VETIS TOOL FOR EDITING AND TRANSFORMING 
SBVR BUSINESS VOCABULARIES AND BUSINESS 
RULES INTO UML&OCL MODELS

Lina Nemuraite, Tomas Skersys, Algirdas Sukys, Edvinas Sinkevicius, 
Linas Ablonskis

Kaunas University of Technology, Department of Information Systems, Studentu 50-313a, 
Kaunas, Lithuania, lina.nemuraite@ktu.lt, tomas.skersys@ktu.lt, sukys.algirdas@gmail.com, 
edvinas.sinkevicius@ktu.lt, vejobrolis@gmail.com

Abstract. The OMG SBVR standard is the most mature abstract representation for Business semantics; 
however, the complexity of SBVR metamodel prevents its broad and rapid usage in business 
communities. There are a few SBVR implementations yet, but the popularity of SBVR is growing as 
different interest-groups are finding the variety of ways for applying the SBVR standard. The goal of this 
paper is to present the VeTIS tool capable for editing SBVR Business Vocabularies and Business Rules 
and transforming them into UML class models supplemented with OCL constraints. It is integrated into 
MagicDraw UML CASE tool and organically combines with simple development process, which pursues 
defining requirements via use cases and modelling business processes via activities.

Keywords: SBVR, UML, OCL, Business Vocabulary, Business Rules, modelling, transformation, 
development process.

1 Introduction

The vision of OMG “Semantics of Business Vocabulary and Business Rules” (SBVR) standard [20] is 
expressing business knowledge in a controlled natural language unambiguously understandable by human and 
computer systems. Such knowledge is captured by business experts or information system analysts who need 
tools that would allow storing SBVR specifications in metamodel repositories (e.g. based on Meta Object 
Facility (MOF) or Eclipse Metamodeling Framework (EMF)) for interchanging and linking them with other 
models (e.g. UML&OCL models). SBVR is fully integrated into the OMG’s Model-Driven Architecture [4].

The precise meaning of SBVR vocabularies allows transforming them to software models of 
information systems without violating business semantics. SBVR vocabularies and business rules are formulated 
in logical formulations that are based on ISO/IEC Common Logic but also they appreciate the grammar of the 
natural language. Currently, the SBVR structured English is the concrete language for SBVR metamodel though 
other languages are possible. The peculiarity of SBVR metamodel is the explicit separation of meaning, 
representation and symbolization: the same meaning can have many representations and the same signifier or 
expression can represent different meanings. For this reason, conceptual models presented in other representation 
languages and having the same interpretation will have many-to-one relationships with the corresponding SBVR 
model. SBVR semantic formulations have formal interpretation and are able to express complex definitions and 
statements that are related with SBVR concepts and fact types. As concepts and fact types are basic elements of 
both natural and software modelling languages, the foundations of SBVR are able to serve as means for better 
alignment of business and information technologies. Another useful quality of SBVR is capability for reflecting 
different contexts – each SBVR Vocabulary represents meaning shared in some Semantic Community. As the 
shared meaning can be represented in multiple languages, the Speech Communities can exist inside the Semantic 
Communities. There are perspectives for having a rich multilingual notation for representing the shared 
semantics defined by SBVR. The complexity of SBVR metamodel prevents its broad and rapid usage in practise.

There are few SBVR implementations yet. The first version of SBVR standard was completed in 2008, but 
preliminary versions existed some years before. Currently, the popularity of SBVR is growing as different 
interest groups are finding variety of ways for applying the SBVR standard in Semantic Web and software 
engineering. As SBVR support in tools is still lacking, we are offering the VeTISI – SBVR editor integrated into 
MagicDraw UML CASE tool capable for generating UML class diagrams with OCL constraints from SBVR 
vocabularies and rules. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 is devoted 
to editing SBVR vocabularies and rules, section 4 – for transforming them into UML&OCL models. Section 5

1 The research is pursued according the project proposal “Methodology and Technology Foundations for Semantically-Based 
Information System Design (SEMIS)”


presents the SBVR based development process. Section 6 draws conclusions and outlines the future work.

