

Verbalising R2ML rules into SBVR

Oana Nicolae and Gerd Wagner
Department of Internet Technology
Institute of Informatics
Brandenburg Technical University at Cottbus, Germany
{nicolae, G.Wagner}@tu-cottbus.de

Abstract

Nowadays, business rules receive a lot of attention from both industry and academia, as they are considered the ideal vehicle for capturing business logic. The purpose of our paper is to raise the level of business logic abstraction with the help of existing, mature enough, R2ML language, in order to obtain higher semantic representations of rules and their basic constructs i.e. SBVR. Our paper focuses on the informal translation from R2ML derivation and integrity rules, into SBVR Structured English in order to provide means for business rules validation and authorisation. SBVR Semantic Formulations of R2ML rules and their basic atoms are also described and exemplified, as they are the unique construct of SBVR Standard, that captures and structures the meaning of rules to enable further translations into executable rule languages/engines, following MDA abstract levels of rule modelling.

1. Introduction

The concept of business rules became so common in the actual literature, as its usage tends to permeate different communities: software architects community i.e. UML[6] modellers, ontology architects, business processes modellers.

Due to the large amount of rule languages/engines that exist on the business market, the problem of rules inter-operability had emerged. From the perspective of Model Driven Architecture¹ (MDA), rules can be considered at three different abstraction levels i.e. CIM², PIM³, PSM⁴, for refining the conceptualisation process

of any business application.

Business rules are for business people and must be specified in a declarative way, as close as possible to natural language i.e. *EU-Rent reviews each corporate account at EU-Rent Headquarters*⁵. Attempto Controlled English (ACE) ([7], [2]), Structured English, RuleSpeak English or SBVR [5] are natural language styles, easily understood by business people and suitable for defining business models. These languages are situated at MDA's CIM level of abstraction and are appropriate for capturing business rules together with their vocabularies.

The main purpose of a markup approach is to provide means for reusing, and publication and interchanging of rules between different systems and tools. Actually, they also play an important role in facilitating business-to-customer (B2C) and business-to-business (B2B) interactions over the Internet. Moreover, an interchange approach always supposes less transformations than PSM-to-PSM translations.

The main standardisation communities, OMG⁶ and W3C⁷ focus their work on providing business rules specification languages for all MDA layers of models in order to obtain rules inter-operability. Their standards are not sustained by most of business rules management system tools, as they implement proprietary rule languages. The reasons for this situation imply the existence of only a few interchange works in the academia i.e. RIF [1] language still has no well defined guidelines of how to implement the transformations and it also do not specify how to test the correction of the translation.

In this context, EU network of Excellence REVERSE⁸ developed an XML-based, general rule markup language i.e. R2ML [8]. R2ML brought together all the best characteristics from OCL, SWRL and RuleML

¹MDA - <http://www.omg.org/mda/>

²CIM - Computational Independent Model

³PIM - Platform Independent Model

⁴PSM - Platform Specific Model

⁵EU-Rent - <http://www.eurobizrules.org/eurentcs/eurent.htm>

⁶OMG - <http://www.omg.org>

⁷W3C - <http://www.w3.org>

⁸REVERSE - <http://reverse.net/>

languages and refines them with rule categorisation: integrity, derivation, production and ECA rules. R2ML complies Web naming concepts like (URI and namespaces), datatype concepts of RDF⁹ and the ontological distinction between objects and data. R2ML is a mature enough, rule interchange language, already tested in practise by means of translators¹⁰. We choose R2ML, as it do not restricts its use only for markup purposes: it can be used for modelling the Semantic Web services and it can give complete XML-based specification of a software agent. It is also an evolving language i.e. actual version is 0.5. Due to these reasons, we conclude that R2ML is actually a suitable rule interchange language to use in our mapping.

At the CIM level of abstraction, business rules are generally expressed in almost natural language e.g. SBVR, although some rules are at times graphically illustrated (see URML¹¹ and Strelka¹² tool). We choose SBVR, as it is the most suitable language to capture business vocabulary and rules in business natural language.

