators [4] e.g. AND, OR, NOT, implies, etc. For logical formulation, the tokens '*not'* or '*no'* are mapped to negation ($\neg$ a). Similarly, the tokens '*that*' and '*and*' are mapped to conjunction (a $\wedge$ b). The token '*or'* is mapped to disjunction (a $\vee$ b) and the tokens '*imply'*, '*suggest'*, '*if*, '*infer'* are mapped to implication (a $\Rightarrow$ b).

Quantification: Quantification [4] 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.

Modal Formulation: Modal formulation [4] specifies seriousness of a constraint. Modal verbs e.g. '*can'* and '*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.

Applying Structured English Notation. The last step in generation of a SBVR is application of the Structured English notation: The object types are underlined e.g. student; the verb concepts are italicized e.g. *should be*; the SBVR keywords are bolded e.g. at most; the individual concepts are double underlined e.g. Patron. The characteristics are also italicized but with different colour: e.g. *name*.

4. A Case Study

To demonstrate the potential of our tool SR-Elicitor, a small case study is discussed from the domain of online ordering systems Cafeteria Ordering System (COS): that was online available at: [16]. Following is the problem statement of the case study:

"The system shall let a Patron, who is logged into the Cafeteria Ordering System, place an order for one or more meals. The system shall confirm that the Patron is registered for payroll deduction to place an order. If the Patron is not registered for payroll deduction, the system shall give the Patron options to register now and continue placing an order, to place an order for pickup in the cafeteria, or to exit from the COS. The system shall prompt the Patron for the meal date. If the meal date is the current date and the current time is after the order cutoff time, the system shall inform the patron that it's too late to place an order for today. The Patron may either change the meal date or cancel the order. The Patron shall specify whether the order is to be picked up or delivered. If the order is to be delivered and there are still available delivery times for the meal date, the Patron shall provide a valid delivery location."

The problem statement of the case study was given as input (NL specification) to the SR-Elicitor tool. The tool parses and semantically interprets English text and extracts the SBVR vocabulary from the case study as shown in table I:

Table 1: SBVR vocabulary generated from English text

Category	*Count*	*Details*
Object Types	05	system, order, payroll, date, time
Verb Concepts	14	let, log, place, confirm, register; pick_up, exit, inform, change, cancel, specify, pick, deliver, provide
Individual Concepts	03	Cafeteria_Ordering_System, Patron, COS
Characteristics	04	meal_date, cutoff_time, delivery_time, delivery_location
Quantifications	08	Universal (01), At least *n* (07)
Unary Fact Types	05	Patron *registers*, , Patron change, order *picked*, order *delivered*, Patron provide
Associative Fact Types	08	Patron *logged* into Cafeteria_Ordering_System, Patron *place* order, system *confirm* Patron, Patron *registered* for payroll, Patron *pickup* order, system *prompt* Patron, System *inform* Patron, Patron *exit* from COS, Patron *cancel* order, Patron specify order,
Partitive fact Types	00	
Categorization Fact Types	00	

Here, Cafeteria_Ordering_System and COS are synonyms of each other but not picked but our system and these are specified as separate individual concepts. One object type has not been picked that cafeteria. Moreover, current date and current time are characteristics but they are picked as object types. In the used case study's problem statement, there were seven requirements as shown in table II:

Table II: SBVR Rule representation of software requirements:

Details
It is obligatory that the system *shall let*, each Patron who *is logged* into the Cafeteria Ordering System, *place* at least one order for at least one or more meals.
It is obligatory that the system *shall confirm* that the 'Patron' *is registered* for payroll deduction to *place* at least one order.
If the Patron *is* not *registered* for payroll deduction, It is obligatory that the system *shall give* the Patron options to *register* and *continue placing* at least one order, to *place* at least one order for *pickup* in the cafeteria, or to *exit* from the COS.
It is obligatory the system *shall prompt* the Patron for the *meal date*.
If the *meal date is* the current date and the current time *is* after the order *cutoff* time, it is obligatory that the system *shall inform* the Patron that it's too late to *place* at least one order for today.
It is possibility that the Patron *may change* the *meal date* or *cancel* the order.
It is obligatory the Patron *shall specify* whether the order *is* to *be picked* or *delivered*.
If the order *is* to *be delivered* and there still *are* available *delivery times* for the *meal date*, it is obligatory that the Patron *shall provide* at least one valid *delivery location*.

