A Set of Ontology Design Patterns for Reengineering SBVR Statements into OWL/SWRL Ontologies

E. Reynares[1,], M.L. Caliusco[1], M.R. Galli[1,1]

[a]*CIDISI Research Center, UTN Santa Fe, B. Lavaysse 610, S3004EWB Santa Fe, Argentina*

[b]*INGAR-UTN-CONICET, Avellaneda 3657, S3002GJC Santa Fe, Argentina*

Abstract

The interest in the use of ontologies for creating more intelligent and effective enterprise information systems has increased considerably in recent years. The most critical aspects during the development of these systems are: (1) to identify the ontology concepts and (2) to make explicit the business rules by means of the ontology axioms.

In order to address these issues, mappings of business rules expressions to ontology statements based on different languages were proposed. Despite the efforts made in this area, some work remain to be done.

This work presents a set of ontology design patterns providing a way to obtain an OWL/SWRL ontology by applying metamodel transformation rules over the SBVR specification of a business domain. Patterns are rooted in the structural specification of the standards, providing a set of mappings readily usable for business people or developers concerned with the implementation of a mapping tool. Moreover, translations from SBVR to SWRL language are presented in order to fill the gap in the expressive power of SBVR and OWL. The theoretical expressions of patterns are illustrated by means of an example depicting the core structure of a fictitious company.

Keywords: ontology-driven information systems, ontology design pattern, SBVR 1.1, OWL 2, SWRL

*Corresponding author. Address: CIDISI Research Center, UTN Santa Fe, B. Lavaysse 610, S3004EWB Santa Fe, Argentina. Tel.: +54 0342 460 2390 int. 258 sub-int. 106

1. Introduction

The application of semantic technologies for creating more intelligent and effective enterprise information systems has increased considerably in recent years. Ontologies in particular proved to provide strong benefits in a wide range of settings and applications. For example, ontology reasoners can automatically prove the consistency of business models (?????). Ontologies intended to be used at the analysis stage of a software development process can be generated from main business knowledge sources (??). Ontologies can also encapsulate the declarative specification of business knowledge into information software systems, enabling unambiguous representation of knowledge and efficient management of highly dynamic environments (?????).
Despite those significant applications, ontology development methodologies remain to be an open research area. Most methodologies provide neither enough details about employed techniques and activities nor detailed recommendations for reusing and reengineering ontologies. Moreover, they opt for conventional strategies for identifying ontology concepts. New methods and techniques should be explored to make this process more efficient and handier for the software engineers, as it plays a critical role in the ontology designing phase. (?).
Recent works highlight that ontological definitions of concepts in a formal language do not differ much from descriptions of terms in natural language: in both cases an expression is constructed by combining symbols according to grammatical rules (??). Following such insight, some authors have proposed the mapping of business rules expressions to ontology statements as a building technique (???????). Such approaches are usually rooted in two languages: (1) the Semantics of Business Vocabulary and Business Rules (SBVR) and (2) the OWL Web Ontology Language (OWL). SBVR provides business people with a linguistic way to semantically describe business concepts and specify business rules independently of any information system design (?). OWL is selected as the receipt language of transformations because it has evolved as a de-facto standard for a broad spectrum of applications (?).
A first intent in that direction is explored in ?. The work combines OWL 1 (?) and SWRL expressions (?) to obtain a Platform Independent Model (PIM) (?) from SBVR 1.0 business vocabulary and rules expressions.? and ? propose a set of transformations rooted in ORM conceptual modeling lan-

guage[1], which is at the core of the SBVR proposal. **?** also depict a tool which implements translations of a set of ORM 2 constraints into a OWL 2 ontology. **?** proposes a reversible mapping between SBVR and OWL 2, offering a way to exchange SBVR vocabularies between tools. An study analysing the suitability of a subset of the SBVR metamodel for representing OWL 2 ontologies is depicted in **?**. However, such proposals fails to provide an explicit representation of the transformations (**?**), or they states logical foundations that not be used directly to map SBVR expressions into OWL 2 statements (**??**), or they take in account a very small set of the elements usually involved in the stating of complex business rules (**?**), or they overlook specific characteristics of the source metamodel of the transformations by following the opposite direction to that considered in this work. **?** and **?** present a set of structural based transformations that allows the automatable generation of an OWL 2 ontology from SBVR specifications of a business domain. An experiment aimed at obtaining empirical evidence about the feasibility of the two latter proposals can be found in **?**. Although the experiment stresses the potential of the approach as an ontology development technique, it also allows to recognize the need to improve the way the semantics of both metamodels is reflected (**??**).
The recent publication of an improved version of SBVR language and the empirical evidence found by the aforementioned experiment has motivated the development of this work: it presents a set of ontology design patterns providing a way to obtain an OWL/SWRL ontology by applying a set of metamodel transformation rules over SBVR specification of a business domain. Transformations are rooted in the structural specification of both standards rather than theoretical considerations of the language, providing a set of mappings readily usable for business people or developers concerned with the implementation of a mapping tool. Although the latest version of OWL provides a set of constructors which enables the mapping of most SBVR concepts, the proposal - different from previous work - also includes some translations from SBVR to SWRL language as a way to fill the gap in the expressive power of SBVR and OWL.
The rest of this paper is organized as follows. Section **??** presents some conceptual foundations of the design pattern approach and its applications to the

[1]Object Role Modeling. The official site for conceptual data modeling. Accessible in: http://orm.net/

ontology engineering field. Section **??** and Section **??** provides an overview of the latest version of SBVR and OWL/SWRL specifications, respectively. Section **??** presents patterns and illustrates them using an example. Finally, Section **??** presents some discussions about the addressed topics, while conclusions and future research directions are presented in Section **??**.

