

# The Common Information Model (CIM) Standard - An Analysis of Features and Open Issues

Vladimir Tomic<sup>\*</sup>, Slobodanka Djordjevic-Kajan<sup>\*\*</sup>

**Abstract** – In this paper an analysis of the Common Information Model (CIM) standard for object-oriented information specification in network and system management is given, including a short discussion of several important potential benefits and open issues, and a presentation of original extensions of the standard - solutions to some open issues.

**Keywords** – network and system management, interoperability, Common Information Model (CIM), methods, invariants

## I. Introduction

The Common Information Model (CIM) [1] is a new standard OO information model in network and system management (NSM), developed by the DMTF committee to enable interoperability and information sharing between various NSM systems. The idea is not to replace existing NSM solutions, but to complement and integrate them by providing abstraction, consolidation, and unification of all management information from various sources. In that respect, CIM provides OO information specification mechanisms independent of management information exchange, as well as standard OO models which can serve as a basis for describing real managed environments. It is intended to be independent both of managed environments and underlying implementations and to preserve investments in existing NSM solutions.

The CIM standard is separated into two parts. The CIM specification document [1] describes and discusses a formal definition (called Meta Schema) of the information model and its basic concepts, a formal definition of the language used to describe CIM constructs (called Managed Object Format - MOF), the concepts of naming in CIM, mapping techniques between CIM and other information models, and some possible usages of CIM. On the other hand, descriptions of standard CIM models, covering the most important aspects of network and system management, are given in separate documents (available at <http://ftp.dmtf.org/cim>). The OO concept of inheritance is used for hierarchical organization of CIM models. In the root of the hierarchy is the Core model, describing a small set of basic concepts applicable to all areas of NSM. Apart from the Core model, the standard CIM schema also contains the Common model. The Common model is a set of CIM (sub)models describing concepts common to particular management areas (currently: systems, devices, networks, and applications), but independent of any particular

NSM technology or implementation. Classes of the Common model should be specializations of Core model classes, and serve as base classes for classes of Extension schemas which are technology-specific CIM models describing particular managed entities.

## II. The Interoperability Problem

Today, a large number of different NSM standards, like SNMP, DMI, the system of ISO OSI standards around CMIS/P, TMN, TINA, etc., exist. Unfortunately, different standardizing committees defining these standards have not (in most cases) coordinated their work. This resulted in a huge incompatibility between standards, regarding goals, concepts, terminology, level of abstraction, provided solutions, etc. Furthermore, even the most prominent of these standards are (more or less) accepted only in particular management areas and only for particular management aspects. Therefore, for NSM covering several management areas (e.g. in an enterprise), management software supporting several standards must be used. However, due to the incompatibility of standards, it becomes very hard to achieve interoperability (the most important aspect of which is information sharing) between programs supporting different standards and, consequently, to consistently manage modern complex and heterogeneous networks and systems as single entities. Due to the lack of interoperability, the complexity and the price of NSM solutions are significantly increased, while the overall power is decreased.

In spite of significant differences, it can be noticed that the core of SNMP, CMIS/P, and DMI standards are protocols for management information exchange between managing and managed entities. The basic functionality of such protocols is comparable in the sense that they do not define high-level management information operations, but define only basic manipulations like access, addition, deletion and modification. However, standard syntax of messages exchanged between managing and managed entities is not enough to achieve interoperability [2, 3, 4]. Additionally, it is needed to further standardize semantics in the form of appropriate information models, defining specification of management information and operations on that information independently from management information exchange. A good information model can significantly alleviate development of management software and improve its understandability, flexibility, and extensibility.

SNMP, CMIS/P, and DMI standards all provide, directly or indirectly, some kind of information model or description of

<sup>\*</sup> Faculty of Electronic Engineering, Beogradska 14, 18000 Nis, Yugoslavia; E-mail: vladat@kalca.junis.ni.ac.yu

<sup>\*\*</sup> Faculty of Electronic Engineering, Beogradska 14, 18000 Nis, Yugoslavia; E-mail: sloba@kalca.junis.ni.ac.yu

concepts used for management information specification. Unfortunately, contrary to certain similarities between standard protocols for management information exchange, there are drastic differences between these information models. Additionally, none of the information models of the most prominent NSM standards is without serious technical shortages [4]. Apart from other technical shortages, these information models are either too simple (SNMP, DMI) or too complex (the set of ISO OSI management standards), which in both cases makes their usage quite complex and prevents efficient specification of complex management information. Although mappings between different information models are defined in some cases, the practical usage of such mappings is very hard. The differences in information models are one of the main causes of NSM standards incompatibility, and, consequently, of the interoperability problem. While incompatible information models prevent interoperability even when similar exchange protocols are used, with standardized specification of management information and operations on that information a degree of interoperability (primarily, information sharing) can be achieved even with different exchange protocols. Therefore, a very strong need to somehow unify and standardize specification of management information and operations on that information emerges.

