A Grammar for ADVANCED SBVR Editor

Algirdas Šukys, Linas Ablonskis, Lina Nemuraitė, Bronius Paradauskas

Kaunas University of Technology, Department of Information Systems
Studentų St. 50-308, LT-51368 Kaunas, Lithuania
e-mail: algirdas.sukys@ktu.lt, linas.ablonskis@ktu.lt, lina.nemuraite@ktu.lt, bronius.paradauskas@ktu.lt

crossref http://dx.doi.org/10.5755/j01.itc.45.1.9219

Abstract. Semantics of Business Vocabulary and Business Rules (SBVR) is the richest knowledge model allowing to create specifications that are understandable for business people and also interpretable by computers. Existing SBVR editors still lack capabilities that could allow generating formal SBVR models, adapting SBVR to several languages or making SBVR extensions for various purposes (e.g., implementing transformations to software modelling languages) without changing the original SBVR metamodel. The goal of the paper is to present a grammar for SBVR structured language and a prototype of SBVR editor, created on the base of this grammar. An experiment conducted with the prototype has shown that it allows defining business vocabularies, business rules and questions in SBVR structured English and Lithuanian languages; producing formal SBVR models; using concepts from several vocabularies, and extending SBVR without changing its metamodel.

Keywords: SBVR; business vocabulary; business rules; SBVR questions; structured language; Xtext grammar; SBVR editor.

1. Introduction

This research was inspired by the desire to provide opportunities to carry out a semantic search using a language closer to humans than ordinary query languages (e.g., SQL or ontology query language SPARQL), and by the acquaintance with the Semantics of Business Vocabulary and Business Rules (SBVR) [1], [2]. SBVR has attracted attention from a lot of researchers and practitioners as it can bridge the gap between natural and formal languages, thus giving possibility of conceptualizing business and its supporting information systems. SBVR origins are based on Guide project [3] of Business Rules Group (BRG). According to BRG, business rule is “a rule that can be interpreted by computer and defines or restricts some aspect of business”. BRG categorizes business rules to structural assertions, action assertions, and derivations. Structural assertions restrict static aspects of business domain; action assertions are constraints or conditions that limit or control the actions; derivations allow to derive new facts from the existing ones. All these kinds of business rules can be specified on the base of SBVR metamodel following the Business Rules mantra [1], which expresses the way how business rules are constructed: “Rules are based on facts, and facts are based on terms”. Facts, terms and verb concept wordings, representing business concepts, are specified in SBVR business vocabulary, which allows to define business rules. According to SBVR, business rule is a rule that is under business jurisdiction [1], [2]; SBVR structural rules correspond to structural assertions and derivations, operative rules – to action assertions of BRG.

SBVR is the OMG metamodel for specifying business vocabulary and business rules using a kind of controlled natural language. SBVR offers a methodology to define meaning of business vocabularies and business rules using concepts and semantic formulations; it is attractive for its potential to raise information system models and their development processes into the level understandable both for business participants and computers. Currently, SBVR is the richest knowledge model based on the principle of separating the meaning of business concepts, propositions and questions from their representations. This allows to use a variety of representations for the same meaning, and to share meanings of things in different languages [4]. However, such ambitious goals yet require a lot of investigative effort.

SBVR bridges the gap between business and information technologies by providing business oriented language to specify business knowledge transformable to software models and other artefacts. SBVR specifications are generally used for creating various
software artefacts: Web Services [5], [6], [7], [8], UML&OCL models [9], [10]; event based environments [11], RDB Schemas [6] and SQL queries [12]; OWL 2 ontologies [13], [14], [15], [16], [17]; and, vice versa, there are experiments to transform other languages to SBVR [18], [19]. Besides that, the ideas were announced to use SBVR for querying and analysing software models; however, these ideas were not implemented yet [20]. Querying have not received much attention in the SBVR related research.

In our previous works, we have shown a possibility to formulate questions using SBVR structured language and to transform them to queries in ontology query language SPARQL [21], and presented a framework for implementing such solution [22]. First of all, business vocabulary and business rules of a particular domain are specified and transformed to OWL 2 ontology [16]. Business participants can formulate questions in SBVR structured language using terminology of business vocabulary. These questions are transformed into SPARQL queries [23] and executed in OWL 2 ontology, whose scheme is obtained via transforming SBVR business vocabulary and business rules.

The complete implementation of this framework requires creating the overall infrastructure of semantic querying. The essential tool required for such framework is a robust SBVR structured language editor, capable of parsing structured language specifications and producing SBVR business vocabularies and business rules; to transform these vocabularies and rules to OWL 2 ontologies, and to transform SBVR questions to SPARQL queries. There are several SBVR editors implemented; however, none of them, to our knowledge, are capable to meet these requirements. Moreover, the analysed SBVR editors are designed for Structured English and cannot be adapted to Lithuanian or other languages.

Adjusting SBVR to specific natural languages is very important when semantic technologies are becoming the only means for overcoming the information overload and complexity of computerized systems. Conceptualization of Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Business Process Management Systems, Governmental Resources, etc. in human-friendly language were announced in SBVR vision, e.g., [24]. Business participants usually prefer analysing business activities in their native languages. Whereas English or other popular languages are the most often used in information technologies, small languages are at risk of disappearing from the cyberspace, and this presents a risk of preventing the better understanding of alignment between business and its supporting information systems.

