SBVR Use Cases

Mark H. Linehan

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

Abstract

Semantics of Business Vocabulary and Rules (SBVR) is a new standard from the OMG that combines aspects of ontologies and of rule systems. This paper summarizes SBVR, reviews some possible use cases for SBVR, and discusses ways that vocabularies and rules given in SBVR could relate to established ontology standards, to rules technologies, and to other IT implementation technologies. It also describes experience with an SBVR prototype that transforms a subset of SBVR rules types to several types of runtime implementations.

SBVR

The *Semantics of Business Vocabulary and Rules* (SBVR) (Object Modeling Group, 2007c) standard is very new, so it may be useful to summarize it for the benefit of readers who are unfamiliar with it. The standard specification document is very large at 422 pages, of which about half are normative and half are supporting and explanatory material. The length of the standard is due to the fact that it combines ideas from multiple topics, including modeling, ontologies, mathematics, philosophy, and linguistics. More accessible descriptions of SBVR are available at (BRC, 1997-2007), (Chapin, 2005), (Chapin and Hall, 2007), (Linehan, 2006), and (Linehan, 2007).

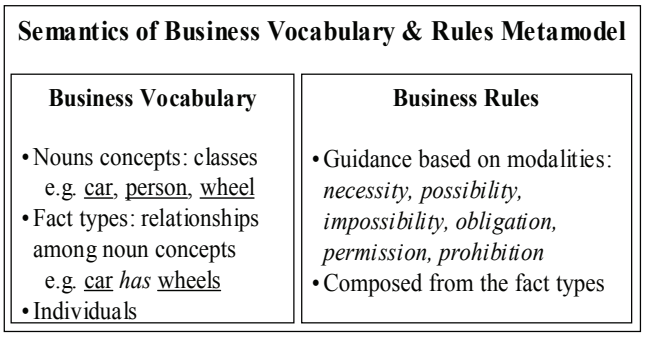

Figure 1: Major Features of SBVR Metamodel

This paper describes only those SBVR features that are most significant to the use cases described below, and to relating SBVR to other standards. Figure 1 summarizes some of the major features of the standard.

The SBVR standard defines a metamodel that formally captures aspects of business vocabularies and business rules. The term "business" is used in the Model-Driven Architecture sense of a business-focused, versus implementation-oriented, approach to the vocabulary and rules. In that sense, SBVR applies not only to commercial businesses such as finance and manufacturing, but also to other domains such as education, government, medicine, and law.

The SBVR metamodel gives formal semantics to these aspects, to enable consistent interpretation of the vocabulary and rules. The standard also defines an interchange format to enable exchange among tools.

In this paper, SBVR rule examples are given in "Structured English" (explained below), using several font styles:

nouns are underlined
verbs are given in italics
literal values and instance names are shown with double underlines
keywords are shown in bold font
uninterpreted text is shown in normal font style

Business vocabularies: SBVR business vocabularies define noun concepts (the equivalent of UML or OWL classes), fact types (the equivalent of UML associations or OWL relationships), individual instances of both noun concepts and fact types, as well as various specialized concepts such as categorizations and reference schemes. Noun concepts form class hierarchies via subtype relationships, thus providing the basis for subsumption reasoning.

SBVR fact types identify relationships among one or more roles, but every fact type has a fixed number of roles. Unary fact types, called "characteristics", capture aspects of a single instance, such as "order *is paid*". Binary fact types characterize relationships among two roles, such as "customer *places* order". Ternary and larger arity fact types are also possible. For example: "buyer *purchases* house *from* seller *through* broker" has four roles.

Attributive fact types capture mereological relationships, that is the relationship between parts and a whole. For example, "order *has* items" is the equivalent of specifying "items" as a property of an "order" class in UML.

Fact type roles identify the kinds of things that can participate in the corresponding relationships. For example, if a "firm order" is a kind of "order", then an instance of a "firm order" can play the role of "order" in an instance of the fact type "customer *places* order". Thus, SBVR utilizes a form of typed logic.

Vocabulary concepts may have definitions given as rules that describe derivations for the concepts. For example, the fact type "grandfather *of* person" might be defined as "a parent *of* a parent *of* the person". Such definitions formally specify the derivation of concepts from other concepts, and can support inferencing.

Concepts may also have constraints, given as structural rules. For example, the fact type "person *has* parent" may be constrained by a necessity rule that "**Each** person *has* **exactly** two parents". Constraints may give cardinalities, as in this example, or specify other structural aspects of concepts. For example: "**No** parent *of* **a** person *is* **a** child *of* **the** person".

Concepts may have synonyms, which enable the common phenomenon of multiple names for the same idea. For example, a vocabulary may specify client as a synonym for the noun concept customer, and "customer *orders* product" for the fact type "customer *purchases* product". Synonyms may be associated with individual human languages to provide a basis for multilingual support (discussed in more detail, below). Synonyms may also identify "subject fields" that specify the topics or contexts that employ the synonyms. For example, the medical profession has its own terms for many concepts.

