On the Formalization of the Common Information Model Metaschema

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

[1] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Escuela Politécnica Superior, Francisco Tomás y Valiente, 11, E 28049 Madrid, Spain
jorge.lopez_vergara@uam.es

[2] Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, ETSI de Telecomunicación, Ciudad Universitaria, s/n, E 28040 Madrid, Spain
{villagra, berrocal}@dit.upm.es

Abstract. Integrated network management frameworks include a common definition of the managed resources, known as an information model, which is a key factor to describe the domain to be managed. In this scope, it is important to understand the semantics each information model provides to allow interoperation among different integrated management architectures. For this, ontology languages have recently been proposed, because thanks to their formalization they can deal with the semantics of information. Nevertheless, they need to be adapted to meet the management requirements. An alternative to the use of ontology languages can be the formalization of the management information languages to cope with the semantics of the information models. This paper provides a way to formalize one of these management languages: the Common Information Model metaschema. The formalization is based on the use of the Object Constraint Language to define in a formal way the set of natural language rules that describe this metaschema, improving its semantics, comparing also this solution to those based on ontologies.

1 Introduction

Network and service management has been a field in which traditionally proprietary solutions from different vendors were usually imposed. In these solutions the management of those equipments could only be performed with those vendor products. Then, integrated network management architectures appeared that defined standard protocols and information models allowing the interoperability between multiple vendors managers and managed elements.

Due to historical reasons, two different management frameworks have survived the standardization process: Internet network management framework (also known as SNMP, Simple Network Management Protocol) and OSI network management framework (also known as its protocol: CMIP, Common Management Information Protocol). These frameworks are incompatible, so finally each one has got its own application field, even though both frameworks have to coexist in some environments, such as telecommunication companies.

Later on, other integrated network management architectures have appeared that use other technologies for resources management, different to SNMP or CMIP. The most significant example is the Web-Based Enterprise Management (WBEM) and its associated Common Information Model, CIM.

Each integrated management architecture deals with its own information, defined in a different language: same concepts can be defined to model a resource using incompatible formats, which cannot be directly translated. This issue is a combination of syntax and semantic problems. One way to deal with the semantics of the management information is the use of ontologies: they are formal [1], and thus, the meaning of this information is machine-interpretable.

By applying this knowledge representation technique, the work presented in [2] provided a way to analyze management information languages, being useful to identify their semantic expressiveness. One of the results obtained was that the Common Information Model (CIM) had most of the elements usually contained in ontology languages. However, it was not a model appropriate to deal with the semantics of the information because of its lack of formalization: rules about its structure have been defined in natural language, which is not machine interpretable, so that they cannot be processed and checked.

Then, ontology languages have been proposed to describe the management information [3, 4, 5, 6]. In this case, these languages have a formalized semantics. Still, they have to be adapted to the management scope, as there are some constructs they do not include.

Another way to deal with the semantics of management information is the formalization of the CIM metaschema: in this case a management specific information model is used, and a computer would be able to interpret the information defined in such way. With respect to the formalization of a management language, some works have been found [7, 8], but they are related to GDMO (Guidelines for the Definition of Managed Objects), the language used for OSI Systems Management, which had less constructions in common with ontology languages than CIM, as stated in [2]. The formalization of the CIM metaschema also reinforces the information defined in the CIM schema, which currently includes in its last release more than a thousand classes that base their relationships on that metaschema.

This paper presents an approach to formalize the CIM metaschema. For this, first of all, an analysis of this metaschema is given. Then, it is also compared to UML (Unified Modeling Language) metamodel. Next, a set of rules defined in OCL (Object Constraint Language) are shown that match natural language rules about CIM elements, providing a formalization of the metaschema. After that, this approach is compared with the use of a formal ontology language. Finally, conclusions and future works are also presented.