Our contribution on the field is to provide, in the context of existing rule-based PSMs to R2ML translators, a way to abstract executable rule languages at the CIM level using R2ML as interchange language. One of the main benefits of our work is providing means for the *validation and authoring of executable rule languages that serialise to R2ML*. We mention here the commercial implementation i.e. RuleExpress¹³ and the research prototype SBeaVeR¹⁴, the Business Modeler Editor implemented by Tommasi et al. The tool is a plugin implementation for the Eclipse platform, and provides creation, editing, validation, verification, and export of both vocabulary and rules using SBVR Structured English style.

Our work is to be considered also in the context of existing URML visual rule modelling language, implemented by Strelka tool. Strelka allows visual rule modelling on top of UML vocabularies and generates R2ML markup by representing rules as formal statements, expressed in some formalism or computational paradigm, which can be directly mapped to executable statements of a software system. The generated R2ML derivation rules will be further translated into SBVR derivation rules.

⁹RDF - <http://www.w3.org/RDF/>

¹⁰R2ML Translators - <http://oxygen.informatik.tu-cottbus.de/reverse-il/?q=node/15>

¹¹UML based Rule Modelling Language (URML) - <http://reverse.net/I1/oxygen.informatik.tu-cottbus.de/reverse-il/@q=urml.htm>

¹²Strelka - <http://oxygen.informatik.tu-cottbus.de/reverse-il/?q=Strelka>

¹³RuleExpress - <http://www.ruleexpress.com/index.php>

¹⁴SBeaVeR - <http://sbeaver.sourceforge.net>

The translation focuses only on R2ML integrity and derivation rules, basic atoms and main terms. The mapping is challenging, as it refers an informal translation from an XML-based language into a controlled, plain, English one i.e SBVR Structured English. The paper also provide SBVR Semantic Formulations for R2ML rules and basic atoms, in order to prove SBVR capacity in capturing enough information for further representation of business rules in PIM and PSM rule languages.

The remainder of this paper is organised as follows: Section 2 provides a review on the SBVR standard. Section 3 discusses the approaches of other research works and the differences from our works. Section 4 describes the SBVR verbalization of R2ML derivation and integrity rules and the last section discusses conclusions and future works.

2. Briefly about SBVR Standard

Even adopted in 2006 as a standard, OMG body published SBVR as formal Specification only recently, in January 2008. SBVR is presented as the first standard from OMG stack that explicitly provides a model of formal logic. It also claims to have a well defined semantics based on a MOF model that captures the meaning of vocabulary and rules that is also used for XMI-based interchange format.

SBVR emerged from ORM/NIAM and from ISO terminology work (particularly terminology science standards: ISO 704 and ISO 1087-1), therefore the fact-orientation together with the concept of **fact type** come from ORM/NIAM. Besides the MOF meta-model, which is not well presented in the specification (i.e. a bad splitting of UML class diagrams, therefore the main concepts loose their complete representation). SBVR Specification provides its own vocabulary (see pp. 45 from SBVR Specification) intended to describe the semantic structures of conceptual schema i.e. business communication of concepts, facts, and rules. In other words, SBVR is defined in terms of itself. SBVR do not involve rules enactment, therefore it does not address the need of an inference engine. Being classified as belonging to the CIM level, it focuses on structuring the meaning independently of any possibilities for automating business rules completely or partially i.e. SBVR Semantic Formulations.

SBVR Specification do not impose any normative or mandatory SBVR notation. It can be: MOF/XMI compliant XML, SBVR Structured English, a graphical model such as UML Profile for SBVR (see SBVR specification Annex H), ORM, CogNIAM or proprietary language, e.g. RuleSpeak. Our translation work uses

