SBEAVER: A Tool for Modeling Business Vocabularies and Business Rules

Maurizio De Tommasi and Angelo Corallo

e-Business Management School - ISUFI
University of Lecce
via per Monteroni sn, 73100 Lecce (Italy)
{maurizio.detommasi, angelo.corallo}@ebms.unile.it

Abstract. Methodologies in software development are typically applied when a problem is already formulated and described. Software developers transform requirements into code with a relatively repetitive process. The actual difficulty lies in describing business needs and expected functionalities. Stakeholders involved in software development can express their ideas using a language close to them, but they usually are not able to formalize these concepts in a clear and unambiguous way. In this paper, we introduce a new tool intended primarily for business analysts and modelers who want to formalize their business knowledge using a business oriented notation based on natural language and fact-oriented approach. Moreover, the capability to map models to formal logic allows automatically generation of IT system design artifacts bridging the existing language gap between business and IT.

1 Introduction

E-business adoption represents a unique opportunity for enterprises to gain competitiveness by improving their business performance at a local level and helping them take advantage of global market opportunities. A major stumbling block for enterprises (especially Small and Medium Enterprises) in adopting new ICT is the lack of a common operational notion describing business needs and ICT solutions. The need for natural language as a means for business modeling is particularly important in order to allow enterprises to author, validate, and dynamically define and redefine in the underlying IT-systems their products, services, prices, policies and terms [6]. As such, following the MDA roadmap [10], the direct involvement of business actors as well as the adoption of a business oriented formal language in business modeling would represent a concrete attempt to align the business strategy with IT infrastructure [5]. The main purpose of natural language modeling approach is hence to make natural language suitable for conceptual modeling. In other words, it aims at designing analytic processes able to produce a simple syntax and to reduce ambiguity and vagueness, preserving language completeness and essential meaning [3].

In addition, formalizing business policies and rules allows for automated transformations enabling creation of corresponding Platform Independent or Platform

Specific Model elements (PIMs or PSMs) in different software and systems architectures to support the design and construction of the system (business applications, workflow, etc.). Furthermore, formal business models support evolution of the system design as business facts and rules change, thus allowing alignment between business strategy and the underlying IT infrastructure [9].

In this paper we present a tool called SBEAVER which allows business analysts or experts to create formal and interchangeable business models in accordance with the OMG's *Semantics of Business Vocabulary and Business Rules* (SBVR) standard. It is a rich text editor which allows the business modeler to type structured sentences and business rules through an easy-to-use graphical interface. The editor guides the modeller in the process of creating a business model in a computation-independent fashion, avoiding technical modeling formalisms typically based on the object oriented modeling paradigm as used by IT system designers and technical people. The resulting models serves two different purposes. On the one hand, they are semantically rich shared representation of business knowledge useful as guidance for business and input to IT system specifications. On the other hand, the models are suitable for easy interchange among people and organizations or software tools and repositories.

2 Business Modeling Approach

The Semantics of Business Vocabulary and Business Rules [12] defines a metamodel conceptualized for business people and designed to be used for business purposes, independently of information systems designs. Its first aim is to allows business vocabularies construction and business rules definitions by business people in business language (i.e. natural language, common graphics, and tables), enabling their interchange among organizations.

SBVR is self-describing: the foundation that makes up SBVR itself is represented through SBVR Vocabularies and their related rules. The SBVR Vocabulary is extensible: since it is a vocabulary, it can be included in other vocabularies in order to create an extended SBVR Vocabulary. The latter can, for example, add new symbol for existing concepts or add new concepts along with symbols that represent them. In this way, even if the SBVR Vocabulary is based on English language, it is possible to create an alternative SBVR Vocabulary based on a different language; it should provide symbols from the different language for the concepts represented in the SBVR Vocabulary.

2.1 Business Vocabularies and Rules

A business vocabulary contains all the specialized terms and definitions of concepts that a given organization or community uses in their talking and writing in the course of doing business. A SBVR business vocabulary provides capabilities to define taxonomies, categorization schemes, Thesauri including synonyms, abbreviations. It provides also the ability to specify definitions formally and unambiguously in terms of other definitions in the business vocabulary as well as the ability to define connections between concepts (fact types). The SBVR follows a common-sense definition of business rule: *rule that is under business jurisdiction* [12]. 'Under business jurisdiction' is taken to mean that the business can enact, revise and discontinue business rules as it sees fit. Rules serve as criteria for making decisions. They are statements that define or constrain some aspect of the business [14]. In SBVR, rules are always constructed by applying necessity or obligation to fact types.