2 CIM Metaschema Analysis

CIM [9] is the information model defined by DMTF to be used in the Web Based Enterprise Management architecture, and has a considerable acceptation in the industry. This model is object-oriented and much more powerful than SNMP SMI (Structure of Management Information). However, its complexity is lower than

GDMO, as discussed in [2]. With this format, classes can have properties (the name they use for attributes) and methods. Other facets can be defined, thanks to the possibility of specifying new qualifiers [9]. This information model can also be expressed in XML (Extended Markup Language) to exchange the information.

As stated before, CIM has the information model metaschema with a largest number of elements usually included in ontology languages. It includes these characteristics, when comparing it to ontology languages [2]:

- Concepts or classes: They are a collection of instances with the same properties and methods. CIM can define:
  - Metaclasses: This item deals with the possibility of defining classes as instances of other ones. In CIM it is possible to define new statements with qualifiers, which indirectly makes feasible the redefinition of classes.
  - Attributes: Concepts usually have attributes. In CIM they are defined in the local scope of a class and can be instance attributes, class attributes, and polymorph attributes.
  - Facets: Attributes usually have a set of predefined properties or facets. In CIM default value, data type constraint, cardinality constraint, and documentation can be found among other facets such as the access, the key or index, and the identifier. In addition, CIM can define new facets by using qualifiers.
- Taxonomy: Concepts are usually organized in taxonomies, with generalization/ specialization relationships among them. CIM allows the definition of subclasses with simple inheritance.
- Relations and functions: Relations represent a type of interaction between concepts. Functions provide a unique value from a list of valued arguments. CIM can define both relations among classes and functions for every class, with data type constraints.
- Instances: They represent elements of a given concept, a relation or an assertion. CIM allows the definition of class and relation instances.
- Axioms: They model expressions that are always true, and are usually used to define constraints. CIM does not currently support constraints, although a qualifier could be defined with this purpose.

Also, CIM schemas are structured in a similar way to ontology libraries [10]. In this way CIM schemas could be considered an ontology except for their lack of formalism. On the other hand, CIM uses the Unified Modeling Language (UML) class diagrams to model the management information, and several works [11, 12] have identified UML as a valid ontology modeling language.

This set of reasons presents CIM as a good candidate to define management information from a semantic viewpoint. Nevertheless, there is a problem that has to be solved to achieve this goal: as stated before, CIM is not formal (the rules about its metaschema are written in natural language, which cannot be processed and checked by computers), so it is not valid for the definition of heavyweight ontologies. To solve this problem it will be necessary the formalization of its metaschema. For this, the Object Constraint Language (OCL) [13], used in UML to define constraints can be applied, rewriting CIM metaschema rules, avoiding existing ambiguities that are caused because they are currently written in natural language. Other rule languages such as SWRL (Semantic Web Rule Language) [14] would also be useful for this task, but OCL has been chosen because of its integration with UML, and because it is being studied by the DMTF to specify constraints for management classes and objects in the CIM schemas.

3 CIM and UML

CIM semantics has been defined in its metamodel or CIM metaschema, depicted in **Fig. 1**, which describes the elements existing in this model by representing them in a UML class diagram and defining a set of rules about these elements in natural language.

