An Editor for Micro-Concept Rules Design

Lyazid Sabri[1], Abdelghani Chibani[1], Jerome Beck[2], Gian Piero Zarri[1],

Yacine Amirat[1], Jean-Sebastien Brunner [2], Patrick Gatellier[2]

[1]University Paris-Est , Laboratory of LISSI Paris XII
[2]Theresis Innovation Center - Thales Security Solutions & Services Campus
{lyazid.sabri, chibani, gian-piero.zarri,amirat}@univ-paris12.fr
{jerome.beck, jean-sebastien.brunner, patrick.gatellier}@thalesgroup.com

Abstract. Weak support tools for the development of rule bases constitute the main motivations leading the researchers to find new ways to standardize rule languages. For example, the lack of the notion of 'variable' in OWL makes it very difficult to rely on the W3C languages in their 'native' form to build up 'real' inference engines for rule processing. For practical applications, especially in an industrial context, a solution to the problem of finding an efficient 'rule engine' for executing the so-called 'business rules' consists then in making use of 'expert systems' tools like JESS: the production rules engines based on the RETE algorithm seem to be, in fact, particularly well adapted in this context. We describe in this paper our MCL and show a demo of MRE allowing us to make use of some syntactic/semantic structures usually not handled by business rules languages, like handling properties with multiple values. These tools constitute then an attempt to support any possible user in the task of creating and editing effective business rules in a simple, safe and valuable manner.

Keywords: Ontology, Rule Standard Language,

1 Introduction

A reasoning system based on the "production rules" paradigm makes use of an "inference engine" (or "rule engine") to try to unify the "condition" part (the "antecedent") of the rules with information included in a "working memory" where the "facts" are inserted. The production rules paradigm – introduced by the logician Emil L. Post in the early '40 [1] – conforms to the classic "if/then" rule format. If the unification is successful, the corresponding rule "fires" and new knowledge (inference) is derived by executing the operations proper to "action" part (the consequent) of the rule. It is possible that the production of this new knowledge lead to some changes into the working memory: in the next "cycle of operations" performed by the inference engine, other rules can fire producing then new knowledge and inducing new changes in the working memory, etc. It is then obvious that the inference engine plays a central role in the implementation of any project that makes use of business rules according to the production rules paradigm.

Moreover, we can note that – independently from the specific business rules applications – "rules" and "production rules" play a fundamental role in many other scientific/technical domains. For example, interoperability is one of the primary goals of the Semantic Web research and the work on rules and on their standardization represents a key move towards the realization of that goal. Authors in [10], e.g., present a Framework-based production rule for the discovery and composition of Web services using the CLIPS [12] rule engine. They use rules to: a) realize the matching between semantic search queries and OWL-S [11] descriptions of Web services, and b) implement the algorithm for service composition. Thus, the rules are considered as composite services or Template where the premise part of a rule contains a set of conditions.

As we will see in more details in the next Section, however, the situation in the Semantic Web rule domain is still particularly moving, in spite of the emergence of several " reasoner" like RACER [2], Pellet [3], Fact++ [4], KAON [5], JENA [6], Hoolet [7] and so on that, all based on the weak "inference by inheritance" reasoning paradigm, can only solve, in practice, the most common classification ("subsumption") problems. We can also note that, in a strict W3C languages (OWL[23], RDF(S)[29]) context, building up 'true' rule systems is a really complex task given that i) on the one hand, the lack of the notion of 'variable' in OWL makes it impossible to rely on this language in its 'native' form to build up 'real' inference engines for rule processing, and ii) on the other hand, no support for rules and rule processing has been introduced in the standard descriptions of these languages at the time of their conception. The consequence is that the whole Semantic Web rule domain seems to be in an early stage of development. Languages like the Semantic Web Rule Language

(SWRL) [13] – all based, roughly, on extensions of the inferential properties of Horn clauses and (Unary/Binary Datalog) to deal with OWL-like data structures appear to be, for the time being, as quite limited with respect to the range of their possible applications and particularly complicated to be used in practice.

The paper is organized as following; Section 2 will discuss related work and will introduce some important methodological issues. Section 3 will describe the SEMbySEM [20] project and the proposed semantic language, Section 4 will emphasize the innovations introduced by our rule editor and the paper ends by evoking some future perspectives.