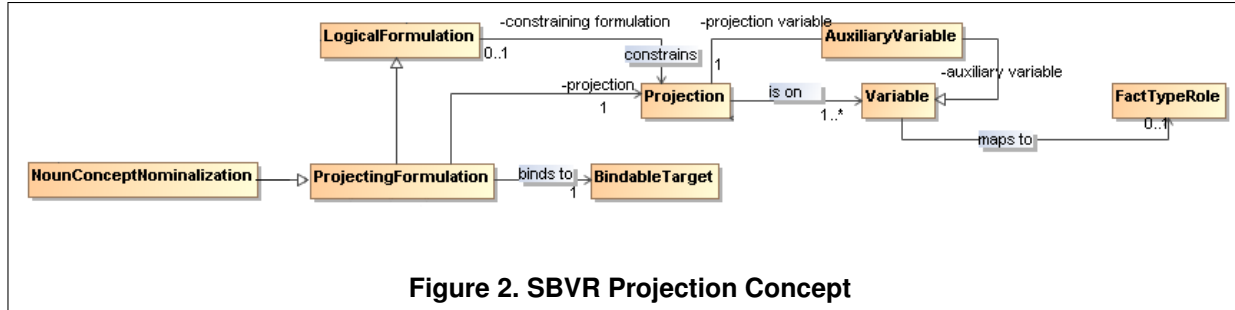


Figure 2. SBVR Projection Concept

- A way of structuring the meaning using: definitions, rules that govern the operation of an organization and questions (i.e. queries)
- Recursive
- Not intended for business people in general, but rather for capturing enough information in order to bridge SBVR to executable rule-based platforms
- Hierarchical indentation structure
- Many semantic formulations are possible, but there is one unique meaning for a statement

3. Related works on the field

Tommasi et al. (University of Lecce, Italy) began in 2007 a challenging *SBVR to Drools*¹⁵ translation. In order to accomplish the purpose, two assumptions were taken into consideration. First, the creation of a so called SBVR2DRL translation tool that can be embedded into SBeaVeR plugin for Eclipse, allowing the generation of Java code starting from the vocabulary and DRL rules from the ruleset.

Second, the implementation of an open-source Java library that allows to convert SBVR Structured English business rules into DRL rules. Moreover, the library could be used by SBeaVeR to provide new DRL generation/export functionality as well as by Drools to import SBVR business rules.

The work was stopped, as the translation implied the translation of integrity/derivation rules into Drools production rules, that can not distinguish between necessity and obligation modalities.

The literature had also treated the mapping from UML conceptual schema to SBVR, OCL rules (integrity-alethic rules for defining with the purpose of structuring and derivation rules) to SBVR semantic formulations i.e. [3], [4]. They try to provide a bridge

¹⁵Drools - <http://www.jboss.org/drools/documentation.html>

between software architects and business modelers. We try to offer this alternative also for the Semantic Web.

An R2ML rule set refers to a vocabulary¹⁶ which can be R2ML own vocabulary¹⁷ or an external one, such as: UML, RDF(S) and OWL. R2ML own vocabulary represents a serialisation of an UML fragment of class diagrams. Taking into account the paper space limits and the fact that this subject (i.e. UML/OCL to SBVR) was previously treated in the literature, we will focus on the following Section only on the mapping from R2ML to SBVR rules.

4. R2ML to SBVR

This Section describes the general mapping principles from R2ML rules (REVERSE I1 Rule Markup Language) serialisation to SBVR Structured English and Semantic Formulations. In this context, we focus on capturing the main patterns for translating R2ML integrity rules (alethic/deontic) and R2ML derivation rules.

4.1. Mapping R2ML Derivation rules

R2ML Derivation rules have an identifier, conditions and conclusions. Their semantics is: *the conclusions are to be derived every time the condition part holds*. In R2ML language the conditions of a derivation rule comprise `AndOrNafNegFormulae` i.e. a conjunction of `r2ml:qf.AndOrNafNegFormula`. `AndOrNafNegFormula` represents *quantifier-free* (i.e. `qf`) logical formula with conjunction, disjunction and negation. Conclusions are restricted to quantifier-free, disjunctive normal forms without NAF (i.e. weak negation). The language elements can be either specialisations of `r2ml:Atom` or the `r2ml:qf.LiteralConjunction` container that implies a conjunction of R2ML atoms. The concrete syntax of a R2ML derivation rule example implies the following XML representation:

¹⁶R2ML Examples - <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/17>

¹⁷R2MLV="http://www.reverse.net/I1/2006/R2ML/R2MLV"

```

1.<r2ml:DerivationRule r2ml:ruleID="InvalidDriver">
2. <r2ml:conditions>
3. <r2ml:ObjectClassificationAtom r2ml:class="Driver">
4. <r2ml:ObjectVariable r2ml:name="driver"/>
5. </r2ml:ObjectClassificationAtom>
6. <r2ml:DatatypePredicateAtom
7. r2ml:datatypePredicate="swrlb:lessThan">
8. <r2ml:dataArguments>
9. <r2ml:AttributeFunctionTerm r2ml:attribute="age">
10. <r2ml:contextArgument>
11. <r2ml:ObjectVariable r2ml:name="driver"/>
12. </r2ml:contextArgument>
13. </r2ml:AttributeFunctionTerm>
14. <r2ml:TypedLiteral r2ml:lexicalValue="16"
15. r2ml:datatype="xs:integer"/>
16. </r2ml:dataArguments>
17. </r2ml:DatatypePredicateAtom>
18. </r2ml:conditions> <r2ml:conclusion>
19. <r2ml:ObjectClassificationAtom r2ml:class="InvalidDriver">
20. <r2ml:ObjectVariable r2ml:name="driver"/>
21. </r2ml:ObjectClassificationAtom>
22. </r2ml:conclusion> </r2ml:DerivationRule>