2. Background

This paper presents a set of ontology design patterns providing a way to obtain an OWL/SWRL ontology by applying a set of metamodel transformation rules over SBVR specification of a business domain. The term *design pattern* was originally conceived in the architectural field for naming a set of shared guidelines that help in solving design problems. Each pattern describes a problem that occurs over and over again in an environment, and describes the core of the solution to that problem (**??**). Design patterns have evolved as a widely accepted notion in the software engineering area (**?**) and their applicability has been extended to the ontology engineering field (**?????**). In such contexts, the reusable nature of patterns assists with the modeling of the common issues and improves interoperability by means of proven solutions also known as *best practices*. Several experiments have been performed showing that reusing ODP facilitates the development process and improves the quality of the resulting ontology (**?**). So, the following definition can be posed according to **?** and **?**:

Definition 1. An *Ontology Design Pattern* (ODP) is a modeling solution aimed at solving a recurrent ontology design problem, which provides (1) reuse, (2) guidance, and (3) communication benefits.

Moreover, ODPs have been grouped into six families for organization purposes(**?**):

1. *Structural* family comprises patterns that are compositions of logical constructors, solving a problem of expressibility and affecting - internally or externally - the overall shape of the ontology.
2. *Correspondence* patterns provide solutions to the problem of transforming a conceptual model into an ontology. Patterns for creating semantic associations between two existing ontologies also belong to this family.

3. *Content* patterns solve design problems for a specific domain. They are content dependent, defining an explicit non logical vocabulary for a specific domain of interest.
4. *Reasoning* patterns are oriented to obtain certain reasoning results, based on the behaviour implemented in a reasoning engine, i.e.: classification, subsumption, inheritance, etc.
5. *Presentation* patterns deal with usability and readability of ontologies from a user perspective: they are conceived as good practices that support the reuse of ontologies by facilitating their evaluation and selection.
6. *Lexicon-Syntactic* patterns are linguistic structures or schemas that consist of certain types of words following a specific order, allowing to generalize and extract some conclusions about the meaning they express.

Furthermore, ontology design patterns can be separated into either the *logical* or the *conceptual* group. While logical patterns are aimed at solving design problems independently of any particular conceptualization, conceptual ones solve design problems for specific domains. Most of the work in the field has been performed in the conceptual group, proposing a wide spectrum of content patterns for solving modeling issues in several domains[2]. Important contributions to the logical group are presented in **?** and **?**. OWL 2 representations of three patterns - i.e., *diamond structures*, *reification of relations* and *sequences of entities* - are depicted in **?**, also presenting how they can be applied to various domains. A model, a method, and a technology for reusing and reengineering non ontological resources when building ontologies by means of reengineering patterns are presented in **?**. However, the patterns are conceived to translate classification schemes, thesauri, and lexica; and they do not provide support for the mapping of complex expressions in business rules.

This work is focused on the specification of a set of *Schema Reengineering Patterns*, which belong to the aforementioned *Correspondence* family. A *Schema Reengineering Pattern* (SRP) provides a way to obtain an ontology by applying a set of metamodel transformation rules over a conceptual model. The source conceptual model can be either an ontology, a thesaurus concept, a data model pattern, a UML model, a linguistic structure, etc

[2]A list of the available ODPs can be found in http://ontologydesignpatterns.org

(?). SRPs presented in this paper define a set of rules for the generation of an OWL/SWRL ontology from SBVR specification of a business domain. The following subsections provide a brief description of the source and target metamodels.

2.1. SBVR Overview

SBVR 1.1 is the latest version of a fact-oriented language for modeling business domain information. It defines the vocabulary and grammar for documenting the semantics of business vocabularies, business facts, and business rules by means of a controlled vocabulary readily understandable by business people. SBVR has a sound theoretical foundation of formal logic: it is based on first-order predicate logic with extensions into modal logic, i.e., some deontic forms for expressing obligation and prohibition, and alethic forms for expressing necessities and possibilities.
The fact-oriented approach of SBVR stems from the Business Rules Manifesto[3] which states that *rules are built on facts, and facts are built on concepts as expressed by terms*. In other words, *terms express business concepts, facts make assertions about these concepts, and rules constrain and support these facts*. In this way, the core of SBVR is composed of *noun concepts* and *verb concepts* corresponding to the notions of *terms* and *facts* respectively, in addition to rules as restriction statements.
A *noun concept* is a concept that is the meaning of a noun or noun phrase, which is specialized by: (1) *general concepts*, which are noun concepts classifying things on the basis of their common properties; (2) *individual concepts*, which are concepts corresponding to only one object thing and (3) *roles*, which are noun concepts corresponding to things based on their playing a part, assuming a function or being used in some situation. Additionally, *verb concept roles* are defined as those roles that specifically characterizes its instances by their involvement in an instance of a given verb concept. A *verb concept* is a concept that is the meaning of a verb phrase that involves one or more noun concepts. It can be used to represent unary relations - i.e.,*characteristic/unary verb concepts* -, binary relations - i.e., *binary verb concept* and even n-ary relations. Figure 1 shows the structural organization of the aforementioned core concepts.
Finally, a SBVR rule is an element of guidance that introduces an obliga-

[3]http://www.businessrulesgroup.org/brmanifesto.htm

tion or a necessity. It is built by imposing restrictions over verb concepts by means of modalities, quantifiers, and logical operators. All rules have an associated modality determined by the logical modal operator that functions explicitly or implicitly as its main operator; i.e.: an alethic modality of necessity is assumed if no modality is explicitly specified. Consequently, two fundamental categories of rules can be distinguished: (1) *structural rules*, which describe the way chosen by the business to organize the elements it deals with, and (2) *operative rules*, which govern the conduct of business activity and, in contrast to the structural ones, it can be violated in the affairs of the business. Table **??** shows the alethic and deontic modal operators that may be used by rule formulations. Figure 2 and Figure 3 depict quantifiers and logical operators, respectively.

