World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:2, 2008

Defining a Semantic Web-based Framework for Enabling Automatic Reasoning on CIM-based Management Platforms

Fernando Alonso, Rafael Fernández, Sonia Frutos, and Javier Soriano

Abstract**—** CIM is the standard formalism for modeling management information developed by the Distributed Management Task Force (DMTF) in the context of its WBEM proposal, designed to provide a conceptual view of the managed environment. In this paper, we propose the inclusion of formal knowledge representation techniques, based on Description Logics (DLs) and the Web Ontology Language (OWL), in CIM-based conceptual modeling, and then we examine the benefits of such a decision. The proposal is specified as a CIM metamodel level mapping to a highly expressive subset of DLs capable of capturing all the semantics of the models. The paper shows how the proposed mapping provides CIM diagrams with precise semantics and can be used for automatic reasoning about the management information models, as a design aid, by means of new-generation CASE tools, thanks to the use of *state-of-the-art* automatic reasoning systems that support the proposed logic and use algorithms that are sound and complete with respect to the semantics. Such a CASE tool framework has been developed by the authors and its architecture is also introduced. The proposed formalization is not only useful at design time, but also at run time through the use of rational autonomous agents, in response to a need recently recognized by the DMTF.

Keywords**—** CIM, Knowledge-based Information Models, Ontology Languages, OWL, Description Logics, Integrated Network Management, Intelligent Agents, Automatic Reasoning Techniques.

I. Introduction

THE growing complexity, heterogeneity and dynamism inherent in emerging telecommunications networks, distributed systems and advanced information and communication services, as well as their increased criticality and strategic importance in the networked economy, calls for the adoption of increasingly more sophisticated technologies for their management, coordination and integration to assure adequate levels of functionality, performance and reliability.

Of the available technologies, those associated with the autonomous agent-based computation paradigm [16], [10] are precisely the ones that are better accepted for conceiving new techniques for developing management solutions with a higher level of automation, greater potential for interoperability within open environments and better capabilities of cooperation. Autonomous agent technology and, particularly, Multi-Agent Systems provide in this respect a series of new and exciting possibilities in the field of network operations and management [6], [7], such as formal semantic-level knowledge representation, automatic reasoning and learning capabilities, high-level communication languages and protocols, frameworks for automated negotiation, goal-driven proactive behavior or rational decision making.

The formalisms used in management information modeling and representation are closely related to the capabilities of automation, interoperation and cooperation of the management solutions developed on their basis. The success of the process of incorporating autonomous agents, capable of reasoning and dynamically integrating knowledge and services, as an enabling technology for new management solutions, largely depends on the evolution of the information models of existing management architectures [8] towards explicit declarative-type semantic models, equipped with a solid formal basis, that can capture the semantics of the management information models, as well as their formal specification, communication and automatic reasoning about these models. *Knowledge Representation* and *Conceptual Modeling* [1] are the fields of *Artificial Intelligence* that have progressed most in this respect. However, they have had hardly any impact on any of the management information models built to date.

Considering the advances achieved in the field of *Knowledge Representation* by the international research community, the strategy followed for building the existing management information models should be reconsidered and the possibility of including techniques related to the field of *Knowledge Representation* should be examined, as should the benefits of such a decision. In this paper, we demonstrate the adequacy of the use of *Description Logics* [11] and the *Web Ontology Language OWL* [18] for formally defining the structure and constraints of management information in the context of the information model of a management architecture.This model determines the modelling approach and notation used to describe the managed elements, which includes their identification, structure, behavior and relations to other elements.

Common Information Model (CIM) is the chosen information model. CIM is the standard formalism for modeling management information developed by the Distributed Management Task Force DMTF in the context of its WBEM proposal [2], designed to provide a conceptual view of the managed environment. There is widespread agreement on the need to provide CIM diagrams with precise semantics that can be used to establish a common understanding of the formal meaning of the CIM metamodel constructs used for

Manuscript received March 28, 2006. This work is being supported in part by the Spanish Ministry of Science and Technology (contract TIC2001-3451); and the Spanish Ministry of Industry, Tourism and Commerce under its National Program of Service Technologies for the Information Society (contract FIT-350110-2005-73).

F.Alonso, R. Fernández, S.Frutos and J.Soriano are with the with Department of Computer Science, Technical University of Madrid (UPM), Spain. (e-mail: {falonso,sfrutos,jsoriano}@fi.upm.es, rfdez@pegaso.ls.fi.upm.es)

the purpose of enabling interoperation and cooperation. This point has been repeatedly recognized by the DMTF since a keynote address presented at the IEEE Policy 2003 Conference [17]. To our knowledge, however, no specific proposal for CIM model formalization has yet been made. Although there are proposals for formalizing structural UML diagrams [3], [4] that are easily adaptable to CIM diagrams, none of these proposals amounts to a solid foundation for the development of automatic reasoning techniques based on algorithms that are sound and complete with respect to the semantics.

In this paper, we propose the inclusion of formal knowledge representation techniques, based on DLs, in CIM-based conceptual modeling. The proposal is specified as a CIM metamodel level mapping to a highly expressive subset of DLs called $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$. The aim is to be able to automatically reason about the management information models conceptualized by CIM both in the design phase (to verify formal properties of the models, such as their satisfiability, extract logical implications from and detect inconsistencies or redundancies in the models) and at run time, through the use of rational agents that are able to exploit the DL-OWL expressions of CIM models and their instances as domain ontologies in their deduction, coordination and action processes. To achieve this latter aim, the proposal contemplates the use of the Semantic Web ontology language OWL for XML-based representation and exchange of the CIM models previously formalized by means of DLs. This latter point amounts to a significant advance over the use of the MOF (Managed Object Format) textual specification language or CIM/XML mapping proposed by the DMTF. The proposal as a whole makes up an original Management Information Model, called CIMOnt, and has been developed in the context of a novel Management Architecture called Nesmarq.

The remainder of the paper is organized as follows. Section II presents the findings of an analysis that takes all the information models proposed to date into account and justifies the need for a semantically-enabled information model. Section III argues the adequacy of DLs as a representation formalism capable of capturing the semantics of CIM models. Section IV is the core of the paper and describes the proposed mapping of CIM to the $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ DL. Section V describes the CIMOnt framework architecture, the set of new-generation CASE tools that we have developed to help to demonstrate the ideas presented in this paper. Section VI presents the new reasoning services available as a result of the formalization process for both the conceptual design phase and at run time. Finally, section VII discusses the main conclusions of this work.

II. Management Information Models and the need for a semantics-based approach

It is essential when constructing a management solution to select the best architecture, as this architecture determines to a large extent its capabilities. If reference management architectures are available, open management platforms can be constructed that ease the development of the final applications.

One of the primary goals of a management architecture is to provide a conceptual framework for standardizing the key elements that will later support the management platforms and solutions developed on its basis. This framework is vital for implementing an integrated management solution in an open and strongly distributed environment like the Internet. Management information specification lies therefore at the heart of any management architecture. This is made accessible as managed object definitions built according to a specific model. The managed objects represent an abstraction of those features of the resources and services involved that are considered relevant for the management process. The key elements for standardization, then, include the modeling methods used and the description of the management domain objects that make up the management architecture *information model*. The information model should provide consensus on how to identify these objects, how to represent their state, how to define their behavior and how they can be manipulated.

In [15] we analyzed and qualitatively compared the different management architectures there are from the viewpoint of (a) the information model, in terms of the expressiveness of its respective models and the level of abstraction with which the tasks are specified and (b) the organizational model adopted, as well as the type and granularity of the permitted delegation. This analysis enabled us to classify the existing architectures with respect to the four key requirements needed to cover new management needs: automation, interoperability, cooperation and scalability. In this section we summarize the findings of this analysis as regards information models and justify the approach proposed in this article.