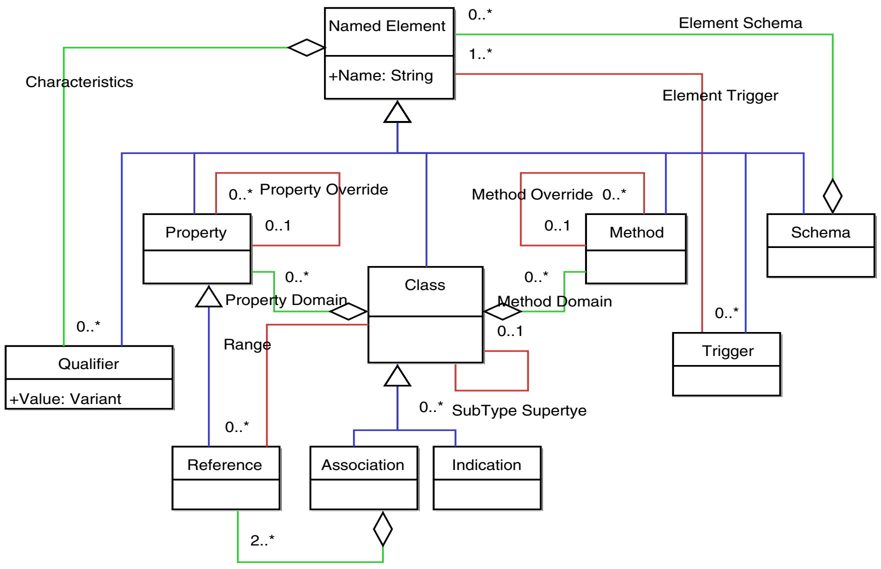


Fig. 1. CIM Metaschema [9]

Given that UML metamodel [15] includes a set of constraints defined in OCL that formalizes the behavior of its elements, a question arises: if CIM metaschema uses UML, are its elements as formal as UML? To answer it, a comparison between CIM and UML is provided, which shows that there are important differences between them.

The first difference is related to abstraction levels: **Table 1** shows a comparison between CIM and the four-layer metamodel architecture used in UML: CIM meta-metamodel (the model used to define the CIM metaschema) is directly UML; the metamodel (the model used to define the models) is the CIM metaschema; the set of CIM schemas are in the model level; finally, CIM schema class instances are the user objects.

If the comparison is focused in the metamodel layer, CIM metaschema is also different to UML metamodel. **Fig. 2** shows a subset of the UML metamodel that includes a set of elements which could be equivalent to CIM metaschema, shown in **Fig. 1**. Although they share a similar structure there is a different number of elements in both figures, as there are more specialization degrees in UML.

Table 1. Comparison of UML and CIM layers

Layer	UML	CIM
Meta-metamodel	OMG MOF meta-metamodel	UML
Metamodel	UML metamodel	CIM Metaschema
Model	UML models	CIM schemas
User objects	UML model instances	CIM class instances

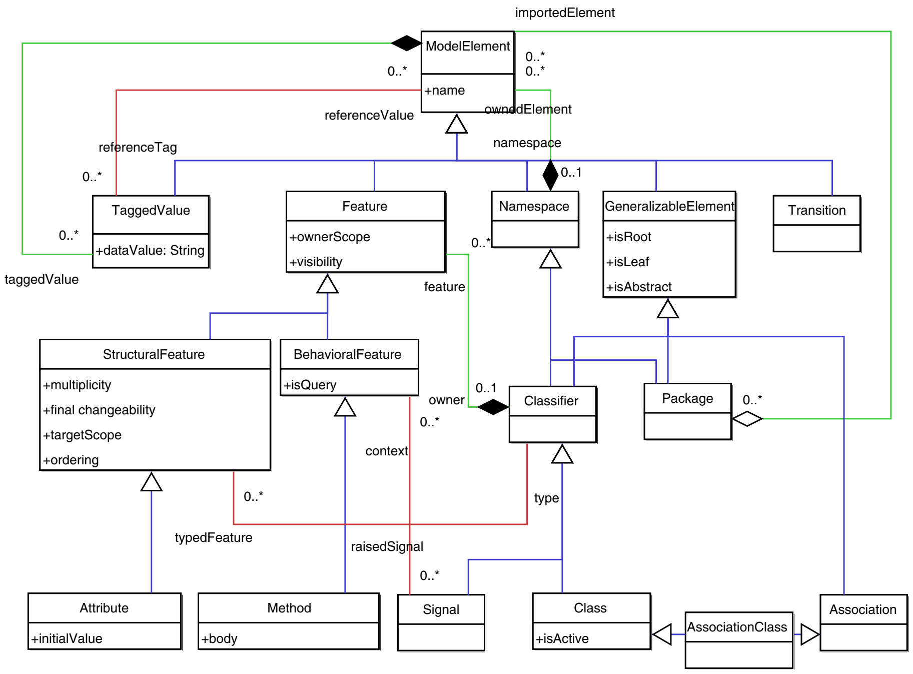


Fig. 2. UML metamodel subset

As a result, UML formalization rules are not directly applicable to CIM metaschema. New rules have to be defined to achieve this goal, as shown in next section.

