

Semantic Modeling of Management Information: Enabling Automatic Reasoning on DMTF-CIM

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 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 DLs [11] and 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 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

Manuscript received February 25, 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 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 remainder of the paper is organized as follows. Section II argues the adequacy of DLs as a representation formalism capable of capturing the semantics of CIM models. Section III is the core of the paper and describes the proposed mapping of CIM to the $\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{HR}+\circ}^-$ DL. Section IV describes the CIMOnt framework architecture, a set of new-generation CASE tools that we have developed to help to demonstrate the ideas presented in this paper. Section V 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 VI discusses the main conclusions of this work.

Like object-oriented modeling, CIM modeling is derived from classical set theory and classification theory. CIM's



DLs (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.

The decision to use the $\mathcal{AL}\mathcal{E}\mathcal{N}\mathcal{O}\mathcal{Q}_{\mathcal{H}\mathcal{R}+\circ}$ DL is a compromise between the expressiveness of the language used to build the *terminology knowledge bases* (TBox), which contains

TABLE I
SUMMARY OF THE CIM- $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}^{-}$ MAPPING

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

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 DLs, 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.

Table I summarizes the proposed CIM- $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}^{-}$ 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. 2.

A. Mapping CIM classes

A CIM *class* denotes a set of objects with common characteristics in terms of *properties*, *methods* and *associations*, which means that it is represented as a concept C in $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}^{-}$ 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 $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}^{-}$ concepts, taking advantage of the fact that the semantics of the inclusion

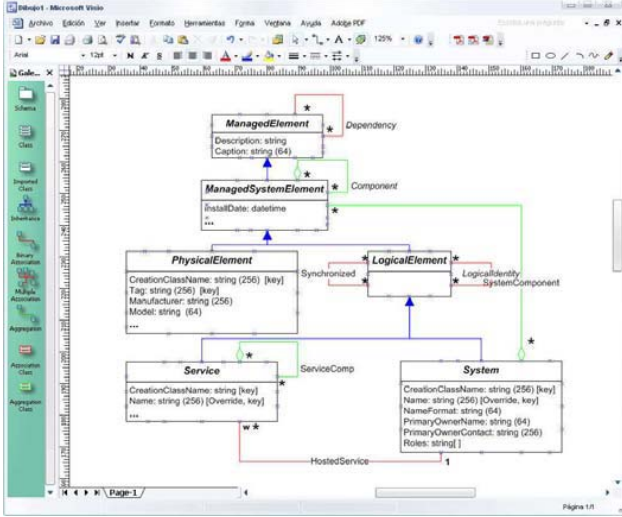


Fig. 2. CIM Core Model 2.6 (extract) edited with CIMOnt CASE tool

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*.

The *Total and Disjoint* generalization is the relationship type least used in the context of knowledge representation languages owing to the intrinsic openness of ontologies. However, it is the type that contributes more semantics to the CIM model. The meaning of this constraint is: $C_i^T \subseteq C^T$, $i = 1, \dots, n$; $C_i^T \cap C_j^T = \emptyset$, $\forall i \neq j$; $C^T \subseteq \bigcup_{i=1}^n C_i^T$, 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 (11) in Table I specifies this constraint in $\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{HR}^+}^-$ DL.

The meaning of the constraints represented by the other three types of generalization relationships and their translation to *first-order logic* formulae have been omitted for reasons of space. Nevertheless, expression (8), (9) and (10) specify these constraints in $\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{HR}^+}^-$.

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.

$ManagedSystemElement \sqsubseteq Class \sqcap$
 $(PhysicalElement \sqcup LogicalElement)$
 $PhysicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap$
 $\neg LogicalElement$
 $LogicalElement \sqsubseteq Class \sqcap ManagedSystemElement \sqcap$
 $\neg PhysicalElement$

$\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{HR}^+}^-$ 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 >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 >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 $C^T \subseteq \{x \in \Sigma \mid \#(A^T \cap (\{x\} \times Y_D)) \geq 1\}$, 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 (1) for single value properties, (3) for $A[]$ type properties, and (4) for properties with cardinality $(n_i..n_j)$ in Table I, where C is a concept, A is a binary role and D is a data type, express this relationship in $\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{HR}^+}^-$ DL.

The CIM mapping to DLs described above can only express the semantics of properties acting as simple keys by including the axiom (2) in Table I.