```

We formalise a general pattern for translating an R2ML Derivation rule into an SBVR *projecting formulation*. We introduce *projecting formulations* as logical formulations that have a *projection* and binds to a *bindable target (variable, individual concepts and expressions)*. The SBVR *projection* is a semantic formulation that introduces one or more variables corresponding to actualities (i.e. facts) and that is possibly constrained by a logical formulation representing the behavior (i.e. condition part) of the derivation rule i.e. Figure 2.

In SBVR, Derivation rules indicate how the population of a fact type may be derived from the populations of one or more fact types or how a type of individual may be defined in terms of other types of individuals and fact types. Our derivation rule example comply with the second case, as we define the concept of *InvalidDriver* as a specialisation concept of the *Driver* concept and the fact type i.e. characteristic driver is of age.

Below, we represent the above derivation rule in SBVR Structured English notation style:

If *driver age is less than 16*, then *driver is invalid driver*.

As the verbalization of R2ML basic atoms will be described in the following Subsection, we only present here one of the possible SBVR Semantic Formulation for a R2ML rule pattern. In this case, for R2ML derivation rule, we have i.e.

```

The definition is formalised by a projection
. The projection is on a first variable
. . The first variable ranges over the concept 'driver'
. . The first variable maps to the one role of the characteristic
. The projection is constrained by a first universal quantification
. . The first universal quantification introduces a second variable
. . . The second variable ranges over the concept 'age'

```

4.2. R2ML Integrity rules

Integrity rules in R2ML, also known as integrity constraints, consist of a constraint assertion, which is a sentence in a logical language such as FOL or OCL.

R2ML framework supports two kinds of integrity rules: alethic and deontic integrity rules. An alethic integrity rule can be expressed with a phrase, such as *it is necessarily that*, whereas a deontic one can be expressed with phrases, such as *it is obligatory that* or using the negations of the above mentioned constructs, expressing the *non-necessity* and *non-obligation* constraints i.e. the `r2ml:isNegated='true'` attribute.

A constraint assertion is a logical sentence that must necessarily, or that should, hold in all evolving states and state transition histories of the discrete dynamic system to which it applies. An integrity rule cannot have free variables, i.e. all variables from this formula are quantified. An example of an R2ML alethic integrity rule i.e.

```

1.<r2ml:AlethicIntegrityRule r2ml:ruleID="LocalAreaConstr">
2. <r2ml:constraint>
3. <r2ml:UniversallyQuantifiedFormula>
4. <r2ml:ObjectVariable r2ml:name="country"
5. r2ml:class="OperatingCountry"/>
6. <r2ml:Implication><r2ml:antecedent>
7. <r2ml:AtLeastAndAtMostQuantifiedFormula>
8. <r2ml>DataClassificationAtom r2ml:datatype="xs:double">
9. <r2ml>DataVariable r2ml:name="localArea"/>
10. </r2ml>DataClassificationAtom>
11. </r2ml:AtLeastAndAtMostQuantifiedFormula>
12. </r2ml:antecedent><r2ml:consequent>
13. <r2ml:AttributionAtom r2ml:attribute="localAreaForCountry">
14. <r2ml:subject><r2ml:ObjectVariable r2ml:name="country"/>
15. </r2ml:subject>
16. <r2ml:dataValue> <r2ml>DataVariable r2ml:name="localArea"/>
17. </r2ml:dataValue></r2ml:AttributionAtom>
18. </r2ml:consequent></r2ml:Implication>
19. </r2ml:UniversallyQuantifiedFormula>
20. </r2ml:constraint></r2ml:AlethicIntegrityRule>