By default, concepts are assumed to be open-world, but may be explicitly modeled as closed-world. For example, a business might specify the concept "employee" as closed-world because it always knows exactly who is and who is not an employee. The same business would probably model the concept "phone number *of* employee" as open world, since one can never exactly know all the phone numbers of all the employees.

Vocabulary entries may have supporting information such as examples, notes, and source references. These provide documentation for persons using the vocabularies.

To date, there are no standard "foundational vocabularies" for common concepts such as date and time, quantities, currencies, and units of measure. The need for such vocabularies is shown in Annex E.2.3 of the SBVR specification, which defines concepts about date and time. These concepts are necessary for a use case studied in detail in that Annex, and will be needed by many other business scenarios.

Tools may display SBVR vocabularies in a variety of textual or graphical formats such as variations of UML class diagrams. Many of the appendices of the SBVR specification describe formats favored by individual contributors to the document. The specification uses a textual format akin to a dictionary to define the concepts used by SBVR itself. Figure 2 gives an example.

Section 10.2 of the SBVR specification gives a partial mapping of SBVR concepts to OWL and to ISO Common Logic. In some cases, the mapping is one-to-one, but in many others, a single SBVR concept is represented as a composition of multiple OWL or Common Logic Constructs.

In (Linehan, 2007), this author argues that SBVR vocabularies meet the six criteria given by Atkinson (Atkinson, 2006) that qualify models as ontologies. Furthermore, I propose that SBVR vocabularies are *reference ontologies* using the terminology of (Guizzardi, 2006), meaning that their primary intended use is descriptive rather than execution. Despite that, some use cases (as described below) go beyond descriptive applications of SBVR vocabularies to employ them as the basis of implementations.

customer
Definition: one that purchases a commodity or service
Source: Merriam Webster Collegiate Dictionary

customer *submits* order *to* company
Definition: **the** customer *transmits* **the** order *to* **the** company

firm order
Definition: order **that** *is final*

letter of intent
Definition: order **that** *is* **not** *final*
Synonym: LOI
Note: A customer may submit an LOI to get a price quote, delivery schedule, or other terms of sale.

order
Definition: customer *request for* items

order *has* item
Necessity: **Each** order *has* **at least** one item.

Figure 2: Example Business Vocabulary

Business Rules: SBVR rules are distinguished from most other rule systems by the use of alethic and deontic *modalities* from the world of philosophy and logic (Halpin, 2006). The alethic modalities enable structural rules such as "**It is possible that an** order *has* **more than** one line item". The deontic modalities describe behavioral rules such as "**Each** order **must** *be paid within* 24 hours". The primary distinction between these is that structural rules define character-

istics of a model itself and thus cannot be violated, whereas behavioral rules specify expectations of humans or automated systems with the understanding that such expectations are not always met. (Linehan, 2007) gives more detail about the SBVR modalities.

SBVR models rules as predicate formulas, extended with modalities. Formulas may incorporate multiple modalities, but their meaning is formally defined only for formulas that have a single modality. Most SBVR rules employ first-order semantics, but second-order logic is possible, when restricted to Henkin semantics (OMG, 2007c, section 10.1.1.9). For example, one could say "**any** rule **that 'all** orders **must** *be paid within* 24 hours**'** *does* **not** *apply to* gold customers". Henkin semantics restricts the range of quantifications to a known set. For example, presuming that all the rules of business "xyz" are known, it would be legitimate to say "**if all** rules *of* business xyz *are compliant with* regulation abc ...", but not to say "**if all** rules *are compliant with* regulation abc ..." because the former limits the range of the condition, while the latter is open-ended.

Predicate logic formulas form the basis of SBVR rules. This means that well-established symbolic analysis techniques should be applicable to these rules. Potential benefits include rule validation and consistency checking to automatically recognize problems such as conflicting rules, and situations where one rule dominates another (i.e. a second rule never has effect because a first rule applies more generally).

The formal semantics of SBVR rules also create the possibility of rule simulation, particularly when combined with process models as in the MDBT project discussed below.

The SBVR specification is not concerned with issues of computational tractability, and indeed it seems likely that users might define vocabulary and rule combinations that imply impractical computations. The operational solution to this issue should be tools that warn users about such combinations, just as tools should warn about circular generalization hierarchies.

SBVR also supports "exception" rules, which are akin to defeasible logic. These are particularly important in authorization or access control rules. For example, one may have a default rule that says "**The** gate **may not** *be opened*", supplemented by a rule that states exceptions "**The** gate **may** *be opened* **only** *for* authorized users." Defining an "employee" as an "authorized user" in the business vocabulary would then mean that the gate may be opened for "employees".

In the SBVR specification, rules are given using "Structured English", a restricted form of the natural language. This is similar to techniques employed by (Fuchs and Schwitter, 1996), (Bernstein and Kaufmann, 2006), and (Sowa, 2006). The approach is related to "controlled language" methods used for several decades to support translation of reference manuals among natural languages (Altwarg, 2006). (Schwitter, 2005) describes a similar approach to using a "controlled natural language ... as an interface language to the Semantic Web".