Taking into consideration the size of investments made in management solutions based on existing standards, the lack of cooperation between various standardization committees and numerous technical problems it becomes obvious that a new standard intended to replace all other NSM standards would encounter a considerable resistance. Several recent standardization attempts of this kind have failed and even aggravated the interoperability problem. Therefore, it would be more reasonable to try to standardize only an information model, which would be used as a complement and an integration point of existing NSM standards. Such solution would offer both interoperability and investment protection and could lead to more open and less expensive NSM solutions. However, regarding the fact that none of the information models of the most prominent NSM standard is a convenient basis for such standardization, a new, technically better and easier to use, standard information model in NSM was needed. The CIM standard was developed with intention to answer that need.

### III. CIM - Potential Benefits and Open Issues

As discussed above, the most important potential benefit of CIM is achieving both interoperability (primarily, information sharing) and investment protection. Two CIM features, object-orientation and support for mappings to and from other NSM information models, have an extremely important role in achieving the goal of CIM as an integration point of different NSM information models [4].

On one hand, the combination of potential benefits of object-orientation (like improved modeling of semantics, abstraction, modularity, integrity, understandability, reusability, maintainability, flexibility, decreased redundancy, etc.) is not provided by other methodologies. For example, the concept of

inheritance enables vendors to offer standard compliant management information, while adding value and differentiating their products through appropriate extensions and sophistications. Also, the ability to specify data types of arbitrary complexity as appropriate classes can overcome the problem of different data types existing in various information models.

On the other hand, CIM also provides an important support for investment protection by means of technique, recast, and domain methodologies for mapping management information to and from other information models. With this support and implementation independence, CIM can be used in a variety of ways (e.g. for a repository). DMTF has put an accent on mappings between DMI MIFs and CIM, while some work involving other information models, like SNMP MIBs and ISO OSI GDMO, is in progress. However, as noticed in [5], mapping activities must be conducted very carefully, because with more extensive mapping the possibility for information losses also rises.

A short comparison of CIM with information models of the most prominent NSM standards (SNMP, ISO OSI management standards, and DMI), presented in [4], showed that CIM is in a lot of aspects technically superior. For example, medium complexity of information specification concepts in CIM produced, due to object-orientation, high expressive power and medium complexity of usage. As it can be seen from Table I, such results could not be achieved by other compared information models.

TABLE I  
A COMPARISON OF CIM WITH INFORMATION MODELS OF OTHER NETWORK AND SYSTEM MANAGEMENT STANDARDS

Standard	Complexity of Concepts	Power of Concepts	Complexity of Usage
SNMP	Low	Low	High
ISO OSI Management Standards	High	High	High
DMI	Low	Low	Medium
CIM	Medium	High	Medium

Another analysis presented in [4] explored how consistently are OO concepts applied in CIM. The general conclusion of that analysis is that OO concepts are applied in CIM relatively consistently (for comparison, much more consistently than in ISO OSO management standards), but that not all possible OO features are supported. First, the concept of information hiding is not completely supported - CIM properties and references are visible outside their containing CIM instances and need not be accessed with appropriate methods of the same instance. Additionally, the very important concept of methods (as well as the related concept of method parameters) is not worked out in detail and there are some open issues regarding method specification and invocation in CIM. The issue of methods in CIM will be in more detail discussed in the next section, where an appropriate extension of the CIM standard will also be presented. Next, the concept of unique object identifiers is supported in a (not strictly OO) way that

instances are uniquely identified through a set of constant key values, which must not be modified during the whole object lifetime. Also, while the single inheritance concept is supported, the multiple inheritance concept is not.