Fig. 1 shows the results of comparing the information models of the different management architectures there are from the viewpoints of their level of abstraction and their expressiveness. As regards their abstraction, we find that all the proposed architectures have need of a full specification of the tasks to be done, except the incipient policy-based management proposal. This proposal nonetheless requires a high level of supervision. In terms of expressiveness, we find a trend towards increasingly more expressive models (Time evolves to the upper right-hand corner of the graph). Hence, there is a move from the use of just data types, through relationship modeling and the provision of graphical notations, to being able to specify and exchange metadata and even behavior.

We also find that none of the existing architectures makes provision for automatic reasoning about the models built and only one can model behavior, and its proposal is confined to a mere graphical notation. Similarly, any metadata modeling is done at a very elementary level. The conclusion is that if the information model is to be semantically richer, with the resulting improvement in terms of flexibility, automation and interoperability, the object paradigm adopted by many architectures should be abandoned in favor of a more evolved theory like the cooperative paradigm. This will call for a restatement of how concepts, their interrelations, metadata should be specified and exchanged, and behavior should be modeled, as well as new graphical modeling languages, along the lines proposed in this paper. Fig. 1 also shows where the architecture containing the proposed semantic information model is positioned with respect to these classification param-

eters.

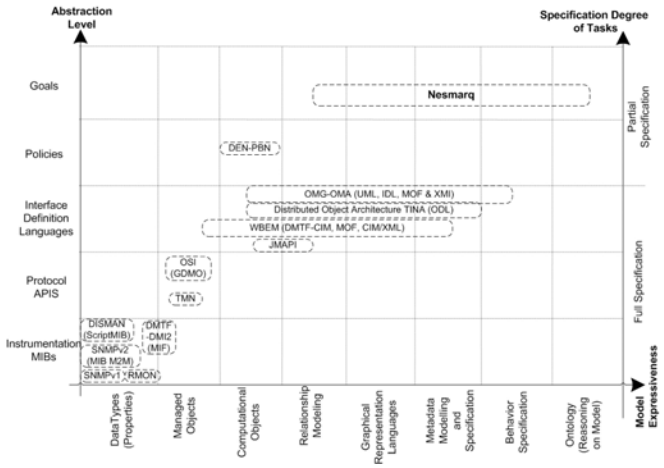

Fig. 1. Comparison between the information models of existing management architectures and CIMOnt (referred to as Nesmarq in the figure).

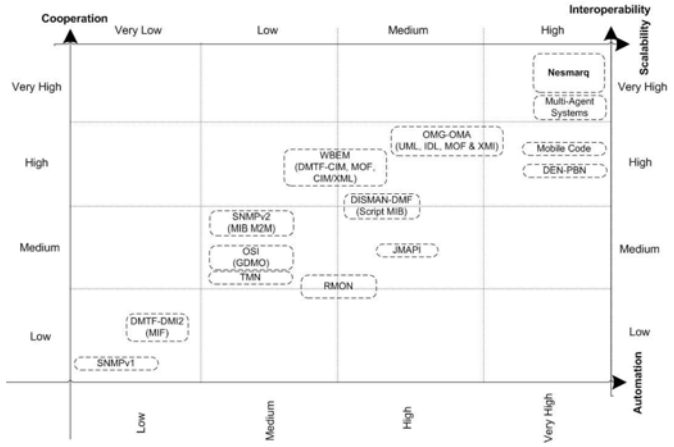

Fig. 2. Qualitative comparison between existing management information models and CIMOnt).

Fig. 2 illustrates how the architectures have evolved over time, improving in terms of automation, interoperability, cooperation and scalability, the optimal levels of which fall in the upper right-hand corner of the graph (Actually, the plotted evolution also depends on the other criteria addressed in the study, apart from the ones we are dealing with here. This is why the two graphs (Fig. 1 and Fig. 2) differ, but the explanation for this difference is outside the scope of this paper). However, the technologies used to date have made this impossible to achieve.

Fig. 2 shows that the existing management architectures have a marked tendency to develop models enabling a high level of management automation, with the resulting increase in scalability. Fig. 2 also shows how this trend implies the need to provide a greater degree of cooperation and interoperability through these models. All this leads to the need to introduce new technologies into the development of the architecture models to achieve of these objectives. One such technology is agents. For the process of incorporating rational autonomous agents, capable of reasoning and dynamically integrating knowledge and services, to be successful, the management architecture information model needs to evolve towards explicit declarative-type semantic models or ontologies based on a sound formal foundation.

In this paper, we describe the semantic information model that we developed for the Nesmarq architecture from the viewpoint of the management information structure. Issues related to the dynamic modeling of the interactions between agents and to the organization model are left for later papers.

CIMOnt is based on DLs, formal ontology languages by themselves, enabling a greater amount of management automation than traditional architecture models, thanks to its use by rational autonomous agents, capable of reasoning, inferring and dynamically integrating knowledge and services conceptualized using the CIM model and formalized semantically by means of DLs. Using the developed model, we have been able to develop a new generation of visual modeling tools capable of reasoning about the models at design time thanks to automatic reasoning engines based on DLs. CIMOnt also includes a metamodel mapping of CIM to OWL (the Semantic Web ontology language), which is a significant advance in the field of XML-based representation and exchange of management models and metadata.

III. Adequacy of DLs for CIM-based Conceptual Modeling

The CIM information model, developed by the DMTF in the context of its WBEM proposal [2], is formally described by means of an object-oriented metamodel based on the UML modeling language [12], [14], called CIM Metaschema, which defines the elements used to express the model, as well as their use and their semantics. These elements are *schemas*, *classes*, *properties*, *methods*, *indications*, *associations* and *references*. Fig. 3 describes the CIM Metaschema.

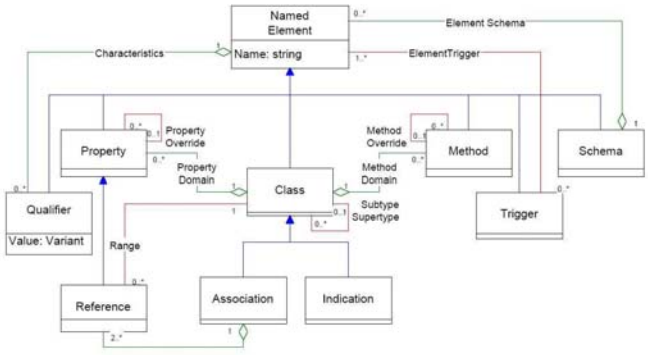

Fig. 3. CIM Metaschema

The *classes* can be organized as *generalization* hierarchies that form a directed graph which rules out single inheritance. The *associations* are class types (i.e. they can also be organized as generalization hierarchies) that represent the relationships between two or more objects. The roles performed by each object that participates in an association are defined by a particular type of property called *reference*.

The model also includes *qualifiers* that characterize other elements and provide a controlled mechanism for extending the metamodel. Accordingly, the *association* and *indication* elements are defined by two standard qualifiers.

Like object-oriented modeling, CIM modeling is derived from classical set theory and classification theory. CIM's abstraction and classification capabilities mean that it can define the fundamental concepts of the management domain (objects), and group these objects by types (classes), identifying their common characteristics (properties), their interrelationships (associations) and their behavior (methods). This makes the use of DLs suitable as a representation formalism, based on *concepts* (classes) and *roles* (relationships), capable of capturing and expressing the semantics present in the CIM models, as well as formally representing other additional aspects not accounted for by the CIM metamodel, such as class disjunction, full class partitioning into subclasses or association navigability. In particular, the $\mathcal{ALE CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ logic proposed in this paper is especially suited for the highly expressive CIM information structuring mechanisms.

DLs are decidable subsets of *first-order logic*, making them an effective formalism for management knowledge representation. This eases the design of intelligent management solutions, furnished with inductive-style reasoning services capable of reaching implicit consequences from the explicitly represented management knowledge.

IV. Mapping the CIM Metamodel to $\mathcal{ALE CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ DL

In this section, we describe a CIM mapping to DLs designed for the use of description-logic based automatic reasoning systems, such as [5], [9].