At least three commercial implementations (KnowGravity, Inc., 2004-2007), (RuleArts, LLC, 2006), (Unisys, 2008) of SBVR exist already. In addition, there is an open source SBVR tool (SBeaVer, 2006), although work on it appears to have stopped for the past year.

Summary: The previous sections have described SBVR features for modeling business vocabulary and rules. The remainder of this paper describes some use cases for these features. The scenarios are organized in two main categories: Modeling Use Cases, and Transformation Use Cases. The final section below describes experience with a specific top-down transformation technology that uses SBVR.

Modeling Use Cases

One main use of SBVR is to precisely describe concepts and rules in a manner oriented to business users. The prime motivation is clarity in communication among business users and also between business and IT staff.

This section describes three use cases that address this need for clarity by drawing upon the descriptive capability of SBVR. These examples assume that vocabulary and rules are modeled by professionals who have the ability to think abstractly about these ideas, and who are supported by appropriate tools. These professionals may be called "business consultants" or "business analysts" or "business architects" or even "business engineers". Their output is vocabularies and rules expressed in "Structured English" or some other easy-to-understand form. The reason for using "Structured English" is to enable review and feedback about the vocabulary and rules by ordinary business people.

Modeling Business Vocabulary and Rules

Consider the contents of legal contracts, which are typically organized as "terms" and "conditions". The terms define and name concepts that are used throughout the contracts. The conditions specify constraints upon the terms (i.e. structural rules) or upon the behavior of the contract participants (i.e. behavioral rules). Both the terms and the conditions may be stated more or less precisely. Public laws and regulations typically have a similar structure.

Consider what must happen when a business intends to support a contract, a law, or a regulation. First, a business must understand the legal terms using its own concepts, and deal with any ambiguity. For example, a contract may broadly define a term such as "order". A business might

need to consider whether or not a "letter of intent" is an "order". Second, a business must interpret the legal conditions in terms of the business context. For example, a contractual condition might be that "all orders must *be paid within* 30 days". But is that "within 30 days of order receipt" or "within 30 days of order fulfillment" or within 30 days of some other point in the lifecycle of an order? SBVR is about clearly and precisely describing these details, when it matters to a business.

One output of SBVR-style modeling can be a formal business vocabulary, perhaps presented as a dictionary of nouns and relationships. Figure 2 gives an example that shows that "firm order" and "letter of intent" are both kinds of "order". A "letter of intent" is also called an "LOI". A customer can submit either kind of order. In all cases, an order must have at least one item. This kind of business dictionary can help enterprises reduce misunderstandings (e.g. by distinguishing "letter of intent" from "firm order") and clarify expectations (e.g. that an "order" always has at least one "item").

Rule 1: **Each** order **always** *has* **at least** one item**.**

Rule 2: **No** sales rep *of* **a** company **may** *attempt to dislodge* **a** firm order *of* **another** company**.**

Synonymous Statement: **A** sales rep *of* **a** company **must** *stop selling* **if a** customer *submits* **a** firm order *to* **another** company**.**

Description: To avoid claims of "unfair competition", a sales rep should abandon any selling activity once a customer has placed a firm order.

An order **may** *be shipped* **only if all the** items *of* **the** order *are in inventory***.**

Figure 3: Example Rules

Structural rules may be embedded in the vocabulary, as shown in Figure 2 with the necessity rule "**each** order *has* **at least** one item". Both behavioral and structural rules may also be presented in sets, as shown in Figure 3. The first of these rules simply restates the structural rule from Figure 2. The second rule is given in two forms: as a prohibition and as an obligation. The description adds an explanation.

SBVR enables considerable flexibility in the degree of modeling completeness. For example, vocabulary entries can have informal definitions (e.g. for "customer") that use terms not otherwise defined. Noun concepts can also be partially formal: one can say that a noun is a subtype of another noun while stating the distinguishing characteristics informally.

Nevertheless, modeling of any kind is labor-intensive, and hence expensive. Why would this level of detail matter to a business? Large companies, partnerships, and governments have tremendous problems with ambiguity, leading to confusion, inefficiency, loss of time, and sometimes contract or regulatory violations. Another risk is excessive or unpredictable information system development time and effort. The practical impact can be increased business costs, loss of business agility, or even jail for business executives. Those risks often motivate enterprises to make the investment needed to pin down their terms and explicate their business rules.

This author co-wrote a paper (Nayak et al. 2007) that proposes SBVR business rules as one component of a comprehensive business architecture modeling framework. The approach "... consider[s] five main *domains*: business value, structure, behavior, policy [rules], and performance" (ibid, p. 1). In this approach, the SBVR rules complement and extend the modeling of the other four domains. The rules add detail to business aspects such as service specification, process constraints, and information models.

Requirements Management

Requirements management is a well-known software engineering discipline involving gathering, articulating, and verifying business and user needs. Requirements typically are documented as a combination of use cases (e.g. as in UML) and text. One problem is that text in any human language is unavoidably ambiguous. Another problem is that most lists of requirements are incomplete.

