Ontology-based Integration of Management Behaviour and Information Definitions using SWRL and OWL

Antonio Guerrero[1], Víctor A. Villagrá[1], Jorge E. López de Vergara[2], Julio Berrocal[1]

[1]Dpto. de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid.
[2]Dpto. de Ingeniería Informática, Universidad Autónoma de Madrid.
antonio.guerrerocasteleiro@telefonica.es, villagra@dit.upm.es, jorge.lopez_vergara@uam.es, berrocal@dit.upm.es

Abstract. Current network management architectures are using different models to define management information objects. These definitions actually also include, in a non-formal way, the definition of some behaviour information that a manager should accomplish related to the managed objects. So, a manager is not able to make an automatic processing of this behaviour information. Prior research work proposed the use of formal ontology languages, such as OWL, as a way to make a semantic integration of different management information definitions. This paper goes further proposing a formal definition of the different management behaviour specifications integrated with the management information definitions. Thus, usual behaviour definitions included implicitly in the management information definitions and explicitly in policy definitions can be expressed formally, and included with the information definitions. This paper focuses on the definition of behaviour rules in management information with SWRL, a rule language defined to complement OWL functionality.

1 Introduction

The heterogeneity of the resources found in telecommunications networks and services has led to the definition of several integrated management architectures, which are intended to manage a heterogeneous environment by using a common mechanism for accessing the management information. Initially, two frameworks emerged: the OSI management architecture and the Internet management architecture. While the Internet management architecture has gained widespread acceptance, OSI management is still used in some cases, especially in telecommunications networks managed with the TMN architecture.

In addition, by the end of the 1990's, a new framework, which makes use of the web protocols, came onto scene with the same goal of achieving integrated management: the WBEM architecture (Web Based Enterprise Management). Nevertheless, this new architecture did not exclude the previous ones, including some interoperability with different integrated network management architectures.

In this environment with multiple integrated management frameworks, the initial problem that those architectures tried to solve arises once again, since one manager has to interact with different resources using several different mechanisms. Even so, the manager cannot achieve a truly integrated management of the environment: since it cannot know about the semantics of the managed resources defined in the different models, it cannot apply common policies to manage those resources, which are in fact the same. Policies should be generic and independent of the model in which the resources are defined.

In order to solve this problem, a proposal of the so-called Ontology-based Semantic Management is included in [1]. This semantic management allows a manager working with a unique information model, which integrates all the different definitions of the managed resources, taking into account the semantic aspects of those definitions (i.e. their meaning). Another advantage of this approach was also pointed out in that proposal: it allows the integration, in that same unified management information model, of the definition of behavioural characteristics of the manager and of the resources. These behavioural aspects are usually found as comments inside the definitions, or explicitly declared outside of the definitions of the managed resources. With this approach, all the definitions (information and behaviour) are integrated, so they can be jointly processed automatically.

This paper proposes a starting point for this integration: beginning with the unified information model, defined in an ontology language such as OWL, it proposes inclusion in this model of the behaviour definitions, expressing them by means of the Semantic Web Rule Language, SWRL. For this purpose, next section shows the semantic management architecture and the SWRL language. This language will allow definition of the behaviour as part of the definitions of the management information, as will be shown. Also, the different types of behaviour are explained: implicit restrictions, explicit behaviour of the manager, and behaviour of the managed elements. Example definitions in SWRL are included for each one of these types. Lastly, the most relevant concepts are summarized.

2 Semantic Management

The global architecture proposed in [2] is based on a manager that works and reasons with a unique information management model, represented by means of ontologies. This system manages elements from different domains (SNMP, CIM, etc.) from a common and neutral perspective. **Fig. 1** shows the proposed architecture of the semantic manager.

The main goal is the usage by the manager of only one information model, but, in many cases, the different network resources are defined in different management domains (SNMP MIBs, CIM schemas, etc.), so they have to be accessed by using the predefined protocols for those domains. Therefore, it is necessary to translate and integrate those definitions into one unified specification, taking into account the semantics of the initial definitions. In other words, the same resource, defined twice in two different models, will have one single representation in the unified model, and will have two translations to the original models. So, the goal is not just to make a