SBVR metamodel supports multilingual vocabularies by providing concepts of language and speech community. However, for adjusting SBVR editor to human written natural (although structured) language, linguistic processing is needed, especially for inflectional languages like Lithuanian. Integration with linguistic processing allows using SBVR Structured Lithuanian Language for semantic analysis and search in ontologies and unstructured information resources, e.g., Lithuanian Internet corpora, provided with annotations. Such capabilities were tested in SemantikaLT project [25] but they still require a lot of additional effort for reaching the full-fledged functionality for asking questions in Structured Lithuanian Language.

In this paper, we present a context-free grammar, required for our editor, which is specified in Xtext grammar [26], [27]. From the start, the grammar was designed for SBVR Structured English language, but we will show how it can be adapted to Lithuanian and, hopefully, to other similar languages. However, such an adaptation cannot be achieved without linguistic processing, which still is not perfect and requires further improvements.

The rest of this paper is structured as follows. Section 2 presents an overview of existing SBVR editing solutions. Section 3 presents the SBVR structured language grammar. Section 4 is devoted to the experimental evaluation of quality of SBVR structured language editor. Section 5 draws conclusions and describes directions for future work.

2. Overview of SBVR editing tools

In this section, we present an overview of existing SBVR editing tools concentrating on their capabilities to specify business vocabularies, business rules, and questions, and to produce formal SBVR models.

The first SBVR tool was an open source SBVR editor SBeaVeR, created in 2006 [28] as Eclipse plugin. It allows to specify and validate business vocabularies and business rules using SBVR Structured English, to transform them into formal SBVR specifications, and to serialize them into XMI Schema to support interchange of specifications between software tools. SBeaVeR editor has syntax highlighting following SBVR Structured English style, hierarchical vocabulary navigation panel, and embedded WordNet dictionary supporting of synonyms, hypernyms, hyponyms, meronyms, and informal definitions. Specifications are parsed and validated using the LL parser – a top-down parser for a context-free languages constructing a leftmost derivation. Unfortunately, the tool is no longer supported.

Another open source tool VeTIS [10] was created on a base of SBeaVeR. It provides all functionality of SBeaVeR plus better capabilities of specifying business vocabulary (e.g., individual concepts) and recognizes a wider variety of business rules. In VeTIS, business vocabularies and business rules are serialized into SBVR XMI models for further transforming them into UML models with OCL expressions. However, both of the mentioned tools lack flexibility for further extensions, necessary to realize the full potential of the SBVR knowledge model.
Marinos et al. presented a SBVR editor with syntax highlighting and auto-completion functions [29]. Unlike SBeaVeR and VeTIS, it allows to write terms or verbs consisting of multiple words without joining them with a dash as it is recommended by the SBVR style [2] and implemented in other tools, and automatically recognizes terms in plural form. The grammar was written in OMeta, an object oriented language for matching patterns, based on expression grammars [30]. The editor was implemented to work in a Web browser using CodeMirror library. It is able to specify terms, fact types with attributes, complex business rules (based on several fact types), and quantifiers.

RuleXpress [31] is a commercial framework for developing and managing business vocabularies and business rules. RuleXpress allows to create conceptual models that comply with the SBVR metamodel, using graphical interface. It does not use any controlled language for writing business rules. Instead, it allows using a natural language and recognizes only those concepts that are specified in a vocabulary. RuleXpress is not an execution tool, but rather a tool to help people to organize and understand their business.

The SBVR lab 2.0 is a commercial SBVR editor working in a Web browser [32]. This tool allows to specify business vocabulary and business rules in the Structured English language. Each vocabulary term and business rule should be indicated by specifying their type and features. The SBVR lab 2.0 allows using binary verb concepts and characteristics. Like in Marinos et al. editor [29], verbs of verb concepts can consist of several words without joining them. The editor has a unique function of graphical visualization of business vocabulary and publishing it to the Web.

Table 1 presents the comparison of analysed SBVR tools with regards to their essential features. The SBeaVeR and VeTIS use context-free grammars to parse SBVR specifications. Marinos et al. editor uses expression grammar, which is similar to context free grammar, but with ambiguity eliminated by prioritizing alternatives [33].

Table 1. Comparison of SBVR editors

<table>
<thead>
<tr>
<th>Language used in business vocabulary and business rules</th>
<th>SBeaVeR</th>
<th>VeTIS</th>
<th>Marinos et al. SBVR Editor</th>
<th>RuleXpress</th>
<th>SBVR Lab 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic highlighting and auto completion</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>Language parser’s grammar</td>
<td>Context-free grammar</td>
<td>Context-free grammar</td>
<td>Extended expression grammar</td>
<td>Uncontrolled natural language</td>
<td>Unknown</td>
</tr>
<tr>
<td>Formalization in compliance with SBVR metamodel</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>Model interchange format</td>
<td>XMI XSD</td>
<td>XMI model</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Possibility to write questions</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Language Grammar and to create a desirable SBVR Editor using Xtext framework [26], [35], [36], which allows to generate not only the language parser, but also a full set of language-processing tools (in our case, the entire user interface with auto completion, text highlighting and colouring features, and capabilities to recognize text, compliant with the defined grammar). Xtext [27] uses the context-free grammar but allows to analyse context following the two-phase process: parsing a text in the first phase, and using a linking service in the second phase for analysing links among newly introduced verb concepts and previously introduced noun concepts thus assuring compatibility of vocabulary entries. Xtext framework is integrated with Eclipse Modelling Framework (EMF) and Eclipse User Interface [37]; therefore, it allows to transform SBVR Structured Language text to EMF abstract syntax tree (AST), which can be transformed to SBVR formal model and used in other transformations.