2 Related Work

The goal of SBVR standard is specifying business semantics without any considerations about its implementation. It can serve many purposes, for example, ensuring semantic interoperability between different Semantic and Speech Communities or distributed information systems on the Web [22]; or engineering the software; often it is related with service oriented systems [9, 14]. The approaches of business rule automation are endeavouring to build software systems directly from vocabularies and rules. In the context of Model-driven Architecture, the straightforward way for doing this is to transform SBVR into UML&OCL models. The early proposals for such transformations are simplified and limited in scope [12, 17]. The most sophisticated matching between SBVR and UML&OCL is the reverse transformation between UML schemas and SBVR vocabularies and rules in order to verbalize and validate UML schemas [1]. However, it also has drawbacks; for example, it does not consider UML operations and underestimates the actual difficulties of verbalizing conceptual languages as UML. Another important development concerns SBVR transformation into UML with advanced parser for SBVR structured language [10]. Also, there is a parallel research in transforming SBVR specifications into Web Ontology Language OWL and Semantic Web rules R2ML [6]. It is worth to mention that whereas the correspondence between OWL and SBVR is declared in SBVR specification, currently there are no established tools for transforming SBVR vocabularies and rules to ontologies.

The Model Driven Enterprise Engineering (MDEE) methodology created by KnowGravity is one of the first efforts to apply OMG SBVR in the holistic IS development process [21]. MDEE supports the smooth transition from SBVR structural and operative rules to PIM Constraints, ECA (Event-Condition-Action) and CA (Condition-Action) rules. There is another work that incorporates SBVR rules into IBM Model-driven Business Transformation process integrating interaction among rules, processes, and ontologies [11].

SBVR metamodel is used as a modeling language for building the so called generative information systems, built on the declarative technologies [13]. This architecture proposed by Marinos and Krause is based on REST\textsuperscript{IV} architectural style that focuses on resources identified by names, a fixed number of methods with known semantics to manipulate these resources and stateless interactions between client and supplier. As SBVR does not explicitly specify business processes, the resulting architecture is capable of implementing atomic operations on resources, which are necessary but not sufficient for the operation of an information system. For this purpose Razavi et al. extend SBVR specification for declarative representation of business processes and compose them with business rules [18].

The prerequisites for SBVR implementation presented in the current paper were described in [15, 2, 3]. In [14], the possibilities of representing different types of business rules in UML models supplemented with OCL expressions were identified. The paper [2] outlined the methodological points for modelling business processes and business rules: how to separate and relate these concerns for obtaining simple and manageable software systems. The paper [3] presents the SBVR based software development process devoted for service-oriented information systems. It should be noted that our current implementation is more universal and simpler. We present the VeTIS tool that allows editing SBVR business vocabularies and business rules, and transforming them into UML class model with OCL constraints. The VeTIS tool is integrated with the MagicDraw UML CASE tool and organically combines with the incident development process by allowing definition of requirements as use cases and modelling business processes as activities – the usual practice of designing information systems (e.g. [7]) that may be combined with business rule implementation techniques (e.g. [14]).

3 Editing SBVR Business Vocabularies and Business rules in the VeTIS tool

The VeTIS editor was implemented on the Eclipse 3.4.1 platform on the basis of SBVR 1.0 metamodel, whereas the user interface was adapted from SBeaVeR tool [5, 19] that was extended for generating SBVR 1.0 XMI schema, specifying rule sets, validating them etc. Also, we have reused Cabot’s [1] EMF SBVR XMI schema while extending it at some points and merging the five metamodels into one. There are five basic SBVR concept types that can be presented in a Business vocabulary using VeTIS tool: \textit{object types} (or general concepts), \textit{individual concepts}, \textit{fact types} (or verb concepts), and \textit{roles}. All these concept types are represented in different styles for easy recognition: the \textit{term} font is devoted for object types and roles, the \textit{verb} font renders fact types, the \textit{”Name”} – individual concepts (slightly differently from original SBVR style), and \textit{”each”} font is for keywords; individual concepts start with the capital letter to distinguish them from object types. The Business Vocabulary and Business Rule views are presented in the Figure 1.

Following the SBVR, fact types are defined using the existing noun concepts (object types or individuals), which have already been defined in the business vocabulary. \textit{Verbs} (or fact symbols) represent fact

\textsuperscript{IV} REST is a competing paradigm for Web Service technologies as the latter suffer from theirs complexity and lack of adoption for distributed applications on the World Wide Web
types that create relationships between these nouns or specify their characteristics. For example, “bank gives loan” presents an active sentential form of the fact type. Also, fact types can have passive sentential forms (e.g. “loan is given by bank”) or noun forms (e.g. “loan of the bank”). All these forms are synonymous forms of one and the same fact type (similarly, noun concepts can have synonyms), while the verb in the expression “bank gives loan” is accepted for the primary designation of that fact type. Normally, the primary designation of a concept corresponds to its preferred designation. In reverse cases, the description of a concept includes a reference to the preferred representation under the caption “See”. Synonymous forms of fact types are mostly used in business rules statements. Only primary designations of the meaning are assessed during transformation of SBVR specifications into UML models.