The semantics of a compound key is defined by the relationship $\mathcal{R} : C \rightarrow (A_1 \times A_2 \times \dots \times A_n)$, 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 $(A_1 \times A_2 \times \dots \times A_n)$ as a concept such that its instances could constitute the range of the relationship $\mathcal{R}$. In DLs, 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.

$System \sqsubseteq Class \sqcap LogicalElement \sqcap$
 $\exists CreationClassName.string \sqcap$
 $\langle \leq 1 CreationClassName \rangle \sqcap$
 $\exists Name.string \sqcap \langle \leq 1 Name \rangle \sqcap$
 $\exists NameFormat.string \sqcap \langle \leq 1 NameFormat \rangle \sqcap$
 $\exists PrimaryOwnerName.string \sqcap$
 $\langle \leq 1 PrimaryOwnerName \rangle \sqcap$
 $\exists PrimaryOwnerContact.string \sqcap$
 $\langle \leq 1 PrimaryOwnerContact \rangle \sqcap$
 $\forall Roles.string \sqcap \langle \geq 1 Roles \rangle$

D. Mapping CIM Associations and Dependencies

The constraint imposed by the interrelationship of n classes $C_1 \dots C_n$ by means of an association R is $R^T \subseteq C_1^T \times \dots \times C_n^T$, which can be translated to a *first-order logic* formula $\forall x_1, \dots, x_n \cdot R(x_1, \dots, x_n) \rightarrow C_1(x_1) \wedge \dots \wedge C_n(x_n)$. This

constraint can be expressed in DL by means of an n-ary role or by mapping the association R to a concept A and n roles $r_1 \dots r_n$, as shown in the first axiom (6) in Table I. This latter option can easily represent *properties* and *qualifiers* associated with this association by means of new roles as described in section III-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.

As a CIM association is a specialization of a CIM class, there may be relationships of generalization between associations, apart from properties whose domain is an association. 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.

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

$$C_i^I \subseteq \{x \in \Sigma \mid m_i \leq \#(R^I \cap (\Sigma \times \{x\} \times \Sigma)) \geq n_i\}$$

$$i = 1, \dots, n$$

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

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

$$\exists^{\leq p} x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n \cdot R(x_1, \dots, x_n)$$

with

$$\exists^{\leq n} x \cdot R(x, y) \equiv \forall x_1, \dots, x_n, x_{n+1} \cdot R(x_1, y) \wedge \dots \wedge$$

$$R(x_n, y) \wedge R(x_{n+1}, y) \rightarrow$$

$$(x_1 = x_2) \vee \dots \vee (x_1 = x_n) \vee (x_1 = x_{n+1}) \vee$$

$$(x_2 = x_3) \vee \dots \vee (x_2 = x_n) \vee (x_2 = x_{n+1}) \vee$$

$$\dots \vee (x_n = x_{n+1})$$

Likewise for $\exists^{\geq n} x \cdot R(x, y)$.

Therefore, an n-ary association with cardinalities $(k_i..l_i)$ can be expressed in $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}$ DL by means of the axioms set out in expression (6) in Table I. 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.

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 (7) in Table I.

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.

$$\begin{aligned} \text{System} &\sqsubseteq \forall \text{HostedService_A}^- . \text{HostedService} \sqcap \\ &\quad \forall \text{HostedServiceRole} . \text{Service} \sqcap \\ \text{Service} &\sqsubseteq \forall \text{ServiceComp_G}^- . \text{ServiceComp} \sqcap \\ &\quad \forall \text{ServiceComp_P}^- . \text{ServiceComp} \sqcap \\ &\quad \forall \text{ServiceCompRole} . \text{Service} \sqcap \\ &\quad \exists \text{HostedService_D}^- . \text{HostedService} \sqcap \\ &\quad (\leq 1 \text{HostedService_D}^-) \sqcap \\ &\quad \exists \text{HostedServiceRole}^- . \text{System} \sqcap \\ &\quad (\leq 1 \text{HostedServiceRole}) \\ \text{HostedService} &\sqsubseteq \text{Association} \sqcap \\ &\quad \exists \text{HostedService_A} . \text{System} \sqcap \\ &\quad (\leq 1 \text{HostedService_A}) \sqcap \\ &\quad \exists \text{HostedService_D} . \text{Service} \sqcap \\ &\quad (\leq 1 \text{HostedService_D}) \\ \text{HostedServiceRole} &\equiv \text{HostedService_A}^- \circ \text{HostedService_D} \\ \text{ServiceComp} &\sqsubseteq \text{Aggregation} \sqcap \\ &\quad \exists \text{ServiceComp_G} . \text{Service} \sqcap \\ &\quad (\leq 1 \text{ServiceComp_G}) \sqcap \\ &\quad \exists \text{ServiceComp_P} . \text{Service} \sqcap \\ &\quad (\leq 1 \text{ServiceComp_P}) \\ \text{ServiceCompRole} &\equiv \text{ServiceComp_G}^- \circ \text{ServiceComp_P} \end{aligned}$$

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.