3. SBVR Structured Language grammar

3.1. SBVR meanings and representations

The principle of separating meaning and representation is illustrated by the excerpts of SBVR metamodel in Figures 1 and 2. Meaning is defined in [2] as “what is meant by a word, sign, statement, or description; what someone intends to express or what someone understands”. Meaning can have many representations. Each representation relates meaning with one expression, which can be a sequence of characters or speech sounds, a picture, etc. Different kinds of representations are used for different kinds of meaning. E.g., designations represent general and individual concepts, verb concept wordings represent verb concepts, statements express propositions, etc. Representations have different syntax to express various kinds of meaning. All designations and verb concept wordings are included into the namespace of the SBVR vocabulary. The SBVR Structured Language grammar defines syntactic structure of SBVR representations – designations, verb concept wordings, statements, etc.

For recognizing representations of different kinds of meaning, Structured English notation [2] uses the following standard font styles: “term” style is used for designations of general concepts (it starts with a lowercase letter); “Name” style is used for designations of individual concepts (it starts with an uppercase letter or number); “verb” is used for designations of verb concepts; “keyword” style is used for additional words used to formulate statements (‘is_greater_than’, ‘at_least’, ‘What’, ‘It is obligatory that’, etc.).

The SBVR Structured Language styles can be applied to various natural languages; for each concrete natural language, keywords will be different. The grammar is primarily based on recognition of terms having the meaning of general concepts, roles, and verb concept roles; then it recognizes verb concept wordings having meaning of verb concepts. The grammar is limited to unary verb concepts (characteristics) and binary verb concepts; n-ary verb concepts can be specified in the grammar and recognized by the SBVR editor, but n-ary relations are not further supported by CASE tools, ontology editors, database management systems, etc. Therefore, we strive immediately expressing semantics of n-ary relations via binary verb concepts thus leaving fewer opportunities to lose that semantics in the following transformations from SBVR to software artefacts.

Facts, according to BRG and SBVR, are propositions that are taken as true (e.g., Vilnius is capital of Lithuania). The most specific logical formulation, that formalizes the SBVR Structured language statement, is an atomic formulation, based on a verb concept (e.g., “city is capital of country”). Atomic formulations are primary building blocks of the formal SBVR model. Meaning of propositions including facts and business rules is formulated by

![Figure 1. SBVR representations of concepts (excerpt from [2])](image-url)
closed logical formulations (modal formulations, atomic formulations, instantiations, quantifications, logical operations, etc.). Meaning of questions and definitions is formulated by closed projections. On the base of these constructs of SBVR metamodel, we have categorized SBVR Structured Language (SL) grammar rules into four categories that are presented in Figure 3.

On the top there is a grammar for specifying business vocabulary entries, i.e., terms, names and verb concept wordings, which express relations among terms. Terms (meaning general concepts, roles, verb concept roles, and individual concepts), names (meaning individual concepts), and verb concept wordings (meaning verb concepts), are the basis for defining facts and statements, formalized by atomic formulations.

The grammar rules for statements of atomic formulations are used both for business rules and questions, which are expressed by more complex statements, formalized by more complex logical formulations and projections. In the further sections, we present SBVR Structured Language grammar rules in detail.

3.2. Structured grammar rules for business vocabulary

Before introducing production rules for vocabulary entries, we present terminal rules (also called lexer rules) of the grammar. These rules represent terminals, i.e., fundamental building blocks of a language. Xtext grammar allows using terminal fragments as reusable parts. In our grammar, terminal rules represent syntax of terms, verb concept wordings, names, and various keywords. Main terminal rules of the grammar define Latin and Lithuanian letters, digits, syntax of SBVR terms, verb concept wordings, and names.

Terminal rules define the syntactic structure of primary representations of vocabulary entries. However, these rules are not directly used. Xtext has a feature to define references between objects in a grammar. The type of each referenced object is defined by a production rule. Therefore, we use additional production rules (Term, Name, and VerbConceptWording) for defining primary representations to be able to establish crosslinks between them. Terms and verb concept wordings share the same terminal rule defining their syntax. “Name” syntax is defined by other terminal, starting with uppercase letter or digit.

Besides primary representations, vocabulary entries can have optional specifications, containing a set of captions. Captions of vocabulary entries are used to provide textual description; synonyms; synonymous forms; more general concepts for specifying concept hierarchy, which can be defined for general concepts, verb concepts, roles, etc. Some
captions are applicable only to particular entry types. E.g., a caption “synonymous form” can be used only for a verb concept wording. Therefore, we have defined different sets of captions for each concept type in our grammar:

```plaintext
GeneralConcept:    
  primaryRepresentation=Term
(captions+=CaptionForGeneralConcept)*;  
IndividualConcept:   
  primaryRepresentation=Name
(captions+=CaptionForIndividualConcept)*;  
VerbConcept:        
  primaryRepresentation=VerbConceptWording
(captions+=CaptionForVerbConcept)*;  
```

The verb concept wording is a general template to represent verb concepts; it can be expressed in sentential form or in noun form. Sentential form is used to represent verb concepts, having a designation of verb concept (verb symbol) and one or two placeholders. Placeholders are represented by terms already specified in the grammar. Noun form template represents verb concepts consisting of two placeholders.