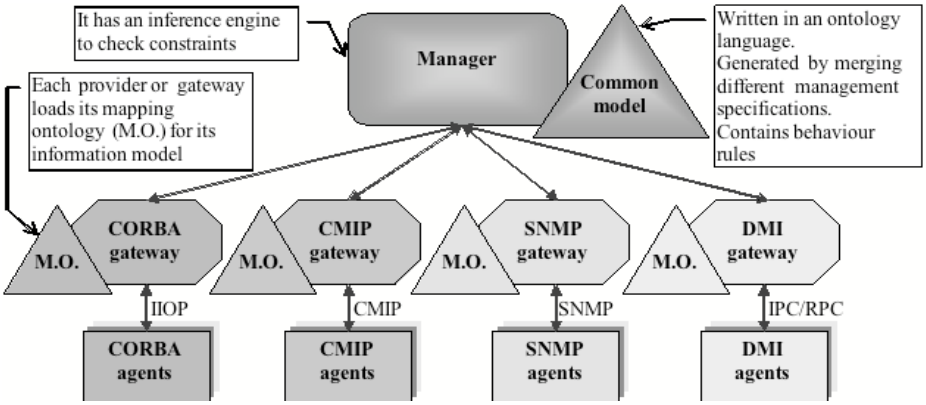

Fig. 1. Proposed architecture for the semantic manager [2]

syntactic translation of the definitions, but also about their integration from a semantic viewpoint.

The resulting mappings from the original definitions to the ontology specification are then used by the so-called "providers" or gateways in **Fig. 1**. They are responsible for translating the information of the network elements in the unified model back into the information in their original management models.

Using an ontology language for defining the management information has additional advantages, such as the possibility to use existing tools in order to work and to reason with the ontologies (for example inference engines used in artificial intelligence).

Another advantage, already mentioned in the previous section, is that a management ontology allows the integration of rules defining the expected behaviour of the management information. In this way, the behaviour definitions that are usually implicitly included in the management information definitions (declared in natural language, or tacitly supposed), can now be formally expressed as an integrated part of the management information definitions, and in their same language (ontology language). This approach implies that the behaviour definitions are now formally expressed in the same definition language, and that they can be interpreted and validated or enforced by the same semantic manager, in order to work and reason with them. All definitions, information management (MIBs) and behaviour rules (policies), are now integrated in the same network management ontology. One of the main advantages of this approach is that it will permit generic managers to behave depending on definitions, instead of having embedded encoded behaviour.

OWL [3] is proposed [4] as the ontology language for the definitions, a general purpose ontology language defined for the Semantic Web that contains all the necessary constructors to formally describe most of the information management definitions: classes and properties, with hierarchies, and range and domain restrictions. SWRL [5] extends the set of OWL axioms in order to include conditional rules (Horn clauses), of the form *if… then …*

Axioms and rules can be used in this management framework in order to:

1. Further constrain or define more precisely the behaviour of the OWL management information. This will guarantee the correct use and implementation of the management information.
2. Formally define the behaviour of the manager. This allows the declaration of the manager actions when certain conditions are met on the managed elements.
3. Formally define the behaviour of the managed objects. This allows the definition of what the managed resources should do upon certain events or conditions.

The purpose of this paper is to integrate the definition of the management information (expressed in OWL) with the definition of the management behaviour (expressed in SWRL). The next section explains briefly the SWRL language, used to define rules for OWL ontologies.

3 SWRL: definition of rules for OWL ontologies

An OWL ontology contains a sequence of axioms and facts. It includes several types of axiom, such as subclass axioms, equivalent Class axioms and property constraints. SWRL proposes to extend these with rule axioms.

A rule axiom consists of an *antecedent* (body) and a *consequent* (head), each of which consists of a (possibly empty) set of atoms.

In the "human-readable" syntax of SWRL, a rule has the form:

`antecedent ⇒ consequent`

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

`Person(?x) ∧ isFather(?x,?y) ∧ isBrother(?y,?z) ⇒`
`isUncle(?x,?z)`

An SWRL rule has therefore the form of an implication relationship between the head and the body. The SWRL specification [5] provides an abstract syntax that extends the OWL abstract syntax described in [6] to include this relationship in the ontology language. The XML labels used for defining rules include:

- <ruleml:imp>: it is the element that relates the body of the rule with the head (the label of the relationship).
- <ruleml:_body>: it is the element that contains the atoms that form body of the rule.
- <ruleml:_head>: it is the element that contains the atoms that form the head of the rule.
- <ruleml:var>: it allows the definition of the variables used in the evaluation of rules.
- <swrlx:individualPropertyAtom>: it allows the definition of atoms that refer to specific properties. It is also possible to define atoms that refer to classes, data

ranges, valued properties, or typed functions such as mathematical, dates and strings.