4 CIM Metaschema Formalization

This section presents a formal specification of CIM metamodel. For this, a set of rules have been defined in OCL, trying to cover the set of rules defined in natural language in CIM specification [9]. During this process, some incongruities were found among existing rules, which were solved when possible. This formalization does not modify existing information defined in CIM, but on the other side it allows its validation with OCL constraints.

To carry on this formalization, all information given in CIM specification has been taken into account: This specification first describes the CIM metaschema with a UML class diagram (as shown in **Fig. 1**). Then, it provides a set of rules written in natural language (and thus, not formal). Next, it specifies in ABNF (Augmented Backus-Naur Form) the MOF (Managed Object Format) syntax. Finally, it presents the CIM metaschema written in MOF format. This specification has been revised by DMTF [16], but defined rules are mostly similar to prior version, and are still in natural language. Some conflicts have been found among these sections:

The UML class diagram that models the CIM metaschema is not complete. It does not include constraints related to each element, neither other elements named in the natural language rules or in the MOF syntax (e.g. Instance element).

Some natural language rules are incongruous, as there exist different properties for an element in different rules (e.g. rules about the Qualifier element).

Other rules are redundant with respect to the class diagram (e.g. cardinality relationships, or element specialization), so that it is not necessary their definition.

Taking into account these conflicts, a formalization has been performed on the CIM metaschema, as shown in **Fig. 3**.

The formalized diagram includes these points:

1. All elements named in rules or in MOF syntax have been added, including those that were not depicted previously (e.g. DataType and Instance elements, and some associations between elements).
2. All association ends have been named when they start and finish in the same element, to improve the diagram semantics and to make easier the definition of OCL rules (e.g. Overriding and Overridden in Property and Method elements, or Subtype and Supertype in Class element). For the rest of associations the name of the association end is directly the name of the associated element, except when the constraint rule uses other name (e.g. Range in the association end of Class with Reference, or Domain in the aggregation end of Class with Property and Method).

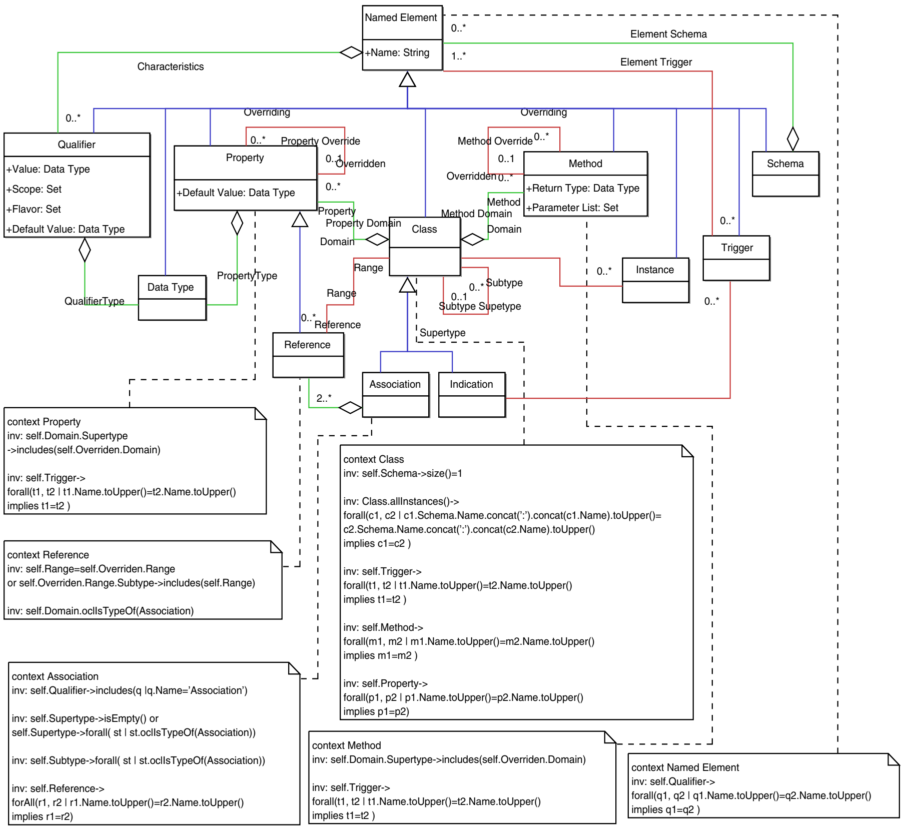
All those rules defined in English that could be formalized have been written in OCL, defining invariants inside the scope of each element. Other rules about the utility of each element were not formalized. Following lines present most important ones:

The rule that says that "A Class must belong to only one schema" has been specified as:

```
context Class
  inv: self.Schema->size()=1
```

The rule about overriding properties "The Domain of the overridden Property must be a supertype of the Domain of the overriding Property" has been defined as:

```
context Property
  inv: self.Domain.Supertype->includes(self.Overriden.Domain)
```



```
context Property
inv: self.Domain.Supertype
->includes(self.Overriden.Domain)

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )
```

```
context Reference
inv: self.Range=self.Overriden.Range
or self.Overriden.Range.Subtype->includes(self.Range)

inv: self.Domain.oclIsTypeOf(Association)
```

```
context Association
inv: self.Qualifier->includes(q |q.Name='Association')

inv: self.Supertype->isEmpty() or
self.Supertype->forall( st | st.oclIsTypeOf(Association))

inv: self.Subtype->forall( st | st.oclIsTypeOf(Association))

inv: self.Reference->
forAll(r1, r2 | r1.Name.toUpper()=r2.Name.toUpper()
implies r1=r2)
```

```
context Class
inv: self.Schema->size()=1

inv: Class.allInstances()->
forall(c1, c2 | c1.Schema.Name.concat(':').concat(c1.Name).toUpper()=
c2.Schema.Name.concat(':').concat(c2.Name).toUpper()
implies c1=c2 )

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )

inv: self.Method->
forall(m1, m2 | m1.Name.toUpper()=m2.Name.toUpper()
implies m1=m2 )

inv: self.Property->
forall(p1, p2 | p1.Name.toUpper()=p2.Name.toUpper()
implies p1=p2)
```

```
context Method
inv: self.Domain.Supertype->includes(self.Overriden.Domain)

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )
```

```
context Named Element
inv: self.Qualifier->
forall(q1, q2 | q1.Name.toUpper()=q2.Name.toUpper()
implies q1=q2 )
```

Fig. 3. CIM Metamodel formalized with OCL

A similar rule has been defined for methods: “The Domain of the overridden Method must be a superclass of the Domain of the overriding Method”.

```
context Method
  inv: self.Domain.Supertype->
    includes(self.Overridden.Domain)
```

There are some interesting rules about associations, such as “Associations are classes with an Association qualifier”, “An Association cannot inherit from a non-association Class”, or “Any subclass of an Association is an association”.

```
context Association
  inv: self.Supertype->isEmpty() or
    self.Supertype->forall( st |
      st.oclIsTypeOf(Association))
  inv: self.Qualifier->includes(q |
    q.Name='Association')
  inv: self.Subtype->forall( st |
       st.oclIsTypeOf(Association))
```