Grammar rules of verb concept wordings use cross references to objects, whose types are specified in square brackets (for the sake of simplicity, we do not specify syntax of referenced objects as required in Xtext grammar). Here, cross references link placeholders of verb concept wordings with primary representations of general concepts. These references help to check whether verb concepts are specified using terms existing in a vocabulary.

```plaintext
VerbConceptWording:    
  SententialForm|NounForm;  
  SententialForm:  
    BinaryVerbConcept|Characteristic;  
    BinaryVerbConcept:  
      placeholder1=[Term]  
      verbSymbol=[VerbSymbol]  
      placeholder2=[Term];  
    Characteristic:  
      placeholder1=[Term]  
      verbSymbol=[VerbSymbol];  
  NounForm:  
    placeholder1=[Term]  
    placeholder2=[Term];  
```

Vocabulary entries are specified by pairs of captions and values. The grammar allows using the same syntax for different captions. For example, the synonym of general concept can be defined using syntax of term whereas the description is written using free text:

```plaintext
CaptionForGeneralConcept:  
  caption={(DESCRIPTION|SYNONYM) ... );  
  terminal DESCRIPTION: "Description: " STRING;  
  terminal SYNONYM: "Synonym: " Term;  
```

Our grammar allows using in a vocabulary metaconcepts, specified in a metavocabulary. Metaconcepts can be elements from the SBVR metamodel or SBVR metamodel extensions. E.g., the SBVR verb concept “concept incorporates characteristic” allows to specify that a particular concept incorporates a necessary characteristic that makes up a concept. Concepts used as values of metaconcepts are written between quotes. Such a construction is realized using the following grammar rule:

```plaintext
MetaVerbConceptConstruction:  
  placeholder1=[Term]  
  ""(concept1=VerbConceptWording  
  concept1=Term)""  
  verbSymbol=[VerbSymbol]  
  placeholder2=[Term]  
  ""(concept2=VerbConceptWording  
  concept2=Term)"";  
```

### 3.3. Grammar rules for statements of atomic formulations

Atomic formulation is a simplest logical formulation, based on a particular verb concept. Atomic formulations are used to compose statements for business rules and questions. The atomic formulations are composed by binding roles of verb concepts to bindable targets – individual concepts, variables or variables restricted by other logical formulations. They are represented by statements based on verb concept wordings, including placeholders. A placeholder is a place for an expression that can be replaced by designations, representing individual concepts, quantifiers, or quantity comparisons. For example, verb concept wording 'event is organized by organizer' has two placeholders: event and organizer. Placeholders can be replaced by designations of individual concepts. E.g., the placeholder organizer can be replaced by `Events_Ltd` or organizer `Events_Ltd`. If the placeholder is not replaced it is interpreted as a variable of a certain type (e.g., event, organizer, etc.).

Statements can be simple, based on one verb concept wording (e.g., 'event is organized by organizer Events_Ltd') or complex, based on several verb concept wordings connected via conjunction or disjunction. E.g., the complex statement 'event is organized by organizer Events_Ltd' and 'event has number of sold tickets greater than 100' is based on two verb concept wordings: 'event is organized by organizer' and 'event has number of sold tickets' that are connected using conjunction. However, this statement can be rewritten in a more elegant way: 'event is organized by organizer Events_Ltd and has number of sold tickets greater than 100'. It is also based on the same verb concept wordings, but the first placeholder of the second verb concept wording is omitted (the first placeholder of the first verb concept wording (i.e., event in the presented example) is used instead it.)
Verb concept wordings also can be combined by restricting verb concept roles. E.g., the statement ‘organizer organizes event that takes place in sports arena Snow’ is based on the verb concept wording ‘organizer organizes event’. Role event is further restricted by the statement ‘event takes place in sports arena Snow’.

In summary, our grammar can represent the following types of statements for atomic formulations:

- BothPlaceholdersReplacedByName:
  - 1stPlaceholderReplacedByName:
  - 2ndPlaceholderReplacedByName:
  - BothPlaceholdersReplacedByQuantityRestriction:

- BothPlaceholdersNotReplaced:
  - 1stPlaceholderReplacedByTerm:
  - 2ndPlaceholderReplacedByTerm:
  - BothPlaceholdersNotReplacedByTerm:

**Statements with both placeholders not replaced.** The statements of this type correspond to verb concept wordings. The atomic formulation such a statement represents is based on verb concept having both roles bound to variables. The meaning is true when all referent things of the first variable have corresponding referents of the second variable. E.g., the proposition ‘organizer which organizes event’ represents meaning, which is true if each organizer organizes at least one event. The grammar rule of such statements is as follows:

- BothPlaceholdersNo3Replaced:
  - placeholder1= [Term]? (PRONOUN)? verbSymbol= [VerbSymbol] placeholder2= [Term]
  - ((conj=CONJUNCTION|disj=DISJUNCTION)? restriction= StatementOfAtomicFormulation)?;
  - event takes place in event venue

This grammar rule allows writing complex statements using recursive restrictions. The recursive restriction is expressed by an optional call of a StatementOfAtomicFormulation connecting it by conjunction or disjunction, and allows to represent projecting formulations, defined by the corresponding PRONOUN (e.g. “which”, “that”, etc.). Recursive restrictions can be used for all types of statements.