The need for better requirements management has been identified by many sources. The United States Government Accountability Office reported in March 2006, (GAO, 2006, p. 4) that:

> For example, ill-defined or incomplete requirements have been identified by many experts as a root cause of system failure. As a case in point, we recently reported that the initial deployment of a new Army system intended to improve depot operations was still not meeting user needs, and the Army expected to invest $1 billion to fully deploy the system. One reason that users had not been provided with the intended systems capability was a breakdown in requirements management.

1. Training costs are US$300 per student per course.
2. All training material will be provided to student on first day of class.

Figure 4: Initial Requirements

Consider the two requirements listed in Figure 4. Some questions that might be asked about these requirements include:

- Is a “course” the same thing as a “class”?
- Who provides the training material?
- Who pays the training costs?

SBVR can help answer such questions by enabling formal definitions of the terms used in the requirements, and of the requirements themselves. Figure 5 shows how an SBVR vocabulary and rules could clarify these requirements. Specifying "course" as a synonym of "class" makes explicit that these terms mean the same thing. The first rule identifies who pays for each course. The second rule specifies that the "school" provides the "training material". Detailing the vocabulary and the rules in this manner eliminates many ambiguities.

Vocabulary

class
Synonym: course

class *has* training material
Synonymous Form: training material *for* class

student *enrolls in* course

student *pays* amount *for* course

Rules

Each student *enrolled in* a course **must** *pay* $300 *for* **the** course.

Each school **must** *provide* all training material *for* **a** class *to* **each** student *of* **the** class *on* **the** first day *of* **the** class.

Figure 5: Requirements Refined as SBVR Vocabulary and Rules

SBVR-style modeling can thus be seen as an extension of requirements management. When this degree of formality is desired, requirements given informally can be restated as rules that use formal business vocabularies. This should drive out ambiguity, thus making the requirements clearer and more precise.

Multi-Lingual Support

One key challenge for many businesses is operating in multilingual contexts. Governments and companies need to apply their rules consistently in multiple languages, whether within individual countries, such as Canada and Belgium, or in global operations.

SBVR provides the basis for multilingual capability by distinguishing names from concepts. A concept may have multiple names (synonyms), and the names may be identified with specific languages. For example, a vocabulary might include a concept that in English is called "order" and in French "ordre". Language-aware tools can then display the same rules in multiple controlled natural languages, such as "Structured English" and "Structured French".

Of course, across multiple languages, the semantics of words such as "order" and "ordre" may not exactly match. This problem has been addressed in methods and tools for translating maintenance manuals and other text between languages, for example as described in (Altwarg, 2006). Solutions include vocabularies for restricted domains where the terms can be defined precisely, and linguistic techniques such as morphological analysis and lexical disambiguation.

One multilingual scenario comes from a Japanese company that defines its rules in Japanese but wants to have them applied by customer support representatives in India. The company needs techniques to overcome the language barrier between Japanese and Hindu or English. SBVR promises one way to address such needs.

Tools Interchange

A fundamental feature of the SBVR specification is a MOF- and XMI-based document format for exchanging SBVR vocabularies and rules among SBVR tools. This enables potential collaboration among tools that address different user cases. Another goal that appeals to many commercial users is the potential avoidance of "vendor lock-in".

Mapping / Transformation Use Cases

Figure 6 illustrates how SBVR fits in the Computation Independent Modeling (CIM) or Business Modeling layer of the OMG's three-layer Model Driven Architecture (Miller and Mukerji, 2003). That is, SBVR describes business concepts and requirements without addressing their implementation. For example, an SBVR obligation rule specifies what a business must do, possibly under some condition, without saying how the business should do it.

SBVR complements another rules standardization effort at the OMG, called *Production Rules Representation* (PRR) (OMG, 2007b). The latter standardizes a cross-industry model for if-then rules as used in forward chaining and sequential execution. PRR is at the Platform Independent Modeling (PIM) layer because PRR rules define how rules should implement a solution in a vendor-independent manner.

Other activities at the PIM layer include the *RuleML Initiative* (Boley, Tabet, and Wagner, 2001) and the World Wide Web Consortium's *Rules Interchange Format* (RIF) (Welty and de Sainte Marie, 2006). Each of these take a

vendor-independent approach to some aspect of rules used in solutions. RuleML attempts to define a whole family of rule languages that share core concepts. RIF addresses exchange of rules among different rule systems.