```

SBVR imposes a peculiar perspective on its integrity constraints. For example, alethic rules or Definitional /Structural Business Rules use alethic logic operators and they specify what the organization takes things to be. These constraints cannot be broken, as they are *true by definition* e.g. *Local area is in exactly one operating country*. The vocabulary of such a rule reveals the characteristics of noun concepts and the constraints on fact types. SBVR refines its alethic rules with modality operators expressing: necessity, non-necessity, possibility, impossibility and contingency (i.e. possible, but not necessary) i.e.

```

The rule is a proposition meant by a necessity formulation
. That necessity formulation embeds an universal quantification
. . The universal quantification introduces a first variable
. . . The first variable ranges over the concept 'operating country'

```

An SBVR Deontic rule, also known as *behavioral rule* or *operative rule*, uses deontic logic operators and govern what the organisation does i.e. what actions it takes.

SBVR Deontic rules are intended for people: actionable, but not necessarily automable, can be broken and need an enforcement regime. The enforcement regime is out of the scope of SBVR specification. SBVR Deontic modality operators express: obligation, non-obligation, permission, prohibition and optionality.

```

1.<r2ml:DeonticIntegrityRule r2ml:ruleID="ReturnWayRental">
2.<r2ml:constraint> <r2ml:UniversallyQuantifiedFormula>

```

```

3. <r2ml:ObjectVariable r2ml:name="rental" r2ml:class="Rental"/>
4. <r2ml:Implication><r2ml:antecedent><r2ml:Conjunction>
5.   <r2ml:ObjectClassificationAtom r2ml:class="Rental">
6.     <r2ml:ObjectVariable r2ml:name="rental"/>
7.   </r2ml:ObjectClassificationAtom>
8.   <r2ml:ObjectClassificationAtom r2ml:class="OneWayRental">
9.     r2ml:isNegated="true"
10.   </r2ml:ObjectClassificationAtom>
11.   <r2ml:ObjectVariable r2ml:name="rental"/>
12. </r2ml:Conjunction>
13. <r2ml:ReferencePropertyAtom
14.   r2ml:referenceProperty="hasReturnBranch">
15.   <r2ml:subject>
16.     <r2ml:ObjectVariable r2ml:name="rental">
17.       r2ml:class="Rental"/>
18.   </r2ml:subject> <r2ml:object>
19.     <r2ml:ObjectVariable r2ml:name="returnBranch">
20.       r2ml:class="Branch"/>
21.   </r2ml:object></r2ml:ReferencePropertyAtom>
22. <r2ml:ReferencePropertyAtom
23.   r2ml:referenceProperty="hasPickupBranch">
24.   <r2ml:subject>
25.     <r2ml:ObjectVariable r2ml:name="rental">
26.       r2ml:class="Rental"/>
27.   </r2ml:subject><r2ml:object>
28.     <r2ml:ObjectVariable r2ml:name="pickupBranch">
29.       r2ml:class="Branch"/>
30.   </r2ml:object></r2ml:ReferencePropertyAtom>
31. </r2ml:Conjunction>
32. /r2ml:antecedent><r2ml:consequent>
33. <r2ml:EqualityAtom>
34.   <r2ml:ObjectVariable r2ml:name="returnBranch"/>
35.   <r2ml:ObjectVariable r2ml:name="pickupBranch"/>
36. </r2ml:EqualityAtom>
37. </r2ml:consequent></r2ml:Implication>
38. </r2ml:UniversallyQuantifiedFormula>
39. </r2ml:constraint></r2ml:DeonticIntegrityRule>

```

An example of integrity rule from Eu-Rent in SBVR Structured English is: If rental is not one way rental then return branch must be the same as pick-up branch of rental.

The rule is a proposition meant by an obligation formulation
. The obligation formulation embeds an universal quantification
. . The universal quantification introduces a first variable
. . . The first variable ranges over the concept 'rental'

4.3. R2ML Atoms

r2ml:ObjectClassificationAtom is used to capture the *instanceOf* relationship between objects and classes. Any R2ML object classification atom consists into a mandatory attribute r2ml:class (xs:QName) and an object term as an argument.

```

<r2ml:ObjectClassificationAtom r2ml:class="Driver">
  <r2ml:ObjectName r2ml:object="John"/>
</r2ml:ObjectClassificationAtom>

```

SBVR Structured English style will express the above atom in: John is driver.