As is shown, many of these labels are not defined inside the SWRL namespace, but in RuleML's [7], a rule language previously defined that has been taken as the base for SWRL definition. SWRL mainly brings the definition of atoms and the integration of these rules into an ontology written in OWL.

4 Definition of Management Behaviour in OWL+SWRL

4.1 Types of management behaviour

In order to define management behaviour let us first classify which types of behaviour can exist. Concretely, three types of management behaviour have been identified:

1. Implicit constraints and rules about the behaviour of the modelled objects in the MIBs and CIM schemas
2. Explicit behaviour of the manager, in the traditional manager-agents architecture, which defines how the manager should behave upon analysing the information obtained from the agents
3. Explicit defined policies to specify the dynamic behaviour or dynamic configuration of the managed resources (Policy-Based Management [8])

In this semantic management framework, policies can be defined in the same management language – OWL+SWRL – in which the objects of the management information base (MIB) are defined, with the advantage of working with a unified model.

Going back to the management architecture proposed in Fig. 1, the constraints, rules, and policy definitions would be stored in the rule information base – part of the common model, whose purpose is verifying the integrity of the information, and automating the control of the managed elements.

The following sections focus on each of the identified behaviour types.

4.2 Implicit restrictions on the management information

This kind of rules refers to restrictions in the data type, cardinality, or access. They are typical restrictions upon the properties and classes of the managed objects. In this case SWRL rules complement the existing mechanisms in OWL to form the management information definitions, in the sense that they allow expressing more complex restrictions: values that depend on the values of other variables, relationships among objects, state-machine behaviour of the values, etc.

In a general way, SWRL allows the representation of behaviour restrictions that can be expressed in a natural language as conditional clauses (*if… then …*). This includes, for example, values that depend on other values, state-machine behaviour, temporal behaviour, or complex types such as composed functions. The following subsections present some examples of these kinds of behaviour found in SNMP

MIBs, and how SWRL is used to formally define these restrictions. In many cases, the definition of the restrictions will require the creation of new classes and properties that would extend the management ontology.

Example 1

This first example shows how to specify in SWRL that the value of one variable depends on the value of another variable. It is based on a definition from SNMP's MIB II, which restricts the value of the mask for route entries, in the routing table:

```
ipRouteMask OBJECT-TYPE
        SYNTAX IpAddress
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
           "… If the value of the ipRouteDest is 0.0.0.0 (a
default route), then the mask value is also 0.0.0.0 …"
        ::= { ipRouteEntry 11 }
```

This implicit restriction is expressed in natural language in the DESCRIPTION clause, but it is not formally defined in SMIv2, so it cannot be automatically implemented by a manager when the MIB is compiled.

If theMIB II has been mapped and integrated in the management information base in OWL, the SWRL rule to define this example restriction would be the following:

```
ipRouteEntry(IR?) ∧ swrlb:equal (ipRouteDest(IR?), "0.0.0.0")
⇒ swrlb:equal(ipRouteMask(IR?), "0.0.0.0")
```

where ipRouteDest and ipRouteMask are properties of the class ipRouteEntry.

It is interesting to notice that this kind of restriction, hereby expressed in a simple manner, is not currently supported by existing management definition languages. In fact, the IETF's SMIng Working Group proposed the following objective for the next generation of the SMI definition language, stated in [9]: "*SMIng should provide mechanisms to formally specify constraints between values of multiple attributes*", but its implementation was not accomplished: "*This objective as is has been rejected as too general, and therefore virtually impossible to implement*".

On the other hand, further reinforcing the idea under study, it so happens that OWL without SWRL can neither express this constraint: a restriction on a property (owl:restriction clause) cannot be made to depend on the value of another property of the same object.