![Figure 1. Business Vocabulary and Business Rule views in the VeTIS tool](image)

The examples presented above are of the most common associative fact type that has two or more roles involved. A role is a noun concept that corresponds to things based on their playing a part, assuming a function or being used in some situation. For example, in the fact type “debtor requests loan” the “debtor” is the role played by a person. A role is identified by the vocabulary entry, which has the field “Concept_type:” set to “role”, and the “General_concept:” field set to the object type, which plays that role in the corresponding fact type. Role is represented by a placeholder – a place for expression of what fills a role in the fact type form. VeTIS supports only two roles for associative fact types. Though SBVR allows fact types with many roles, interpretation of n-ary fact types is more complicated; such concepts are not well supported in other languages (such as OWL) or tools (such as MagicDraw UML). It is always possible to represent fact types with several roles by using several fact types with two roles. Usually, there are two noun concepts playing specific roles in the relationship defined by the associative fact type. Only in the case of a reflexive relationship, there is one noun concept playing both roles. A partitive fact type is the binary fact type stating that one noun concept (all of its instances) is in the composition of a given whole, i.e. another noun concept. Partitive fact types are identified by the verb phrases “includes” and “is_included_in” (for active and passive forms respectively).

Is_property_of fact types define the essential qualities of a given noun concepts. They are identified by the verb phrases “has” and “is_property_of” (for active and passive forms respectively), e.g. “loan has amount” or “amount is_property_of loan”. It is unnecessary to define “Concept_type: role” for roles that are used in is_property_of fact types. Roles of is_property_of fact types have predefined elementary object types text, integer and number as their general concepts, e.g. role “amount” will have the “number” in the “General_concept:” field. Characteristic is a fact type having only one role, e.g. “loan is_returned”.

Categorization fact type represents relationship between the more general noun concept and more specific noun concept, which is a category of the first concept. VeTIS tool allows specifying simple categorization fact types, e.g. “accepted_loan is_a loan” by using the following pairs of verbs and verb phrases: “specializes” and “generalizes”, “is_category_of” and “is_of_category”, “is_a” and “is_a” (VeTIS interprets is_a as a relationship between a specific concept and the general concept, and creates the synonymous form with the verb phrase is_a between the general concept and the specific concept by default). Simple categorization fact types can also be represented in noun concept entries with additionally specified “General_concept:” fields. More complex structures involving categorization types and categorization schemes are also supported by VeTIS. Categorization scheme is a set of categories that subdivides instances of a general concept into subsets specialized by some feature (categorization type) (Figure 2). Such a structure is mandatory, if you want to specify the complete and correct information about a categorization scheme. Similarly, segmentation is a
According to SBVR, *business rules* are rules under business jurisdiction. Business rules are constructed as *closed modal formulations* that have recursively embedded logical formulations and are based on fact types. Modal formulations may be *alethic* modal formulations, i.e. necessities, possibilities or impossibilities used in Structural Business Rule statements expressing Structural Business Rules, or *deontic* modal formulations, i.e. obligations, permissions or prohibitions used in Operative Business Rule statements expressing Operative Business Rules. *Structural Business Rules* are rules that govern the conduct of business activity (dynamic or action rules). In contrast to Structural Rules, Operative Rules can be *directly* violated by people involved in the affairs of the business. VeTIS supports four types of business rules (you can define impossibilities and prohibitions using remaining types of rules):

- **Necessities**: “It is necessary that a **loan** owns exactly one **bail**.”
- **Possibilities**: “It is possible that a **debtor** gets at most 3 **loan**.”
- **Permissions**: “It is permitted that a **debtor** requests a **loan**.”
- **Obligations**: “It is obligatory that a **bank** gives a **loan** if the **loan** is a **valid_loan** and the **loan** is a **reliable_loan**.”

## 4 Transforming SBVR specifications into UML&OCL models

One of the core features of the VeTIS tool is the transformation of the SBVR specification (i.e. Business Vocabulary and Business Rules) in SBVR Structured English into SBVR 1.0 XMI format and subsequently – into the UML class model with OCL constraints (in EMF UML 2.1.2 XMI). For that purpose ATL transformation language [8] and ATL transformation engine 3.0.0 were used. In the last step UML model is visualized in the MagicDraw UML tool using Open API. The transformation process is presented in Figure 3.