Table I sets out the concepts and roles constructors used to develop the proposed mapping for the purpose of configuring a sufficiently expressive Description Logic. From the above table, it is clear that a $\mathcal{ALE CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ logic had to be used. This decision is a compromise between the expressiveness of the language used to build the *terminology knowledge bases* (TBox), which contains the models, and the complexity involved in the reasoning processes both on the TBox and on the instance or *assertion knowledge bases* (ABox). In this respect, the use of other types of Description Logic, such as the family of logics derived from $\mathcal{DLR}$ logic, which eliminate the binary roles constraint and introduce constructors for n-ary roles, would have allowed us to develop a more intuitive mapping than the one proposed, but at a much greater computational cost. The tests run during the results generation phase [15] showed that the RACER tool [5] classified a Tbox knowledge base with all the CIM version 2.6 models built by the DMTF expressed in DL according to the proposed mapping in a matter of a few seconds, whereas this same tool was unable to classify the knowledge base expressed in $\mathcal{DLR}_{reg}$, that is, the extension of $\mathcal{DLR}$ with the constructors of union, composition and transitive closure of binary roles as a mapping of the n-ary roles on two of its components.

The selected logic is really a subset of the minimum logic required for mapping, as we have the following equivalences:

$$C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$$
$$\exists R.C \equiv \neg\forall R \neg C$$

which means that $\mathcal{C} \equiv \mathcal{U}\varepsilon$. However, we have opted not to use these simplifications so as to make the mapping clearer without introducing further computational complexity in doing so, as the two logics are equally expressive.

Table I does not set out the permitted axioms for building TBox knowledge bases. Apart from the traditional axioms of *concept subsumption* ($C \sqsubseteq D$) and *concept equivalence* ($C \equiv D$), constrained in the sense that only D can be a concept expression (and, therefore, C must be an atomic concept), we have also used the *role subsumption* axiom to be able to create roles hierarchies. Hence, the subindex $\mathcal{H}$.

The formal semantics of the mapping of the CIM metaschema to $\mathcal{ALE CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ is based on a *Model Theory*, in which $\mathcal{I} = \langle D, \cdot^{\mathcal{I}} \rangle$ is an *interpretation* where:

- $D \neq \emptyset$ represents a domain or universe of discourse such that $D = \Sigma \cup \Upsilon$ with $\Upsilon = \bigcup_{i=1}^{n} \Upsilon_{D_i}$, $\Upsilon_{D_i} \cap \Upsilon_{D_j} = \varnothing$, and $\Sigma \cap \Upsilon = \varnothing$, such that Σ is the domain of managed objects and Υ_{D_i} is the set of values associated with each basic data type D_i supported by CIM (integer, string, etc).
- $\cdot^{\mathcal{I}}$ is the interpretation function that maps:
  - $D_i^{\mathcal{I}} = \Upsilon_{D_i}$.
  - $C_i^{\mathcal{I}} \subseteq \Sigma$.
  - $A_i^{\mathcal{I}} \subseteq \Sigma \times \Upsilon$.
  - $R_i^{\mathcal{I}} \subseteq \Sigma \times \cdots \times \Sigma \equiv \Sigma^n$.

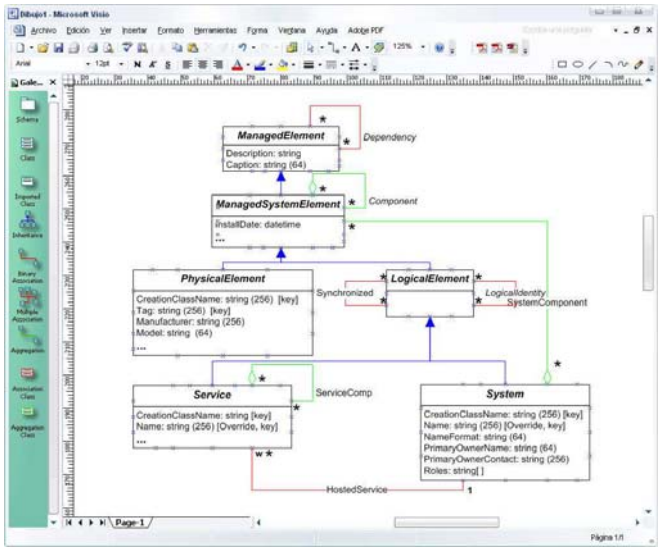

Fig. 4. CIM Core Model 2.6 (extract) edited with CIMOnt CASE tool

Table II summarizes the proposed CIM-$\mathcal{ALE CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ mapping. The following sections describe this mapping. To illustrate the results, we present, incrementally, part of the process of translating the CIM Core Model shown in fig. 4.

A. Mapping CIM classes

A CIM *class* denotes a set of objects with common characteristics in terms of *properties*, *methods* and *associa-*

TABLE I
TYPES OF DESCRIPTION LOGICS USED TO DEVELOP THE PROPOSED MAPPING

Constructor	Syntax	Semantics	DL
Atomic Concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$\mathcal{FL}_0$
Domain | Empty	$\top \mid \bot$	$\Delta^{\mathcal{I}} \mid \emptyset$	
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
Universal	$\forall R.C$	$\{x \mid \forall y : R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$	
Existential	$\exists R.\top$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y)\}$	$\mathcal{FL}^-$
Atomic Negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$	$\mathcal{AL}$
Qualified Existential	$\exists R.C$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\}$	$\mathcal{E}$
Concept Negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$	$\mathcal{C}$
Enumeration	$a_1 \cdots a_n$	$a_1^{\mathcal{I}} \cdots a_n^{\mathcal{I}}$	O
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
Cardinality	$\geq nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \geq n\}$	$\mathcal{N}$
	$\leq nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} \leq n\}$	
	$= nR$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y)\} = n\}$	
Qualified	$\geq nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \geq n\}$	$\mathcal{Q}$
Cardinality	$\leq nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \leq n\}$	
	$= nR.C$	$\{x \mid \sharp\{y \mid R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} = n\}$	
Selection	$f : C$	$\{x \in Dom(f^{\mathcal{I}}) \mid C^{\mathcal{I}}(f^{\mathcal{I}}(x))\}$	$R_{\mathcal{F}}$
Transitive Roles	R^+	$\bigcup_{n \geq 1}(R^{\mathcal{I}})^n$	$()^+$
Inverse Roles	R^-	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(a,b)\}$	$()_{R^-}$
Role Composition	$R \circ S$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} = \{(x,z) \mid \exists y \in \Delta^{\mathcal{I}} \cdot R^{\mathcal{I}}(x,y) \wedge S^{\mathcal{I}}(y,z)\}$	$()_{R^\circ}$

tions, which means that it is represented as a concept C in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^-_{\mathcal{H}R^{+\circ}}$ DL.

B. Mapping Generalization Hierarchies

A *generalization* or inheritance relationship between two CIM classes specifies that each instance of the *child* class is also an instance of the *parent* class, and that the instances of the child class inherit properties present in the parent class (and satisfy other additional properties) and can participate in its associations. The CIM *generalization* relationship is expressed as an inheritance between $\mathcal{AL}\varepsilon\mathcal{CNOQ}^-_{\mathcal{H}R^{+\circ}}$ concepts, taking advantage of the fact that the semantics of the inclusion assertions ($C_i \sqsubseteq C_j$) is based on set inclusion and respects the concept of substitution.

The CIM generalization relationship can distinguish between four types of situation with different semantics: *Partial and non-disjoint*, *Partial and disjoint*, *Total and non-disjoint*, and *Total and Disjoint*.