Example 2

This second example, also obtained from SNMP's MIB II, can be used to show how to define state-machine behaviour with SWRL. It is based on the column of the TCP connection table which shows the state of each connection.

```
tcpConnState OBJECT-TYPE
 SYNTAX INTEGER {
    closed(1),
    listen(2),
    synSent(3),
    synReceived(4),
    established(5),
    finWait1(6),
    finWait2(7),
    closeWait(8),
    lastAck(9),
    closing(10),
    timeWait(11),
    deleteTCB(12)
  }
 ACCESS  read-write
 STATUS  mandatory
 DESCRIPTION "…"
::= { tcpConnEntry 1 }
```

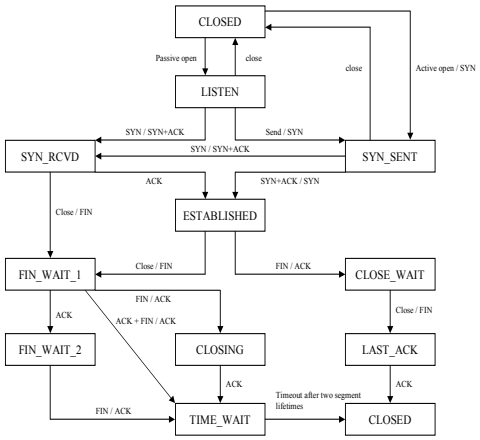

Fig. 2. TCP state machine and definition

Since this parameter has a read/write access, this example is intended to define the state-machine diagram shown also in **Fig. 2**, so that a manager could check that all connections comply with the state machine.

For this purpose, in the management ontology we define a new class of objects called tcpConnEntry, with a property that indicates the actual state of the connection, tcpConnEntry, and auxiliary properties tcpConnPreviousState and tcpConnNextState, which indicate the list of possible before and after states. The following is an example rule that could be used to define the state-machine:

```
tcpConnEntry(cx?) ∧ swrlb:equal(tcpConnState(cx?),
"fin_wait_1") ⇒ swrlb:member(tcpConnNextState(cx?),
nextStatesForFin_Wait_1_List)
```

where nextStatesForFin_Wait_1_List would be a list (rdf:list) with the values “closing”, “time_wait” and “fin_wait_2”.

Example 3

The third example is a case of defining a temporal constraint, also described in natural language in SNMP MIB II. It is a constraint upon the value of the column of the interfaces table which indicates the latest time at which the interface changed. The constraint definition for the property ifLastChange that can be represented in SWRL is marked in bold letters:

```
ifLastChange OBJECT-TYPE
        SYNTAX TimeTicks
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The value of sysUpTime at the time the interface
entered its current operational state.  If the current state
was entered prior to the last re-initialization of the local
```

```
network management subsystem, then this object contains a
zero value."
        ::= { ifEntry 9 }
```

To implement this constraint it is also needed to use auxiliary classes and properties. In this case, the temporal comparison can be made making use of the property sysUpTime of the class system.

```
system(?x) ∧ ifEntry(?y) ∧ isInterfaceOf(?y, ?x) ⇒
swrlb:lessThan(ifLastChange(y?), sysUpTime(?x))
```

4.3 Explicit behaviour of the manager

The behaviour of the manager when certain conditions are met on the network or the managed systems, can also be specified by means of conditional rules *if condition then action*, which become the following in SWRL

```
condition ⇒ action
```

or, making use of the broader definition of SWRL:

```
condition set ⇒ action set
```

Actions

As a management information definition language, OWL lacks the explicit ability to define operations or methods on the managed objects that are typical in other management information languages such as CIM or GDMO. In order to solve this problem, actions in OWL can be represented by means of the OWL-Services ontology, OWL-S [10], which allows an easy integration with the rest of definitions. One of the classes that this ontology defines, among many others, is the Process class that can be used for this purpose. In this way, "executing an action" can be interpreted as "calling a service" that executes that action, and so it is possible to define that action as an OWL-S Process, and then instantiate an object of the class Perform, with such process as its argument:

```
Perform(MyProcess)
```

The class Perform is an auxiliary class used in OWL-S to represent the execution of atomic processes inside a composite process, e.g. Perform(Reset), Perform(SendAlarm(…)), Perform(SetIPRoute(…)), etc.

These processes can be either newly defined actions or the result of integrating existing operations or methods of the merged information models (CIM methods, SNMP "set" operations, etc.)

In the proposed architecture in Fig. 1, which follows the traditional manager-agent paradigm, the manager would invoke the service calls that the providers would offer, and these providers would in turn execute the requested operations upon the managed elements, in their native languages and protocols.