**Statements with placeholders replaced by names.** Designations, representing individual concepts (names) can replace first, second, or both placeholders of verb concept wordings. Such statements represent atomic formulations that are based on verb concepts and have one of roles bound to an individual concept; other, unrestricted role, is bound to a variable. The meaning represented by such structure is true when each referent of the variable is an individual concept that is bound to the role of the verb concept. E.g., the statement ‘organizer Events Ltd organizes event’ formulates meaning, which is true for each event that organizer Events Ltd organizes. The grammar rules for such statements are as follows:

- 1stPlaceholderReplacedByName:
  - placeholder1= [Term]? role1Replacement= Name (PRONOUN)? verbSymbol= [VerbSymbol] placeholder2= [Term]
  - ((conj=CONJUNCTION|disj=DISJUNCTION)? restriction= StatementOfAtomicFormulation)?;

- Examples:
  - organizer Events Ltd organizes event
  - Events Ltd organizes event

- 2ndPlaceholderReplacedByName:
  - placeholder1= [Term]? role1Replacement= Name (PRONOUN)? verbSymbol= [VerbSymbol] placeholder2= [Term]? role2Replacement= Name ((conj=CONJUNCTION|disj=DISJUNCTION)? restriction= StatementOfAtomicFormulation)?;

- Examples:
  - event that takes place in sports arena Snow
  - event takes place in Snow

The following grammar rule is designed for restricting statements with both placeholders replaced by names. In such cases, the statements express facts – propositions that have meaning true.

- BothPlaceholdersReplacedByName:
  - placeholder1= [Term]? role1Replacement= Name (PRONOUN)? verbSymbol= [VerbSymbol] placeholder2= [Term]? role2Replacement= Name ((conj=CONJUNCTION|disj=DISJUNCTION)? restriction= StatementOfAtomicFormulation)?;

- Examples:
  - organizer Events Ltd organizes event Tern
  - Events Ltd organizes Tern

**Statements with the second placeholder replaced by quantity restriction.** In statements of this type, the second placeholder of a verb concept wording is replaced by some quantity restriction expression. Such a statement represents meaning of atomic formulation that is based on a verb concept, which has one role bound to a variable and the second role restricted by logical formulation of quantity restriction. Its meaning is true for each referent of the variable, which satisfies the restriction. E.g., the statement ‘event that has number of sold tickets greater than 100 formulates meaning, which is true for each event, which has number of sold tickets greater than 100. There are predefined keywords to express various quantity restrictions: ‘is less than’, ‘is greater than’, ‘is less or equal to’, ‘is greater or equal to’, ‘equals’.

- Examples:
  - organizer Events Ltd organizes event
  - Events Ltd organizes event
The grammar rule for statements with quantity restrictions is as follows:

```
2ndPlaceholderReplacedByQuantityRestriction:
  placeholder1=[Term]?(PRONOUN)?$,
  verbSymbol=[VerbSymbol],
  role2Replacement=QuantityRestriction,
  placeholder2=[Term];
```

Examples:
- `event that has number_of_sold_tickets that is_greater_than 100`;
- `event has number_of_sold_tickets that is_less_than 100`;
- `number_of_sold_tickets of event equals 100`;

Statements with placeholders replaced by quantification representations. Statements with placeholders, replaced by quantification representations, express logical formulations of cardinality quantifications. The quantification can restrict one of roles of verb concept wording. E.g., the meaning of statement 'organizer organizes at_least 2 events' is true for each organizer that organizes at least two events. Statements of this type are used in a structural rules and questions.

There are five types of quantifications for statements with cardinality restrictions: universal, at least n, at most n, exactly n, and numeric range. Grammar rules for quantifications are as follows:

```
Quantification:
  UNIVERSAL_QUANTIFICATION|
  AT_LEAST_N_QUANTIFICATION|
  AT_MOST_N_QUANTIFICATION|
  EXACTLY_N_QUANTIFICATION|
  NUMERIC_RANGE_QUANTIFICATION;
terminal UNIVERSAL_QUANTIFICATION:
  "a"|"an"|"each";?
terminal AT_LEAST_N_QUANTIFICATION:
  "at_least" valueN=DIGIT;
terminal AT_MOST_N_QUANTIFICATION:
  "at_most" valueN=DIGIT;
terminal EXACTLY_N_QUANTIFICATION:
  "exactly" valueN=DIGIT;
terminal NUMERIC_RANGE_QUANTIFICATION :
  "at_least" valueN=DIGIT "and_at_most" valueM=DIGIT;
```

The grammar rules for modal operators are as follows:

```
ModalOperator:
  NECESSITY|POSSIBILITY|IMPOSSIBILITY|
  OBLIGATION|PERMISSION|PROHIBITION;
terminal NECESSITY:
  "It is necessary that";
terminal POSSIBILITY:
  "It is possible that";
terminal IMPOSSIBILITY:
  "It is impossible that";
terminal OBLIGATION:
  "It is obligatory that";
terminal PERMISSION:
  "It is permitted that";
terminal PROHIBITION:
  "It is prohibited that";
```

3.4. Grammar rules for business rules

Grammar rules for business rules use statements of atomic formulations that were described in the previous section. The grammar is suitable for writing structural and operative business rules and definitions. Structural business rules are written using alethic modal operators 'It is necessary that', 'It is possible that', 'It is impossible that'. Operative business rules are defined by deontic modal operators 'It is obligatory that', 'It is permitted that', 'It is prohibited that'. Derivations are statements for deriving new facts from existing ones; they can be specified as SBVR structural business rules and definitions.