**Business Vocabulary** (along with Business Rules) is transformed into a UML *package*, while the vocabulary name is transformed into the package name. Two elements are included in that newly created package: “VeTIS” profile that supports visualization of constraints in UML class diagrams, and “Constraints” package for holding OCL constraints obtained from SBVR. “VeTIS” profile includes single stereotype `<<constrained>>`, base metaclass of which is “Element”, so it is applicable for classes, operations and other UML elements.

**Object type** (represented by term) is transformed into a UML *class* having the same name as the object type. **Primitive object types** as numbers, texts, integers etc are transformed into UML *data types*. **Role** (represented by a placeholder) is transformed into UML property having the same name as its placeholder and the type of the property is set to a class obtained from the object type playing the corresponding role in that fact type.
Associative fact type is transformed into an association relationship between two classes. The names of properties corresponding to the ends of that association are obtained from names of roles of that fact type. A verb (or a verb phrase) used by the fact type becomes a name of that association. If there are no business rules constraining the number of occurrences of instances of a fact type, multiplicity bounds of the association are set to “0..*” by default. Is_property_of fact type is transformed into a property presenting an attribute of a class similarly as the associative fact type. Roles in SBVR are ordered, thus a class owning the attribute is obtained from object type that plays the second role in the corresponding is_property_of fact type. The type of an attribute is set to a data type obtained from the “General_concept:”, which specifies a primitive object type.

Characteristic (fact type having only one role) is transformed into an attribute of the Boolean type. Partitive fact type is transformed into a composition relationship between two classes. Categorization fact type is transformed into a generalization relationship between two classes. Categorization scheme is transformed to a generalization set. Categorization type is transformed to a class, which is set as a powertype of that generalization set. The name of categorization scheme becomes the name of the generalization set and the generalization set is referenced in generalization relationships obtained from categorization fact types included in that categorization scheme. Segmentation is transformed in the same manner as a categorization scheme except that generalization relationships are generated with the {complete, disjoint} constraints instead of the {incomplete, disjoint} constraints used in the case of the categorization scheme.

4.2 Transforming Business Rules into UML

Some types of business rules are transformed into elements of UML class diagrams. Structural business rules formulated by alethic modal formulations having directly embedded quantifiers (different from universal quantifier) are transformed into multiplicity bounds of the corresponding associations or compositions (Figure 4). In VeTIS, it is possible to define the overall variety of multiplicity bounds specified by SBVR structural business rules.

Figure 4. Transforming alethic modal formulations

Operative business rules (obligations and permissions) formulated as closed deontic formulations are transformed into UML operations. Verbs denoting fact types on which these formulations are based are transformed into operation names, roles – into parameter names, object types playing these roles – into parameter types. For example, the business rule: “It is permitted that a debtor requests a loan.” is based on the fact type “deborah requests loan” and it is transformed into the following UML operation: “Loan::requests(debtor:Person, loan:Loan):void” (the operation of the class Loan in Figure 8).

4.3 Transforming Business Rules into OCL

Structural Business rules formulated by alethic modal formulations that do not have directly embedded implications are transformed into OCL class invariants expressing integrity constraints. For example, structural business rule “It is necessary that the account is_owned_by exactly one person” is transformed to:

   context Account
   inv invAccount1: self.owns(account) implies
       self.owns(account) <- 1

Some types of business rules are transformed into OCL class invariants expressing derivation rules. It is necessary to describe derivation rules for all specialized concepts that cannot be categorized only by playing roles in fact types.

   “It is necessary that the requested_loan is_a_reliable_loan if each issued_loan that_is_of_the debtor that_receives_the issued_loan is_returned.” is transformed to:

   context RequestedLoan
   inv invRequestedLoan3: self.oclIsTypeOf(ReliableLoan) implies
       self.debtor.issuedLoan→forAll(it1|it1.isReturned=true)

   \footnote{We will use “directly embedded” for the sake of simplicity in the cases like this. Factually, these “quantifiers different from universal quantifier” are embedded into a universal quantifier that is embedded into a corresponding modal formulation}
Operative business rules (obligations and permissions), formulated as deontic formulations with directly embedded implications, are transformed into operations and operation preconditions. Implication consequent is transformed into operation, antecedent – into precondition:

“It is obligatory that bank checks_reliability_of the requested_loan if the requested_loan is a valid_loan.” is transformed to

```ocl
Context RequestedLoan::checksReliabilityOf(bank:Bank, requestedLoan:RequestedLoan):
  OclVoid pre preChecksReliabilityOf1: self.oclIsTypeOf (ValidLoan)
```

There are further possibilities to automatically obtain attribute default values, operation results and post conditions from SBVR definitions. Though mathematical calculations are difficult to express in SBVR, the standard is extensible. Therefore, such improvements depend on the willingness of developers.