The following example shows this type of definition of the behaviour of the manager upon the existence of certain conditions on the network:

Example 4

This example makes use of the class CIM_SystemDevice from the CIM schema, with two instances that are port devices. With this SWRL rule, the manager will activate the second port if the first port is not working, i.e.:

```
If LogicalPort #1 is "Operatively Down", then enable
LogicalPort #2
```

In SWRL:

```
CIM_SystemDevice(LP1?) ∧ swrlb:equal(deviceName(LP1?),
"Lport1") ∧ CIM_SystemDevice(LP2?) ∧
swrlb:equal(deviceName(LP2?), "Lport2")  ∧
swrlb:equal(StatusInfo(LP1?), "OPERATIVELY_DOWN") ⇒
Perform(SetAdminAvailability(LP2?, "ENABLE"))
```

In this case, the rule is applied upon certain instances of a class, not upon all elements of the class.

In the same way as it occurred with implicit restrictions, this type of explicit definitions could neither be expressed in OWL without SWRL, as it is a restriction upon the values of a property. Also, there is a clear difference in this example with the example #1 in the previous section, because in this case the related values are the values of the same property, from two different objects of the same class, instead of values from different properties of the same object.

4.4 Explicit behaviour of the managed elements: application to policy-based management

If the rules being defined in the information model are not rules for the behaviour of the manager, but rules that define the behaviour of the managed elements, then we need an architecture that supports the distribution of the rules or policies to those managed elements. This type of architectures is defined in the PBM framework (Policy-Based Management) or policy-based networking (PBN). This work references the following elements of the PBM architecture, proposed by the IETF/DMTF [8]:

- PEP Devices (Policy Enforcement Point): those elements that can apply or execute the policies.
- PDP Devices (Policy Decision Point): they act as proxy between the PEPs and the policy repository, being responsible for interpreting the policies from the repository and indicating the corresponding actions to the PEPs.
- Policy Repository: it stores the defined policies that will be distributed to the PDPs.

A policy-based architecture presents the following key characteristics:

- Centralization: definition of behaviour is made in a single point and can be massively distributed throughout the network, instead of being defined and applied individually for each element
- Abstraction Levels: policies can be defined at different level: high level policies (i.e. business rules), intermediate level (i.e. service level rules), and lower level (i.e. policies applied by the network elements). It might be necessary to translate policies among levels, in order to convert high level rules into the lower level policies that will be applied by the network elements. For this task it will be useful

to define models (business models, service models, network models). Existing policy definition languages can be oriented to policy definition at determinate levels. For example, PONDER and RBAC would be classified as high-level policy definition languages, since the information model is closer to natural language and human thinking. On the other hand, PCIM and COPS-PR PIB allow the definition of lower level policies, closer to the management languages used by the network elements.

In this section, we will attempt a first approach to the possibility of using ontologies in OWL+SWRL as a definition language for those kinds of high and low level policies, through the following examples.

Example 5: high level policy in PONDER

This example matches in SWRL a policy previously defined in PONDER, extracted from [11]:

```
type rel ReportingT (ProjectManagerT pm, SecretaryT secr) {
        inst oblig reportWeekly {
           on timer.day ("monday") ;
           subject secr ;
           target pm ;
           do mailReport() ;
        }
        // . . . other policies
}
```

The obligation policy reportWeekly specifies that the subject of the SecretaryT role must mail a report to the subject of the ProjectManagerT role every Monday. In SWRL it could be expressed like this

```
ProjectManagerT(pm?) ∧ SecretaryT(secr?) ∧ Timerday(t?) ∧
swrlb:equal (t?, "monday") ⇒  Perform(mailReport(secr?, pm?))
```

In a similar way, other kinds of policies could be mapped to SWRL: authorization policies, obligation policies, filtering policies, etc.

Example 6: low level policy

Next, an SWRL definition will be shown of a low level policy that could be executed by a network element.

The following rule would set to “0100” the ToS (Type Of Service) field of each IP packet whose destination address belonged to the specified address range.

```
IPpacket(ippacket?) ∧ InsideIPRange(DestAddress(ippacket?),
"192.168.1.0 – 192.168.1.255") ⇒
Perform(SetTOSlabel(ippacket?, "0100"))
```

While this example demonstrates the possibility of defining these kinds of lower level policies, it might be too simple and not very useful in a real case. A better approach to integrate these kind of policies into the unified management ontology would be to attempt the mapping of the existing PIBs (Policy Information Base), or mapping the existing policy definition languages that the network elements or PDPs understand. Two kinds of necessary mappings can be identified: on one side, the mapping of the conditions that trigger the policies, or condition sets; on the other side,

the mapping of the operations to be performed, or action sets. If the same kinds of policy are defined in different definition languages, the integration approach could use the M&M (Merge and Map) method [2], proposed for the integration of management information definitions coming from different management models to a common ontology. The application of this method would in turn allow a reverse translation, from the policies represented in the OWL+SWRL ontology, to the native management languages that the PDPs would understand.