For the purpose of illustrating this criterion, the following shows the expression of part of the CIM metaschema described in Fig. 1 in $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}$ according to the formalization process presented above.

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.

$$\begin{aligned} \text{Aggregation} &\sqsubseteq \text{Association} \sqcap \forall \text{hasReference} . \text{Aggregate} \sqcap \\ &\quad (= 2 \text{hasReference}) \\ \text{Aggregate} &\sqsubseteq \text{Reference} \end{aligned}$$

The $\mathcal{AL}\varepsilon\mathcal{CN}\mathcal{OQ}_{\mathcal{HR}^{+o}}$ expression of the *NamedElement* concept is

$$\begin{aligned} \text{NamedElement} &\sqsubseteq \exists \text{Name} . \text{String} \sqcap (\leq 1 \text{Name}) \sqcap \\ &\quad \exists \text{ElementSchema} . \text{Element}^- . \text{ElementSchema} \sqcap \\ &\quad (\leq 1 \text{ElementSchema} . \text{Element}^-) \sqcap \\ &\quad \forall \text{Characteristics} . \text{Qualifier} \sqcap \\ &\quad \forall \text{ElementTrigger} . \text{Element}^- . \text{ElementTrigger} \sqcap \\ &\quad \forall \text{ElementTriggerRole} . \text{Trigger} \end{aligned}$$

Below, this expression is extended with the standard *qualifiers* defined by DMTF, defining new constraints on the existing concept:

$$\text{NamedElement} \sqsubseteq \forall \text{Description.String} \sqcap (\leq 1 \text{Description}) \sqcap \forall \text{DisplayString.String} \sqcap (\leq 1 \text{DisplayString})$$

The concepts and roles created for expressing the CIM metaschema are used as a basis for establishing the $\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{H}R^{+o}}^{-}$ representations of other CIM schemas, such as the CIM Core model described above (remember that the proposal is specified as a CIM metamodel level mapping). 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.

IV. 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 both the visual development of CIM models using MS Visio and the formalization in DL of these models and their OWL specification according to the described 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. Figure 3 shows the proposed architecture for this framework.

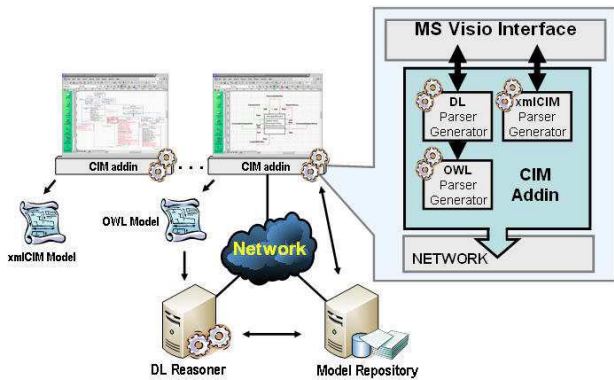


Fig. 3. CIMOnt Framework Architecture

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.

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.

V. AUTOMATIC REASONING SERVICES ABOUT A CIM CONCEPTUALIZATION

As mentioned above, the $\text{CIM-}\mathcal{AL}\mathcal{E}\mathcal{CN}\mathcal{O}\mathcal{Q}_{\mathcal{H}R^{+o}}^{-}$ 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 DL 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 on 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 the CIMOnt 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 prototype services such as concept satisfiability, subsumption, equivalence and disjunction [11].

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 prototype services such as ABox consistency, instance checking, membership (extension) and implementation (instance classification)[11].

VI. CONCLUSIONS

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 $\mathcal{AL}\varepsilon\mathcal{CNOQ}_{\mathcal{HR}^+}^-$ to be a highly expressive subset of DLs capable of completely formalizing the semantics of CIM models. For this purpose, a mapping to $\mathcal{AL}\varepsilon\mathcal{CNOQ}_{\mathcal{HR}^+}^-$ 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- $\mathcal{AL}\varepsilon\mathcal{CNOQ}_{\mathcal{HR}^+}^-$ "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 ar-

chitecture 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 Northern 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. Miller. 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.