The statement is formulated by an instantiation formulation
. The instantiation formulation considers the concept type 'driver'
. The instantiation formulation binds to the individual concept 'John'

Another example, using this time the r2ml:ObjectVariable term (see R2ML Derivation rule example - lines 19-20) verbalizes into SBVR Structured English: driver is invalid driver.

The statement is formulated by an instantiation formulation.
. The instantiation formulation considers the concept type 'invalid driver'
. The instantiation formulation binds to the variable 'driver'

r2ml:DataClassificationAtom consists of a data term and refers to a datatype. Its role is to classify data terms (data variables, data literals and functional data terms) with respect to a datatype indicated by a reference i.e. see R2ML Alethic rule example - lines 8-10.

SBVR nu dot distinguish between object types and data types. Therefore, for each datatype except for pre-defined SBVR elementary concepts (number/ integer/ nonnegative integer/ positive integer/ text) all XML Schema datatypes will be mapped into SBVR Object-Type concept. Therefore, SBVR conceptual schema contains an object of type double i.e.

The conceptual schema includes an object type, ob
. ob has a designation whose signifier is the text 'local area'

Below, the SBVR semantic formulation classify the variable local area as being of type double i.e.

The statement is formulated by an instantiation formulation.
. The instantiation formulation considers the concept type 'double'
. The instantiation formulation binds to the variable 'local area'

An R2ML r2ml:ReferencePropertyAtom associates an object term as subject with other object term as object.

```

<r2ml:ReferencePropertyAtom r2ml:referenceProperty="isCoveredBy">
  <r2ml:subject>
  <r2ml:ObjectVariable r2ml:name="vehicle"
    r2ml:class="userv:Vehicle"/>
  </r2ml:subject><r2ml:object>
  <r2ml:ObjectVariable r2ml:name="policy"
    r2ml:class="InsurancePolicy"/>
  </r2ml:object></r2ml:ReferencePropertyAtom>

```

SBVR expresses this association using its basic unit: *atomic formulation* that is based on a *binary fact type*. This translation always implies exactly two role bindings to the appropriate variables. The SBVR atomic formulation can be considered as an invocation of a predicate. A specialization of a binary fact type is the *partitive fact type* that denotes a binary fact type where each instance is an actuality that a given part is in the composition of a given whole i.e. the binary fact type has a designation whose signifier expresses UML composition i.e. contains/includes.

The statement is meant by an atomic formulation
. The atomic formulation is based on the binary fact type 'vehicle is covered by policy'
. . The atomic formulation has a role binding
. . . The role binding is of the role 'vehicle' of the binary fact type
. . . . The role binding binds to the first variable
. The first variable ranges over the concept 'vehicle'
. . The atomic formulation has a second role binding
. . . The second role binding is of the role 'policy' of the binary fact type
. . . . The second role binding binds to the second variable
. The second variable ranges over the concept 'insurance policy'

SBVR Structured English: Vehicle is covered by insurance policy.

r2ml:AssociationAtom is used with the purpose to support common fact types of natural language. It represents the concept of n-ary predicate from first order logic and comprise a collection of data terms as "data arguments" and a collection of object terms as "object arguments" i.e.

```

<r2ml:AssociationAtom r2ml:associationPredicate="delivers">
  <r2ml:objectArguments>
    <r2ml:ObjectVariable r2ml:name="manufacturer"
      r2ml:class="CarManufacturer"/>
    <r2ml:ObjectVariable r2ml:object="consignment"
      r2ml:class="Consignment"/>
    <r2ml:ObjectVariable r2ml:object="branch"
      r2ml:class="Branch"/>
  </r2ml:objectArguments>
</r2ml:AssociationAtom>

```

The statement is meant by an atomic formulation

- . The atomic formulation is based on the associative fact type 'manufacturer delivers consignment to branch'
- . . The atomic formulation has a role binding
- . . . The role binding is of the role 'manufacturer' of the associative fact type
- . . . The role binding binds to the first variable
- The first variable ranges over the concept 'car manufacturer'
- . . The atomic formulation has a second role binding
- . . . The second role binding is of the role 'consignment' of the associative fact type
- . . . The second role binding binds to the second variable
- The second variable ranges over the concept 'consignment'
- . . The atomic formulation has a third role binding
- . . . The third role binding is of the role 'branch' of the associative fact type
- . . . The second role binding binds to the second variable
- The second variable ranges over the concept 'branch'

SBVR Structured English: A car manufacturer delivers consignment [to] branch.