Modalities and their readings		
alethic	*necessity*	It is necessary that p
	non-necessity	It is not necessary that p
	possibility	It is possible that p
	impossibility	It is impossible that p
	contingency	It is possible but not necessary that p
deontic	*obligation*	It is obligatory that p
	non-obligation	It is not obligatory that p
	permission	It is permitted that p
	prohibition	It is prohibited that p
	optionality	It is permitted but not obligatory that p

Table 1: SBVR Modal Operators

2.2. OWL 2 / SWRL Overview

OWL 2 Web Ontology Language (informally OWL 2) is the latest version of an ontology language for Semantic Web with formally defined meaning **?**. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. It has been proposed by World Wide Web Consortium (W3C) for the development of Semantic Web, but it has gradually evolved into a de-facto standard for a broad spectrum of applications.

An OWL 2 ontology is a formal description of a domain of interest interpreted under a standardized semantics and allows useful inferences to be drawn. Following, the three syntactic categories an OWL 2 ontology is based on are briefly depicted.

1. *Entities* such as classes, properties, and individuals. They are the basic elements of an ontology. For example, a class *:Person* can be used to represent the set of all people; the object property *:parent-of* can be used to represent the parent-child relationship; and the individual *:Peter* can be used to represent a particular person named *'Peter'*. Figure 4 depicts the structural specification of entities.
2. *Expressions*, representing complex notions in the described domain. For example, a class expression describes a set of individuals in terms of the restrictions on the individuals characteristics. Structural specification of expressions is depicted in Figure 5.
3. *Axioms*, which are statements asserted to be true in the described domain. For example, a subclass axiom state that the class *'Boy'* is a subclass of the class *'Person'*. Figure 6 depicts the structural specification of axioms.

OWL 2 ontology language defines several concrete syntaxes that can be used to serialize and exchange ontologies. Among them, the functional style syntax is used in the OWL 2 structural specification **?** with the aim of stating the semantics of OWL 2 constructors and allowing a compact writing of ontologies.
Regarding the semantics of OWL 2, the language provide two different ways of assigning meaning to ontologies: (1) the Direct Semantics (**?**) and (2) the RDF-based Semantics (**?**). Direct Semantics provides a semantic compatible with the model theoretic semantics of the SROIQ fragment of Description Logic (**?**). However, some syntactic conditions must be imposed over an ontology structure in order to ensure that it can be interpretable under such semantic [4]. In this case, the reasoning procedures applied over the ontology are sound - i.e., only correct answers to queries are computed - and complete - i.e., all correct answers are computed -. *"OWL 2 DL"* is used informally to refer to OWL 2 ontologies satisfying the syntactic conditions and interpreted

[4]A full list of the conditions can be found in http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Ontologies

by means of the Direct Semantics. RDF-based Semantics can be applied to a OWL 2 ontology without the restrictions before mentioned. *"OWL 2 Full"* is used informally to refer to OWL 2 ontologies interpreted by means of the RDF-Based Semantics. Nevertheless, a reasoning procedure applied over an ontology interpreted under this semantic could not be complete: i.e., it is not ensured that all correct answers to queries are computed.
The rest of this paper follows the structural specification insight to generate OWL DL ontologies: it uses the functional style syntax to state OWL 2 ontology expressions, assigning meaning to them by means of the Direct Semantics.
Semantic Web Rule Language (SWRL) is a proposal aimed at extending the set of OWL 2 axioms to include Horn-like rules, enabling their combination with an OWL 2 knowledge base (?). It defines an OWL model theoretic semantics to provide a formal meaning for OWL ontologies including such rules.
The rules are of the form of an implication between an antecedent - *body* - and consequent - *head* - . The intended meaning can be read as: whenever the conditions specified in the body hold, then the conditions specified in the head must also hold. Both the body and the head consist of zero or more atoms. An empty antecedent is treated as trivially true - i.e., satisfied by every interpretation - so the consequent must be also satisfied by every interpretation. An empty consequent is treated as trivially false - i.e., not satisfied by any interpretation -, so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction. The rules with conjunctive consequents could be easily transformed into multiple rules each with an atomic consequent. Atoms in these rules can be of the form *'C(x)'*, *'P(x,y)'*, *'sameAs(x,y)'* or *'differentFrom(x,y)'*, where *'C'* is an OWL 2 description, *'P'* is an OWL 2 property, and *'x'* and *'y'* are either variables, OWL 2 individuals, or OWL 2 data values.

3. Schema Reengineering Patterns

The theoretical expressions of the proposed patterns are presented in the following subsections by grouping and sequencing them according to the inherent logical order of the subject matter itself. Additionally, the patterns are illustrated by means of an example depicting the core structure of a fictitious company named INTCO (Figure 7). The company is organized into departments, each of which is run by a chief and has a traditional hierarchical organization: each employee is under one boss's charge and each boss can be in charge of 25 employees at the most - where *employee* and *boss* are roles played by human resources of the company -. The amount of human resources working in a given department determines its size. Moreover, a set of skills and the existence of a senior degree is settled for each human resource. Interns are a special kind of human resource: they do not have a senior degree and cannot be the chief of any department. Even though the example is quite simple, it turns to be complete enough to depict the usefulness of the proposal[5].

