$\mathcal{MDR}$Ontology : An Ontology for Managing Ontology Changes Impacts on Business Rules

Amina Chniti[1,2], Patrick Albert[1], Jean Charlet[2,3]

[1] CAS France, IBM
{amina.chniti,albertpa}@fr.ibm.com
[2] Inserm UMRS 872 Q.20, Ingénierie des Connaissances en Santé, Paris, France
Jean.Charlet@upmc.fr
[3] AP-HP, Paris, France

Abstract. In this paper, we focus on the impact of ontology changes on rules, or more specifically on business rules authored over ontologies. For this we develop the $\mathcal{MDR}$ framework ($\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair) based on the $\mathcal{MDR}$Ontology and a set of rules. As rules depend on the ontology entities, ontology changes may make them inconsistent. The $\mathcal{MDR}$ framework analyses an ontology change to detect its impact (*e.g.* inconsistency) on rules and propose solutions, called repair, to repair the change impact. The $\mathcal{MDR}$Ontology includes, among others, a change model, ontology changes classification and the impact of such changes on rules.
Keywords: Ontology Change, Business Rule, Change Impact, Rule Inconsistency

1 Introduction

Integration of ontologies and rules is a research area of interest especially for the semantic web community. Ontologies changes also take the attention of many research works, which propose solutions to manage the changes of the ontologies while maintaining their consistency. In this paper we focus on the integration of OWL-DL ontologies and business rules and on the impact on ontology changes on rules authored over ontologies. Business Rules are a description of a business policy, encoded in a natural controlled language. They define or specify constraints of some aspect of the business[4] and enable automating business decisions. An example of a business rule is given in the following :

IF the yearly income of the borrower is less than 20000
and the amount of the loan is more than 60000
THEN reject the loan request;

The purpose of integration ontologies and business rules is to bring the expressiveness of OWL ontologies to business users by means of business rules authored in a controlled natural language. In [3] the authors present a prototype which enables authoring and executing business rules over OWL ontologies.

[4] http://www.businessrulesgroup.org

This prototype, called *OWL plug-in for WODM*, is based on the Business Rule Management System (BRMS) WebSphere Operational Decision Management (WODM). Due to this new dependency of business rules on ontologies, changes to this ontologies have an impact on them and may cause inconsistencies.

To cope with the problem of inconsistencies, caused by ontology changes, we develop[5] a new approach, $\mathcal{M}$odel-$\mathcal{D}$etect-$\mathcal{R}$epair ($\mathcal{MDR}$), which supports consistency maintenance of business rules when ontologies evolve. The general idea of $\mathcal{MDR}$ consists of modeling the ontology change, detecting the inconsistencies that could be caused by it and propose solutions, called repairs, to resolve the inconsistencies.

2 $\mathcal{MDR}$Ontology

The *MDROntology*, represents the different entities that are important to detect the impact of ontology changes on rules. It contains the concepts to model business ontology changes, business rule inconsistencies and inconsistency repairs. The $\mathcal{MDR}$Ontology contains more than one hundred concepts, the most important are:

- `Entity`: represents the entities of an ontology (*i.e.* concept, property, and individual);
- `ChangeObject`: represents the OWL-DL construct on which the change operates. Six categories of OWL-DL constructs are represented based on the W3C guide [6]:
  - `ClassAxiomConstruct`: *rdfs:subClassOf*, *owl:equivalentClass*, *owl:disjointWith*;
  - `ClassDescription`: *owl:oneOf*, *owl:intersectionOf*, *owl:unionOf*, *owl:complementOf*;
  - `PropertyRestriction`:
    - Cardinality restriction: *owl:maxCardinality*, *owl:minCardinality*, *owl:cardinality*;
    - Value restriction: *owl:allValuesFrom*, *owl:hasValue*, *owl:someValuesFrom*.
  - `PropertyRelation`: *owl:equivalentProperty* and *owl:inverseOf*
  - `PropertyCharacteristic`:*owl:FunctionalProperty*,*owl:InverseFuctionalProperty*, *owl:SymmetricProperty*,*owl:TransitiveProperty*;
  - `RDFConstruct`: *rdfs:subPropertyOf*, *rdfs:subClassOf*, *rdfs:domain*, *rdfs:range*.
- `Change`: represents ontology changes that have impacts on the rules. We distinguish between three categories of change
  - `OWLConstructChange`: consists of changes that impact a construct. We define six categories of `OWLConstructChange`which are `ClassAxiomConstructChange`, `ClassDescriptionChange`, `PropertyRestrictionChange`, `PropertyRelationChange`, `PropertyCharacteristicChange` and `RDFConstructChange`.