2 Business Rules Vs Semantic Web rules

Semantic web and the business rules communities have their roots in the field of knowledge representation in Artificial Intelligence. They are both looking for mechanisms and models to formalize and reason with domain knowledge using logic and logical inference. In these last years, we observe an explicit adhesion from researchers to the semantic web community projects (ontology research field and reasoning on the web), and the standardization efforts of the semantic web community at the W3C and the OMG. We completely agree with Silvie Spreeuwenberg observation [26]; where she said "*the business rules community and semantic web community talk about the same things, but by people with a different background; the business rules community is driven by the practical experiences of business people and business consultants while the semantic web community is a vision of scientists driven by (mostly) scientific publications*".
SWRL is the first W3C accepted proposal for setting up a Semantic Web rule language by combining sub-languages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language. This rule language enhances OWL by allowing a user to create 'if-then rules' written in terms of OWL classes, properties and individuals. Making use of the 'open world' assumption, it does not support 'negation by failure' (NAF) – but also, among other things, it does not support classical negation, disjunctions and non-monotonic. SWRL is 'semi-decidable'; moreover, SWRL variables can only be bounded to known individuals in a knowledge base (in OWL ontology). This last property is too restrictive for many applications where variables in the rules must also be bound, when a specific instance is unknown, to the general concept subsuming the specific instance [14]. We can add that the current OWL/SWRL [9] editors, which are based on formal logics and general purpose graphical user interface, are often seen as too complicated and confusing for domain experts with no background in formal methods [15]. ACE View [16] tries to resolve this problem by introducing a novel approach with respect to the OWL/SWRL paradigm that makes it radically different from the current SWRL editor, with respect now to REWERSE [21], using this tool to edit a complex rule appears as very complicated. It must be noted, however, that REWERSE represents an easy mechanism to translate rule from R2ML [22] into JESS[17].

3 A seamless model for business rules

For practical problems, especially in an industrial context, the (at least provisional) solution to the problem of finding an efficient 'rule engine' for executing the business rules consists then in making use of 'expert systems' tools like JESS and DROOLS [18], i.e., of production rules engines based on the RETE algorithm. This is the solution adopted in the SEMbySEM project; SEMbySEM aim to describe and manage an environment where heterogeneous objects can evolve over the time and interact with each other, in particular in the context of the internet of objects. Indeed, SEMbySEM will provide designers and business managers a full framework, according to a semantic approach, for managing their assets and infrastructures,— enabling the development of a generic software allowing the aggregation of information from "communicating" objects (RFID tags, industrial sensors, servers, simulated objects, devices, etc.) and applications (video acquisition system, supervision system, IT monitoring system, etc.) under an end-user ergonomic form and actions from the end-user on these objects and applications. This software allows several actors to get "system awareness" of their working system and perform actions on it.

In order to cope with the heterogeneity of knowledge and rule bases, SEMbySEM proposes a semantic model called µConcepts (Micro-concept) that allows to make use of some syntactic/semantic structures usually not handled by business rules languages, like handling properties with multiple values, a way to edit rules as naturally and easily as possible. It is then indispensable to set up a rule editor whose main function is that of facilitating as much as possible the task of the users, i.e., to allow them to write and modify the business rules in a simple, safe and effective manner. In particular, the rules are able to directly address the semantic objects of the model (concepts, instances, properties) and benefit from the logic of the model.

OWL is the reference standard in the domain of semantic web, where it is possible to describe entities and their relationships and performing reasoning over them. In the infrastructures and assets management OWL standard is not very well suited, because in OWL we deal with open world description where ontology of a domain is interrelated with other ontologies from different domain. Otherwise, in OWL we can't assume the uniqueness of concepts names: i.e. two different instances can refer to the same object. In the management use case, the main characteristic of SEMbySEM Micro-Concept model is its suitability for a closed world (which is the same assumption in Data Bases World) and concepts name must be unique to avoid inconsistencies. Compared to OWL, constraints in Micro-Concept model must be checked before addition to guarantee model consistency and if a fact is not asserted, it is considered as false, while in OWL it is considered as unknown. For interoperability issues, syntactic/semantic structures of the Micro-Concept model is built on top of RDF(S) which make it semantically and syntactically compatible with RDF(S), the only exception being that instances and classes are disjoint (an element must not be simultaneously an instance and a concept). Besides this point, an RDFS document must directly be used as a micro-concept model [28]. It may be extended using some attributes specific to the micro-concept model. Thus, OWL representation is based on description logic while Micro Concept is closer to conceptual models such as Entity/Relationship model.