In order to enable the use of SBVR artifacts in information systems design, SBVR is underpinned by First Order Predicate Logic, but it also provides an extended formalization for higher-order types, that uses a restricted version of higher-order logic. The formal semantics of SBVR is based on various formal approaches: typed predicate logic, arithmetic, set and bag theory, with some results from modal logic.

2.2 Semantic Interchange

As stated above, a business vocabulary provides a means of recording and communicating facts. Following OMG's Model Driven Architecture [11], a business vocabulary developed as an information system independent model (Computation Independent Model or CIM) is used to drive the creation of a platform independent MOF model representing these facts. The MOF model is, in turn, used to drive generation of Java interfaces and an XML schema [7]. The SBVR Metamodel is intended to provide for standardized data interfaces and data interchange among tools that collect, organize, analyze and use Business vocabularies and rules, as well as tools that bind business vocabularies and rules to other models and implementations.

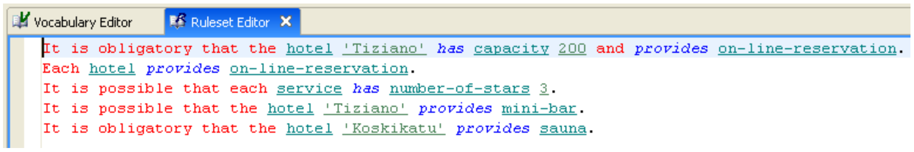
In order to address semantic interchange of business vocabularies and rules SBVR specification defines a Vocabulary-to-MOF/XMI Rule Set that governs how a business vocabulary is mapped to a MOF 2 model. An XML Schema could be then generated based on XMI 2.1. The resulting SBVR model is intended, not for business people, but for software engineers that build tools for business people. The SVBR Metamodel is includable and extendable in models that address various business domains. That the SBVR Metamodel is generated without manual intervention guarantees that it accurately represents the concepts of the SBVR Vocabularies.

SBVR explicitly uses the fact-oriented approach in order to standardize business-level data interchange. This approach implies the creation of a separate class for each type of fact that can be expressed; in other words, each fact is represented by its own object, and not by an attribute of some other object.

2.3 Structured English Notation

The most common means to express definitions and business rules is through statements. Even if there are numberless ways to use a language in order to express them, SBVR specification introduces a small number of English structures and common words in order to provide a simple and automatic mapping to SBVR concepts. In fact, since SBVR aims to be a powerful means to express and interchange business semantics, its primary focus is to provide the means to express each possible statement in an unambiguous form. For this reason, SBVR specification defines a SBVR Structured English. This means that each statement represented in terms of SBVR Vocabulary has a structured form and, in particular, can be represented through a logical formulation (i.e. the SBVR representation of formal logic). Consequently, all formal definitions and rules stated using the SBVR Structured English can be automatically interpreted in order to create MOF and/or XMI representations. However, it is important remembering that SBVR Structured English is just one of many possible notations that can map to the SBVR Metamodel.

The SBVR notation is characterized by some specific font styles and some keywords, each with its formal meaning. Moreover, it describes some structures to be used in order to define each vocabulary entry and rule. Fig. 1 shows some example of Structured English business rules.



```
It is obligatory that the hotel 'Tiziano' has capacity 200 and provides on-line-reservation.
Each hotel provides on-line-reservation.
It is possible that each service has number-of-stars 3.
It is possible that the hotel 'Tiziano' provides mini-bar.
It is obligatory that the hotel 'Koskikatu' provides sauna.
```