Neither R2ML, nor SBVR do not succeed to capture the full meaning of a n-ary predicate from FOL, as it needs more than one associative predicate i.e (we add the [to] signifier to denote the possible preposition needed).

r2ml:AttributionAtom captures data valued properties of objects. SBVR verbalizes the r2ml:AttributionAtom using the concept of quantity i.e. the binary fact type 'quantitative property equals quantity', in the sense that the quantitative property is mathematically equivalent to the quantity. In the first order logic, the equality testing becomes an attribution if the quantitative property represents an unbounded variable, which is our case i.e. see R2ML Alethic rule example - lines 13-17.

The SBVR concept quantity can be elaborated into mathematical systems, such as integers and real numbers, and into systems of measures. This specification elaborates only the concepts for integer, because they are commonly used in structural rules. The proposition is based on the fact type 'quantitative property equals quantity'. The quantitative property is the noun concept expressed as: country's local area.

```

The statement is formulated by an existential quantification
. The quantification introduces a unitary variable
. . The variable ranges over the concept 'quantitative property'
. . . The variable is restricted by a noun concept nominalization
. . . . The noun concept nominalization binds to the first variable
. . . . The noun concept nominalization considers a projection
. . . . . The projection is on a second unitary variable
. . . . . The second variable ranges over the concept 'local area'
. . . . . The projection is constrained by an atomic formulation

```

```

. . . . . The atomic formulation is based on the fact type
. . . . . . 'country has local area'
. . . . . The 'country' role is bound to the first variable
. . . . . The 'local area' role is bound to the second variable
. . The quantification scopes over an atomic formulation
. . The atomic formulation is based on the fact type
. . . . 'quantitative property equals quantity'
. . . The 'quantitative property' role is bound to the second
. . . . . variable
. . . The 'quantity' role is bound to the third variable

```

SBVR Structured English: Local area is in exactly one operating country.

r2ml:DatatypePredicateAtom is designed to capture the use of relational operators in R2ML language. Until this version, R2ML had not declared its own built-in constructs, but it allows the use of external ones, such as SWRL built-ins, for representing the predicate type of the relational operations (i.e. swrlb:lessThen) - (see R2ML Derivation rule example - lines 6-17). We also use the construct of r2ml:DatatypePredicateAtom to represent literal field constraints that test equality / inequality of data types properties. When serialising object types literal field constraints we use the r2ml:EqualityAtom to express the concept of equality and the r2ml:InequalityAtom to express the concept of inequality (!=) (see R2ML Deontic rule example - lines 32-35).

SBVR expresses the relational operators using quantifications. The above proposition is meant by atomic formulation. The atomic formulation is based on the fact type quantitative property is less than quantity i.e. Driver 's age is less than 16.

```

The statement is formulated by an existential quantification
. The quantification introduces a unitary variable
. . The variable ranges over the concept 'quantitative property'
. . . The variable is restricted by a noun concept nominalization
. . . . The noun concept nominalization binds to the first variable
. . . . The noun concept nominalization considers a projection
. . . . . The projection is on a second unitary variable
. . . . . The second variable ranges over the concept 'age'
. . . . . The projection is constrained by an atomic formulation
. . . . . The atomic formulation is based on the fact type
. . . . . . 'driver has age'
. . . . . The 'driver' role is bound to the first variable
. . . . . The 'age' role is bound to the second variable
. . The quantification scopes over an atomic formulation
. . The atomic formulation is based on the fact type
. . . . 'quantitative property is less than quantity'
. . . The 'quantitative property' role is bound to the second
. . . . . variable
. . . The 'quantity' role is bound to the individual concept '16'

```

4.4. Mapping R2ML Terms

Following RuleML, R2ML framework defines the generic concepts of variable. However, R2ML make a clear distinction between object terms and data terms. Typed terms are either *object terms* standing for *objects*, or *data terms* standing for *data values*. R2ML also provides the concept of the *variable*, and distinguishes between object and data variables i.e. between

references that are instantiated with objects and data values, respectively.

Usually, a R2ML term is contained, as a child of a particular role element, in other R2ML atoms. Therefore, the SBVR verbalization will always take into consideration, the relation between the terms and the R2ML atoms that comprise them. Is not our intention to provide in this paper a complete verbalization of the entire set of R2ML terms. We only discuss few peculiar verbalisation cases.