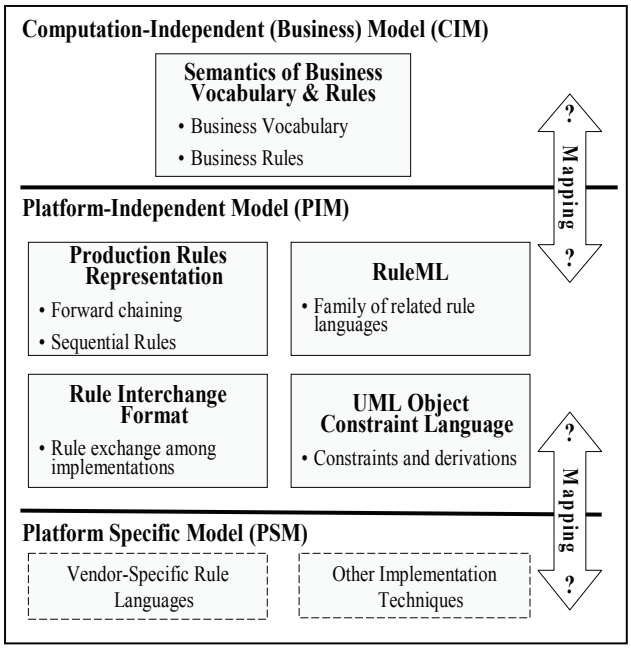

Figure 6: Rules and Model Driven Architecture

The Object Constraint Language (OCL) (Warmer and Kleppe 2003) of UML also fits at the PIM level. Although not often viewed as a "rule" language, OCL enables modeling of conditional constraints on UML operations, and also derivation of those operations. As discussed below, at least some SBVR rules may be mapped to OCL constraints.

A plethora of rule languages and engines exist both in academia and as commercial products. These fit at the Platform Specific Modeling (PSM) layer, in that there is no ability to exchange rules among implementations. Depending upon how liberally one defines the term "rule", one could extend the view of PSM-layer rules to include things like conditional statements in programming languages, referential integrity constraints in relational database systems, and guards in state machines.

As illustrated in Figure 6, transformations or mapping among these layers is an open issue. Mappings could occur between the CIM and PIM layers, and also between the PIM and PSM layers. The next few sections describe how these transformations may map from an upper layer to a lower layer, or the reverse. "Meet-in-the-middle" use cases also exist. Finally, there is a strong need for traceability of rules across these layers.

Top-down Mapping

The term "top-down" refers to a process of defining a business function at a very abstract level, and then successively refining the definition to add more detail. In the MDA methodology, an abstract representation of a business fits at the top (CIM or business) layer. When an automated solution is desired, refinement of a CIM business model should lead successively to a PIM-layer model, and then a PSM-layer implementation.

Requirements management, as discussed above, fits the business or CIM layer. Because they are more formal than traditional requirements statements, SBVR vocabulary and rules add rigor and detail to requirements management at this layer. That detail can be a step towards other kinds of formal modeling. For example, SBVR business vocabularies can provide some of the information one finds in UML class models, such as the names and relationships among classes. But vocabularies need not be fully specified to be useful for business modeling. And SBVR vocabulary entries do not address programming details such as integer versus floating point datatypes because these are not meaningful to businesses. Such information must be added at the lower modeling layers, when required.

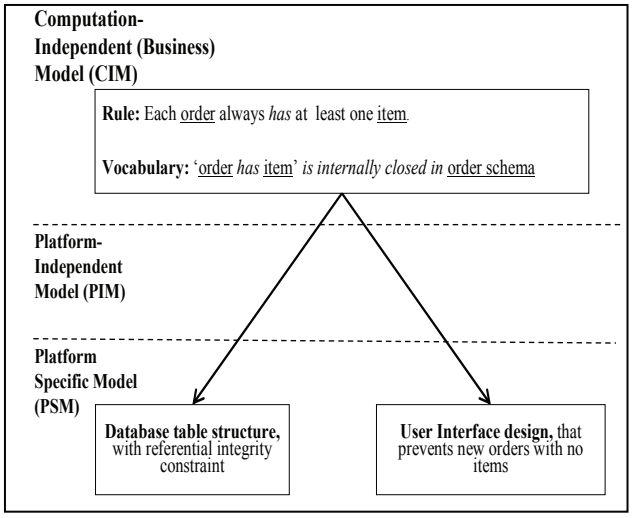

Figure 7: Example Top-Down Transformation

Top-down transformation of rules often produces a 1:*n* mapping, as illustrated in Figure 7. Business-layer rules map to multiple implementation artifacts. For example, the rule "**Each** order **always** *has* **at least one** item" implies a database table structure with a foreign-key reference among items and orders, and a user interface that supports multiple items per order. The closed-world assumption (via the entry that "'order *has* item' *is internally closed in* order schema"), implies that the database should have a referential integrity constraint specifying that each order must be referenced by at least one item, and that the user interface

should require the entry of at least one item for each new order.

Top-down rules transformation of rules may also generate *n*:*m* mappings. These occur when multiple business-layer rules affect a single implementation-layer design feature. For example, both Human Resource rules and national laws may affect security and privacy aspects of personnel systems.

Usually, top-down mapping targets the automatic creation of executable artifacts, but another valuable output can be test plans. Model-based test generation is an area of active study as exemplified in the A-Most workshop series (A-Most 2008). SBVR vocabularies and rules can add detail to use cases or other models, thus making automated tests, or test plans, more complete.

In (Nayak et al. 2007), this author and his colleagues argue that multiple modeling types, including vocabulary and rules, should be combined to enable comprehensive top-down transformations. For example, consider the rule "A clerk **may** *pack* **an** *order* **only if all the** items *of* **the** order *are on hand*." This rule is perfectly legitimate when taken as a standalone requirement. Coupling the rule with an order handling business process model can enable a more comprehensive transformation. Then the business rule can be understood as a constraint on one or more business process steps. The discussion of the "MDBT Top-Down Transformation", below, describes experiments with this kind of top-down transformation.