5 Development process based on SBVR

The process of defining business vocabularies and business rules is not easy. It is difficult to formulate consistent and complete sets of concepts and propositions governing business. SBVR specifications are of declarative nature and they define business constraints – constraints on structure and on activities, but not on control flows of these activities, i.e. business processes. Though it is possible to predefine sequences of activities by declarative constraints or even extend SBVR for specifying business processes, such a practice is not recommended by business rule methodologists. In contrast, they propose the “separation of concerns” – keeping models of business processes and specifications of business rules separately and not intertwining them, because they are changing independently [2]. Also, visual modelling is better suited for definition of business processes. Additionally, modelling of business processes helps define right and consistent business rules as well. Let’s take a look at how all this integrates into the software development process (Figure 5).

In the presented process, we recommend writing software requirements (i.e. use case specifications) in alignment with defining business processes (e.g. in the form of BPMN or UML activity diagrams), business vocabularies and business rules. We will demonstrate the process with the simplified example of a Bank Information system (Figure 6). In the first step „Define use cases“ (Figure 5. SBVR based development process) we choose the use cases of the system and in the second step we define straightforward processes representing steps of main success scenarios. We draw a UML activity diagram for each use case and represent our desired business processes (Figure 6).

In the 3rd and 4th steps we define business vocabulary and business rules. Now we can try to transform the vocabulary and rules to UML&OCL model though the model would not be complete. In the 6th step “Define alternative scenarios” we consider how the process will look in the cases where requested loan is not valid or unreliable. Refined activities representing use cases include specific concept categories corresponding to
different states of object types: requested loan, issued loan, rejected loan, etc. We supplement business vocabulary by adding these categories and specify categorization schemes if needed. Then we can augment the business vocabulary with fact types corresponding to activities of alternative scenarios (Figure 7).

Figure 7. Activity diagrams representing alternative scenarios of Bank information system

Now we can add or refine operative business rules regarding alternative scenarios providing when needed the corresponding synonymous forms. Also, we describe additional structural rules for derivation of categorized object types and all required constraints. After exporting to MagicDraw UML, we obtain the class diagram as shown in Figure 8. In reality, the process would not be so straightforward. We would make many refinements and iterations till moving to the next phase(s) of Model-driven software development.

Figure 8. UML class diagram with OCL constraints obtained after transformation

6 Conclusions and Future Work

The implementation of VeTIS has shown the possibility of transforming SBVR vocabularies and rules into non-trivial conceptual models. We have found that UML language is capable for representing all concepts of SBVR vocabularies (we have explained earlier why we have abandoned fact types with many fact type roles, though their representation in UML is feasible) and some kinds of business rules: alethic formulations having directly embedded quantifiers expressing multiplicity bounds, and simple deontic rules (having no directly embedded implications) expressing unconstrained UML operations. Our transformations from SBVR rules into OCL are yet incomplete for the following reasons:

– It is difficult to recognize the right formulations of complex rules in SBVR structured language. In order to generate SBVR XMI schema for complex rules, we are now considering several possible approaches: creating the advanced parser, which could present several possible variants of meaning for confirming to user; proposing rule wizards; implementing visual editor for rules, etc.
SBVR metamodel is not fully comprehensive with regards to specifying rules: it does not allow representing simple computations, iterations, if-then-else constructs etc. The extensibility of SBVR makes it possible to supplement it; however, it is difficult to anticipate encompassing all possibilities of OCL. From the other side, OCL is not capable to expressing SBVR modalities; however, modalities are not very important for implementing software.

Our future work is directed to enhancing the VeTIS interface for entering complex rules, and to implementing multilingual vocabulary (at first only for Lithuanian language). For this purpose, SBVR terms, names and other designations should be extended with attributes for identifying the corresponding language, and with synonymous forms for representing cases. Furthermore, there are a lot of research problems in managing and reusing large vocabularies, enhancing transformations etc.

References