We have defined two templates for representing business rules, i.e., rules based on statements of atomic formulations, and rules based on implications. Each template contains two parts: modal operator and restricting statement. A fragment of the grammar for business rules (examples are presented in Section 4) is as follows:

```
Rule:
  RuleBasedOnStatementOfAtomicFormulation
  |ImplicationRule;
RuleBasedOnStatementOfAtomicFormulation:
  modality=ModalOperator
  statement=StatementOfAtomicFormulation;
ImplicationRule:
  modality=ModalOperator
  consequent=StatementOfAtomicFormulation
  "if"
  antecedent=StatementOfAtomicFormulation;
```

Grammar rule for statements with quantifications is as follows:
3.5. SBVR structured language grammar rules for questions

Grammar rules for questions also use statements of atomic formulations. The question starts with a keyword, indicating expected answer type. Some keywords are used for listing information units (i.e., what, find), others return yes/no answer (i.e., does). Detailed taxonomy of question types is presented in [38].

One of the problems of interpreting questions written in SBVR structured language is to identify concepts that should be given in the answer. It can be done automatically in questions that are based on verb concepts with single unrestricted role. For example, if one writes question 'Find organizer that organizes event Tern', she or he obviously wants to retrieve a list of organizers.

When question contains more unrestricted roles it is not easy to automatically determine concepts that should be given in the answer. E.g., the answer to the question 'Find person that organizes event' can consist either of persons or of persons and events. The grammar allows to enumerate concepts that are required in the answer: 'Find person that organizes event. Show person'. The grammar rule for questions (examples are presented in Section 4) is as follows:

```java
Question:  
  startKeyword=(WH|HOW_MANY|FIND)  
  statement=StatementOfAtomicFormulation  
  ("?"|".") ("Show " (showTerm+=[Term]),  
  "?")+ ";");
```

Terminal rules of question grammar are as follows:

```java
terminal WH: "What"|"When"|"Where"|...;  
terminal HOW_MANY: "How many";  
terminal FIND: "Find"|"Search for"|...;
```

4. Experimental evaluation

4.1. Implementation of the SBVR Structured Language Editor

We implemented SBVR Structured Language Editor (SBVR SLE) using the presented grammar. The SBVR SLE consists of three parts for specifying business vocabulary, business rules, and questions. The editor was implemented in Eclipse environment [35] using Xtext framework [36], which generates parser, AST metamodel and full-featured Eclipse editor from EBNF grammar descriptions. Xtext generated editor automatically creates the AST from structured language specifications thus making

![Figure 4. Graphical interface of SBVR SLE with package explorer (1), vocabulary outline (2), and editing areas for business vocabulary (3), business rules (4), and questions (5)](image)
4.2. The investigated model of business domain

Using the created editor, the experiment was conducted for checking the relevance and completeness of SBVR structured language grammar for specifying and verifying SBVR business vocabulary, business rules, and questions, and generating XMI schema. As a representative example of business domain, we have chosen organization of events. The conceptual model of event organization domain is presented in Fig. 5 as UML class diagram.

SBVR general concepts are expressed as classes and verb concepts as associations. For the sake of completeness, it contains class hierarchy, generalization between associations, generalization between roles, bidirectional association, aggregation, cardinality constraints etc. Further, we will show that the created grammar is capable for specifying and querying this model using SBVR Structured English notation.

4.3. Specifying business vocabulary

Examples of SBVR vocabulary entries are presented in Table 2. These examples demonstrate that the grammar is suitable for specifying all types of SBVR concepts, creating concept hierarchy, categorization schemes and segmentations, and definitions.

Table 2. Examples of business vocabulary concepts

<table>
<thead>
<tr>
<th>General concepts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td></td>
</tr>
<tr>
<td>person</td>
<td></td>
</tr>
<tr>
<td>General concept: agent</td>
<td></td>
</tr>
<tr>
<td>Synonym: human</td>
<td></td>
</tr>
<tr>
<td>company</td>
<td></td>
</tr>
<tr>
<td>General concept: agent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb concept roles</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>organizer</td>
<td></td>
</tr>
<tr>
<td>Concept_type: verb_concept_role</td>
<td></td>
</tr>
<tr>
<td>General concept: agent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>organizer organizes event</td>
<td></td>
</tr>
<tr>
<td>Synonymous_form: event is_organized_by organizer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property associations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>General_concept: text</td>
<td></td>
</tr>
<tr>
<td>Concept_type: role</td>
<td></td>
</tr>
<tr>
<td>ticket_price</td>
<td></td>
</tr>
<tr>
<td>General_concept: number</td>
<td></td>
</tr>
<tr>
<td>Concept_type: role</td>
<td></td>
</tr>
<tr>
<td>organizer has name</td>
<td></td>
</tr>
<tr>
<td>Concept_type: property_association</td>
<td></td>
</tr>
<tr>
<td>event has ticket_price</td>
<td></td>
</tr>
<tr>
<td>Concept_type: property_association</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>event is_finished</td>
<td></td>
</tr>
<tr>
<td>event venue is_suitable_for_sport_events</td>
<td></td>
</tr>
<tr>
<td>concept 'sports arena' incorporates characteristic 'event venue'</td>
<td></td>
</tr>
</tbody>
</table>