Figure 7 avoids the question of what kind of model is appropriate at the PIM layer when doing top-down transformation. The easy answer in the MDA context is some form of UML model. The MDBT toolkit that is described below produces a UML model as an intermediate step. The UML model describes a true platform-independent solution in the sense that it can be mapped to multiple different PSM level implementations. The UML model serves as an indirection layer, isolating the considerations of the business model from the practical details of any particular implementation.

In principle, one could perform top-down transformations directly from a business (CIM) model to a PSM model. Clearly, there is a tradeoff between having a single-step mapping and the two steps shown in figures 6 and 7. Further experience is needed to determine whether the benefits of having the intermediate PIM layer are worth the complexity of a second transformation step.

Bottom-up Mapping

Bottom-up mapping of rules is about abstracting rules from software code. This puts it in the space of "Architecture Driven Modernization" (ADM), an OMG task force targeted to analysis and "revitalization" of software.

The concept here is to scan and parse existing software, extract the if-then statements, somehow separate the conditionals that represent business rules from those representing the mechanics of the software, and then recover the original intent of the rules. Clearly this is a significant challenge, even when performed manually. Automating this process, beyond the basics of "mining" the software statements, is very difficult. However, the cost of doing the work manually is so high that even limited automation can have value.

The ADM task force has produced a *Knowledge Discovery Metamodel* (KDM) (OMG, 2007a) intended as the basis for such technologies. KDM incorporates a multi-layered structure compatible with MDA ideas, but intended for bottom-up software analysis. The most abstract KDM layer – called the "Abstractions Layer" – explicitly claims alignment with SBVR in order to enable mapping of KDM elements to SBVR rules.

(Putrycz and Kark, 2007) describe a tool that partially automates bottom-up mapping of rules. The tool combines ontological techniques, source code scanning, and human input, to reconstruct business-layer rules. The technique holds promise as a way to reduce, but not eliminate, human effort in bottom-up mapping.

The kind of situation shown in Figure 7 makes the bottom-up challenge even greater. In addition to analyzing multiple kinds of software implementations – database table configurations and UI software in this example – the abstraction process needs to notice the common aspects. In this case, the fact that the database expects an item for each order, and the UI requires an item before an order can be entered, should be related to the single common business rule shown in the figure. This author is not aware of any automated technologies to accomplish that.

Meet-in-the-Middle

One of the key questions business leaders ask is how well their operations implement their business policies and rules. Public companies are required to attest to this in formal documents, signed by corporate executives. In the United States, fraudulent attestations can constitute crime and can lead to jail time for the executives. This generates considerable business for corporate auditors.

One way to think about this issue is to imagine a scenario where business rules are formally modeled using SBVR, while business operations are analyzed using bottom-up modeling using something like KDM. Then one could imagine tools and techniques that permit one to compare the expected rules (modeled in SBVR) with the operations (captured in KDM).

This is a fantasy today, since we have fairly limited bottom-up modeling capabilities. If that problem could be solved, then meet-in-the-middle rules comparisons would

be very attractive both as a way to avoid executive jail time, and a means to verify software implementations.

Traceability

Traceability is the maintenance of book-keeping records that link business-layer rules to implementations. These records enable a business to answer these two questions:

1. What is the operational impact of a change to a particular corporate rule?
2. If we change a particular implementation detail, are we still consistent with relevant corporate rules?

Answering these questions is key to regulatory requirement compliance. Automatically maintaining the book-keeping records needed to show traceability has the potential to reduce the risk of non-compliance and the cost of demonstrating compliance.

Traceability may also enable rule explanations (error messages or help text) such as "you can't do that because it would break the corporate rule that ...." The concept is that the "Structured English" used to display rules to rule reviewers can be the basis for providing such explanations to persons who attempt to violate the rules.

MDBT Top-Down Transformation

MDBT – Model Driven Business Transformation – is a project at the IBM TJ Watson Research Center to do automated top-down transformation of business-layer models into executable implementations. (Kumaran, 2004) describes MDBT in some detail. (Linehan, 2006) and (Linehan, 2007) describe an extension of MDBT that adds limited SBVR rules modeling. The work is summarized here as an example of top-down business transformation.