`r2ml:TypedLiteral` is mapped into SBVR using the values of the `r2ml:lexicalValue` and `r2ml:datatype` attributes. Currently, in SBVR, the valid data types are: integer / non negative integer / positive integer (see Figure 1). Therefore, all the other XML qualified names from R2ML language will be translated into SBVR object types.

Exception makes the XML qualified name `xs:boolean`. It will translate into SBVR characteristic (i.e. unary fact type), while all the other will translate into binary fact types. The below example represents a boolean property (i.e. `isLicensed`) of the object variable `driver`.

The SBVR verbalization supposes the existence of the above mentioned characteristic in the set of characteristics, for an SBVR object type. The characteristic is an abstraction of a property of each instance of the concept. Every characteristic incorporated by a concept is a necessary characteristic of the concept, but not every necessary characteristic of the concept is incorporated by the concept. For example, the concept 'qualified driver' incorporates the characteristic: *driver being licensed* because it is necessary (by the definition of qualified driver that each qualified driver is licensed).

R2ML variables are provided in the form of `r2ml:ObjectVariable` and `r2ml:DataVariable`. `r2ml:ObjectVariable` are variables that can be only instantiated by objects. `r2ml:DataVariable` are variables that can be only instantiated by data literals. They collapse into SBVR `variable` concept (see R2ML Alethic rule example - line 14 and line 16).

SBVR variables are logic variables and are introduced by quantifications and projections, so that embedded formulations can refer to instances of concepts. A logic variable used in a formulation is free within that formulation if it is not introduced within that formulation. A formulation is closed if no variable is free within it.

Only a closed semantic formulation can formulate a meaning. If a formulation has a variable that is free within it, then it can be part of a larger formulation of a meaning (one that introduces the variable) but it

does not by itself formulate a meaning.

5. Conclusion and Future Works

Our paper brought into discussion the verbalization of R2ML integrity/derivation rules into SBVR Structured English and SBVR Semantic Formulations. Our approach is based on the concepts of MDA abstract cycles and focuses on two areas: obtaining a controlled natural language i.e. SBVR Structured English, suitable for business people that can validate and authoring the rules, bridging this way the Semantic Web community and business modellers, and obtaining a formal, structured form of rules in SBVR Semantic Formulations that further can be translated into executable rule languages.

The main idea of MDA is that the design models at different levels of abstraction CIM/PIM/PSM are derived from one to another.

The paper presents an ongoing work that focuses on putting executable business rules at higher levels of abstraction, and also discusses the difficulties of such a mapping.

References

- [1] H. Boley, M. Kifer (2007), *RIF Core Design*, W3C Working Draft, March 30, 2007 <http://www.w3.org/TR/rif-core/>
- [2] N. E. Fuchs, U. Schwertel, R. Schwitter, *Attempto Englisch als (formale) Spezifikationsprache*, In F. Bry, B. Freitag, D. Seipel (Eds.) Proceedings of the Twelfth Workshop on Logic Programming WLP '97, Mnich, September 1997.
- [3] F. Meziane, N. Athanasakis and S. Ananiadou, *Generating Natural Language Specifications from UML Class Diagrams* Requirements Engineering Journal 13(1) (2008) 1-18.
- [4] R. Pau, J. Cabot *Paraphrasing OCL expressions with SBVR* In Proc. 13th Int. Conf. on Applications of Natural Language to Information Systems, LNCS, 2008, (to appear).
- [5] (OMG) *Semantics of Business Vocabulary and Rules (SBVR) Specification* OMG Adopted Specification (dtc/06-03-01). (2006)
- [6] (OMG) *Unified Modeling Language ver. 2.1.1 (UML)*, 2007, <http://www.omg.org/technology/documents/formal/uml.htm>
- [7] G. Wagner, S. Lukichev, N. E. Fuchs, S. Spreeuwenberg *First-Version Controlled English Rule Language* deliverable, REWERSE project, IST506779/Eindhoven/I1-D2/D/PU/b1 (2005)
- [8] G. Wagner, A. Giurca and S. Lukichev, *R2ML: A General Approach for Marking up Rules*, Dagstuhl Seminar Proceedings 05371, In F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, ISSN:1862-4405.