[5] This work is partially founded by the European Commission under the project ONTORULE (IST-2009-231875).

[6] http://www.w3.org/TR/owl-ref/

  - `StructuralChange`: consists of adding or removing entities;
  - `RenameOntologyEntity`.
- `Rule`: represents the business rules authored by business users, which are impacted by a change. They have a `ConditionPart` and an `ActionPart` and they are authored over the ontology entities (represented by the concept `Entity`);
- `Inconsistency`: represents the inconsistencies that could impact rules due to an ontology change and we focus on 4 types on inconsistency [6]; `Contradiction`, `Domain Violation`, `Invalid Rule` and `Rule Never Apply`.
- `Repair` : represents the repairs that are proposed to resolve an inconsistency.

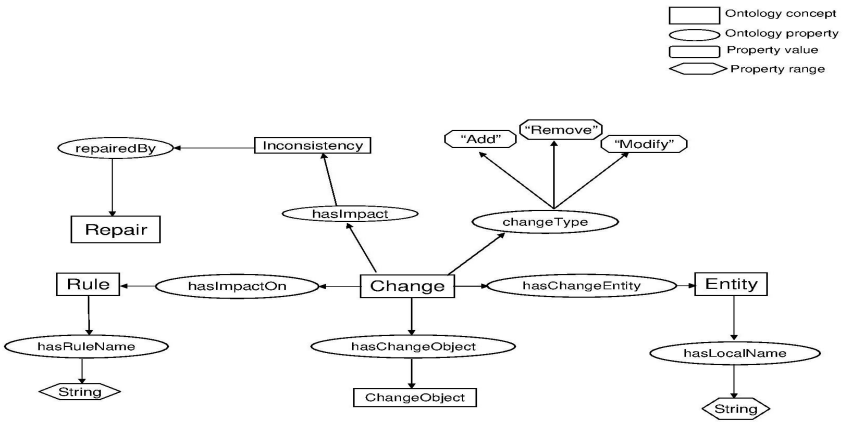

Fig. 1. $\mathcal{MDR}$Ontology main concepts

Within the $\mathcal{MDR}$Ontology a change is defined as follows (see Figure 1):

- a change has a *ChangeObject* which is the OWL construct impacted by the change (*i.e.* owl:subclassOf, owl:allValuesFrom, owl:oneOf...);
- a change is applied to a `Change entity` that can be a concept or a property;
- a change has a change type: {Add, Remove, Modify};
- a change has impact on `Rule`;
- a change has impact an *Inconsistency* hat is repaired by a *Repair*.

Nevertheless, depending on the change to apply to the ontology, more information should be provided to enable detecting its impact on rules. For example, in case of an enumeration change, the new collection of the enumeration is provided (see Figure 2), in case of a rename entity change, the new name of an entity, in case of a range change, the new range of a property...

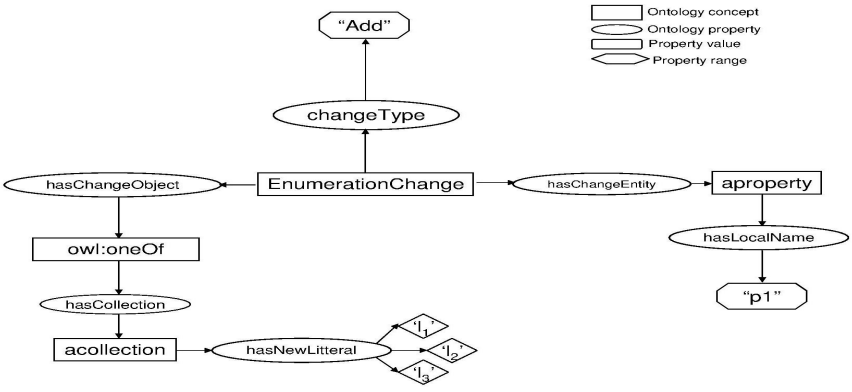

Fig. 2. Enumeration change template

3 $\mathcal{MDR}$ Framework

The $\mathcal{MDR}$ approach is inspired by ONTO-EVO^{A}L [4], which deals with the consistency maintenance of OWL ontologies while they evolve. In this context

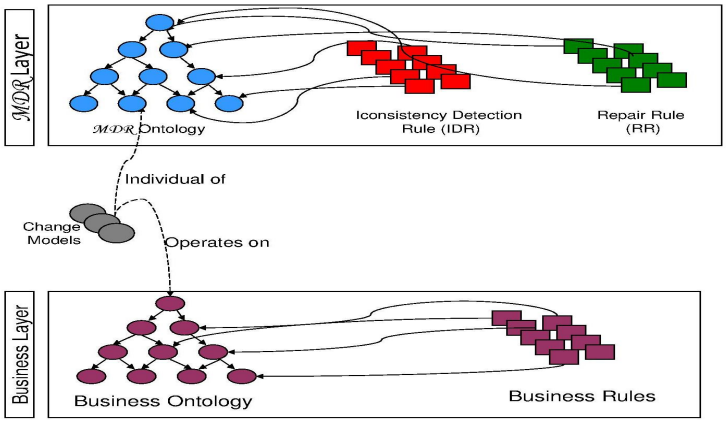

Fig. 3. General architecture

we assume that inconsistencies in the ontology have been resolved. Our focus is on the inconsistencies that emerge in the rule set as a consequence of the change in the ontology over which the rule set is authored.

The general idea of the $\mathcal{MDR}$ approach is: (1) to model the changes that the business user wants to apply to an ontology, (2) to detect which changes produce inconsistencies and (3) to provide solutions to repair them.

Figure 3 gives an overview of our approach. We distinguish two different layers : the Business Layer and the $\mathcal{MDR}$ layer. The Business layer consists of the business ontology which is subject to changes and the business rules authored over this ontology by the business user. These business rules are possibly impacted by the ontology changes. The $\mathcal{MDR}$ layer depicts the architecture of our system. It contains the $\mathcal{MDR}$Ontology which models our problem domain (*e.g.*ontology change, rules inconsistencies and inconsistency repairs). Additionally, it contains two rule sets, the Inconsistency Detection Rules (IDR) and the Repair Rules (RR). These rules are authored over the *MDROntology* using the *OWL plug-in for WODM*.The IDR detects which inconsistencies in the business rules are caused by the change, while the RR proposes the appropriate repairs for any combination of a change and an inconsistency. These repairs can be either changes to the business rule set or to the business ontology. The $\mathcal{MDR}$

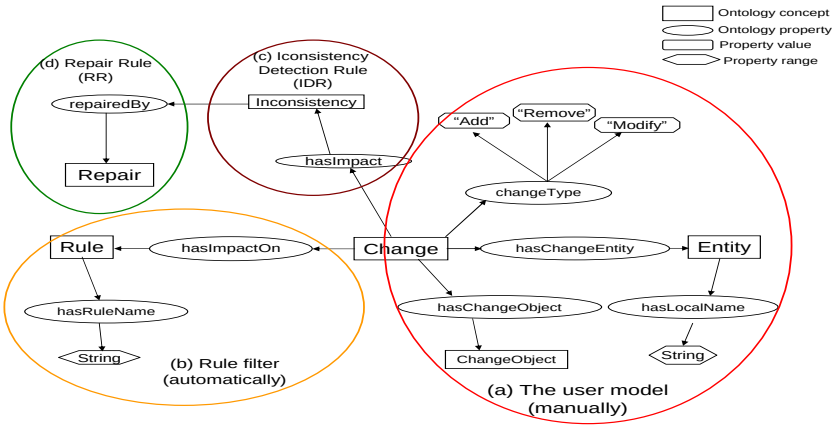

Fig. 4. $\mathcal{MDR}$ Process

process to detect the impact of an ontology change on rules is as follow (see Figure 4); (a) It starts by modeling the required business ontology changes . This is done by the business user, by creating individuals of the $\mathcal{MDR}$Ontology. These change instances refer to entities in the business ontology that are subject to change. This change model serves as input for the consistency process. (b) The Rule Filter module is executed to detected rules that are impacted by a change. This module updates the change model by adding the impacted rules. (c) The updated model is used to trigger the IDRs, in order to detect the type of inconsistencies on the impacted business rules. This step updates the change model with the detected inconsistencies for each impacted business rule. (d) Fi-

nally, this new updated model is used to trigger the RRs in order to find possible repairs for each inconsistency.

4 Experimentation

The $\mathcal{MDR}$ approach is implemented as a WODM plug-in called the *Change Management plug-in*. To illustrate our work, we present a business scenario that stages two agents representing business users who are involved in the process of consistency maintenance.

Marc is the business analyst. His mission is to formalize the business knowledge. He has to make sure that the business model is correct, complete and valid. Alice is the domain expert. She is in charge of authoring the business rules. She understands quite well their formalization and uses business rules tools to update, create and test them.

The business ontology we will use in this section is an extract of the use case used in the ONTORULE project [7]. The use case, provided by Audi, models a scenario of a car development. One domain constraint that must be verified is

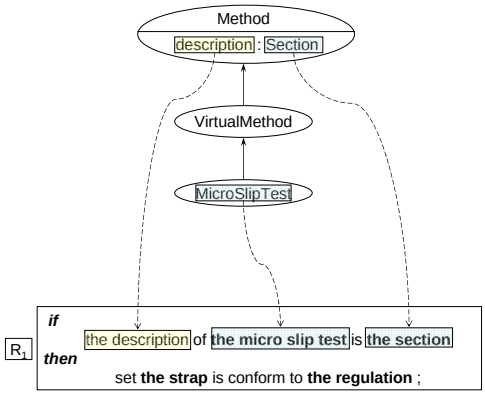

Fig. 5. Example of a portion of the Audi use case ontology and the rule authored over it.

that the **SolutionConcept**s are conform to **Regulation**s. The **MicroSlipTest** is a **Method** that is used to test the quality of a seat belt. When a **Method** is based on a **Section**, the required **SolutionConcept** for the **Process** that uses this **Method** must be *conform to* a **Regulation**. For this, we authored the rule described in Figure 5 using the *OWL plug-in for WODM*. This rule expresses,

[7] http://ontorule-project.eu/

that the required **Strap** (which is a **SolutionConcept**) for the **MicroSlipTest** `conforms to` a **Regulation**.

Let us consider an ontology change that consists of removing the subclass relation between **VirtualMethod** and **MicroSlipTest**. In this case, the rule R_1 will be invalid as it tests the property `description` of **MicroSlipTest**, an inherited property that will be lost after the change application.

The application of the $\mathcal{MDR}$ approach will be as described in the following :

1. **Change modeling**
   Marc requests the change by modeling its structure. An instantiation of the change concept corresponding to the subclass change that he wants to apply is given in Figure 6.

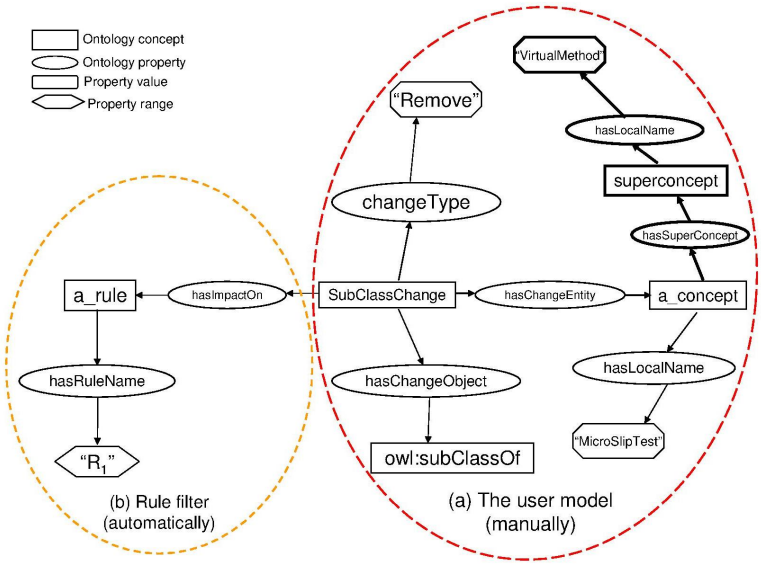

Fig. 6. Subclass Change structure

When Marc activates the *Change Management plug-in*, a list of the requested changes is shown. Then, when Alice selects the change(s) to apply, the rule filter is triggered to detect the impacted rules. The impacted rules are detected based on the change entity and the change object. For example, in our case the impacted rules are the one using the change entity with an inherited property.

2. **Inconsistency detection**
   Depending on the modeled change, the IDRs are executed. Each change has to satisfy constraint(s). For the subclass change, the constraint to verify is

that, in the rule set, there is no rule using the change entity with a property of its super class. If this constraint is not satisfied an inconsistency of type `Invalid Rule` will be detected by the following rule.

```
IF change.objectChange = subclassConstruct
&& change.type = "Remove"
&& change.hasImpactOn(a_rule)
&& a_rule.hasRuleEntity(changeEntity)
&& changeEntity.hasRuleProperty(prop)
&& super_concept in changeEntity.superClass
&& super_concept.hasProperty(prop)
THEN
change.add_inconsistency(invalid_rule);
```

3. Inconsistency repair
   Depending on the change to apply and the detected inconsistency one or more repair are proposed. For this example, the proposed repairs are :
   - an alternative ontology change; a proposition of additional ontology change that will enable to avoid the inconsistency;
     In this case, one of the proposed repairs is to add the missing property (i.e `description`) to the concept **MicroSlipTest**. This repair is proposed by the following RR.
     ```
     IF invalid_rule in change.impact
     && change.changeObject = subclassConstruct
     then
        addMissingProperty();
     ```
   - common repairs, which consist of removing the rule or the rule part that causes this inconsistency. These repairs are proposed independently of the requested change and the detected inconsistency.

5 Conclusion

In this paper, we presented the $\mathcal{MDR}$ framework built on top of the $\mathcal{MDR}$Ontology. This approach enables the impact of ontology changes on rules authored OWL ontologies. This approach analyses the impact of a change, detects the inconsistencies that could be caused by its application and proposes repairs to resolve these inconsistencies. Nevertheless, in the current state of the work, the IDRs detect only three types of inconsistencies which are the Invalid Rule, Domain Violation and Rule Never Apply caused by enumeration change or subclass changes.

The IDRs defines the constraints that each change should verify. The development of this kind of rules is not an easy task as it is necessary to cover the changes that impact an ontology and to define manually the constraint that each change should verify. It would be rewarding to develop a module enabling to automatically generate these rules depending on the change to apply in a way to detect more inconsistencies that could impact business rules.

Another important point to improve is the change modeling step. It would be interesting to develop a module that detects automatically an ontology change and models its structure, which will avoid the change modeling step.

References

1. G. Antoniou, C. V. Damasio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, and P. F. Patel-Schneider. Combining rules and ontologies: A survey. *Technical Report IST506779/Linkoping/I3-D3/D/PU/a1, Linkoping University*, 2004. http://rewerse.net/publications/.
2. B. Berstel and M.Leconte. Using constraints to verify properties of rule programs. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on Software Testing, Verification and Validation.*
3. A. Chniti, S. Dehors, P. Albert, and J. Charlet. Authoring business rules grounded in owl ontologies. In M. Dean et al. (Eds.), editor, *RuleML 2010 : The 4th International Web Rule Symposium: Research Based and Industry Focused.* LNCS 6403, Springer-Verlag Berlin Heidelberg 2010, 2010.
4. R. Djedidi and M.A. Aufaure. Onto-evoal an ontology evolution approach guided by pattern modelling and quality evaluation. *Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2010)*, 2010.
5. T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits. Reasoning with rules and ontologies. *Reasoning Web 2006*, pages 93–127, 2006.
6. M. Fink, A. El Ghali, A. Chniti, R. Korf, A. Schwichtenberg, F. Lévy, J. Pührer, and T. Eiter. D2.6 consistency maintenance. final report. *ONTORULE Delivrable, http://ontorule-project.eu/deliverables.*, 2011.
7. Aikaterini K. Kalou, T. Pomonis, Dimitrios A. Koutsomitropoulos, and Theodore S. Papatheodorou. Intelligent book mashup: Using semantic web ontologies and rules for user personalisation. In *ICSC 2010, Fourth IEEE International Conference on Semantic Computing.*
8. Mark H. Linehan. Ontologies and rules in business models. *Enterprise Distributed Object Computing Conference Workshops, IEEE International*, pages 149–156, 2007.
9. L. Stojanovic. *Methods and Tools for Ontology Evolution.* PhD thesis, University of Karlsruhe, 2004.