Fig. 1. SBEAVER: Structured English business rules

3 SBEAVER: A Business Modeling Tool

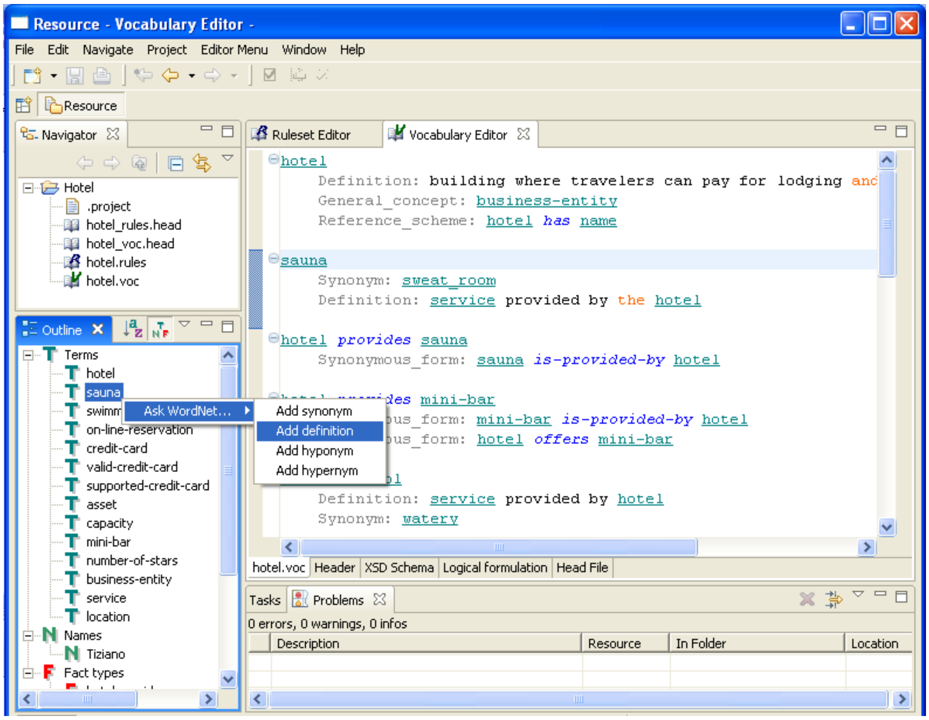
In order to enable creation of SBVR business models (in terms of business vocabulary and business rules, as explained in the previous section) we developed an open source tool called SBEAVER. It allows to define concepts, fact types and business rules following the SBVR semantics in accordance with the Structured English notation and its mapping to formal logic. Sentences representing fact types and business rules are parsed and validated using a *LL* parser with multi-token lookahead[1] in order to have an in-memory representation of the model conforming to the SBVR Metamodel. The resulting in-mamory model representation is then suitable for further transformations.

SBEAVER is designed primarily for business analysts and modelers working in enterprises, who want to specify business policies and rules precisely and using a non technical notation. Their business view is the enterprise business view, or perhaps a view of part of the business. Nevertheless, mapping to formal logic

[1] An *LL* parser is a table-based top-down parser for a subset of the context-free grammars. It parses the input from Left to right, and constructs a Leftmost derivation of the sentence. An *LL* parser is called an *LL(k)* parser if it uses *k* tokens of look-ahead when parsing a sentence.

allows to translate natural language based artifacts into different representation formats (e.g. MOF/XMI, XML or UML).

Currently we support automatic generation of XMI XML Schema and Prolog. This feature makes the tool suitable for users such as system engineers, integrators and developers responsible for designing, integrating and implementing IT systems following the business rules and contents provided by business people. Automating transformation of software artifacts from business rules represents a basis for validating the system design by business people at the business level of understanding.



```
hotel
    Definition: building where travelers can pay for lodging and
    General_concept: business-entity
    Reference_scheme: hotel has name

sauna
    Synonym: sweat_room
    Definition: service provided by the hotel

hotel provides sauna
    Synonymous_form: sauna is-provided-by hotel

                 des mini-bar
              ous_form: mini-bar is-provided-by hotel
              ous_form: hotel offers mini-bar

               l
    Definition: service provided by hotel
    Synonym: watery
```

```
0 errors, 0 warnings, 0 infos
```

Fig. 2. The SBEAVER main window

3.1 Main Features

SBEAVER key features are:

- Business knowledge representation: SBEAVER provides a tool for formalizing the semantics of business knowledge using the Structured English notation.
- Portability: SBEAVER is written in Java for portability and could be installed on many different operating systems without code changes. SBEAVER comes with all source code with EPL licence which can be modified or tailored to meet a user's specific needs.