Further transformations simpler. Xtext generated editor is easily configurable and extensible with new functions such as syntax highlighting or external linguistic services. Another feature of Xtext is the ability to define cross references between objects of the language. In our case, it allows to create links between vocabulary concepts (e.g., verb concept roles based on general concepts), and between vocabulary concepts and statements of business rules or questions. Automatic error marker helps to find misspelled words in business vocabulary, business rules, and questions.

The graphical interface of SBVR SLE is presented in Figure 4. It consists of package explorer to manage packages and files (1), outline block for showing the tree of vocabulary concepts (2), and editing areas for business vocabularies (3), business rules (4) and questions (5).
SBVR editor allows writing vocabularies in several separate files and reusing concepts from these vocabularies in other vocabularies. Also, it is possible to define vocabulary of metaconcepts, existing in SBVR metamodel, or extending concepts of SBVR metamodel. This flexibility allows declaring specific concepts required for particular purpose, e.g., one can specify metavocabulary for OWL 2 transformations, containing concepts, extending SBVR metamodel, such as ‘transitive verb concept’, ‘symmetric verb concept’, etc. [17]. Such metaconcepts can be used in a particular vocabulary as concept types, e.g., to declare that a particular verb concept should be transformed into transitive or symmetric object property in OWL 2 ontology:

\[
\text{transitive_verb_concept}
\]

This feature allows extending functionality of the editor for various purposes without modifying the grammar, e.g., for specifying BPMN process vocabularies [39].

### 4.4. Specifying business rules

Examples of business rules are presented in Table 3. It shows that the grammar is suitable for specifying operative, structural, and derivation business rules.

**Table 3. Examples of business rules**

<table>
<thead>
<tr>
<th>Operative rules</th>
<th>Structural rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is obligatory that organizer sells_tickets_for event if event is paid event.</td>
<td>It is necessary that event takes_place_in exactly 1 location.</td>
</tr>
<tr>
<td>It is obligatory that organizer cancels event if number of sold tickets of event is less than 50.</td>
<td>It is necessary that auditorium includes exactly 1 stage.</td>
</tr>
</tbody>
</table>

**Derivation rules**

<table>
<thead>
<tr>
<th>Derivation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is necessary that organizer is company that organizes at least 5 events.</td>
</tr>
<tr>
<td>It is necessary that organizer is experienced if organizer organizes at least 20 events.</td>
</tr>
<tr>
<td>It is necessary that event is_paid if ticket_price of event is greater than 0.</td>
</tr>
<tr>
<td>It is necessary that event takes_place_in location if event takes_place_in event venue which is located in location.</td>
</tr>
</tbody>
</table>

### 4.5. Writing questions

Question examples show that the grammar is suitable for querying models of business domain by writing questions starting with “what” pronouns such as “what”, “where”, etc. The grammar also allows writing question starting with “Find” and “How many” keywords. We do not analyse “Yes/no” questions. SBVR knowledge model is based on open world assumption so answering such questions is complicated. Unlike in conventional databases built with closed world assumption, the absence of required data does not give rise to a negative response. E.g., we cannot say that company A has not organized events if we have no facts about events, organized by company A. We can say so only when we have a fact that company A has not organized any event.
for Lithuanian language

For using SBVR SLE in Lithuanian language, keywords in the grammar were translated and the editor was adjusted to morphological rules of Lithuanian language. While specifying business vocabulary, business rules and questions, it is important to keep relations between concepts, which are related by syntactic matching: therefore, morphologically inflected concepts should be recognized.

Lithuanian language belongs to inflectional languages. In Lithuanian language, nouns can be inflected by a case, number, gender, etc. Single morphemes can have several meanings. E.g., in Lithuanian word teatras the ending us means plurality and a genitive case. As Lithuanian language is morphologically rich, we use morphological analyser created in SemantikaLT project [25] for relating vocabulary concepts by their lemmas (a canonical or citation form) which are found from the morphologically inflected form. E.g., in verb concept spectaklis vyksta auditorijoje general concept auditorija is used in a locative case. By finding lemma (auditorija), verb concept role is related with general concept.

There are words in Lithuanian language that have several meanings. E.g., kasa can be used as a noun or a verb. In Lithuanian language, lemmas of adjectives and nouns are written in a singular form and a nominative case, whereas lemmas of verbs are in the infinitive form. This shows that it is important to analyse appropriate lemmas in ambiguous situations.