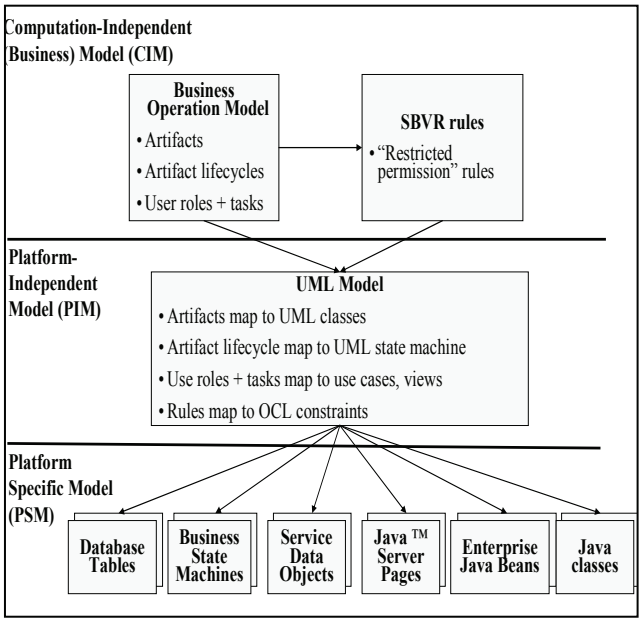

Figure 8: MDBT Transformation

Figure 8 illustrates the MDBT top-down transformation. Artifacts are the business documents that record real things or actions. For example, an "order" artifact might represent a customer making an order. At the business layer, MDBT models the structure of artifacts and their lifecycle. For example, an order may start out as initialized, then be complete, paid, shipped, received, returned, and so forth.
The (limited) SBVR prototype takes the artifacts as business vocabulary, and models "restricted permission" rules. These rules specify constraints on the actions of the user roles according to conditions that test the attributes of the artifacts. The example given above, "A clerk may *pack* an *order* only if all the items of the order are *on hand*" is a restricted permission rule. It constrains the clerk to *pack* an order only when the condition is true.

The MDBT transformation converts the combination of the Business Operation Model and the rules to a UML model. The artifacts are converted to UML classes, and the artifact lifecycles become UML state machines. The transitions in these state machines invoke operations on the classes. For example "*pack*" becomes a UML operation on the "order" class. The rules become OCL pre-constraints on the corresponding operations. The example rule converts to an OCL constraint that prevents the clerk user role from performing the "*pack*" operation if the condition is not true. Figure 9 shows the OCL equivalent of the example rule. Note that the reference to "the clerk" in the original rule is omitted (as here) if the state machine model permits only the "clerk" role to perform the "pack" action on "orders".

```
Context: order:: pack()
pre:     self.items --> forAll (i |
         i.onHand)
```

Figure 9: Example OCL Created from Rules

The MDBT transformation further converts the UML PIM-layer model to a variety of PSM-layer components as shown in Figure 8. The UML state machines become runtime state machines, using the Business State Machine functions of IBM WebSphere Process Server (IBM WebSphere). The use cases become user interface pages implemented as Java™ Server Pages (JSPs). Transitions in the state machine may be triggered by action buttons in the user interface or by web service requests.

The OCL constraints are implemented as Java methods that test the corresponding conditions. The consequent parts of the rules are implemented in two ways:

1. As guards in the state machines, to prevent the actions from succeeding if the conditions are not met.
2. As user interface functions that enable or disable the corresponding action buttons.

Thus, each business-layer rule automatically constrains both the user interface and the business logic.

Experience with this prototype clearly shows that business-layer rules nicely complement and extend the existing MDBT technology. They augment business information and business process modeling as specifications of a business. The rules transformation works smoothly and produces the expected results. The main problem is the very limited subset of SBVR that is supported so far. I hope to solve that problem over time.

Conclusions

This paper has outlined multiple use cases for business vocabulary and rules as conceived in SBVR. The common theme among these scenarios is the use of SBVR to "raise the abstraction level" of business solutions. Instead of describing rules in terms of implementation concepts such as if-then statements, SBVR captures business requirements – what a business wants. This continues the long-standing Computer Science tradition of searching for more abstract ways to understand technical and business issues.

SBVR makes these scenarios possible through a combination of predicate logic, modal logic, ontologies, an XMI- and URL-based interchange format, and the potential to express rules and vocabulary using diagrams or "Structured English". These elements have previously been employed by various other rule systems. The unique contribution of SBVR is their combination into an integrated specification.

The practical question is whether practitioners will find that the value obtained from a higher degree of abstraction is worth the effort involved in formulating such abstractions. Significant tools development effort will be needed to maximize the value and minimize the effort of SBVR-style modeling. This author believes that the eventual result will be very worthwhile for large enterprises.

References

A-Most. 2008. 4th Workshop on Advances in Model Based Testing. See http://kimba.mat.ucm.es/AMOST08/.

ADM. Architecture Driven Modernization task force at the Object Management Group (OMG). See http://adm.omg.org/.

Altwarg, R. 2006. Controlled Languages, an Introduction. Centre for Language Technology website on Controlled Natural Languages, Macquarie University. Available at http://www.shlrc.mq.edu.au/masters/students/raltwarg/clwhatisa.htm.

Bernstein, A.; and Kaufmann, E. 2006. GINO - A Guided Input Natural Language Editor. In 5th International Semantic Web Conference, ISWC 2006. Athens, GA: Springer Lecture Notes in Computer Science, Vol. 4273. Available at http://iswc2006.semanticweb.org/items/Bernstein2006tg.pdf.

Boley, H., Tabet, S., and Wagner, G. 2001. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In Proceedings of the Semantic Web Working Symposium (SWWS '01). Stanford: 381-401. See http://www.ruleml.org/.

Business Rules Community, 1997-2007. Website at http://www.brcommunity.com/.

Chapin, D. 2005, Semantics of Business Vocabulary & Business Rules. Presented at the W3C Workshop on Rule Languages for Interoperability. Available at http://www.w3.org/2004/12/rules-ws/slides/donaldchapin.pdf and http://www.w3.org/2004/12/rules-ws/paper/85/.