For references, rules like "The Class referenced by the Range association of an overriding Reference must be the same as, or a subtype of, the Class referenced by the Range associations of the Reference being overridden" or "The Domain of a Reference must be an Association" have been defined as follows:

```
context Reference
  inv: self.Range=
    self.Overridden.Range or
    self.Overriden.Range.Subtype->
      includes(self.Range)
  inv: self.Domain.oclIsTypeOf(Association)
```

There are some rules about qualifiers, but they reference elements that are not in the CIM metaschema class diagram, such as "A Qualifier Type (not shown in **Fig. 1**) is a Named Element and must be used to supply a type for a Qualifier (that is, a Qualifier must have a Qualifier Type). A Qualifier Type can be used to type zero or more Qualifiers" or "A Qualifier is a Named Element and has a Name, a Type (intrinsic data type), a Value of this type, a Scope, a Flavor and a default Value. The type of the Qualifier Value must agree with the type of the Qualifier Type". They could be defined as:

```
context QualifierType
  inv: self.oclIsKindOf(NamedElement)
  inv: self.Qualifier->size()>=0
context Qualifier
  inv: self.QualifierType->size()=1
  inv: self.oclIsKindOf(NamedElement)
  inv: self.attributes()->
    includesAll(Set { 'Name', 'Type', 'Value', 'Scope',
'Flavor', 'Default-Value' })
  inv:
    self.Value.oclIsTypeOf(self.Type)
```

Other rules have been defined mainly to constraint that element names are case insensitive.

Redundant rules about multiplicity or specialization have not been included, because metaclasses associations already define them graphically in the diagram.

This formalization allows a compiler to check the defined information, by automatically processing the UML diagram with OCL rules, with just one possible interpretation, avoiding rules with multiple meanings.

5 Comparison with an Ontology Language

The approach given in this paper can be compared to other one presented in [4], where the use of the Web Ontology Language (OWL) [17] has been proposed to define management information. In this other work, the elements of this language have been studied, mapping them with management language constructions, and adding those facets not included in OWL that are common in management languages.

Both approaches could be valid depending on the application scope to enhance the semantic expressiveness of the information, as they provide different advantages and drawbacks, which can be taken into account when choosing the language more suitable for each case:

- CIM formalized version allows a smooth transition from the network management domain to the ontology domain. Moreover, with this approach it is not necessary to translate every CIM schema to other language, as they can be directly validated with defined OCL rules because the metamodel has been formalized. OCL can also be used in CIM to define constraints about the behavior of classes, methods and properties. Other UML artifacts different from OCL have also been used to describe behavior in [18]. Nevertheless, currently there are not tools to work with this information model from a semantic viewpoint. Another drawback is that this solution is CIM-centric: other management information defined in other language (for instance, SNMP MIBs) cannot directly profit from this approach.
- An ontology language such as OWL provides all the expressiveness of this kind of languages, because they are formalized. Also, there are many tools developed to use and validate it. In addition, other rule languages such as the Semantic Web Rule Language (SWRL) [14] can be used to define constraints about the behavior of that information. However, its main drawback is that all already defined management information has to be translated to OWL. Moreover, OWL does not allow the definition of class methods, so that part of the information can get lost.

These advantages and drawbacks can be compared in **Table 2**. As a conclusion, it can be said that CIM is better for current management tools, but OWL is better if ontology engines are used that analyze information to infer knowledge. The final decision can be based on the tools that are going to be used to handle the management information.

Table 2. Comparison of formalized CIM metaschema and OWL approaches

	Advantages	Drawbacks
Formalized CIM metaschema	• Smooth transition to the use of ontologies • CIM schemas are kept the same • OCL can also be used to define constraints for CIM schemas	• Semantic tools have to be developed • It only deals with CIM information
OWL	• Already formalized • Many developed tools • Definition of constraints with SWRL	• All management information has to be translated to OWL • Class methods cannot be defined in OWL

6 Conclusions

This paper has presented a proposal to formalize the CIM metaschema. For this, OCL has been used, rewriting the rules defined in natural language. With this, a compiler can load and interpret these rules to automatically check the semantics of defined information. This approach has also been compared with the use of an ontology language, obtaining that both solutions can be valid, providing each one some advantages and drawbacks.