3.1. Core Mappings

The present subsection depicts the patterns for the translation of the concepts that constitute the core of SBVR metamodel, which are fundamental for the definition of the remaining ones.

3.1.1. General Concept

SBVR defines *general concept* as a *"noun concept that classifies things on the basis of their common properties"*. A general concept is mapped to an OWL 2 class, which is understood as a set of individuals.
Following the presented example, the *'Human Resource'* general concept - representing the people working at the company - is mapped to the *'Human-Resource'* OWL 2 class.

3.1.2. Unitary Concept

SBVR defines *unitary concept* as a *"noun concept that always has one instance at the most"*. Although such definition corresponds to the widely known notion of *singleton* in the Software Engineering field, OWL 2 does not define a constructor to model concepts with one individual at the most. However, it is possible to specify a SWRL statement to cover such lack of expressibility:

Definition 2. SWRL restriction over singleton concepts:
Singleton C(?x), Singleton C(?y) $\rightarrow$ *SameAs (?x, ?y)*

[5]A complete SBVR specification of the example and the full ontology generated by the application of patterns can be found in https://code.google.com/p/ontology-development-srp-example/

As a consequence, an unitary concept is mapped to an OWL 2 class plus the above presented SWRL sentence, stating that if more than one individual is defined for such a class, then those individuals are actually the same.
According to the domain being modeled in the example, the INTCO organization is the only individual of the *'Company'* general concept. So, the existence of individuals belonging to the *'Company'* OWL 2 class is restricted by a similar SWRL rule to that of the previous filed: *Company(?x), Company(?y)* $\rightarrow$ *SameAs (?x, ?y)*.

3.1.3. Verb Concept Role

SBVR defines *role* as a *"noun concept that corresponds to things based on their playing a part, assuming a function or being used in some situation"*. Rooted in that insight, a SBVR *verb concept role "is a role that specifically characterizes its instances by their involvement in an actuality that is an instance of a given verb concept"*. A verb concept role is understood as a point of involvement in actualities that correspond to a verb concept. It incorporates characteristics from the verb concept - i.e., what the verb concept requires of role instances - .
A SBVR verb concept role is mapped to an OWL 2 defined class in terms of (1) the relationship that provides it with meaning and (2) the general concept allowed to fill the role. In this way, the verb concept role is reified as an OWL 2 class defined according to two object properties. The first one represents the verb concept in which the role is involved. The second one establishes the role player.
Binary verb concept *'employee respond to boss'* binary verb concept used in the example establishes the relation between employee and boss roles, which are filled by instances of human resources. OWL 2 modeling of such roles by the application of the proposed pattern is depicted in Figure 8.

3.1.4. Binary Verb Concept

SBVR defines *binary verb concept* as a *"verb concept that has exactly two roles"*. The expressions *'employee respond to boss'* and *'human resource has skill'* are examples of binary verb concepts. The first one establishes the relation between employee and boss role, while the second one represents the relation between a human resource and a skill - which is modeled by means of a string -. As a consequence, the translation of a binary verb concept into an OWL 2 statement is made according to the type of the involved roles.

If the verb concept relates two concepts, then the expression is mapped to an OWL 2 object property with the corresponding domain and range. Therefore, binary verb concept *'employee respond to boss'* is mapped to *'respond-to'* OWL object property, with *'Employee'* OWL 2 class as the domain and *'Boss'* OWL 2 class as property range.
If the verb concept relates a concept with a literal, then the expression is mapped to an OWL 2 data property stating that the class representing the first role is connected to the corresponding datatype. So, binary verb concept *'human resource has skill'* is mapped to *'has-skill'* OWL data property, with *'Human-Resource'* OWL 2 class as the domain and *'string'* datatype as property range.

3.1.5. Characteristic (Unary Verb Concept)

SBVR defines *characteristic* as a *"verb concept that has exactly one role, an abstraction of a property of an object or a set of objects"*. The *'human resource has senior degree'* characteristic of the example represents the level of experience of a given human resource.
A characteristic is mapped to an OWL 2 data property ranging over the boolean datatype. As a consequence, the *'human resource has senior degree'* characteristic is mapped to *'has-senior-degree'* OWL 2 data property, with *'Human Resource'* OWL 2 class as domain and boolean datatype as property range.

3.1.6. Individual Concept

SBVR defines *individual concept* as a *"concept that corresponds to only one object"*. *'John Smith'* is an example of an individual concept representing the instance with such a name that is a human resource of INTCO company.
An individual concept is mapped to an OWL 2 named individual, e.g., *'John Smith'* individual concept is mapped to *'John Smith'* OWL 2 named individual. However, it is important to note that the named individual created by the application of this pattern does not belong to any particular class, i.e., a given individual belongs to *'Thing'* OWL 2 built-in class , which represents the set of all individuals. Assignment of an individual to a given class requires a SBVR classification sentence (Section **??**).

3.1.7. Classification

As shown in Section **??**, the assignment of an OWL 2 named individual to a given class requires a SBVR classification sentence. As a consequence, a SBVR *classification* is mapped to an OWL 2 class assertion axiom *'ClassAssertion(C I)'* stating that a given individual *'I'* belongs to a given class *'C'*. Application of this pattern adds semantics to the modeling of an individual by stating its membership to a particular class.