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 SR-Elicitor.

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 II.

Table III: Results of NL to SBVR Translation by SR-Elicitor

#	*Type/Metrics*	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$
1	Object Types	05	3	2	1
2	Verb Concepts	14	14	0	0
3	Individual Concepts	02	2	1	0
4	Characteristics	06	4	0	2
5	Quantifications	08	8	0	0
6	Unary Fact Types	05	5	0	0
7	Associative Fact Types	08	8	0	0
8	Partitive fact Types	00	0	0	0
9	Categorization Fact Types	00	0	0	0
	Total	48	44	3	3

In table III, the average recall for SBVR software requirement specification is calculated 91.66% while average precision is calculated 93.61%. 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.

Table III: Recall and Precision of SR-Elicitor for NL software requirements

Type/Metrics	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$	*Rec%*	*Prec%*
Software Requirements	48	44	3	3	91.66	93.61

6. Related Work

A few controlled natural language (CNL) representations are introduced in last two decades such as Attempto Controlled English (ACE) [8], Processable English (PENG) [9], computer Processable Language (CPL) [10], Formalized-English (Martin, 2002) [11], etc. All above languages are human-oriented CNLs [12], while a machine-oriented CNL [13] can be more helpful in modern software modelling practices. Furthermore, the available CNLs are general purpose and not specifically designed for natural language based software requirement specifications.

An automated approach was presented in [16] to generate SBVR representation from English language description. However, English is difficult to machine process and translate to formal languages [15], [17]. SBVR based controlled natural language is not a brand new proposal as it has been previously presented and implemented in a tool RuleXpress [5] but it is specifically designed for business people to express and communicate business rules. The related work shows that currently there is no approach and tool available that can automatically translate natural language software requirements to a CNL representation such as SBVR.

7. Conclusion and Future Work

The primary objective of the paper was to automate the process of software requirement elicitation and specification by overcoming ambiguous nature of natural languages (such as English) and generating a controlled representation. To address this challenge we have present a NL based too SR-Elicitor that is based on an automated approach to parse English software requirement specifications and generated a controlled representation using SBVR. The output of out tool can be used for automated object oriented analysis and design from natural language software requirements. Additionally, our 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.

References

[1] Denger, C., Berry, D.M. Kamsties, E. 2003. Higher Quality Requirements Specifications through Natural Language Patterns. In Proceedings of IEEE International Conference on Software-Science, Technology \& Engineering (SWSTE '03):80-85

[2] Ormandjieva O., Hussain, I., Kosseim, L. 2007. Toward A Text Classification System for the Quality Assessment of Software Requirements written in Natural Language. in 4th International Workshop on Software Quality Assurance (SOQUA '07):39-45.

[3] Kuhn, Tobias. 2010. Controlled English for Knowledge Representation. Doctoral Thesis. Faculty of Economics, Business Administration and Information Technology of the University of Zurich.

[4] OMG. Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0. Object Management Group, Available: http://www.omg.org/spec/SBVR/1.0/, 2008

To insert individual citation into a bibliography in a word-processor, select your preferred citation style below and drag-and-drop it into the document.

[5] Spreeuwenberg, S., and Healy, K.A. 2010. SBVR's Approach to Controlled Natural Language. CNL 2009 Workshop, LNCS Volume 5972/2010:155-169

[6] Ilieva, M.G., Ormandjieva, O. 2005. Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation. In NLDB 2005: 392-397.

[7] 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.

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

[9] White, C. and Rolf S. 2009. An Update on PENG Light. In: Proceedings of ALTA 2009, pp. 80–88.

[10] Clark, P., Harrison, P., Murray, W.R, Thompson, J. 2009. Naturalness vs. Predictability: A Key Debate in Controlled Languages. In: 2009 Workshop on Controlled Natural Languages (CNL'09).

[11] Martin, P. 2002. Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English. In: Proceedings of ICCS 2002, LNAI, vol.2393, pp. 77–91.

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

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

[14] 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.

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

[16] Bajwa, Imran Sarwar, Lee, Mark G., Bordbar, Behzad. 2011. SBVR Business Rules Generation from Natural Language Specification. AAAI Spring Symposium 2011, San Francisco, USA. pp.2-8

[17] Bajwa, I. S., Choudhary, M.A. (2006) "A Rule Based System for Speech Language Context Understanding" *Journal of Donghua University, (English Edition)* 23 (6), pp. 39-42.