There are some open issues. For instance, this formalization has been applied to CIM metaschema qualifiers, but not to qualifiers instances. This can be a problem, because these elements are used to extend the metaschema. However, in ontology languages every element is formalized. Thus, it would be necessary to carry out a formalization for every qualifier instance as performed above, specifying which invariants must be true for every element that have such qualifiers. This task is more complicated, because due to the qualifiers nature, metamodel and model levels get mixed.

Another future task is related to the measurement units. Currently, CIM only defines a list of values for the Units qualifier, but not their relationship. If they are formalized a property can be directly translated from a measurement unit to another. This formalization is useful if different classes are going to be compared. Then, for instance, two classes that measure the throughput of a channel can be mapped, even if one is in bits per second and the other in Megabits per second. Existing ontology libraries such as Ontolingua STANDARD-UNITS or DAML GNU Units can be leveraged with this purpose.

Acknowledgements

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

References

1. R. Studer, V.R. Benjamins, and D. Fensel: Knowledge Engineering: Principles and Methods. Data & Knowledge Engineering. Vol. 25 (1998) 161-197.
2. 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.
3. E. Lavinal, T. Desprats, Y. Raynaud: A Conceptual Framework for Building CIM-Based Ontologies. In: Proc. of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003), Colorado Springs, Colorado, U.S.A., (2003)
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. G. Lanfranchi, P. Della Peruta, A. Perrone, D. Calvanese: Towards a new landscape of systems management in an autonomic computing environment. IBM Systems Journal, Vol. 42, No. 1 (2003) 119-128

6. S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, E. Stokes: Toward a Formal Common Information Model Ontology. Lecture Notes in Computer Science, Vol. 3307, Springer Verlag (2004) 11-21
7. S. Bapat: Towards Richer Relationship Modeling Semantics. IEEE Journal on Selected Areas in Communications, Vol. 11, No. 9 (1993) 1373-1384
8. T. Zhang, PanosGavriil Tsigaridas: A Knowledge-based Model for Network Service Management. In Proceedings of the First IEEE Symposium Global Data Networking (December 1993)
9. Distributed Management Task Force, Inc.: Common Information Model Specification, Version 2.2. DMTF Standard DSP0004 (June 1999)
10. J. E. López de Vergara, V. A. Villagrá, J. Berrocal, J. I. Asensio, R. Pignaton: Semantic Management: Application of Ontologies for the Integration of Management Information Models. In: Proc. of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003), Colorado Springs, Colorado, U.S.A. (2003)
11. S. Cranefield, M. Purvis: UML as an Ontology Modelling Language. In Proc. of the Workshop on Intelligent Information Integration, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden (1999)
12. P. Kogut, S. Cranefield, L. Hart, K. Baclawski, M. Kokar, J. Smith: UML for Ontology Development. Knowledge Engineering Review Journal, Special Issue on Ontologies in Agent Systems, Vol. 17, Issue 1 (2002) 61-64
13. Object Management Group: Object Constraint Language Specification. OMG document formal/03-03-13 (March 2003)
14. 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)
15. Object Management Group: Unified Modeling Language (UML), version 1.5. OMG document formal/03-03-01 (March 2003)
16. Distributed Management Task Force, Inc.: Common Information Model (CIM) Infrastructure Specification, Version 2.3 Preliminary. DMTF Standard DSP0004 (October 2004)
17. D. L. McGuinness, F. van Harmelen: OWL Web Ontology Language Overview. W3C Recommendation (10 February 2004)
18. M. Sibilla, A. Barros de Sales, J. Broisin, P. Vidal, F. Jocteur-Monrozier: Behaviour modelling: a contribution to CIM. DMTF Academic Alliance Paper (2004)