Additionally, an analysis of CIM concepts which are not OO (associations, qualifiers and qualifier types, qualifier type flavors, naming concepts, etc.) concluded that such concepts usefully complement OO concepts, but that there are important open issues requiring further work [4]. For example, naming concepts in CIM (the concepts of weak references, propagated keys, weak and scoping classes, scoping hierarchy, etc.) are quite complex and opposite to the concept that CIM associations should not affect associated classes. And the concept of associations is very beneficial. One possible solution to this open issue is outlined in the next section. Next, it has been concluded that, although the concept of qualifiers and qualifier types is very powerful and quite useful, it has been overused in CIM in the sense that in some cases it would have been more appropriate to extend the MOF language instead of defining additional qualifier types. Additionally, the concept of qualifier type flavors has some shortages. One possible solution to this problem is presented in [4]. Apart from this, several very important concepts, like CIM Object Managers, access protocols and APIs, triggers and indications, are only briefly mentioned in the CIM standard, and not worked out in detail. These concepts are very important for NSM activities (for example, concepts analogous to triggers and indications exist in all NSM standards) and, hence, should be completely elaborated as soon as possible. Above all, some very useful concepts are completely missing from CIM (e.g. exception handling in CIM methods, expressions, invariants and method pre- and postconditions, etc.). Useful new concepts can be integrated into CIM as appropriate extensions of the standard, as it will be presented in the next section. There is also a number of other CIM open issues, addressing more specific technical details. Several open issues of this kind are identified and discussed in [4]. Possible solutions for some of them are also presented and advocated in that reference.

It must also be noticed that there are some problems with understandability, precision, and consistency of the CIM specification document. Some CIM concepts are used before they are defined, some definitions and/or descriptions are complicated and hard to understand, and terminology used in the document is in some cases imprecise and/or inconsistent. Consequently, it would be beneficial to reorganize (and partially rewrite) that document in next releases of the standard.

On the other hand, another potential benefit of CIM is that standard CIM models can enforce consistency, uniformity, and integration among various CIM models, and still enable extensions into appropriate technology-specific CIM models. However, some important management areas (e.g. diagnostics, user management, security, printing, mail, etc.) are missing from the Common model, so that definition of new (sub)models of the Common model, covering additional management areas, has already started.

## IV. Some Extensions of the CIM Standard

One of the most important CIM open issues is that concept of methods, although one of the main OO concepts and potentially very beneficial, is neither worked out in detail nor widely used in the standard CIM schema. It is recognized [2, 3, 4] that standardization of NSM functionality (i.e. management activities upon managed entities and management services provided to users) can be very beneficial. Although this could provide a functional architecture allowing reuse of functionality across different NSM technologies, none of the prominent NSM standards (including CIM) addresses this issue appropriately. When information manipulation services are described in different ways by different implementations of similar functionality, there is still a considerable lack of interoperability in spite of standard information description. On the other hand, methods defined in the standard CIM schema can serve as a basis for standard CIM APIs. Consequently, specification of standard information manipulation through the concept of methods in CIM can complement existing CIM concepts for standard information description of managed objects, and further improve interoperability between various NSM applications using CIM.

Unfortunately, there are (in our view unnecessary) restrictions on the method syntax which do not allow specification of an accessing/modifying method pair analogous to a reference or an array property. CIM methods may not return references or array values, nor take references as parameters. Consequently, it is not possible to consider properties and references as shorter notations of appropriate accessing/modifying method pairs. This has a negative effect on information hiding. It is also not possible to specify methods that return no value (void methods), nor to specify default values for method parameters. Additionally, the issue of exceptional situations occurring during execution of CIM methods is not addressed at all. The only way to specify information about such exceptional situations is to use method return values (or appropriate output method parameters). However, this approach can lead to software that is hard to write, understand, and maintain, so that it is abandoned in modern programming languages. All modern OO systems explore potential benefits (e.g. software reliability, robustness, maintainability, understandability, compactness, uniformity, etc.) of the exception handling concept.

Due to all this reasons, it seems that an extension of the concept of methods in CIM is needed, so that a possible solution to this open issue is presented in [4]. The solution introduces into CIM concepts of exceptions and exception types and formally defines new method syntax (of the MOF language) enabling specification of exceptions and removing discussed restrictions. Additionally, syntactic, semantic, and pragmatic aspects are discussed and illustrated on examples. For example, an idea for possible structure of the CIM standard exception type hierarchy is presented.

On the other hand, although most of the present semantics in CIM is specified through qualifiers, only a few basic qualifier types related to methods (including method parameters)

are defined. In order to make method specification more powerful and more precise, there is a strong need for a wide variety of new qualifier types related to methods. In [4], a number of new qualifier types related to methods (including also method parameters, exceptions and exception types, and other CIM constructs when related to methods) is suggested. All these new qualifier types are formally defined, discussed, and potentials for their usage are illustrated on examples.