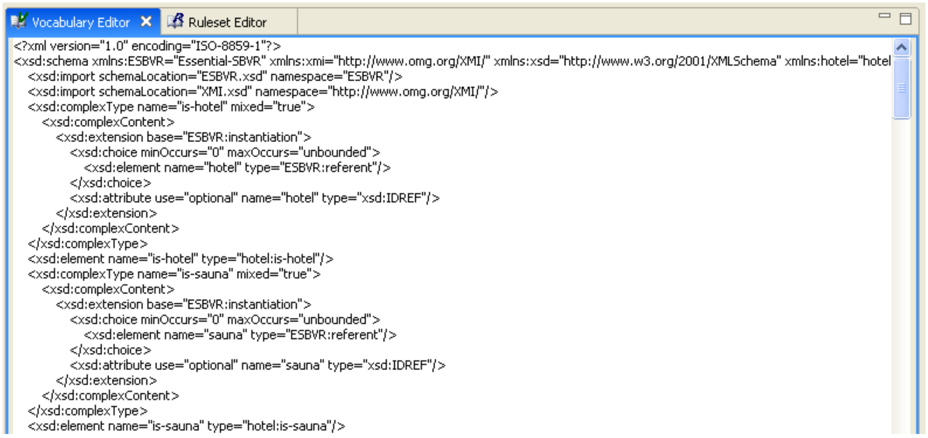
- Integration and Extensibility: SBEAVER is based on Eclipse platform which allows to easily extend or integrate the tool by a user through the use of a well-defined framework.
- Model validation: SBEAVER includes a number of features to support the verification and validation of business models including semantic analysis of rule to determine possible inconsistencies.

3.2 Functional Architecture

Fig. 2 shows the SBEAVER main window. In particular, from a functional architecture point of view SBEAVER provides:

- Presentation and user modification of SBVR Structured English business vocabularies and business rules.
- Standard text editing operations (cut, copy, paste, find, replace).
- Automatic syntax highlighting following the SBVR Structured English font styles.
- Content assist to allow easy and fast creation of contents.
- Hierarchical navigation of vocabulary, as shown in the left bottom column in Fig. 2.
- Dictionary (at present we embed WordNet dictionary) support for synonyms, hypernyms, hyponyms, meronyms, definitions.
- Mapping from in-memory Java representation of vocabulary and rules to logical representation.
- XMI XML Schema serialization allowing interchangeability of models between software tools.

Fig. 3 shows the XSD schema generated automatically starting from the model shown in the main window in Fig. 2.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:ESBVR="Essential-SBVR" xmlns:xmi="http://www.omg.org/XMI/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:hotel="hotel
  <xsd:import schemaLocation="ESBVR.xsd" namespace="ESBVR"/>
  <xsd:import schemaLocation="XMI.xsd" namespace="http://www.omg.org/XMI/"/>
  <xsd:complexType name="is-hotel" mixed="true">
    <xsd:complexContent>
      <xsd:extension base="ESBVR:instantiation">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="hotel" type="ESBVR:referent"/>
        </xsd:choice>
        <xsd:attribute use="optional" name="hotel" type="xsd:IDREF"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="is-hotel" type="hotel:is-hotel"/>
  <xsd:complexType name="is-sauna" mixed="true">
    <xsd:complexContent>
      <xsd:extension base="ESBVR:instantiation">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="sauna" type="ESBVR:referent"/>
        </xsd:choice>
        <xsd:attribute use="optional" name="sauna" type="xsd:IDREF"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="is-sauna" type="hotel:is-sauna"/>
```

Fig. 3. SBEAVER generates automatically XSD Schemas

4 Related Work

As stated in [3], in the past, the use of natural language to influence software design has been mainly tacit and informal.

In the late 1970's, Halsteads [8] categorization of operands and operators from natural language elements raised the possibility that software theory and quantitative analysis might extend to natural language, suggesting the existence of a mapping from natural language to computational primitives.
In the early 1980's, Abbott [1] proposed an approach to Ada program design based on linguistic analysis of informal strategies written in English.

In 1989, Saeki, Horai, and Enomoto [13] proposed a software design process based on natural language, complementing the work by Abbott. They focused on the identification of dynamic system behavior as expressed by the verbs in a natural language description. Their work offers many useful ideas regarding the information needed to represent the relationships between natural language elements and some rules for selecting message senders and receivers.

In 1990, Carasik, Johnson, Patterson, and Von Glahn [4] pointed out the limitations of entity-relationship models for defining semantic intensions arguing for the usage of conceptual modeling languages and knowledge representation techniques to formally represent meaning. However, conceptual models need not diverge so far from natural language as to be unintelligible to stakeholders.

In 1992, Cordes and Carver made one of the first attempts to apply automated tools to requirements analysis and the automatic generation of object models from requirements documents. While the translation of the initial requirements into a suitable knowledge base requires human interaction to resolve ambiguities, the translation of the domain knowledge into object models is automated. However, they recognized that the translation of formalized knowledge into object models is sensitive to the quality of the initial requirements specification. Still, their process and tools can help a requirements analyst begin to bridge the gap between informal requirements and formal software models.

The RECORD (REquirements COllection, Reuse and Documentation) [2] project has recently integrated several tools and techniques, including natural language processing, with the aim of providing a complete solution for requirements collection, analysis, management and object-oriented modeling.

5 Conclusions

Natural language is generally used by business organizations in order to describe themselves and their rules. Nevertheless, even if complex constructs and ambiguous forms of expression provide a great communicative power, they usually make this description unclear and informal. Conversely system requirements gathering and creation of machine-readable documents need a higher degree of precision and formality, with a consequent loss in richness of meaning and expressions.

In this paper, we have presented SBEAVER, an easy-to-use and extensible business modelling tool supporting the Semantics of Business Vocabulary and

Business Rules standard and allowing business modelers to capture and formalize business knowledge in a fact-oriented and natural language approach. We have also shown how semantically rich computation independent business models can be automatically translated into other system design models thanks to the SBEAVER's capability to map Structured English to formal logic. This feature makes SBEAVER suitable for different users which, at different level of understanding, share meanings relevant to the business that drive the creation of IT systems. However, there are still many possibilities for improvement.

First, although we already use one of the most important English dictionary, we need to widen the selection of dictionaries as well as the selection of language notations (currently we support only the Structured English), thus enabling multilinguality.

Second, we recognize that alternative graphical notations are likely important to represent business knowledge in a more compact manner in order to facilitate readability of textual models.

Third, since SBEAVER is in its preliminary phase, implemented funtionalities need to be further tested and stabilized as well as new features need to be developed in order to improve usanility and content development.

Finally, we note that an obvious barrier remains: because of the new approach to business modeling, sufficient knowledge of the main characteristics of SBVR is a necessary prerequisite for effective use of our tool.

References

1. R. Abbott. Program design by informal english descriptions. *Communications of the ACM*, 26(11):882–894, Nov 1983.
2. J. Börstler. User-centered requirements engineering in record - an overview. In *Proceedings NWPER'96, the Nordic Workshop on Programming Environment Research.*, pages 149–156, Aalborg, Denmark, May 1996.
3. N. Boyd. Using natural language in software development. *Journal Of Object Oriented Programming - JOOP*, 11(9):45–55, Feb 1999.
4. R. Carasik, S. Johnson, D. Patterson, and G. Von Glahn. Towards a domain description grammar: An application of linguistic semantics. *ACM SIGSOFT Software Engineering Notes.*, 15(5):28–43, Oct 1990.
5. B. Connell. Web Services Management in Action: Aligning IT with Business Objectives. Westglobal `http://www.westglobal.com`, 2003.
6. M. De Tommasi, V. Cisternino, and A. Corallo. A rule-based and computation-independent business modelling language for digital business ecosystems. In *KES (1)*, pages 134–141, 2005.
7. D. S. Frankel, editor. *Model Driven Architecture Applying MDA to Enterprise Computing.* Wiley Publishing inc., 2003.
8. M. Halstead, editor. *Elements of Software Science.* Elsevier North-Holland, Inc., New York, NY, 1977.
9. Hendryxs & Associates. Integrating Computation Independent Business Modeling Languages into the MDA with UML 2. document ad/03-01-32 `http://www.omg.org/cgi-bin/doc?ad/03-01-32`, Jan 2003.
10. A. Kleppe, J. Warmer, and W. Bast, editors. *The Model Driven Architecture; Practice and Promise.* Addison Wesley, 2003.

11. J. Miller and J. Mukerji. MDA Guide Version 1.0.1. OMG `http://www.omg.org`, Jun 2003.
12. OMG. Semantics of Business Vocabulary and Business Rules (SBVR), ver1.0, draft adopted specification. document dtc/05-11-01 `http://www.omg.org/cgi-bin/doc?dtc/05-11-01`, Nov 2005.
13. M. Saeki, H. Horai, and H. Enomoto. Software development process from natural language specification. In *Proceedings of the 11th International Conference on Software Engineering (ICSE-11): IEEE Computer Society Press*, 1989.
14. The Business Rules Group. Defining Business Rules - What Are They Really? Final Report, revision 1.3. BRG `http://www.businessrulesgroup.org`, 2000.