3.1.8. Atomic Formulation

SBVR defines an *atomic formulation* as a *"logical formulation that is based on a verb concept and has a role binding of each role"*.
Let *'John Smith respond to Ben Arten'* be an example of an atomic formulation based on the previously defined binary verb concept *'employee respond to boss'*. The atomic formulation has two role bindings: the first one binds individual *'John Smith'* to role *'employee'* while the second one binds individual *'Ben Arten'* to role *'boss'*. Another example is *'John Smith has mechanical engineering skills'*, being an atomic formulation based on a verb concept that represents the relation between a human resource and a skill.
As shown in the previous examples, the translation of an atomic formulation into an OWL 2 statement must take into account the type of the involved roles. If the verb concept the atomic formulation is based on relates two concepts, then the expression is mapped to an OWL 2 positive object property assertion *'ObjectPropertyAssertion(OPE I_1 I_2)'* stating that individual *'I_1'* is connected by object property expression *'OPE'* to individual *'I_2'*. *'John Smith respond to Ben Arten'* formulation is an example of this type.
If the verb concept the atomic formulation is based on relates a concept with a literal, then the expression is mapped to an OWL 2 positive data property assertion *'DataPropertyAssertion(DPE I lt)'* stating that individual *'I'* is connected by data property expression *'DPE'* to literal *'lt'*. *'John Smith has mechanical engineering skill'* formulation is an example of this type.

3.2. Logical Operations

SBVR defines *logical operation* as a *"logical formulation that formulates a meaning based only on the truth or falseness of the meanings of one or more logical operands, where each logical operand is a logical formulation playing such a role"*. The different kinds of logical operation are described in the following subsections.

The mappings of the main SBVR logical operations - i.e., *'logical negation'*, *'conjunction'*, *'disjunction'*, and *'equivalence'* operations - are depicted in the following subsections. The remaining formulations - *'exclusive disjunction'*, *'nand'*, *'nor'*, and *'whether-or-not'* - are easily translatable by the logical combination of such mappings.

3.2.1. Logical Negation

SBVR defines *logical negation* as a *"logical operation that has exactly one logical operand and that formulates that the meaning of the logical operand is false"*. A SBVR logical negation is mapped to an OWL 2 expression according to the type of the involved operand.
If the logical operand is verb concept relating concepts, then the formulation is mapped to an OWL 2 *'ObjectComplementOf(CE)'* expression, which comprises all individuals that are not instances of the class expression *'CE'*. Following the example, the expression*'intern does not have a senior degree'* expression presents a logical negation involving a general concept and a characteristic. The sentence is mapped to the OWL 2 expression *'ObjectComplementOf(DataHasValue(:has-senior-degree true))'*. Indeed, the expression *'intern is not a chief of department'* presents a logical negation involving two general concepts related by means of a binary verb concept. Such sentence is mapped to the OWL 2 expression *'ObjectComplementOf(ObjectSomeValuesFrom(:is-chief-of :Department))'*.
If the logical operand is an atomic formulation relating two individuals, then the formulation is mapped to an OWL 2 *'NegativeObjectPropertyAssertion(OPE I_1 I_2)'* expression, stating that individual *'I_1'* is not connected by object property expression *'OPE'* to individual *'I_2'*. The expression *'Peter Tomsom not respond to Ben Arten'* - stating that the first individual is not under the charge of the second one as its boss - is mapped to the OWL 2 expression *'NegativeObjectPropertyAssertion(:respond-to :Peter-Tomsom :Ben-Arten)'*
If the logical operand is an atomic formulation relating an individual with a literal, then the formulation is mapped to an OWL 2 *'NegativeDataPropertyAssertion(DPE I lt)'* expression, stating that individual *'I'* is not connected by data property expression *'DPE'* to literal *'lt'*. Following the example, the sentence stating that *'Peter Tomsom does not have a senior degree'* is mapped to the OWL 2 expression *'NegativeDataPropertyAssertion(:has-senior-degree :Peter-Tomsom true)'*.

3.2.2. Conjunction

SBVR defines *conjunction* as a *"binary logical operation that formulates that the meaning of each of its logical operands is true"*. A SBVR conjunction is mapped to an OWL 2 expression according to the types of the involved operands.
If the conjunction is applied over two verb concepts, then the formulation is mapped to an OWL 2 *'ObjectIntersectionOf(CE_1 CE_2)'* expression, which contains all individuals that are instances of both *'CE_1'* and *'CE_2'* class expressions. The same mapping is applied if the conjunction is formulated over two general concepts.
Also, *'has senior degree and respond to'* is a conjunction example over *'has senior degree'* and *'respond to'* verb concepts. The formulation is mapped to an *object intersection of* sentence where each class expression represents an involved verb concept. In the same way, *'employee and boss'* expression comprising all individuals that play both roles is mapped.
If the conjunction is applied over two datatypes containing a set of literals - e.g., number, string, etc. -, then the formulation is mapped to an OWL 2 *'DataIntersectionOf(DR_1 DR_2)'* expression, which contains all tuples of literals that are contained in both *'DR_1'* and *'DR_2'* data ranges. For example, *'nonNegativeNumbers and NonPositiveNumbers'* formulation that only contains number zero is mapped to a *data intersection of* sentence where data ranges are represented by the corresponding datatypes.

3.2.3. Disjunction

SBVR defines *disjunction* as a *"binary logical operation that formulates that the meaning of at least one of its logical operands is true"*. A SBVR disjunction is mapped to an OWL 2 expression according to the types of the involved operands.
If a SBVR disjunction is applied over two verb concepts, then the formulation is mapped to an OWL 2 *'ObjectUnionOf(CE_1 CE_2)'* expression, which contains all individuals that are instances of at least one class expression *'CE_i'*. The same mapping is applied if the disjunction is formulated over two general concepts.
A disjunction example over such verb concepts is the *'has senior degree or has skill'* expression. The formulation is mapped to an *object union of* sen-

tence where each class expression represents an involved verb concept. In the same way, an *'employee or boss'* expression comprising all individuals that play any of the two roles is mapped.
If the disjunction is applied over two datatypes containing a set of literals, then the formulation is mapped to an OWL 2 *'DataUnionOf(DR_1 DR_2)'* expression, which contains all tuples of literals that are contained in at least one data range *'DR_i'*. For example, *'nonNegativeNumbers or NonPositiveNumbers'* formulation containing all numbers is mapped to a *data union of* sentence where data ranges are represented by the corresponding datatypes.

3.2.4. Equivalence

SBVR defines *equivalence* as a *"binary logical operation that formulates that the meaning of its logical operands are either all true or all false"*. A SBVR equivalence is mapped to an OWL 2 expression according to the types of the involved operands.
If equivalence is applied over two general concepts, then the formulation is mapped to an OWL 2 *'EquivalentClasses(CE_1 CE_2)'* expression, which states that both *'CE_1'* and *'CE_2'* class expressions are semantically equivalent to each other. Let *'human resource is equivalent to person'* be an example of an equivalence among two general concepts, then the formulation is mapped to OWL 2 *'EquivalentClasses(:Human-Resource :Person)'* sentence.
If equivalence is applied over two verb concepts relating concepts, then the formulation is mapped to an OWL 2 *'EquivalentObjectProperties(OP_1 OP_2)'* expression, which states that both *'OP_1'* and *'OP_2'* object property expressions are semantically equivalent to each other. The formulation *'respond to is equivalent to has boss'* is an equivalence example, which is mapped to OWL 2 *'EquivalentObjectProperties(:respond-to :has-boss)'* sentence.
If equivalence is applied over two verb concepts relating concepts with literals, then the expression is mapped to an OWL 2 *'EquivalentDataProperties(DP_1 DP_2)'* expression, which states that both *'DP_1'* and *'DP_2'* data property expressions are semantically equivalent to each other. The formulation *'has senior degree is equivalent to is senior"* is an equivalence example which is mapped to OWL 2 *'EquivalentDataProperties(:has-senior-degree :is-senior)'* sentence.
If equivalence is applied over two individuals, then the formulation is mapped

to an OWL 2 *'SameIndividual(I_1 I_2)'* expression, which states that both individuals *'I_1'* and *'I_2'* are equal to each other. The formulation *'John Smith is equivalent to J.S.'* is an equivalence example which is mapped to OWL 2 *'SameIndividual(:John-Smith :J.S.)'* sentence.

3.3. Quantifications

SBVR defines *quantifications* as *"logical formulations introducing a variable and meaning either: all referents of the variable satisfy a scope formulation or a bounded number of referents of the variable exist and satisfy a scope formulation, if there is one"*.
The mappings of the main SBVR quantifications - i.e., *'universal'*, *'existential'*, *'at most n'*, *'at least n'*, and *'exactly n'* quantifications - are depicted in the following subsections. The remaining quantifications - *'at most one'*, *'exactly one'*, and *'numeric range'* - are easily translatable in terms of such mappings.

3.3.1. Universal Quantification

SBVR defines *universal quantification* as a *"quantification that scopes over a logical formulation and that has the meaning: for each referent of the variable introduced by the quantification the meaning formulated by the logical formulation for the referent is true"*.
If quantification is applied over a verb concept relating concepts, then the expression is mapped to an OWL 2 *'ObjectAllValuesFrom(OPE CE)'* expression, which contains all those individuals that are connected by object property expression *'OPE'* only to individuals that are instances of class expression *'CE'*. The expression *'works in a given department'* is an universal quantification example, which states that a human resource works in a department of the company. Such sentence is mapped to an OWL 2 *'ObjectAllValuesFrom(:works-in :Department)'* statement.
If quantification is applied over a verb concept relating a concept with literals, then the expression is mapped to an OWL 2 *'DataAllValuesFrom(DPE DR)'* expression, which contains all those individuals that are connected by data property expression *'DPE' only* to literals in data range *'DR'*. The expression *'has a given skill'* is an universal quantification example stating that human resources have a skill. Such sentence is mapped to an OWL 2 *'DataAllValuesFrom(:has-skill string)'* statement.

3.3.2. Existential Quantification

An SBVR *universal quantification* scopes over a logical formulation with the following meaning: for at least one referent of the variable introduced by the quantification, the meaning formulated by the logical formulation for the referent is true.
If quantification is applied over a verb concept relating concepts, then the expression is mapped to an OWL 2 *'ObjectSomeValuesFrom(OPE CE)'* expression, which contains all those individuals that are connected by object property expression *'OPE'* to an individual that is an instance of class expression *'CE'*. The expression *'is boss of some employee'* is an existential quantification example, which is mapped to OWL 2 *'ObjectSomeValuesFrom(:is-boss-of :Employee)'* statement.
If quantification is applied over a verb concept relating a concept with literals, then the expression is mapped to an OWL 2 *'DataSomeValuesFrom(DPE DR)'* expression, which contains all those individuals that are connected by data property expression *'DPE'* to literals in data range *'DR'*. The expression *'has some skill'* is an existential quantification example, which is mapped to OWL 2 *'DataSomeValuesFrom(:has-skill string)'* statement.

3.3.3. At Most N Quantification

SBVR defines *at most n quantification* as a *"quantification that has a maximum cardinality 'n' - where 'n' is a positive integer - and that has the meaning: the number of distinct referents of the variable introduced by the quantification that exist and that satisfy a scope formulation, if there is one, is not greater than the maximum cardinality"*.
If quantification is applied over a verb concept relating concepts, then the expression is mapped to an OWL 2 *'ObjectMaxCardinality(n OPE CE)'* expression, which contains all those individuals that are connected by object property expression *'OPE'* to *'n'* different individuals at the most, which are instances of class expression *'CE'*. The expression *'is boss of 25 employees at the most'* is a quantification of this type, which is mapped to OWL 2 *'ObjectMaxCardinality(25 :is-boss-of :Employee)'* statement.
If quantification is applied over a verb concept relating a concept with literals, then the expression is mapped to an OWL 2 *'DataMaxCardinality(n DPE DR)'* expression, which contains all those individuals that are connected by data property expression *'DPE'* to *'n'* different literals at the most, in the data range *'DR'*. The expression *'has at most 1 size'* is a quantification of

this type, which is mapped to an OWL 2 *'DataMaxCardinality(1 :has-size integer)'* statement.

3.3.4. At Least N Quantification

SBVR defines *at least n quantification* as a *"quantification that has a minimum cardinality 'n' - where 'n' is a positive integer - and that has the meaning: the number of distinct referents of the variable introduced by the quantification that exist and that satisfy a scope formulation, if there is one, is not less than the minimum cardinality"*.
If quantification is applied over a verb concept relating concepts, then the expression is mapped to an OWL 2 *'ObjectMinCardinality(n OPE CE)'* expression, which contains all those individuals that are connected by object property expression *'OPE'* to at least *'n'* different individuals that are instances of class expression *'CE'*. The expression *'is boss of at least 1 employee'* is a quantification of this type, which is mapped to OWL 2 *'ObjectMinCardinality(1 :is-boss-of :Employee)'* statement.
If quantification is applied over a verb concept relating a concept with literals, then the expression is mapped to an OWL 2 *'DataMinCardinality(n DPE DR)'* expression, which contains all those individuals that are connected by data property expression *'DPE'* to at least *'n'* different literals in data range *'DR'*. The expression *'has at least 1 skill'* is a quantification of this type, which is mapped to OWL 2 *'DataMinCardinality(1 :has-skill string)'* statement.

3.3.5. Exactly N Quantification

SBVR defines *exactly n quantification* as a *"quantification that has a cardinality 'n' - where n is a positive integer - and that has the meaning: the number of distinct referents of the variable introduced by the quantification that exist and that satisfy a scope formulation, if there is one, equals the cardinality"*.
If the quantification is applied over a verb concept relating concepts, then the expression is mapped to an OWL 2 *'ObjectExactCardinality(n OPE CE)'* expression, which contains all those individuals that are connected by object property expression *'OPE'* to exactly *'n'* different individuals that are instances of class expression *'CE'*. The expression *'respond to exactly 1 boss'* is a quantification of this type, which is mapped to OWL 2 *'ObjectExactCar-*

dinality(1 :respond-to :Boss)' statement.
If quantification is applied over a verb concept relating a concept with literals, then the expression is mapped to an OWL 2 *'DataExactCardinality(n DPE DR)'* expression, which contains all those individuals that are connected by data property expression *'DPE'* to exactly *'n'* different literals in data range *'DR'*. The expression *'has exactly 1 size'* is a quantification of this type, which is mapped to OWL 2 *'DataExactCardinality(1 :has-size integer)'* statement.

4. Discussion

This section presents some considerations about the proposed patterns. First topic to address is about modalities. As stated in Section **??** - all SBVR rules have an associated modality. Furthermore, an alethic modality of necessity is assumed if no modality is explicitly specified; i.e., the rule *'each human resource has at least 1 skill'* is equivalent to the alethic rule *'it is necessary that each human resource has at least 1 skill'*. As a consequence, even though a pattern for the translation of the alethic modality would not be initially needed, the alternatives for the ontological modeling of deontic rules should be explored.
Necessary characteristics of concepts is another important concern: SBVR structural rules often propose necessary characteristics of concepts as stating that something is always true about all instances of the concept. However, it is important not to confuse the use of the word *'each'* with the specification of a universal restriction. Such expression implies the definition of a necessary condition which is modeled as an OWL 2 subclass axiom. So, the statement *SubClassOf(C CE)* relates class *'C'* with its necessary characteristic *'CE'* modeled as a class expression.
It should be highlighted that there is no mapping for the *implication* operation in the patterns proposed in Section **??**. OWL 2 language does not provide primitives enabling the modeling of an implication sentence. However, SWRL language extends OWL 2 by means of Horn-like rules in the form of an implication between an antecedent and a consequent, providing a way to express such operation. The modeling of the SWRL restriction over singleton concepts in Section **??** is an example of an implication sentence.