At this point it is necessary to notice that the policies represented in the examples hereby included (such as *"<condition set> then do <action list>"*) are policies without events. There are no explicit events to trigger the evaluation of the policies, but the agents must do this task whenever there is an implicit event, as would be starting a new session, or periodically. This would be the case of the PCIM definition language, but not of COPS and PONDER, since these languages allow the definition of policies with this kind of event-condition-action.

Policy-based management also deals with translation of policies among different abstraction levels, so policies defined at higher levels (i.e. business level and service level) can be translated down to lower level policies (i.e. network and system levels). In the semantic management approach, policy definitions at all levels would be defined in the same definition language, OWL+SWRL, which could facilitate the process of translating among levels. Nevertheless, this is currently out of the scope of the present work.

5 Conclusions

As it has been shown in previous research work [1,2,4], ontology languages such as OWL include the constructions necessary to define the typical aspects of the network management information that can be found in other management definition languages such as SMI-SNMP, MOF-CIM, etc., so it is possible to make a mapping and merging process which integrates definitions in different languages from a semantic point of view.

In this semantic management framework, the present work has shown how SWRL:

1. adds expressiveness power for defining constraints and rules for the proposed network management information ontology in OWL. This means that more complex constraints and policies can be formally expressed in the OWL management information model, therefore enriching and empowering the overall semantic framework for network management
2. allows explicitly defining the behaviour of the manager, and of managed objects, in the same language – OWL+SWRL – that is used for the definitions in the management information base. This includes:
   - Actions that will be performed by the manager upon certain conditions on the network objects or the manager itself
   - Actions that will be performed by the network elements upon the existence of certain conditions (policies that define the behaviour of these managed elements)

In this semantic management framework, some management architecture elements have yet to be defined, including events, translation and distribution of policies, and other elements from policy-based networking.

Another direction for future work is the approach to be followed for accomplishing the formal definition of the existing restrictions and rules implicitly or explicitly defined. For this work it would be useful to develop tools that propose these restrictions using heuristic searches of constraints described in the "description fields" (e.g. strings including "if*then", "have to" or "must" will have a high probability of being part of these natural language constraint definitions), and their final representation in SWRL making use of the existing OWL definitions.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science under the project GESEMAN (TIC2002-00934).

References

1. J. E. López de Vergara, V. A. Villagrá, J. I. Asensio, J. Berrocal: Ontologies: Giving Semantics to Network Management Models, IEEE Network, Vol. 17, No. 3 (2003) 15-21
2. J. E. López de Vergara, V. A. Villagrá, J. Berrocal: Benefits of Using Ontologies in the Management of High Speed Networks. In *High Speed Networks and Multimedia Communications – Proceedings 7th IEEE International Conference, HSNMC 2004*. Toulouse, France, June 2004. LNCS 3079: 1007-1018, Springer-Verlag (2004)
3. M. K. Smith, C. Welty, D. L. McGuinness: OWL Web Ontology Language Guide, W3C Recommendation (10 February 2004)
4. J. E. López de Vergara, V. A. Villagrá, J. Berrocal: Applying the Web Ontology Language to management information definitions, IEEE Communications Magazine, Vol. 42, Issue 7 (2004) 68-74
5. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission (21 May 2004)
6. P. F. Patel-Schneider, P. Hayes, I. Horrocks: OWL Web Ontology Language Semantics and Abstract Syntax", W3C Recommendation (10 February 2004)
7. Rule Markup Initiative: http://www.ruleml.org/
8. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser: Terminology for Policy-Based Management, IETF Request For Comments 3198 (2001)
9. C. Elliott, D. Harrington, J. Jason, J. Schoenwaelder, F. Strauss, W. Weiss: SMIng Objectives, IETF Request For Comments 3216 (2001)
10. D. Martin, editor: OWL-S: Semantic Markup for Web Services, W3C Member Submission (22 November 2004)
11. N. Damianou, N. Dulay, E. Lupu, M. Sloman: The PONDER Policy Specification Language. In *Proc. International Workshop of Policies for Distributed Systems and Networks (Policy 2001)*. Bristol, UK, January 2001. LNCS 1995: 18-39, Springer-Verlag (2001)