Another problem is related with finding lemmas of words in a plural form. As lemma of a word is normally in a singular form, such lemma cannot be related with concept specified in plural. This situation frequently occurs especially when analysing compound phrases (collocations of words). E.g., the second word of scenos dekoracija is in plural form. The lemma of this general concept is scenos dekoracija, where the second word is in the singular form, so concept recognition error occurs. We recognize such general concepts by using morphological generator for generating plural forms.

We have carried out the experiment for investigating how SBVR concepts are recognized using vocabularies in Lithuanian language. In the experiment, a number of morphologically inflected individual concepts and general concepts were used for specifying verb concepts and facts, and concept recognition errors were counted.

We have analysed business vocabularies and business rules of three domains (agents, events, and e-commerce), and compared two different morphological techniques, for recognition of composite terms: lemma finder and lemma finder with morphological generator, which can generate words in certain number, gender, and pronominal form. Some terms were not recognized for two reasons: lemmas were not found, or morphology generation could not help. Results of the experiment are shown in Table 4, which presents the total number of terms to recognize (i.e., morphologically inflected terms used as verb concept roles in verb concepts and facts); the counted number of recognized terms; precision $P$, recall $R$, and $F$ measures for 3 cases: 1) SBVR SLE with lemma finder ($i=1$); 2) SBVR SLE with lemma finder and morphological generator ($i=2$); 3) SBVR SLE with lemma finder and morphological generator but including only those terms whose lemmas were found ($i=3$). Precision, recall and $F$-measure were evaluated by formulas:

$$P_i = \frac{R_i}{AT_i}$$
$$R_i = \frac{R_i}{RT_i}$$
$$F_i = 2 \times \frac{P_i \times R_i}{P_i + R_i}$$

The impact of morphological generator on the quality of SBVR SLE for recognition of complex terms in Lithuanian language was evaluated by the increase of recall $\Delta R_i$ and $F$ measure $\Delta F_i$:

$$\Delta R_i = R_i - R_{i-1}, \quad \Delta F_i = F_i - F_{i-1}, i=2,3$$

The experiment has shown that the morphological generator, integrated into SBVR SL editor, can improve term recognition results by reducing the number of morphological errors. This improvement can vary in different domains. E.g., in the e-commerce and event domains, there were many titles of product categories and spectacles defined using plural form, while lemma finder returned those concepts in a singular form. Different situation was in the agent domain, containing names and surnames that could not be recognized by lemma finder. Therefore, the least improvement made by the morphological generator was obtained in the agent domain.

Though, in general, the recall seems fairly good, this is not enough for term recognition in business vocabularies. For complete elimination of term recognition errors, the morphology support in the SBVR SLE should be improved.

<table>
<thead>
<tr>
<th>“Wh” questions</th>
<th>Find number of organizers that organize events.</th>
<th>Find events that have location city Kaunas.</th>
<th>Find events whose number of sold tickets is_greater_than 100.</th>
</tr>
</thead>
<tbody>
<tr>
<td>What person organize events that have ticket_price that is_greater_than 150?</td>
<td>Find person that organize at_least 5 events.</td>
<td>What person organize events that are free.</td>
<td>All names of organizers that organize events.</td>
</tr>
</tbody>
</table>

A. Šukys, L. Ablonskis, L. Nemuraitė, B. Paradauskas
5. Conclusions and future works

The ultimate goal of our work is to create a framework for querying business data and even software artefacts using structured natural language. In this paper, we have presented the SBVR structured language grammar that is one step towards reaching this goal.

The analysis of similar tools and methods has shown that currently there are no satisfactory solutions for specifying business vocabularies, business rules, questions, and also producing SBVR semantic models. Similar solutions for specifying and analysing business rules use natural or structured natural languages. Natural language solutions allow writing business rules in a more flexible way, but require linguistic analysis for their interpretation. Due to the language ambiguities, such an analysis is not always deterministic. Structured language solutions are more deterministic but they require familiarization of users with their specific syntax.

We have chosen the SBVR Structured English notation for the first implementation of SBVR editor. We have created a grammar for Structured English and implemented the prototype of the editor that allows to specify business vocabulary, business rules, and questions. The experimental investigation of this editor has shown that the underlying grammar allows to specify all types of SBVR vocabulary concepts, their hierarchies, synonyms and synonymous forms, and other features. Grammar can also be used to specify structural, operative, and derivation rules, definitions, and questions.

For adapting the SBVR SL editor to highly inflective Lithuanian language, the editor was supplemented with lemma finder and morphological generator. The experiment, conducted with SBVR business vocabularies and business rules of different domains, has shown that the morphological generator significantly improves capability to recognize various forms of compound terms but is still not perfect.

The greatest novelty and originality of our SBVR SL editor is the capability to use concepts from several vocabularies created in separate files, and to create metavocabularies of metaconcepts – elements of SBVR metamodel or its extensions for various purposes (e.g., SBVR to OWL 2 transformations, BPMN business process vocabularies, etc.). The last feature allows creating extensions of SBVR without changing the original metamodel. Our future work is directed towards further development of these capabilities, as well as towards improvement of linguistic processing means for complete manipulation with SBVR vocabularies, business rules and questions in SBVR Structured Lithuanian language.

Acknowledgements

The work is supported by the project VP1-3.1-ŠMM-10-V-02-008 “Integration of Business Processes and Business Rules on the Basis of Business Semantics” (2013–2015), which is funded by the European Social Fund (ESF).

References


Received January 2015.