Another very important open CIM issue, not addressed by the standard, is formal specification of invariants (including method pre- and postconditions). Invariants are consistency constraints that are in effect for the entire lifetime of an instance and must be always satisfied in order to consider entities for which they are specified valid. There are two special kinds of invariants for methods: preconditions which are conditions that must be satisfied to start method execution, and postconditions which are conditions which must be satisfied after a valid method execution. Formal specification of invariants (including method pre- and postconditions) in software systems can be very useful because they precisely and unambiguously describe additional semantic aspects (e.g. constraints and dependencies). This can be beneficial in all phases of software development and significantly improve interoperability. Apart from that, the solution to use invariants in order to specify that an instance is unique only in the context of associated instances has multiple advantages (like understandability, maintainability, integrity, etc.) to the present solution to use CIM naming concepts which are not compatible with the concept of associations.

Invariants are often specified informally in natural language, but although this solution is expressive and simple for humans, it is insufficient because it is not machine-parseable (and hence cannot be automatically used by software), is not precise enough, and can lead to ambiguities. Consequently, the means for formal specification of invariants in CIM are needed. Therefore, one possible solution for introduction of formal invariant specification into CIM is developed and presented in [4]. It is based on specification of invariants in the form of Boolean expressions that are values of special new qualifier types. This implies that appropriate new concepts for expression specification also had to be included into CIM. Therefore, an extension of the MOF language called Managed Object Format Expression Extensions (MOFEE) was developed to enrich CIM with expression specification. MOFEE is based on the OCL language of the UML standard for modeling (UML was used for formal definition of the CIM standard and recommended for usage in the process of developing new CIM models), but it is compatible with MOF and appropriate for usage in CIM models. It is primarily intended to be used for invariant specification in CIM, but could be also used for other purposes (e.g. for specification of expressions in default values). This solution for formal invariant specification and specification of expressions formally addresses syntactic (definition of MOFEE) and semantic (definition of three new qualifier types for invariants and method pre- and postconditions) aspects. Also, pragmatic aspects are discussed and a number of examples are used to advocate the solution.

It must be emphasized that the feasibility of all presented CIM extensions is proved by a successful development of experimental extensions of the Hewlett-Packard Public Domain MOF Compiler (Version 2) software (the newer version of the official public domain MOF compiler is available at <http://ftp.dmtf.org/cim>).

## V. Conclusion

Although the CIM standard has features which are in a lot of aspects potentially better than those of information models of other NSM standards, interoperability and information sharing in NSM systems have yet to be achieved. There are a lot of CIM-related open issues, both about management information specification concepts and about standard CIM models, to be solved and the work on improving CIM and resolving open issues is in progress. For example, extensions of the CIM standard that empower the concept of methods or introduce in CIM new concepts like exceptions, expressions, and invariants (including method pre- and postconditions) can further improve interoperability potentials of CIM. Additionally, it must be noticed that in spite of verbal support by major NSM vendors, the potentials of CIM have yet to be proved on the market.

## Acknowledgements

This work resulted from a collaboration with OpenView Software Division of Hewlett-Packard Company, where the first author worked under the supervision of Mr. Holger Dietrich. The authors would like to thank Mr. Dietrich and other colleagues for their help and guidance.

## References

- [1] DMTF, *Common Information Model (CIM) Specification, Version 2.0*, March 3rd, 1998, URL: <http://ftp.dmtf.org/cim/cimdoc20.doc>
- [2] R. H. Glitho, "Management of Heterogeneous Networks", *IEEE Communications Magazine*, Vol. 36, No. 3, p. 34, March 1998.
- [3] S. Poirier and C. Ashford, "Semantics: the Key to Interoperability", in J. Sutherland, D. Patel, C. Casanave, G. Hollowell, and J. Miller (Eds.), *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings*, Berlin: Springer-Verlag, 1997.
- [4] V. Tosic, *On Object-Oriented Information Specification in Network and System Management*, Magisterium Thesis, Faculty of Electronic Engineering, Nis, Yugoslavia, 1998.
- [5] A. K. Larsen, "CIM's Missing Pieces", *Data Communications*, March 1997, URL: <http://www.data.com/tutorials/cim.html>

---

Published in *Proceedings of TELSIKS '99 – 4<sup>th</sup> International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Services*, Nis, Yugoslavia, October 1999, Vol. 2, pp. 677-680, 1999.

ISBN 0-7803-5768-X, IEEE Catalog No. 99EX365