Partial and non-disjoint

The meaning of this constraint is:

$$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n.$$

which can be translated to a set of *first-order logic* formulae:

$$\forall x \cdot C_i(x) \rightarrow C(x), i = 1, ..., n.$$

Expression (1) specifies this constraint in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^-_{\mathcal{H}R^{+\circ}}$ Description Logic.

$$C_i \sqsubseteq C, i = 1, ..., n. \tag{1}$$

Partial and disjoint

The meaning of this constraint is:

$$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n$$
$$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \varnothing, i = 1, ..., n$$

which can be translated to a set of *first-order logic* formulae:

$$\forall x \cdot C_i(x) \rightarrow C(x) \wedge \bigwedge_{j=i+1}^{n} \neg C_j(x), i = 1, ..., n$$

Expression (2) specifies this relationship in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^-_{\mathcal{H}R^{+\circ}}$ Description Logic.

$$\begin{aligned} &C_i \sqsubseteq C, i = 1, ..., n \\ &C_i \sqsubseteq \neg C_j, \forall i \neq j \end{aligned} \tag{2}$$

Total and non-disjoint

The meaning of this constraint is:

$$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n$$
$$C^{\mathcal{I}} \subseteq \bigcup_{i=1}^{n} C_i^{\mathcal{I}}$$

TABLE II
SUMMARY OF THE CIM-$\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ MAPPING

CIM Element	Concepts and roles	DL Axioms introduced	
Class C	Concept C		
Attr. a of C with type T	Binary role	$C \sqsubseteq \langle = 1a \rangle \sqcap \exists a.T$	(5)
Key attribute a of C	Binary role a	$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D \sqcap \langle \leq 1A^{-} \rangle$	(8)
Attr. a of C with type T[]	Binary role a	$C \sqsubseteq \langle \geq 1a \rangle \sqcap \forall a.T$	(6)
Attr. a with card. $(n_i..n_j)$	Binary role a	$C \sqsubseteq \langle \geq n_i a \rangle \sqcap \langle \leq n_j a \rangle \sqcap \forall a.T$	(7)
Dependency A	Binary role A Roles R_1 and R_2	$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^{-}.C_1$ $C_1 \sqsubseteq \forall A.C_2 \sqcap [\geq n_i A] \sqcap [\leq n_j A]$ $C_2 \sqsubseteq \forall A^{-}.C_1 \sqcap [\geq m_i A^{-}] \sqcap [\leq m_j A^{-}]$ $A \sqsubseteq R_1, R_1 \sqsubseteq A, A^{-} \sqsubseteq R_2, R_2 \sqsubseteq A^{-}$	(11)
N-ary association with multiplicity	Concept A Roles $A_r, R_1 \ldots R_n$	$A \sqsubseteq \exists R_1.C_1 \sqcap \cdots \sqcap \exists R_n.C_n \sqcap$ $\langle \leq 1R_1 \rangle \sqcap \cdots \sqcap \langle \leq 1R_n \rangle$ $C_i \sqsubseteq \forall R_i^{-}.A \sqcap \langle \geq n_i R_i^{-} \rangle \sqcap \langle \leq n_j R_i^{-} \rangle$ $i = 1, ..., n$	(9)
Binary association with multiplicity	Concept A Roles A_r, R_1 and R_2	$A \sqsubseteq \exists R_1.C_1 \sqcap \exists R_2.C_2$ $\sqcap \langle \leq 1R_1 \rangle \sqcap \langle \leq 1R_2 \rangle$ $C_1 \sqsubseteq \forall R_1^{-}.A \sqcap \langle \geq m_i R_1^{-} \rangle \sqcap \langle \leq m_j R_1^{-} \rangle$ $C_2 \sqsubseteq \forall R_2^{-}.A \sqcap \langle \geq n_i R_2^{-} \rangle \sqcap \langle \leq n_j R_2^{-} \rangle$ $A_r \equiv R_1^{-} \circ R_2$ $C_1 \sqsubseteq \forall A_r.C_2 \sqcap \langle \geq m_i A_r \rangle \sqcap \langle \leq m_j A_r \rangle$ $C_2 \sqsubseteq \forall A_r^{-}.C_2 \sqcap \langle \geq m_i A_r^{-} \rangle \sqcap \langle \leq m_j A_r \rangle$	(10)
Inheritance relationship (partial and not disjoint)		$C_i \sqsubseteq C, i = 1, ..., n$	(1)
Inheritance relationship (partial and disjoint)		$C_i \sqsubseteq C, i = 1, ..., n$ $C_i \sqsubseteq \neg C, \forall i \neq j$	(2)
Inheritance relationship (total and not disjoint)		$C_i \sqsubseteq C, i = 1, ..., n$ $C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$	(3)
Inheritance relationship (total and disjoint)		$C_i \sqsubseteq C, i = 1, ..., n$ $C_i \sqsubseteq \neg C, \forall i \neq j$ $C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$	(4)

which can be translated to a set of *first-order logic* formulae:

$$\forall x \cdot C_i(x) \rightarrow C(x), i = 1, ..., n$$
$$\forall x \cdot C(x) \rightarrow \bigvee_{i=1}^{n} C_i(x)$$

Expression (3) specifies this constraint in $\mathcal{ALE}\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ Description Logic.

$$C_i \sqsubseteq C, i = 1, ..., n$$
$$C \sqsubseteq \bigsqcup_{i=1}^{n} C_i \tag{3}$$

Total and disjoint

This is the relationship type least used in the context of knowledge representation languages owing to the intrinsic openness of ontologies. However, its presence should be considered, as it is the type that contributes more semantics to the CIM model. The meaning of this constraint is:

$$C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}, i = 1, ..., n$$
$$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \varnothing, \forall i \neq j$$
$$C^{\mathcal{I}} \subseteq \bigcup_{i=1}^{n} C_i^{\mathcal{I}}$$

which can be translated to a set of *first-order logic* formulae:

$$\forall x \cdot C_i(x) \rightarrow \bigvee_{i=1}^{n} C_i(x)$$
$$\forall x \cdot C_i(x) \rightarrow C(x) \wedge \bigwedge_{j=i+1}^{n} \neg C_j(x)$$

Expression (4) specifies this constraint in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ Description Logic.

$$\begin{aligned} &C_i \sqsubseteq C, i = 1, ..., n \\ &C_i \sqsubseteq \neg C_j, \forall i \neq j \\ &C \sqsubseteq \bigsqcup_{i=1}^{n} C_i \end{aligned} \tag{4}$$

This formalization also captures the generalization relationship between CIM associations. The CIM metaschema cannot formally express the above-mentioned generalization relationship types, although they are usually specified graphically in a diagram either using textual or UML notation.Below we give an example of how to formalize the total and disjoint generalization relationship between the *ManagedSystemElement*, *PhysicalElement* and *LogicalElement* concepts.

$$\begin{aligned} &ManagedSystemElement \sqsubseteq Class \sqcap \\ &\quad (PhysicalElement \sqcup LogicalElement) \\ &PhysicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap \\ &\quad \neg LogicalElement \\ &LogicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap \\ &\quad \neg PhysicalElement \end{aligned}$$

$\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ DLs have a direct mapping to OWL [11]. To illustrate this idea, we show below the OWL expression of this DL formalization. The rest of the OWL expression for the CIM Core model is omitted for reasons of space and the verbosity of the OWL notation.

```
<owl:Class rdf:ID="ManagedSystemElement">
   <rdfs:comment>ManagedSystemElement</rdfs:comment>
   <rdfs:label xml:lang="en">Managed System Element
   </rdfs:label>
</owl:Class> <owl:Class rdf:ID="LogicalElement">
   <rdfs:comment>LogicalElement</rdfs:comment>
   <rdfs:label xml:lang="en">Logical Element
   </rdfs:label>
   <rdfs:functionalSubClassOf
      rdf:resource="#ManagedSystemElement"/>
   <owl:disjointWith
      rdf:resource="#PhysicalElement"/>
</owl:Class> <owl:Class rdf:ID="PhysicalElement">
   <rdfs:comment>PhysicalElement</rdfs:comment>
   <rdfs:label xml:lang="en">Physical Element
   </rdfs:label>
   <rdfs:functionalSubClassOf
      rdf:resource="#ManagedSystemElement"/>
   <owl:disjointWith
      rdf:resource="#LogicalElement"/>
</owl:Class>
```

C. Mapping CIM Properties

The constraint imposed by assigning a *property* A with a data type D to a class C is:

$$C^{\mathcal{I}} \subseteq \{x \in \Sigma \mid \sharp(A^{\mathcal{I}} \cap (\{x\} \times \Upsilon_D)) \geq 1\}$$

which can be translated to a *first-order logic* formula:

$$\forall x \cdot C(x) \rightarrow \exists y \cdot A(x, y) \wedge D(y)$$

The axioms set out in (5), (6), and (7) express this relationship in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ DL.

$$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D \tag{5}$$

for single-value properties,

$$C \sqsubseteq \langle \geq 1A \rangle \sqcap \forall A.D \tag{6}$$

for type A[] properties, and

$$C \sqsubseteq \langle \geq n_i A \rangle \sqcap \langle \leq n_j A \rangle \sqcap \forall A.D \tag{7}$$

for properties with cardinality $(n_i..n_j)$, where C is a concept, A is a binary role and D is a data type.