3.1 µConcepts rules

Rules Based Systems have been widely used, primarily, in the context of Expert Systems applications. With respect to the user interaction, these systems focuses on monitoring applications and do not allow then to perform direct actions on the target systems, i.e., they are unable to deal with systems defined as coherent set of objects and grounded on a semantic abstract representation of the system to be supervised or managed.

This language allows user to construct rule following the classic paradigm (see example in section 3.3)

IF *condition* THEN *action*

- condition follows the syntax formalized hereafter:

$$C\ (\sum_0^n S) \qquad (1)$$

Where C may be a Concept in the current ontology (i.e.: a class in the standard W3C languages) or a 'variable' that can be bound to a class. We mean here that a variable references an instance that has a class type.

S may be either a single or a set of constraints applied in order to match Concept instances following some restrictions or/and a variable used to refer a constraint. A variable can also be used as a kind of class. A rule declared without restrictions enable us to match every instance of given Concept. All the types of variables declared in C or S will be automatically deduced from the matched referenced element. Note that the results of the evaluation of Eq. 1 may be linked to a variable V, which can be accessed directly within the current rule as included in the action part. Thus, we can rewrite Eq. 1 as follows

$$V := C\ (\sum_0^n S) \qquad (2)$$

- The 'action' represents operations on instances to be done when a rule is triggered. These operations, O, are the classical ones: create or remove instance in/from the knowledge base, update Property values or global variables; the most important corresponds to a "do" action on an instance J. All these operations follow the pattern by Eq. 3.

$$O\ (\sum_1^n J) \qquad (3)$$

In this manner, we support some missing features in OWL 2 and its related rules language SWRL: the capability to define and execute actions on the objects, the possibility of defining inequality constraints on the properties and the definition of properties calculated as a combination of other properties. It is known, in fact, that RDF(S) and OWL are 'binary' languages, where a property is strictly limited to represent a binary relationship linking two individuals or an individual and a value (Functional Property). This inhibits, in practice, of making use of RDF(S)/OWL when the situation to be represented is characterized by a (relatively) high level of complexity.

Multivalue Property (see example below) can be associated with actions and can be considered as an elegant way of getting rid of the 'binary' limitations associated with W3C languages.

Note, in this context, that we have introduced in MCL the keyword "one" to be used only with this kind of property (therefore, not with Functional Properties) in order to select one value from all values performing multi instances matching.

The following example of rule in the context of the SEMbySEM general use case concerning the transportation of hazardous material by rail in order, e.g., to compare all the 'fixed landmarks' corresponding to locations where the wagons must be checked with each specific landmark associated in real time with each specific wagon.

```
if
{
?w := Wagon(?tPos := hasPosition, ?curLM := one (nearLandmark) ),
?lm := Landmark(?lPos := hasPosition, isnot(?curLM)),
……
}
```

The rule above depicts the role of this specific keyword "one" when comparing each Landmark with each Landmark of each Wagon.
In this example, "Wagon" is concept that belongs to the domain of "nearLandmark" property which is a "set" property (i.e. can have many values for each instance). We select all instances of "Wagon" and then all the values of "nearLandmark" for each instance. We check after if a condition applies to these values. For instance, if there are two wagons and every wagon has three values for "nearLandmark" property, then the rule can be activated six times. In natural language, this rule is made as follows: looking for all instances of "Wagon", then selecting all the values of "nearLandmark" for each instance in such a way to...

A particular attention has been devoted to the definition of the set of properties that can be specifically associated with actions; these properties could be inserted in a proper sub-branch of the "Properties" sub-hierarchy. The properties of an "elementary action" can be reduced to the following seven: Agent, Object, Source, Beneficiary, Modality, Topic and Context. We can note that different, more general properties like Cause, Goal, Coordination, Alternative etc. can also be introduced; in reality, these properties represent ways of linking together elementary actions. Note that all these properties are optional, with the exception of "ActionAgent", whose presence is mandatory in the definition of each of the actions associated with the model and in the description of the corresponding instances.