5. Conclusions and Future Work

This paper presents a set of explicitly formalized ontology design patterns, providing a way to obtain an OWL/SWRL ontology by applying a set of metamodel transformation rules over SBVR specification of a business domain. The goal is to provide a ready to use set of transformations for allowing domain experts to quickly obtain an ontology from a natural language specification of their knowledge of the domain. Such facility is a necessary and fundamental intermediate step into the overall research goal of the authors: to provide an start-to-end environment for the development of ontology-driven information systems, where the ontologies encapsulate the declarative specification of business logic, (1) enabling unambiguous representation of knowledge, (2) providing reasoning services to the software system, and (3) improving the management of highly dynamic environments.
Regarding previous studies in this topic, a first proposal is explored in **?**. However, the mappings between the first versions of SBVR and OWL just are illustrated by an example. Then, transformations are not explicitly formalized and consequently they cannot be generalized for other situations. In the other hand, the works presented in **?** and **?** follow a formally grounded approach by stating the logical foundations of the translations between the underlying theories of SBVR 1.0 and OWL 2: i.e., from ORM to First Order Logic (FOL). But SBVR does not reflect exactly ORM and OWL 2 is based in a fragment of FOL. In consequence, ORM to FOL transformations can not be used directly to map SBVR expressions into OWL 2 statements. **?** proposes a reversible mapping between SBVR and OWL 2 with the goal to allow to map a SBVR vocabulary to OWL 2 statements and back again without loss of semantic information, offering a way to exchange SBVR vocabularies between tools. However, the proposal considers a very small set of the elements usually involved in the stating of complex business rules. An study analysing the suitability of a subset of the SBVR metamodel for representing OWL 2 ontologies is depicted in **?**. Since the analysis follows the opposite direction to that proposed in this work, specific characteristics of the source metamodel of the transformations are overlooked. First, the study does not take into account that the transformation of several SBVR expressions in OWL 2 statements must be made according to the type of the involved entities. Secondly, it does not define a way to fill the semantic gap between SBVR and OWL 2. **?** and **?** also present a set of structural based transformations that allows the automatable generation of an OWL

2 ontology from SBVR specifications of a business domain. Conducting of an experiment aimed at obtaining empirical evidence about the feasibility of such proposals has also allowed to recognize a set of transformations that could be substantially improved in order to better reflect the semantics of the involved metamodels(**??**). The recent publication of an improved version of SBVR language and the empirical evidence found by the aforementioned experiment has motivated the development of this work.
Regarding the evaluation of the proposed transformations, a *"generate/test"* validation approach has been adopted given the inherently iterative nature of the design science. While often an optimal solution remains intractable for realistic problems, an heuristic strategy provides a good design that serves for business purposes. Such perspective in the finding of a "good" solution opens the question of how "goodness" can be measured. An alternative for measuring the goodness of the transformations is to compare the proposed solution with existing artifacts addressing the same problem (**?**). The valuable findings obtained by following such approach on the empirical evaluation of the previous version of the transformations (**?**) have motivated the adoption of the same strategy in the assessing of the set of patterns presented in this paper. A detailed analysis of the experience of several study groups applying the patterns will be presented in the near future.
Such analysis involved the performing of an ontology quality evaluation task by means of OQuaRE (**??**), a framework conceived for that purpose and based on the SQuaRE standard for software quality evaluation (**?**). OQuaRE considers ontologies as artifacts obtained by means of a building process and evaluates them independently of any particular development process, defining a quality model divided into a series of dimensions. The set of characteristics scores is the quality assessment result, enabling the identification of strengths and flaws of an ontology. However, next studies will consider the incorporation of the proposal of **?**: an evaluation framework for the automatic structural assessment of taxonomic ontologies by means of a set of formalised metrics. The work presents an algorithmic methodology for building taxonomies with formally specified content, which the authors claim to be very valuable in settings where taxonomies are developed on the fly by non-computer literate users. Beyond that, a formal evaluation method would provide greater rigour to the proposed patterns. An interesting starting point for a future work is to ensuring semantic equivalence between the ontology and the original set of SBVR statement by means of the definition of a comorphism between the logics theory underlying both metamodels.

Furthermore, it is important to highlight that although the latest version of OWL provides a set of constructors which enables the mapping of most SBVR concepts, some important patterns - *properties*, *partitive relations*, *associations*, *categorizations*, etc. - are still missing. Then, a short-term future work is focused on the definition of patterns for the translation of the aforementioned SBVR expressions. Introduction of such patterns will also require the definition of a procedure to enforce the syntactic conditions imposed over the produced ontology structure, in order to ensure that it can be interpretable under the Direct Semantic: i.e., just OWL 2 DL ontologies should be produced as a way to maintain soundness and completeness of the reasoning procedures. A medium-term future work is about the analysis of the benefits of integrating the patterns approach to EDON (**?**), an evolutionary method for building ontologies intended to be used as an structural conceptual model of an information system. In a early stage, patterns would be useful in a method that makes use of ontologies to encapsulate business rules as a means of raising flexibility, extensibility, and ease in maintaining of enterprise software systems. The implementation of a prototype that performs the automatable translation of SBVR business domain specifications to OWL 2 ontologies by implementing the set of proposed patterns is a practical pursued goal.

Acknowledgements

The authors are grateful to Universidad Tecnológica Nacional (UTN), Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), and Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT) for their financial support.

Figure Captions

F1GURE 1. SBVR Core Concepts
F2GURE 2. SBVR Quantifiers
F3GURE 3. SBVR Logical Operations
F4GURE 4. OWL 2 Entities
F5GURE 5. OWL 2 Expressions
F6GURE 6. OWL 2 Axioms
F7GURE 7. General structure of INTCO company
F8GURE 8. Role example