The CIM mapping to Description Logics described above can only express the semantics of properties acting as simple keys, by including the axiom (8).

$$C \sqsubseteq \langle = 1A \rangle \sqcap \exists A.D \sqcap \langle \leq 1A^{-} \rangle \tag{8}$$

The semantics of a compound key is defined by the relationship

$$\mathcal{R} : C \rightarrow (A_1 \times A_2 \times \cdots \times A_n)$$

such that $\mathcal{R}$ is injective and $\mathcal{R}^{-}$ is functional, which means that it would be necessary to be able to consider the relationship $(A_1 \times A_2 \times \cdots \times A_n)$ as a concept such that its instances could constitute the range of the relationship $\mathcal{R}$. In DL, however, *concepts* and *roles* represent disjoint sets.

Below we show the mapping of the *System* class attributes. They are all single attributes of the type *String*, except *Roles*, which is a multi-valued attribute.

$$\begin{aligned} System \sqsubseteq &Class \sqcap LogicalElement \sqcap \\ &\exists CreationClassName.string \sqcap \\ &\langle \leq 1CreationClassName \rangle \sqcap \\ &\exists Name.string \sqcap \langle \leq 1Name \rangle \sqcap \\ &\exists NameFormat.string \sqcap \langle \leq 1NameFormat \rangle \sqcap \\ &\exists PrimaryOwnerName.string \sqcap \\ &\langle \leq 1PrimaryOwnerName \rangle \sqcap \\ &\exists PrimaryOwnerContact.string \sqcap \\ &\langle \leq 1PrimaryOwnerContact \rangle \sqcap \\ &\forall Roles.string \sqcap \langle \geq 1Roles \rangle \end{aligned}$$

D. Mapping CIM Associations and Dependencies

The constraint imposed by the interrelationship of n classes $C_1 \cdots C_n$ by means of an *association* R is:

$$R^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \times \cdots \times C_n^{\mathcal{I}}$$

which can be translated to a *first-order logic* formula:

$$\forall x_1, ..., x_n \cdot R(x_1, ..., x_n) \rightarrow C_1(x) \wedge \cdots \wedge C_n(x)$$

This constraint can be expressed in description logic by means of an n-ary role or by mapping the association R to a concept A and n roles $r_1 ... r_n$, as shown in equation (IV-D). This latter option can easily represent *properties* and *qualifiers* associated with this association by means of new roles as described in section IV-C. Additionally, this latter option obeys CIM semantics, which states that the associations should not be handled as inverse relationships with references associated

with each participant class, but as a different object that has references associated with the participant classes.

$$A \sqsubseteq \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \sqcap \langle \leq 1 r_1 \rangle \sqcap \cdots \sqcap \langle \leq 1 r_n \rangle$$

As a CIM association is a specialization of a CIM class, there may be, apart from properties whose domain is an association, relationships of generalization between associations. The latter are dealt with like inheritance relationships between concepts.

Unlike CIM associations, a CIM *dependency* can only be binary and has no roles. So, a dependency can be expressed by means of a binary role, as shown in (11) in Table II.

1) Expressing Cardinalities: CIM can associate *cardinalities* with the association roles. This amounts to a new constraint for each role that can be expressed as:

$$C_i^{\mathcal{I}} \subseteq \{x \in \Sigma \mid m_i \leq \sharp(R^{\mathcal{I}} \cap (\Sigma \times \{x\} \times \Sigma)) \geq n_i\} i = 1, ..., n$$

which can be translated to a *first-order logic* formula:

$$\forall x_i \cdot C(x_i) \rightarrow \exists^{\geq p} x_1, ..., x_{i-1}, x_{i+1}, ..., x_n \cdot R(x_1, ..., x_n) \wedge \exists^{\leq p} x_1, ..., x_{i-1}, x_{i+1}, ..., xn \cdot R(x_1, ..., x_n)$$

with

$$\begin{aligned} \exists^{\leq n} x \cdot R(x, y) \equiv \\ & \forall x_1, ..., x_n, x_{n+1} \cdot R(x_1, y) \wedge \cdots \wedge R(x_n, y) \\ & \wedge R(x_{,n+1}, y) \rightarrow \\ & (x_1 = x_2) \vee \cdots \vee (x_1 = x_n) \vee (x_1 = x_{n+1}) \vee \\ & (x_2 = x_3) \vee \cdots \vee (x_2 = x_n) \vee (x_2 = x_{n+1}) \vee \\ & \cdots \vee (x_n = x_{n+1}) \end{aligned}$$

and

$$\begin{aligned} \exists^{\geq n} x \cdot R(x, y) \equiv \\ & \exists x_1, ..., x_n \cdot R(x_1, y) \wedge \cdots \wedge R(x_n, y) \\ & \wedge R(x_{n+1}, y) \wedge \\ & \neg(x_1 = x_2) \wedge \cdots \wedge \neg(x_1 = x_n) \wedge \\ & \neg(x_2 = x_3) \wedge \cdots \wedge \neg(x_2 = x_n) \wedge \\ & \cdots \wedge (x_{n-1} = x_n) \end{aligned}$$

Therefore, an n-ary association with cardinalities $(k_i..l_i)$ can be expressed in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ description logic by means of the axioms set out in expression (9). Note that, generally, cardinalities can only be defined for non-transitive roles that, in turn, have no transitive roles. This is not an intrinsic constraint of the DL used, but of the deductive reasoning systems associated with such logics.

$$\begin{aligned} & A \sqsubseteq \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \sqcap \langle \leq 1 r_1 \rangle \sqcap \cdots \sqcap \langle \leq 1 r_n \rangle \\ & C_i \sqsubseteq \forall r_i^{-}.A \sqcap \langle \geq k_i r_i^{-} \rangle \sqcap \langle \leq l_i r_i^{-} \rangle, i = 1, ..., n \end{aligned} \quad (9)$$

2) Expressing Association Navigability: In the case of a binary association, a new role A_r is introduced to constrain the cardinality of the concept acting as domain and specify association navigability, as shown by the set of axioms set out in (10).

$$\begin{aligned} & A_r \equiv r_1^{-} \circ r_2 \\ & C_1 \sqsubseteq \forall A_r.C_2 \sqcap \langle \geq m_i A_r \rangle \sqcap \langle \leq m_j A_r \rangle \\ & C_2 \sqsubseteq \forall A_r^{-}.C_2 \sqcap \langle \geq m_i A_r^{-} \rangle \sqcap \langle \leq m_j A_r \rangle \\ & A \sqsubseteq \exists r_1.C_1 \sqcap \exists r_2.C_2 \sqcap \langle \leq 1 r_1 \rangle \sqcap \langle \leq 1 r_2 \rangle \\ & C_1 \sqsubseteq \forall r_1^{-}.A \sqcap \langle \geq m_i r_1^{-} \rangle \sqcap \langle \leq m_j r_1^{-} \rangle \\ & C_2 \sqsubseteq \forall r_2^{-}.A \sqcap \langle \geq n_i r_2^{-} \rangle \sqcap \langle \leq n_j r_2^{-} \rangle \end{aligned} \quad (10)$$

Role A_r can also specify the transitivity of an association (bear in mind that r_1 and r_2 are not transitive roles).