In this paper, we do not present in detail the syntax of rules; as an example however; the following Eq. 4 describes the syntax for variables. In Eq. 4, the name of the variable is formed of alphanumerical characters or underscores characters with a starting '?'. The first character after the '?' must be a letter or an underscore. We have also used operators like "and","or", "not", arithmetic symbol (>, ==, etc) and different type: string, integer and so on with a defined syntax.

$$\langle Variable\ Name \rangle ::= \text{'?'}[a\text{-}zA\text{-}Z\_][a\text{-}zA\text{-}Z0\text{-}9\_\$]^* \qquad (4)$$

In Table 1, we provide a fragment of the µConcept grammar allowing the edition of roles in µConcept language format.

```
rulesmcl ::= DECLARATIONS decls_var   | DECLARATIONS decls_var ruleset   | ruleset ;
ruleset ::=rule   | rule ruleset ;
rule ::= RULE STRING CONDITIONS blockcondition ACTIONS   blockactions END
        | RULE STRING attributs CONDITIONS blockcondition   ACTIONS blockactions END ;
decls_var ::= decl SEMILICON     | decl SEMILICON decls_var ;
attributs ::= PRIORITY AFFECT INTEGER  ;
blockcondition ::= predicat  SEMILICON   | predicat SEMILICON blockcondition ;
predicat ::=  affectation     |    condition  ;
condition ::=expres     | filter    ;
expres ::= container  LPAR RPAR     | container LPAR members RPAR  | ONE IN expres
            | container IN expres ;
atomic ::= INTEGER | DOUBLE | VARIABLE|PROPERTY  | STRING | duration |datetime |TRUE | FALSE
            | CONCEPT | COUNT LPAR CONCEPT RPAR  ;
members ::= predicat | predicat COMA members ;
none ::= NONE LPAR exist_members  RPAR ;
exist ::= EXISTS LPAR exist_members  RPAR ;
exist_members ::= exist_expres | exist_expres OR exist_members ;
```

Table 1. Partial fragment of the µConcept grammar.

Mapping each Concept C and each constraint S while editing the rule with the ontology allow us to verbalize rules in English language as shown in Fig 3, note that the reverse operation is not in our scope actually. When there are no constraints, a rule consists of a C ( ) pattern. For example the simple case " if all of C" (i.e.: if C is equal to 'Location'), thus, we verbalize the pattern as "if all Locations…" This mapping between rule and ontology provide an easy way to understand what user is editing in order to validate or to rewrite the rule.

3.2 Rules example: Asset Traking in rail ways

In this section we provide an example of rules design using µConcepts semantic model and rules language. The example concerns the shipments of material by railways. The tracking of material during railways travels is important in order to enhance the protection and the security of the shipped material and train. In general way, the targeted rules concerns for example carrying out real time knowledge about the position of each material carrying railcar on a geographic map, for every train movement, for all stops and for all the start again and when an event occurs on the cargo like opening of a sealed door, increasing temperature, increasing or decreasing pressure in a tank, decreasing gauge for liquids, etc. In table 2, we provide the expression of rule which infers an Alarm triggering when a cargo is late and the door is open.

```
if
{
?w := Wagon(datetime("now")>departureTime + duration("5m"), ?doorSensor :=
hasDoorSensor, ?notif:= all(isSubjectToNotification)),
?doorSensor(isOpened == true),
not(Customer_Arrival in ?notif())
}
then
{
Location_Alarm ?alarm := createInstance(Location_Alarm);
?alarm->hasPosition := ?w->hasPosition;
?alarm->hasDate := datetime("now");
?alarm->doorOpened := true;
}
```

Table 2.. Example of rule in MCL.

In SEMbySEM Framework the production rules will be automatically mapped by the monitoring system to the universal sensors yet available in the infrastructure. For example a motion sensor, door seal sensor, pressure sensor, temperature sensor, gauge, etc... In order to support this scenario, the semantic model was extended with new µConcepts.