Chapin, D. and Hall, J. 2007. Developing Business Models with SBVR and the BMM. Presented at the 6th European Business Rules Conference. Dusseldorf. Available at http://www.eurobizrules.org/Uploads/Files/Chapin_20tutorial.pdf.

Fuchs, N. E.; Schwitter, R. Attempto Controlled English (ACE). CLAW 96, First International Workshop on Controlled Language Applications, University of Leuven, Belgium. See also http://www.ifi.unizh.ch/attempto/description/index.

GAO. 2006. Financial Management Systems – Additional Efforts Needed to Address Key Causes of Modernization Failures. United States Government Accountability Office Report to Congressional Requesters. GAO-06-184. Available at http://www.gao.gov/cgi-bin/getrpt?GAO-06-184.

Guizzardi, G. 2007. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV. Amsterdam: IOS Press. ISBN 978-1-58603-640-8. Available at

http://www.loa-cnr.it/Guizzardi/FAIA.pdf.

Halpin, T. 2006. Business Rule Modality. Presented at EMMSAD'06, the Eleventh International Workshop on Exploring Modeling Methods in Systems Analysis and Design. Luxembourg, 2006. Available at http://www.orm.net/pdf/RuleModality.pdf and http://emmsad06.idi.ntnu.no/EMMSAD06_p3-halpin.pdf.

IBM WebSphere. IBM WebSphere Process Server. See http://www-306.ibm.com/software/integration/wps/.

KnowGravity, Inc. 2004-2007, KnowEnterprise® enterprise modeling tool. See http://www.knowgravity.com/eng/value/knowEnterprise.htm.

Kumaran, S. 2004. Model Driven Enterprise. In Proceedings of Global Integration Summit 2004, Banff, Canada.

Linehan, M. 2006. Semantics in Model-Driven Business Design. Presented at SWPW'06, the 2nd International Semantic Web Policy Workshop. Available at http://www.l3s.de/~olmedilla/events/2006/SWPW06/programme/paper_02.pdf.

Linehan, M. 2007, Ontologies and Rules in Business Models. Presented at VORTE 2007, the 3rd International Workshop on Vocabularies, Ontologies and Rules for The Enterprise.

Miller, J. and Mukerji, J. eds. 2003, MDA Guide Version 1.0.1. Published by the Object Modeling Group at http://www.omg.org/docs/omg/03-06-01.pdf.

Nayak, N., et. al. 2007, Core Business Architecture for a Service-oriented Enterprise. *IBM Systems Journal*, vol. 46, no. 2. Available at http://www.research.ibm.com/journal/sj/464/nayak.pdf.

Object Modeling Group (OMG). 2007a. Knowledge Discovery Metamodel, Draft Adopted Specification. Available at http://www.omg.org/cgi-bin/doc?ptc/2007-03-15.

Object Modeling Group (OMG). 2007b. Production Rules Representation, Draft Adopted Specification. Available at http://www.omg.org/cgi-bin/doc?bmi/07-08-01.

Object Modeling Group (OMG). 2007c. Semantics of Business Vocabulary and Business Rules Specification, Version 1.0. Available at http://www.omg.org/spec/SBVR/1.0/.

Paschke, Adrian. 2006. RBSLA: Rule-based Service Level Agreements. See http://ibis.in.tum.de/projects/rbsla/index.php.

Putrycz, E. and Kark, A. 2007. Recovering Business Rules from Legacy Source Code for System Modernization. In Advances in Rule Interchange and Applications, Proceedings of the RuleML International Symposium. Orlando: Springer Lecture Notes in Computer Science. Vol. 4824. ISBN 978-3-540-75974-4. Slide presentation at http://2007.ruleml.org/docs/erik%20putrycz%20-%20RuleML%202007.pdf.

RuleArts, LLC. 2005-2008, RuleXpress rule creation, visualisation tool & repository. See http://www.rulearts.com/.

SBeaVer. 2006. See http://sbeaver.sourceforge.net/index.php.

Schwitter, R. 2005. Controlled Natural Language as Interface Language to the Semantic Web. Presented at IICAI-05, the 2nd Indian International Conference on Artificial Intelligence. Available at http://www.ics.mq.edu.au/~rolfs/papers/IICAI-schwitter-2005.pdf.

Sowa, J. Common Logic Controlled English. Available at http://www.jfsowa.com/clce/specs.htm.

Unisys, 2008. Unisys Rules Modeler. See http://www.unisys.com/products/software/business__process__software/rules__management/.

Warmer, J. and Kleppe, A. 2003. *The Object Constraint Language: Getting Your Models Ready for MDA*. 2nd edition, Boston, Mass: Addison-Wesley Professional. ISBN 0321179366.

Welty, C. and de Sainte Marie, C. 2006. *Rule Interchange Format* (RIF). World Wide Web Consortium working group report. See http://www.w3.org/2005/rules/.

White, S. 2005. An Introduction to BPMN. Published by the Object Management Group / Business Process Management Initiative at http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf.