# Semantics in Model-Driven Business Design

Mark H. Linehan

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
mlinehan@us.ibm.com

**Abstract.** This position paper describes ongoing work in applying the new OMG standard called *Semantics in Business Vocabulary and Rules* (SBVR) to a model-based approach to business design and implementation. The work explores methods of specifying semantics and rules in SBVR's "Structured English" as extensions of business models that are automatically translated to executable solutions.

## Introduction

The Object Modeling Group's (OMG's) Model-Driven Architecture [13] concept defines a multi-layered approach to defining business solutions, as shown in figure 1.

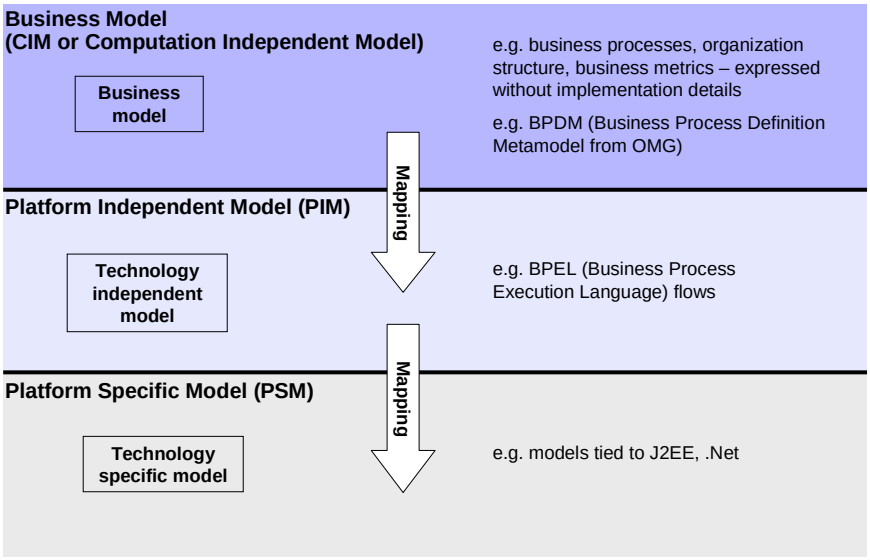


Fig. 1. OMG Modeling Layers

Figure 2 summarizes OMG efforts to define standards for rules at the top two layers. The *Semantics of Business Vocabulary and Rules* [15] activity is defining a "Structured English" approach to vocabulary and rules at the Business Model or Computa-

tion Independent Model layer. The *Production Rules Representation* [14] aims to specify a standard Unified Modeling Language (UML) model for rule structures.

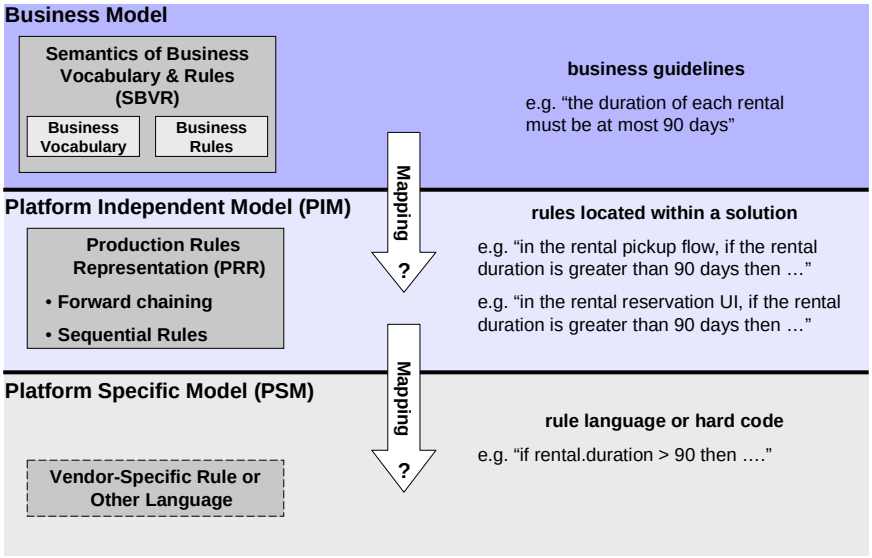


**Fig. 2.** Rules in the MDA Layers

This paper summarizes an ongoing effort to implement a subset of SBVR in the context of an existing *Model Driven Business Transformation* project [10] at IBM Research.

## MDBT – Model-Driven Business Transformation

MDBT is a methodology and matching toolkit for defining a business solution at the business modeling layer, and then semi-automatically transforming the solution into a PIM-layer and then a PSM-layer implementation. A business analyst applies the methodology by defining a business model using the IBM WebSphere Business Modeler [7] tool and the MDBT semantics. The analyst then converts the business model to a PIM-layer model using the IBM Rational Software Architect [6] product, and further transforms the PIM-layer model to an executable implementation using the IBM WebSphere Integration Developer [8] tool and IBM WebSphere Process Server [9] runtime. The generated implementation includes Data Definition Language (DDL) statements to generate relational database tables, state machine definitions for executing the solution, skeleton user interface Java Server Pages (JSPs), and service definitions in the form of Web Services Definition Language (WSDL) files. The implementation incorporates business performance monitoring functions and dashboards, as described in [2].

The transformation process from business layer to implementation can be fully automated in a "rapid prototyping" mode. Manual intervention at the PIM and PSM lay-

ers are needed to produce production-quality user interfaces and adapters for invoking legacy systems as services.

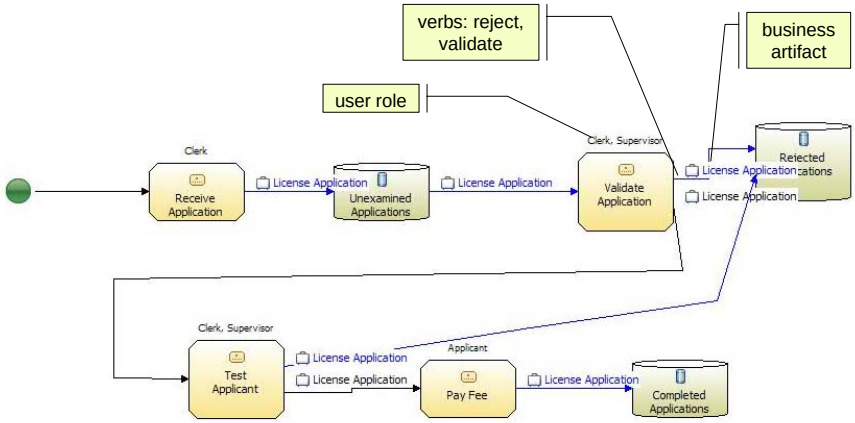


**Fig. 3.** Business-Layer Model of a Driver's License Bureau

Figure 3 shows a simple solution example at the business layer. This shows the processing flow of a Driver's License Bureau which handles License Applications. The flow starts at the dot on the left, and proceeds through the illustrated stages. The rounded squares show processing tasks, while the database icons show repositories for holding in-process work. The rectangular callouts indicate the three primary concepts captured in this model: user roles, verbs associated with the output sides of tasks, and the business artifacts processed by the solution.

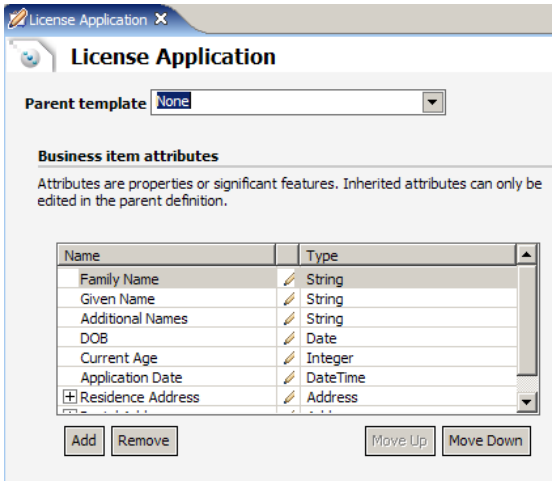


**Fig. 4.** Business Artifact

At the business layer, artifacts are detailed in terms of their attributes. Figure 4 shows that a License Application contains various fields, similar to properties in UML classes.

What's missing from the business-layer model is any concept of business rules. For example, perhaps the applicant must be at least 18 years old to get a driver's license. In the current MDBT approach, such rules must be implemented manually at the PSM layer. A method of specifying such rules at the business layer and then transforming them to the implementation would improve the MDBT methodology. The objective of this project is to examine the suitability of SBVR for this purpose.

**SBVR**

SBVR provides a framework for defining business vocabulary and rules at the business modeling layer using "Structured English" and applying stylized text to four key concepts:

- The 'term' style applies to noun concepts, such as 'License Application'.
- The 'Name' style designates individual concepts, such as a clerk named 'Bill'.
- The '*verb*' style identifies fact types, which define relationships between concepts. For example, 'clerk *validates* application'.
- The 'keyword' style distinguishes various words used to construct vocabulary definitions and rule statements. The keywords designate built-in SBVR concepts such as 'it is permitted that' and 'exactly one'.

SBVR supports standard logical operations ('and', 'or', and so forth) and first order predicate logic (e.g. 'each', 'some'). SBVR also supports certain modal logic concepts such as necessity, possibility, obligation, and permission. Some example rules given in "Structured English" are:

It is permitted that each clerk *validates* each license application only if the current age *of* the license application *is greater than* 18.

It is obligatory that each applicant *pass* the written test.

Note the influence of the vocabulary design on the expression of the rules. The first example references the "the current age *of* the license application," rather than "…*of* the applicant," because "current age" is a field of the "license application" artifact. The vocabulary – and perhaps the underlying application – would have to be restructured to enable a more natural rule statement.

The project described here is creating a prototype tool to evaluate the technical issues involved in writing SBVR rules, and then transforming them to executable implementations.

**Prototype Design**

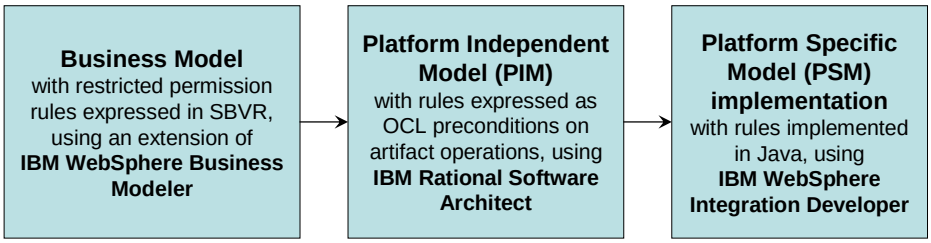


**Fig. 5.** Prototype Summary

The SBVR prototype is designed as an extension of the MDBT project, in order to build upon the existing MDBT modeling and transformation technology. As shown in figure 5, the rules are entered in a new tool added onto the WebSphere Business Modeler, and then transformed into a PIM-layer solution, and then further transformed into a PSM-level implementation.

**Specifying Rules in the Business Model**

The SBVR specification is large and fairly complex. Rather than attempt to support the entire specification, this prototype focuses on a limited subset called "restricted permission rules". These are rules expressed as permissions (someone or something *may* do something) associated with conditions. The first example rule given above is a restricted permission rule. In the prototype, all such rules are associated with an MDBT business model such as the one shown above. Each rule references a user role, an action, and a business artifact in the model.

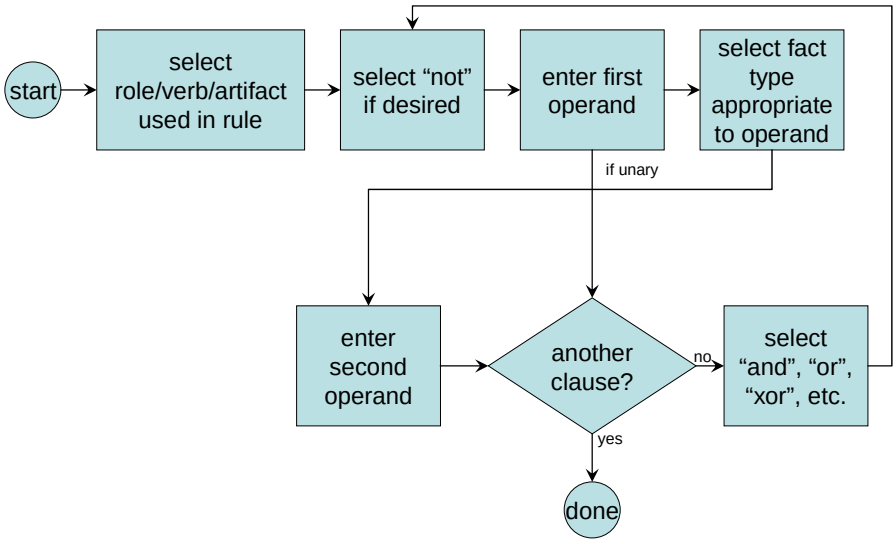


**Fig. 6.** Rule Wizard Navigation Path

Two methods of entering rules could be imagined. One involves a tool that parses "Structured English" text and attempts to discover the underling meaning. The problem with that approach is that all text – even "Structured English" – has ambiguities. Manual user involvement would be required to resolve those ambiguities.

This prototype employs an alternate approach in which rules are entered through a tool "wizard" that guides users, step-by-step, through the process of creating a complete rule. Figure 6 summarizes the wizard steps. Advantages of this approach are two-fold: (a) users can only enter valid rules; (b) the meaning of the rules is explicit.

**Transforming from Business Model to Platform Independent (PIM) Model**

Following the MDBT technology, the rules entered through the wizard are converted to a PIM-layer solution as part of the overall MDBT transformation. MDBT models the PIM layer using UML class, state, and use case diagrams. This prototype extends the class diagrams by converting the rules to pre-conditions on the class operations. These pre-conditions are expressed using the Object Constraint Language [17].

Advantages of OCL for this purpose include the fact that it is an established standard, the potential to convert to any appropriate PIM-layer implementation, and OCL's built-in collection operators. The latter facilitate SBVR's use of universal and existential logic. For example, a rule fragment such as "each line item *of* the order *is complete*" may be converted to an OCL fragment such as "lineItems→select(isComplete)".

**Transforming from Platform Independent Model to Platform Specific (PSM) Model**

The transformation from PIM-layer to PSM-layer potentially converts the pre-conditions to equivalent tests in various aspects of the implementation. These include code that enables or disables buttons in the user interface, guards on state machine transitions, and potentially access control statements in a language such as the eXtensible Access Control Markup Language (XACML) [12]. The first example rule given above might map to all of these. This illustrates the power of SBVR and the MDBT approach: a single rule given at the business layer potentially drives multiple aspects of the ultimate solution.

For simplicity, this prototype converts the OCL preconditions only to Java. The transformation is simple, except that collection operators must be generated as corresponding "for" loops. Mappings to implementations such as rule languages, XACML, script languages, and others are possible and relatively easy with the MDBT approach.

## Related Work

Since SBVR is quite new, relatively little work has been published about it. One announced commercial implementation and a research prototype of SBVR exist:

- RuleExpress is a commercial SBVR tool produced by a collaboration of Business Rule Solutions, LLC [1] and LibRt [11]. The website [16] says RuleExpress provides "Business-people capabilities for business rules … capture, expression, validation, verification, visualization, management, publication, audit."
- SBeaVer [3] is an open-source SBVR tool created by Maurizio De Tommasi and Pierpaolo Cira at the University of Lecce in Italy, in a project funded by the European Digital Business Ecosystem [4] project. This tool runs as a plugin for the Eclipse [5] tools platform, and provides for creation, editing, validation, verification, and export of both vocabulary and rules.

These tools enable entry and modification of rules and vocabulary using "Structured English". In contrast, the work described here focuses upon transformation of the rules to executable code.

## Summary and Outlook

The project experience so far is that entering rules in SBVR "Structured English" seems to be a useful adjunct of the existing Model-Driven Business Transformation (MDBT) technology. Conversion of the limited subset of SBVR supported by this prototype into OCL and Java is straightforward.

This prototype addresses a small portion of the concepts defined by SBVR. Features of particular interest for future work include:

- Synonyms to permit alternative terms and part of speech in rules. For example, a 'License Application' might also be named an 'Application'.
- Noun and fact type definitions to simplify the expression of certain rules. For example, rather than specifying rules for when a clerk may validate an application, one might define a 'valid application' according to a set of conditions.
- Additional modalities, such as necessities and obligations. The second example above gives an example of an obligation rule.

These three require very different kinds of technology. Synonyms are entirely a matter of tools function. Definitions and other modalities require either new transformations among modeling levels or new execution mechanisms such as inferencing.

SBVR brings together concepts from several distinct traditional academic subjects: philosophy (modal logics, taxonomies), linguistics (semantics, pragmatics), and mathematics (first order logic). The application of these concepts to computer science topics such as modeling, model transformation, Description Logics, and rules, offers

rich opportunities for scientific and technical progress. The expression of these concepts in "Structured English" promises to make rules practical and useful for everyday business solutions.

## References


1. Business Rule Solutions, LLC. See http://www.brsolutions.com/.
2. Chowdhary, P., Bhaskaran, K., Caswell, N. S., Chang, H., Chao, T., Chen, S.-K., Dikun, M., Lei, H., Jeng, J.-J., Kapoor, S., Lang, C. A., Mihaila, G., Stanoi, I., Zeng, L. "Model Driven Development for Business Performance Management." *IBM Systems Journal*, Volume 45, Number 3, Page 587 (2006). Available at http://www.research.ibm.com/journal/sj45-3.html
3. De Tommasi, Maurizio, Cira, Pierpaolo. SbeaVer Business Modeler Editor. Available at http://sbeaver.sourceforge.net.
4. Digital Business Ecosystem project, "an Internet-based software environment in which business applications can be developed and used". Available at http://www.digital-ecosystem.org/.
5. "Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software." Available at http://www.eclipse.org/.
6. IBM Rational Software Architect. See http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html.
7. IBM WebSphere Business Modeler. See http://www-306.ibm.com/software/integration/wbimodeler/.
8. IBM WebSphere Integration Developer. See http://www-306.ibm.com/software/integration/wid/.
9. IBM WebSphere Process Server. See http://www-306.ibm.com/software/integration/wps/.
10. Koehler, Jana; Hauser, Rainer; Kapoor, Shubir; Wu, Fred Y.; and Kumaran, Santhosh. *A Model-Driven Transformation Method*. In Proceedings of the Seventh International Conference on Enterprise Distributed Object Computing, pages 186--197. IEEE, September 2003. Available at http://doi.ieeecomputersociety.org/10.1109/EDOC.2003.1233848.
11. LibRT. See http://www.librt.com/.
12. Organization for the Advancement of Structured Information Standards (OASIS). *eXtensible Access Control Markup Language* (XACML). See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
13. Object Modeling Group (OMG). *MDA Guide*, version 1.01, 2003. Available at http://www.omg.org/docs/omg/03-06-01.pdf.
14. Object Modeling Group (OMG). *Production Rules Representation Revised Submission*, June 5, 2006.
15. Object Modeling Group (OMG). *Semantics of Business Vocabulary and Rules Specification Drafted Adopted Specfication*, March 2, 2006.
16. RuleExpress, "The business tool for expressing and communicating business rules." Available at http://www.rulexpress.com/index.php.
17. Warmer, Jos, Kleppe, Anneke. *The Object Constraint Language: Getting Your Models Ready for MDA*. second edition, Addison-Wesley Professional; 2003; ISBN 0321179366.