Concepts like Notification, Geofencing_Notification, Wagon, etc of the AssetTracking ontology are expressed using SEMbySEM MicroConcept model. The Listing 3 gives a partial representation of this ontology in RDF-S. smc means : semantic micro concept.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:j.0="http://www.sembysem.org#AssetTracking/"
    xmlns:smc="http://www.sembysem.org/MicroConcept#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 …..
<smc:SetProperty rdf:about="http://www.sembysem.org#AssetTracking/nearLandmark">
    <rdfs:range rdf:resource="http://www.sembysem.org#AssetTracking/Landmark"/>
    <rdfs:domain rdf:resource="http://www.sembysem.org#AssetTracking/Wagon"/>
  </smc:SetProperty>
  <smc:FunctionalProperty rdf:about="http://www.sembysem.org#AssetTracking/hasDoorSensor">
    <rdfs:range>
      <smc:Concept rdf:about="http://www.sembysem.org#AssetTracking/DoorSensor"/>
    </rdfs:range>
    <rdfs:domain rdf:resource="http://www.sembysem.org#AssetTracking/Wagon"/>
  </smc:FunctionalProperty>
   ….
```

Table 3. Example of AssertTracking ontology descriped in Micro-concept Model

4 µConcept rules Rule Editor Design

µConcept Rules Editor is modular software which is designed to allow on the one hand the edition and tests at design time of the inference rules; on the other hand it can be integrated seamlessly with management software to build run-time management and visualisation views. The inference rules written in the SEMbySEM rules language allows to express business rules, see [27] – which is semi automatically translated to standard RETE[19] based inference languages such as DROOLS or JESS. SEMbySEM rules language allows the direct use of some constructions usually not handled natively by standard rules languages, (e.g. handling properties with multiple values).
In the Fig 2 we provide blueprint architecture of Rules editor developed in the SEMbySEM project. The architecture blueprint is composed of a three main modules (User Interface, SEMbySEM Rules Parser and Class Wrapper) that makes call to both the kernel module and the rule engine (e.g. Drools Engine). The user interface module allows designer to define business rules according to the µConcept model. The defined rules will be wrapped into Java Classes using the Class Wrapper module.

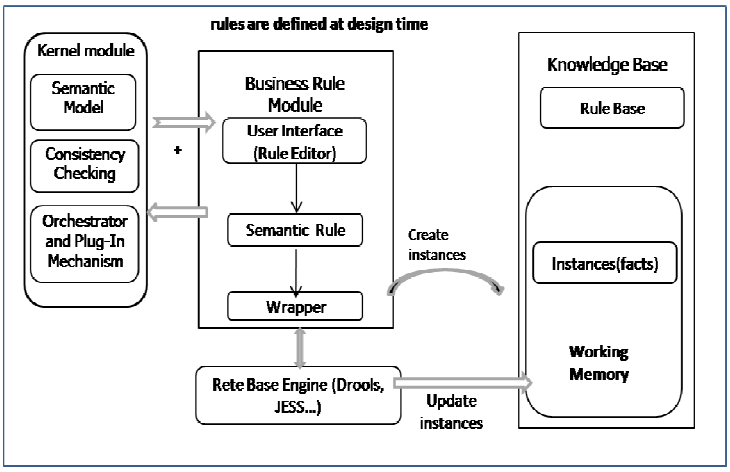

Fig. 1. Architecture of the µConcept Rules editor.

1. **Semantic Model:** maintains the semantic model, the semantic model instantiation and the constraints that apply on it, updating the model according to the messages it receives
2. **Consistency Checking**: checks for semantic representation internal consistency regarding a set of predefined constraints (defined within the semantic model).
3. **Orchestrator and plug-in mechanism:** allows the plug-in of other modules for orchestration, and provides the interfaces for access to the semantic model, for an internal synchronous messaging service and for services publishing.
4. **Working memory** will be dynamic and will allow users to add/remove rules at runtime and create/retract instances (update of semantic model).
5. **SEMbySEM Rule Editor Module:** consist of MRE expounded below.

5 MCL rules Editor

The purpose of our editor is to facilitate the editing of rules in Micro Concept Language. At first sight, a rule depicted in listing (Table 2) seems difficult to write, but the proposed editor that we call MRE (Micro concept Rule Editor) Fig 3 allows editing it easily. After having upload the domain ontology for the target system and having stored it in text and diagram format. A permanent assistance is given to user when editing rule by proposing concepts and property. Note that only constraints and operators related to a property are displayed in similar way as in Eclipse [24] or NetBeans [25]. The MCL Editor is composed of two main blocks: The edititing block and the visualization block, Fig 3. The visualization block depicted in the right hand display the global ontology. User can not only see all diagrams' ontology but he (she) can focus on one concept or property for more details. The edition block: composed of two panels, the top panel displays all the concepts, properties, super concepts and actions related of a selected concept. We use list widget for a seamless navigation among the different concepts. Thus, rules edition becomes easier. After the corresponding Java class is generated and instantiated to be used by the inference engine. The user can execute rules on the targeted inference engine

(Drools in our case), all instances existing in the "working memory" are displayed Fig 5. The second block allows assisting the user in the edition of an MCL rule, Fig 4.

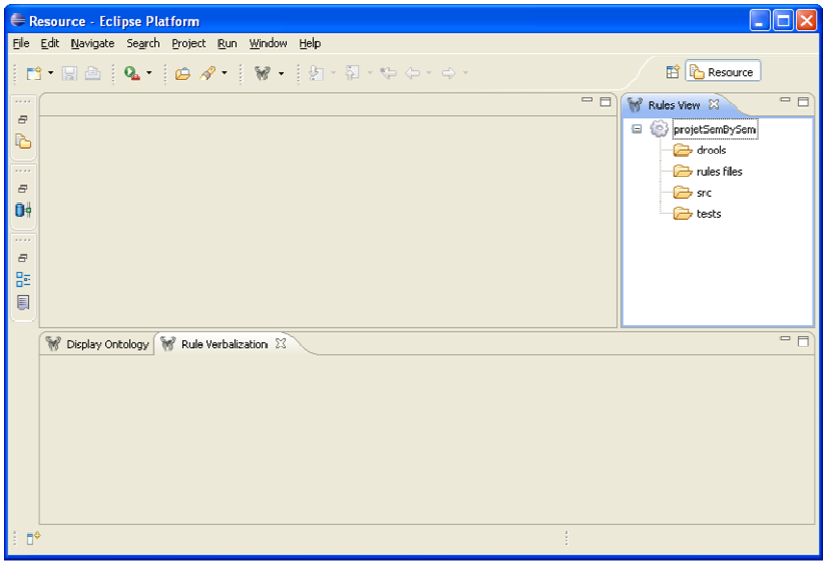

Fig. 2. Global Micro concept Rule Editor Design

MRE provides a contextual help to users when they are editing the rules. It also allows users to select concept properties from a currently loaded knowledge base, to declare variables and to insert them into the rule being edited. This task, see Fig 4, is performed only by clicking on “Ctrl+Space” button, all the operators are automatically added into the rule following a grammar described in partial form into Tab 2. This approach allows MRE to perform syntactic and semantic checking as soon as a rule has been entered. It ensures that a rule is syntactically correct and also ensures that any references to domain ontology are valid and avoid user to makes a mistake while editing a rule. The MRE itself has no inference capabilities – it simply allows users to edit rules, save and load them to and from MCL knowledge bases. However, MRE executes the rules making use of the embedded rules engines (actually we have used DROOLS and we are trying to integrate JESS).
When the user clicks on the “Ctrl+Space” button, a list of all the Concepts is displayed at first; after having selected one of these concepts, all its properties are also displayed and, at the same time, for each Concept selected a fragment of the ontology corresponding to this Concept is shown, see the right hand of Fig 3

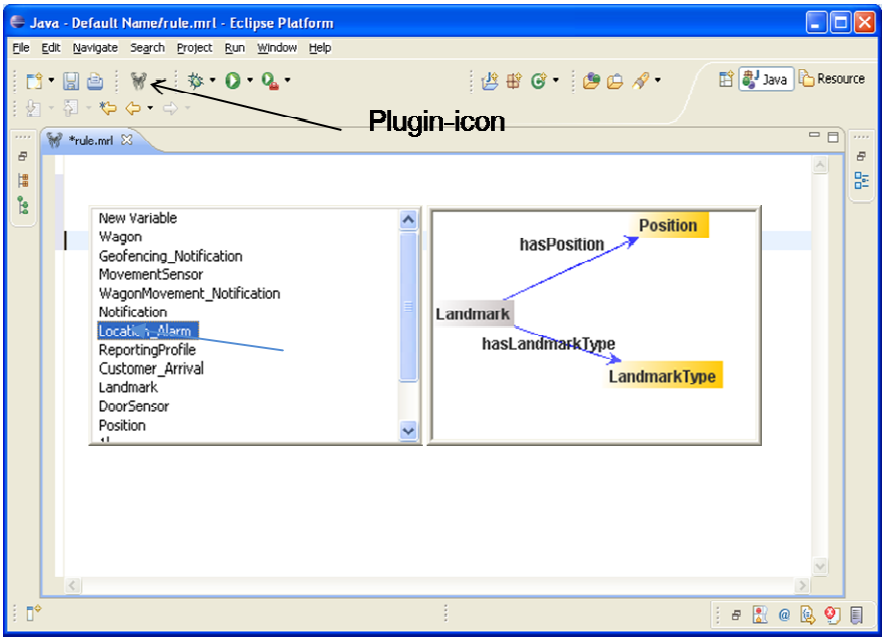

Fig. 3. Assisted edition of rules in the MCL Editor

During design time, rules are exported to a target format related the chosen rule engine. In this release we support the Drools rule format as it is depicted in the Fig 4. The editor allows us to display a selected rule(s) in order to make comparison and test it/them.

The trace of the execution of all rules triggered according to the target rule engine and making use of a fictitious "Working Memory" is reproduced in our rule editor, Fig 5.

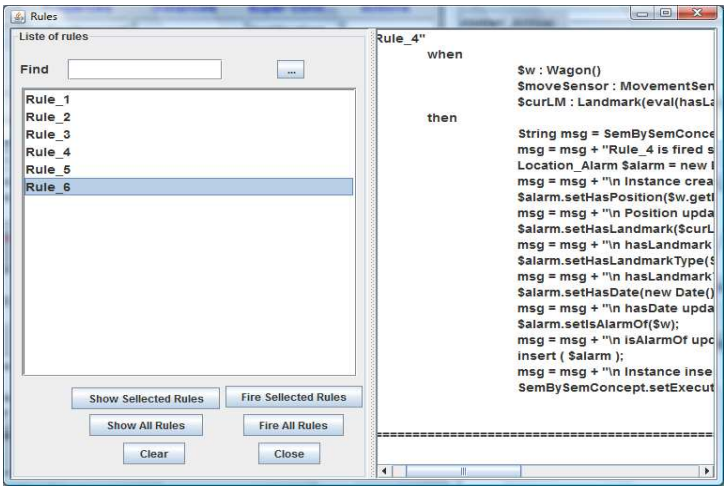

Fig.4 Showing all rules for more comparison and testing.

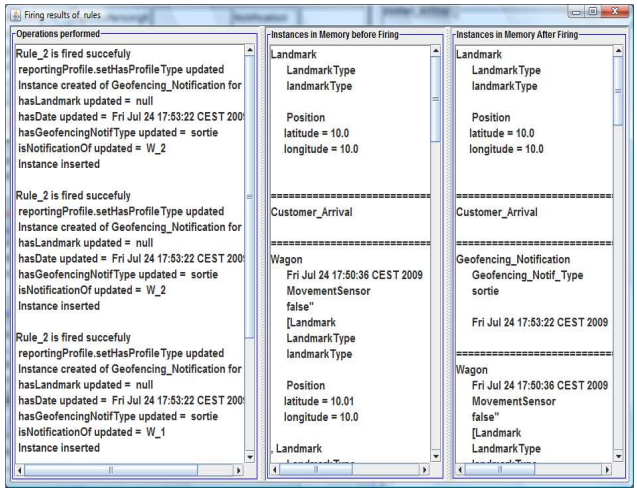

Fig. 5. Working Memory after firing rule

6 Conclusion and ongoing work

We have presented here the core idea of µConcept Rule Language dedicated for management using micro ontologies. We have presented also an implementation of a Micro concept Rule Editor, aiming at facilitating the edition of rules in the context of business-oriented models, namely the µConcepts model developed in the scope of SEMbySEM project. We are now trying to develop an application web server for MRE, and we will continue to work to take into account multilingual issues, to allow the editor to visualize rules in several languages.

Another, more ambitious, direction of research consists in trying to convert automatically or semi-automatically business rules written in a (sort of controlled) natural language into an executable format like that proper to DROOLS or JESS. The facility's solution used in many 'natural language (NL) approaches' to write down business rules consists in making use of pre-established 'templates' that are filled with NL words or expressions. The limitations of this sort of tools are evident, and concern mainly their rigidity and the lack of any sort of interoperability when passing from a specific context to a new one. The introduction of some degree of 'linguistic processing' seems then to be the main road that – even remaining strictly in a controlled language domain and avoiding any theoretical development in the Computational Linguistics style – could lead to supply the user with some really 'friendly' tool for setting up this rule. This is the option chosen, e.g., by Graham Witt [8] who is actually publishing a series of articles about "A Practical Method of Developing Natural Language Rule Statements" in the "Business Rules Community" journal. It is also the option that we will propose some concrete solutions in the "NL Business Rules" domain.