The following is the mapping of the *ServiceComponent* aggregation and the binary association *HostedService* with cardinality (1..1) in its domain and cardinality (0..*) in its range. The example also shows the use of the *ServiceComponentRole* role to specify the navigability of the *ServiceComponent* association. CIM specifies this same semantics informally by means of a naming rule for the references of the respective association class. The suffixes *A*, *D*, *G* and *P* denote *antecedent*, *dependent*, *group* and *part* respectively.

$$\begin{aligned} System \sqsubseteq & \forall HostedService\_A^{-}.HostedService \sqcap \\ & \forall HostedServiceRole.Service \sqcap \\ Service \sqsubseteq & \forall ServiceComp\_G^{-}.ServiceComp \sqcap \\ & \forall ServiceComp\_P^{-}.ServiceComp \sqcap \\ & \forall ServiceCompRole.Service \sqcap \\ & \exists HostedService\_D^{-}.HostedService \sqcap \\ & \langle \leq 1 HostedService\_D^{-} \rangle \sqcap \\ & \exists HostedServiceRole^{-}.System \sqcap \\ & \langle \leq 1 HostedServiceRole \rangle \\ HostedService \sqsubseteq & Association \sqcap \\ & \exists HostedService\_A.System \sqcap \\ & \langle \leq 1 HostedService\_A \rangle \sqcap \\ & \exists HostedService\_D.Service \sqcap \\ & \langle \leq 1 HostedService\_D \rangle \\ HostedServiceRole \equiv & HostedService\_A^{-} \circ HostedService\_D \\ ServiceComp \sqsubseteq & Aggregation \sqcap \\ & \exists ServiceComp\_G.Service \sqcap \\ & \langle \leq 1 ServiceComp\_G \rangle \sqcap \\ & \exists ServiceComp\_P.Service \sqcap \\ & \langle \leq 1 ServiceComp\_P \rangle \\ ServiceCompRole \equiv & ServiceComp\_G^{-} \circ ServiceComp\_P \end{aligned}$$

E. Mapping Qualifiers

The following methodological criterion has been used to map CIM *qualifiers*:

> *If the distinction between two concepts has definite implications for their relationships with other concepts or involves constraints on other properties of*

other concepts, create a new concept. Otherwise, opt for the use of a property to express this distinction.

In the case of the CIM metaschema, the only qualifiers that require the creation of new concepts are *Aggregation* and *Aggregate*. The others can be expressed by means of properties associated with the concepts included in the scope of the qualifier. The CIM metaschema expression shown in the following section sets out the mapping of the CIM qualifiers according to this criterion.

F. Expressing the CIM metaschema in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$

For the purpose of illustrating the CIM qualifiers mapping proposed in the preceding section, the following shows the expression of the CIM metaschema described in Fig. 3 in $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ according to the formalization process presented above. The concepts and roles created are used as a basis for establishing the $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ representations of other CIM schemas. This will ease the construction of advanced CASE tools with built-in reasoning systems that will help to detect inconsistencies. An example is the detection of an incorrect generalization relationship between a class and an association [class]. Even so, the CIM metamodel semantics cannot express this constraint by itself.

$$\begin{aligned}
NamedElement \sqsubseteq\ & \exists Name.String \sqcap \langle \leq 1 Name\rangle \sqcap \\
& \exists ElementSchema.Element^{-}.ElementSchema \sqcap \\
& \langle \leq 1 ElementSchema.Element^{-}\rangle \sqcap \\
& \forall Characteristics.Qualifier \sqcap \\
& \forall ElementTrigger_Element^{-}.ElementTrigger \sqcap \\
& \forall ElementTriggerRole.Trigger
\end{aligned}$$

$$\begin{aligned}
ElementTrigger \sqsubseteq\ & \exists ElementTrigger_Element.NamedElement \sqcap \\
& \exists ElementTrigger_Trigger.Trigger \sqcap \\
& \langle \leq 1 ElementTrigger_Element\rangle \sqcap \\
& \langle \leq 1 ElementTrigger_Trigger\rangle
\end{aligned}$$

$$\begin{aligned}
ElementTriggerRole \equiv\ & ElementTrigger_Element^{-} \circ \\
& ElementTrigger_Trigger
\end{aligned}$$

$$\begin{aligned}
Class \sqsubseteq\ & NamedElement \sqcap \\
& \neg\langle Property \sqcup Qualifier \sqcup Method \sqcup \\
& Trigger \sqcup Schema\rangle \sqcap \\
& \forall PropertyDomain.Property \sqcap \\
& \forall MethodDomain.Method \sqcap \\
& \forall Subtype.Class \sqcap \langle \leq 1 Subtype\rangle \sqcap \\
& \forall Subtype^{-}.Class \sqcap \\
& \forall Range_Class^{-}.Range \sqcap \\
& \langle \leq 1 Range_Class^{-}\rangle \sqcap \\
& \forall RangeRole.Reference
\end{aligned}$$

$$Supertype \equiv Subtype^{-}$$

$$\begin{aligned}
Range \sqsubseteq\ & \exists Range_Class.Class \sqcap \\
& \exists Range_Reference.Reference \sqcap \\
& \langle \leq 1 Range_Class\rangle \sqcap \\
& \langle \leq 1 Range_Reference\rangle
\end{aligned}$$

$$RangeRole \equiv Range\_Class^{-} \circ Range\_Reference$$

$$\begin{aligned}
Association \sqsubseteq\ & Class \sqcap \\
& \forall hasReference.Reference \sqcap \\
& \langle \geq 2 hasReference\rangle
\end{aligned}$$

$$Indication \sqsubseteq Class \sqcap \neg Association$$

$$\begin{aligned}
Qualifier \sqsubseteq\ & NamedElement \sqcap \\
& \exists Value.Variant \sqcap \langle \leq 1 Value\rangle \sqcap \\
& \neg\langle Property \sqcup Class \sqcup Method \sqcup \\
& Trigger \sqcup Schema\rangle \sqcap \\
& \exists Characteristics^{-}.NamedElement \sqcap \\
& \langle \leq 1 Characteristics^{-}\rangle
\end{aligned}$$

$$\begin{aligned}
Property \sqsubseteq\ & NamedElement \sqcap \\
& \neg\langle Method \sqcup Trigger \sqcup Schema\ \rangle \sqcap \\
& \exists PropertyOverride.Property \sqcap \\
& \langle \leq 1 PropertyOverride\rangle \sqcap \\
& \forall PropertyOverride^{-}.Property \sqcap \\
& \exists PropertyDomain^{-}.Class \sqcap \\
& \langle \leq 1 PropertyDomain^{-}\rangle
\end{aligned}$$

$$\begin{aligned}
Method \sqsubseteq\ & NamedElement \sqcap \\
& \neg\langle Trigger \sqcup Schema\rangle \sqcap \\
& \forall MethodOverride.Method \sqcap \\
& \langle \leq 1 MethodOverride\rangle \sqcap \\
& \forall MethodOverride^{-}.Method \sqcap \\
& \exists MethodDomain^{-}.Class \sqcap \\
& \langle \leq 1 MethodDomain^{-}\rangle
\end{aligned}$$

$$\begin{aligned}
Trigger \sqsubseteq\ & NamedElement \sqcap \\
& \neg\langle Schema\rangle \sqcap \\
& \forall ElementTrigger_Trigger^{-}.ElementTrigger \sqcap \\
& \langle \geq 1 ElementTrigger_Trigger^{-}\rangle \sqcap \\
& \forall ElementTriggerRole^{-}.NamedElement \sqcap \\
& \langle \geq 1 ElementTriggerRole^{-}\rangle
\end{aligned}$$

$$\begin{aligned}
Schema \sqsubseteq\ & NamedElement \sqcap \\
& \forall ElementSchema^{-}.NamedElement
\end{aligned}$$

$$\begin{aligned}
Reference \sqsubseteq\ & Property \sqcap \exists Range_Class^{-}.Range \sqcap \\
& \langle \leq 1 Range_Reference^{-}\rangle \sqcap \\
& \exists RangeRole^{-}.Class \sqcap \\
& \langle \leq 1 RangeRole^{-}\rangle \sqcap \\
& \exists HasReference^{-}.Association \sqcap \\
& \langle \leq 1 HasReference^{-}\rangle
\end{aligned}$$

Below, the $\mathcal{AL}\varepsilon\mathcal{CNOQ}^{-}_{\mathcal{H}R^{+}\circ}$ expression of the CIM metaschema is extended with the standard *qualifiers* defined

by DMTF:

- New concepts:

$$\begin{aligned} Aggregation \sqsubseteq\ & Association \sqcap \\ & \forall hasReference.Aggregate \sqcap \\ & \langle = 2 hasReference \rangle \end{aligned}$$

$$Aggregate \sqsubseteq\ Reference$$

- New constraints on existing concepts:

$$\begin{aligned} NamedElement \sqsubseteq\ & \forall Description.String \sqcap \\ & \langle \leq 1 Description \rangle \sqcap \\ & \forall DisplayString.String \sqcap \\ & \langle \leq 1 DisplayString \rangle \end{aligned}$$

$$\begin{aligned} Property \sqsubseteq\ & \forall Key.Boolean \sqcap \\ & \langle \leq 1 Key \rangle \sqcap \\ & \forall MappingStrings.String \sqcap \\ & \forall Alias.String \sqcap \\ & \langle \leq 1 Alias \rangle \sqcap \\ & \forall Counter.Boolean \sqcap \\ & \langle \leq 1 Counter \rangle \sqcap \\ & \forall Gauge.Boolean \sqcap \\ & \langle \leq 1 Gauge\ \rangle \end{aligned}$$

$$\begin{aligned} Class \sqsubseteq\ & \forall Abstract.Boolean \sqcap \\ & \langle \leq 1 Abstract \rangle \sqcap \\ & \forall MappingStrings.String \end{aligned}$$

V. CIMOnt Framework Architecture

For the purpose of demonstrating the utility of the mapping proposed in this paper, we have developed a number of tools, which, together, are termed CIMOnt and make up a framework for experimental design. Specifically, a set of CASE tools have been developed for visual ontologies modelling. These tools enhance (a) the visual development of CIM models using MS Visio, (b) the formalization in DL of these models, and (c) their OWL specification according to the proposed mapping. This way the developed CIM models can be checked for logical consistency (on their own and with respect to the other CIM models proposed by DMTF) at design time, and inconsistencies, such as the presence of non-instantiable classes, non-implementable associations, redundancies, etc., can be detected more easily. Fig. 5 shows the proposed architecture for this framework.

CIM management information models are developed visually using MS Visio, thanks the CIMaddin plug-in developed as part of CIMOnt. Using this plug-in the syntactic representation of these models can be generated following the MOF (Managed Object Format) [13] textual specification language or the CIM/XML mapping proposed by DMTF, as can their semantic representation in both DL and the OWL ontologies language. Figure 2 shows a screenshot of the CIMOnt CASE tool while editing the CIM Core Model 2.6.

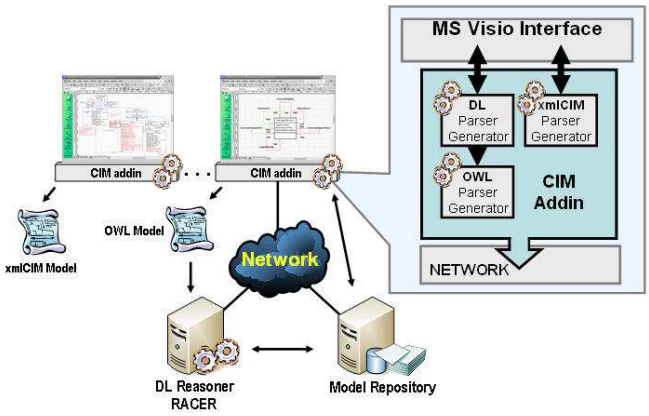

Fig. 5. CIMOnt Framework Architecture

To undertake automatic reasoning about the developed models and check their consistency, CIMaddin accesses the DL reasoner RACER [5] in a distributed fashion. This reasoner provides for the expression of models in both DL and in OWL. The proposed architecture provides for the persistency of models built in a distributed fashion in a centralized Model Repository In this manner, the RACER reasoning engine can dynamically access models stored earlier in later validations that make use of these models. Using the CIMaddin plug-in the CIM models can be retrieved from the Models Repository and their visual representation can be generated automatically from their different syntactic expressions in MOF, DL and OWL.

VI. Automatic Reasoning Services About a CIM Conceptualization

The CIM-$\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+}\circ}$ mapping amounts to a semantic formalization of CIM conceptualizations that can be used to implement automatic reasoning services. The knowledge base semantics likens it to a set of first-order predicate logic axioms. Therefore, like any other set of axioms, it contains implicit knowledge that can be specified through logical inference. The fundamental inference service is *consistency verification* for assertion knowledge bases (ABox) on the basis of which the remainder can be expressed.

A. Reasoning about CIM models

The mapping of a CIM model to a TBox amounts to the construction of a terminology $\mathcal{T}$. During the construction of a CIM model, it is important to discover whether a new class makes sense or, contrariwise, is contradictory to the remainder of the model, in which case it will never be able to be instantiated consistently. From the logical viewpoint, a new concept C makes sense if there exists at least one interpretation $\mathcal{I}$ that satisfies the axioms of $\mathcal{T}$ and for which the concept denotes a non-empty set. This interpretation is called a *model* and is written $\mathcal{T} \models C$. This property of the concept C with respect to $\mathcal{T}$ is called *satisfiability*.

CIM conceptualization designers will use the reasoning services offered by the DL subsystem of their modeling tool

to verify that all the classes created are satisfiable with respect to the remainder of the model and that they comply with the expected generalization/specialization relationships. All the reasoning services will be based on the following prototype services:

Satisfiability

A concept C is satisfiable with respect to a terminology $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \varnothing$. It is written $\mathcal{T} \models C$. The satisfiability of $\mathcal{T}$ itself is expressed as $\mathcal{T} \models$. In other words, satisfiability is the problem of checking whether a concept expression does not necessarily denote the empty concept. Thanks to this, we can infer for example that a class called Router makes sense within the CIM_Network model, because we can have routers as systems. Nevertheless, such a class may make no sense in other domain models.

Subsumption

A concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. It is written $\mathcal{T} \models C \sqsubseteq D$. Determining subsumption is the problem of checking whether the concept denoted by D is considered more general than the one denoted by C. Subsumption can be expressed in terms of satisfiability as $\mathcal{T} \models C \sqsubseteq D \Leftrightarrow \mathcal{T} \not\models C \sqcap \neg D$. Likewise, satisfiability can be expressed in terms of subsumption as $\mathcal{T} \not\models C \Leftrightarrow \mathcal{T} \models C \sqsubseteq \bot$. Using subsumption we can express that those routers whose firmware can be updated via web are subsumed by the set of routers that have an http server running on them.

Equivalence

A concept C is equivalent to a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. It is written $\mathcal{T} \models C \equiv D$. Equivalence can be expressed in terms of satisfiability as $\mathcal{T} \models C \equiv D \Leftrightarrow \mathcal{T} \not\models C \sqcap \neg D$ and $\mathcal{T} \not\models \neg C \sqcap D$, and in terms of subsumption as $\mathcal{T} \models C \equiv D \Leftrightarrow \mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq D$. For example, thanks to the equivalence we can express that the Router class is equivalent to the Gateway class.

Disjunction

Two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \varnothing$ for all models $\mathcal{I}$ of $\mathcal{T}$. Disjunction can be expressed in terms of satisfiability as $\mathcal{T} \not\models C \sqcap D$, and in terms of subsumption as $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$. Thanks to this property, we can distinguish between laser, ink jet, and dot-matrix printers. We will not be able to have a printer that is both a laser and a dot-matrix printer.