References

1. Post, E.L. (1943). :Formal Reductions of the General Combinatorial Decision Problem. American Journal of Mathematics 65: 197-268.
2. http://www.racer-systems.com/
3. http://clarkparsia.com/pellet
4. http://owl.man.ac.uk/factplusplus/
5. http://kaon2.semanticweb.org/
6. http://jena.sourceforge.net/
7. http://owl.man.ac.uk/hoolet/
8. Witt, G. (2009). A Practical Method for developing Natural language Rule Statements (Parts 1-6). Business Rules Journal 10.

9. Martin O'connor, Holger Knublauch, Samson Tu, Mark Musen : Writing Rules for the Semantic Web Using SWRL and Jess by: In 8th International Protege Conference, Protege with Rules Workshop (2005)
10. Georgios Meditskos, Nick Bassiliades :A Semantic Web Service Discovery and Composition Prototype Framework Using Production Rules. Department of Informatics Aristotle University of Thessaloniki, Greece {gmeditsk, nbassili}@csd.auth.gr.2007
11. http://www.w3.org/Submission/OWL-S/
12. http://clipsrules.sourceforge.net/
13. SWRL Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. Technical report, W3C, 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
14. Zarri, G.P. (2009) "Using Rules in the Narrative Knowledge Representation Language (NKRL) Environment", in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, vol. 1, Giurca, A., Gašević, D., and Taveter, K., eds. Hershey (PA): Information Science Reference
15. Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a Controlled Natural Language for Authoring Ontologies. In ESWC 2008, 2008.
16. Kaarel Kaljurand : ACE View — an ontology and rule editor based on Attempto Controlled English OWLED 2008
17. http://www.jessrules.com/
18. http://jboss.org/drools/
19. Forgy, C.L. (1982). RETE : A Fast Algorithm for the Many Pattern/Many Object Match Problem. Artificial intelligence 19: 17-37.
20. http://www.SEMbySEM.org/
21. http://rewerse.net/
22. http://rewerse.net/I1/oxygen.informatik.tu-cottbus.de/rewerse-i1/@q=r2ml.htm
23. http://www.w3.org/TR/owl-features/
24. http://www.eclipse.org/
25. http://www.netbeans.org/
26. Silvie Spreeuwenberg, Rik Gerrits: Business Rules in the Semantic Web, Are There Any or Are They Different? Reasoning Web 2006:152-163
27. Jean-Sébastien Brunner, Jean-François Goudou, Patrick Gatellier, Jérôme Beck, Charles-Eric Laporte June 1st, 2009, SEMbySEM: a Framework for Sensors Management 1st International Workshop on the Semantic Sensor Web Heraklion, Crete, Greece
28. Jérôme Beck, Jean-Sébastien Brunner, Patrick Gatellier, Jean-François Goudou, Charles-Eric Laporte : Micro-Concept: Model Reference SEMbySEM.
29. http://www.w3.org/TR/rdf-schema/