B. Reasoning About Management Agent Assertive Knowledge

The management agents that follow the proposed information model handle both the domain TBox, generally derived from the OWL syntax representation of a CIM conceptualization, and an ABox. While the agents can make use of the reasoning services discussed in the preceding section, mainly *concept classification*, they are much more likely to require inference services associated with the assertion knowledge about the individuals in the environment. The principal service of this type is related to *checking the consistency* of the knowledge representation from a strictly logical viewpoint, as incoherent conclusions could be extracted otherwise. Having verified the consistency of the assertion knowledge base, an agent will be able to infer knowledge about the relationships between concepts, roles and individuals (and, therefore, CIM classes, associations and objects) based on the following prototype services:

Consistency

An assertion knowledge base (ABox) $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$ if there exists at least one interpretation $\mathcal{I}$ that is both a model of $\mathcal{T}$ and of $\mathcal{A}$. In other words, our agents will be able to check if all the instances in the knowledge base (ABox) can be classified in the proper place within the concept hierarchy (TBox). That is, for example, if the ABox contains the assertions Router(CISCO_1) and Printer(CISCO_1), the system will be able to infer that, according to the CIM_Network TBox, these statements are inconsistent since Router and Printer are interpreted as disjoint sets.

Instance checking

This involves checking that an assertion is a logical consequence of the knowledge stored in the ABox $\mathcal{A}$. An assertion α is a logical consequence of $\mathcal{A}$, and is written $\mathcal{A} \models \alpha$, if any interpretation that satisfies $\mathcal{A}$ (that is, any model of $\mathcal{A}$), also satisfies α. If α is $C(a)$, this service can be reduced to the consistency service, since $\mathcal{A} \models C(a) \Leftrightarrow \mathcal{A} \cup \{\neg C(a)\}$ is inconsistent. This service deals with the basic reasoning task in an ABox.

Membership (extension)

An agent will generally want to know all the individuals that are an instance of a given concept (both a basic concept and a concept expression). The membership or extension service can use the description language to formulate this type of queries. Given an ABox $\mathcal{A}$, the extension service retrieves the set of individuals $\{a \mid \mathcal{A} \models C(a)\}$. For example, our agents could be interested in finding out from the system all domains that have at least two border routers.

Implementation (Instance classification)

This is a dual inference service to the extension service: given an individual a and an ABox $\mathcal{A}$, the implementation service finds the set of more specific concepts $\{C \mid \mathcal{A} \models C(a)\}$. In this context, more specific concepts means concepts that are minimum with respect to the order induced by the subsumption operator $\sqsubseteq$. Thanks to this service, our agents will be able to decide if one printer should be classified merely as a printer (general concept) or as an inkjet printer (more specific concept).

VII. Conclusions and results

On the basis of the advances achieved in the *Knowledge Representation* field by the international *Artificial Intelligence* community, this paper has reconsidered the strategy followed to build the information models of the existing management architectures, and has examined the possibility of including formal techniques related to the *Knowledge Representation* research field, as well as the benefits of such a decision.

In particular, the paper has shown $\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ to be a highly expressive subset of DLs capable of completely formalizing the semantics of CIM models. For this purpose, a mapping to $\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ DL for each of the CIM metamodel constructors has been elaborated. The principal advantage of the proposed mapping lies in the decidability and computability of the selected logic. This permits the use of state-of-the-art automatic reasoning systems based on this logic, which use semantically sound and complete algorithms.

The paper has also presented the main automatic reasoning services available both for building new generation CASE tools and for use at run time by rational autonomous agents. These CASE tools can verify the satisfiability of the created models, extract logical consequences from and detect inconsistencies and redundancies in the models,whereas the autonomous agents can use the DL expressions of the models and their instances as domain ontologies in their deduction, coordination and action processes. To further this latter objective, the proposal includes the use of the OWL ontologies language for XML-based representation and exchange of the CIM models previously formalized by means of DLs, which amounts to a significant advance with respect to the use of the MOF textual specification language or the CIM/XML mapping proposed by DMTF. A significant original finding generated by applying the ideas set out in this paper is that we have classified and verified the satisfiability of the entire CIM model (version 2.7) proposed by the DMTF based on the presented CIM-$\mathcal{ALECNOQ}^{-}_{\mathcal{HR}^{+\circ}}$ "mapping". This CIM version is composed of 14 models and 89 submodels, and includes a total of 1069 classes, 2444 attributes and 1044 references, which gives an idea of the magnitude of the problem. As a result of this classification, we have been able to formally verify model consistency for the first time. All in all, we have formalized, classified and reasoned about the properties of all 14 models.

Other highly important results are related to the automatic deductive reasoning capability provided by the described architecture models. Accordingly, by connecting the developed CIM-based visual modeling tools (CIMOnt) with a reasoning engine like RACER, which supports the expressivity level required by the proposed mapping [5], such CASE tools can be given with logical inference capabilities for the developed models. In this respect, a CIM-conceptualized information base manager designer will be able to check the consistency of his or her models with respect to the other CIM models developed by the DMTF and/or third parties according to the elements (classes, associations, triggers, indicators, etc.) of these models to which their own models refer. These ideas are valid and applicable by extension to other modeling tools based on other general-purpose modeling languages like, for example, UML.

References

[1] O. Dieste, N. Juristo, A. M. Moreno, J. Pazos, and A. Sierra. *Handbook of Software Engineering and Knowledge Engineering*, volume 1, chapter Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends. World Scientific Publishing Company, 2000.

[2] Web-Based Enterprise Management (WBEM). Technical report, Distributed Management Task Force, 2003.

[3] A. S. Evans. Foundations of the Unified Modeling Language. In D. Duke and A. S. Evans, editors, *Proceedings of the 2nd Northen Formal Methods Workshop*, LNCS, pages 75–81, Heidelberg, Germany, 1997. Springer Verlag.

[4] A. S. Evans. Reasoning with UML class diagrams. In *Proceedings of the 2nd Workshop on Industrial Strength Formal Specification Techniques*. IEEE Computer Society Press, 1998.

[5] V. Haarslev and R. Mller. RACER system description. In *Proceedings of the IJCAR 2001*, number 2083 in LNAI, pages 701–705, Heidelberg, Berlin, 2001. Springer Verlag.

[6] A. L. G. Hayzelden and J. Bigham, editors. *Software Agents for Future Communication Systems*. Springer-Verlag, Heidelberg, Berlin, 1999.

[7] A. L. G. Hayzelden and R. A. Bourne, editors. *Agent Technology for Communication Infrastructures*. John Wiley and Sons, LTD, 2001.

[8] H. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems: Concepts, Architectures and their Operational Application*. Series in Networking. Morgan Kaufmann, 1998.

[9] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Springer Verlag, editor, *LPAR'99*, number 1705 in LNCS, pages 161–180, Heidelberg, Berlin, 1999. Springer Verlag.

[10] M. D'Inverno and M. Luck, editors. *Understanding Agent Systems*. Springer-Verlag, 2002.

[11] D. L. McGuinness et al. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[12] OMG. Unified Modeling Language Specification Version 1.4. Technical report, OMG, 2001.

[13] OMG. Meta-object facility (MOF) specification. Technical report, Object Management Group, 2002.

[14] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Object Technology Series. Addison-Wesley, 1999.

[15] J. Soriano. *Architectural Model for Distributed Systems and Services Management based on Holons and Autonomous Agents*. PhD thesis, Technical University of Madrid, Madrid, Spain.

[16] G. Weiss, editor. *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 1999.

[17] A. Westerinen. What is policy and what can it be?. (keynote). In *Proceedings of the IEEE Policy 2003 Conference*. IEEE Computer Society Press, 2003.

[18] W3C WebOnt WG. Web ontology language (owl) guide. Last call working draft, World Wide Web Consortium, 2003.