INTERNATIONAL STANDARD

ISO/IEC 19502

First edition
2005-11-01

Information technology — Meta Object Facility (MOF)

Technologies de l'information — Facilité d'objet «méta» (MOF)

Reference number
ISO/IEC 19502:2005(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19502 was prepared by the Object Mangement Group (OMG) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

ISO/IEC 19502 is related to

— ISO/IEC 19501, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

— ISO/IEC 19503, *Information technology — XML Metadata Interchange (XMI)*

— ISO/IEC 14769, *Information technology — Open Distributed Processing — Type Repository Function*

Introduction

This International Standard defines a metamodel (defined using MOF), a set of interfaces (defined using ODP IDL (ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1999), that can be used to define and manipulate a set of interoperable metamodels and their corresponding models. It also defines the mapping from MOF to ODP IDL (ITU rec X920|ISO 14750). These interoperable metamodels include the Unified Modeling Language (UML) metamodel (ISO/IEC 19501:2005), the MOF meta-metamodel, as well as future standard technologies that will be specified using metamodels. The MOF provides the infrastructure for implementing design and reuse repositories, application development tool frameworks, etc. The MOF specifies precise mapping rules that enable the CORBA interfaces for metamodels to be generated automatically, thus encouraging consistency in manipulating metadata in all phases of the distributed application development cycle. Mappings from MOF to W3C XML and XSD are specified in the XMI (ISO/IEC 19503) specification. Mappings from MOF to Java™ are in the JMI (Java Metadata Interchange) specification defined by the Java Community Process.

In order to achieve architectural alignment considerable effort has been expended so that the UML and MOF share the same core semantics. This alignment allows the MOF to reuse the UML notation for visualizing metamodels. In those areas where semantic differences are required, well-defined mapping rules are provided between the metamodels. The UML has been the subject of a separate PAS submission.

The OMG adopted the MOF (version 1.0) in November 1997. It was developed as a response to a request for proposal, issued by the OMG Analysis and Design Task Force, for Metadata repository facility (http://www.omg.org/cgi-bin/doc?cf/96-05-02). The purpose of the facility was to support the creation, manipulation, and interchange of meta models. The most recent revision of MOF, 1.4 was adopted in April 2002, and includes corrections and clarifications to the original 1.3 version, and minor modeling feature additions.

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, *Open Distributed Processing — Reference Model* (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability, and portability can be integrated. RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2.

While not limited to this context, this International Standard is closely related to work on the standardization of Open Distributed Processing (ODP). In particular, the ODP Type Repository Function (ISO/IEC 14769 | Rec. X.960) references the OMG Meta Object Facility, version 1.3. This function specifies how to use the OMG MOF as a repository for ODP types.

Information technology — Meta Object Facility (MOF)

1 Scope

This International Standard specifies the following:

a. An abstract language for specifying, constructing, and managing technology neutral metamodels: A metamodel is in effect an abstract language for some kind of metadata.

b. A framework for implementing repositories & integration frameworks (e.g., tool integration frameworks) that hold metadata (e.g., models) described by the metamodels and which uses standard technology mappings to transform MOF metamodels into metadata APIs.

This International Standard also provides the following:

a. A formal definition of the MOF meta-metamodel; that is, the abstract language for specifying MOF metamodels.

b. A mapping from arbitrary MOF metamodels to CORBA IDL that produces IDL interfaces for managing any kind of metadata.

c. A set of “reflective” CORBA IDL interfaces for managing metadata independent of the metamodel.

d. A set of CORBA IDL interfaces for representing and managing MOF metamodels.

e. An XMI format for MOF metamodel interchange (OMG XMI Specification).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.902 (1996) | ISO/IEC 10746-2:1996, *Open Distributed Processing — Reference Model: Foundations*
- ITU-T Recommendation X.903 (1996) | ISO/IEC 10746-3:1996, *Open Distributed Processing — Reference Model: Architecture*

2.2 International Standards

- ISO/IEC 14769:2001, *Information technology — Open Distributed Processing — Type Repository Function*
- ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*
- ISO/IEC 19503:2005, *Information technology — XML Metadata Interchange (XMI)*

3 Abbreviations and Conventions

The use of IDL conventions is as defined in the IDL standard.

CORBA	Common Object Request Broker Architecture
IDL	Interface Definition Language
MOF	Meta Object Facility
UML	Unified Modeling Language
XMI	XML Metadata Interchange Specification

4 List of Documents

The following is a list of the electronic documents that variously specify the MOF meta-models and MOF IDL APIs and the XMI DTD for MOF meta-model interchange. These documents may be downloaded from the OMG's Web server at:

`http://www.omg.org/technology/documents/formal/mof.htm`

MOF1.4/XMI1.1/Model1.4/Model.xml

This document (ptc/2001-10-05) is *normative*. It expresses the MOF 1.4 Model package as using the XMI 1.1 for MOF Model 1.4 interchange format. The XMI document contains cross-links to the PrimitiveTypes.xml document. It was generated from the Model.modl file below using an automatically generated MOF 1.4 metamodel repository and an automatically generated XMI serializer.

MOF1.4/XMI1.1/Model1.4/PrimitiveTypes.xml

This document (ptc/2001-10-06) is *normative*. It expresses the MOF 1.4 PrimitiveTypes package using the XMI 1.1 for MOF Model 1.4 interchange format. The XMI document was produced by serializing a hard-coded representation of the package using an automatically generated XMI serializer.

MOF1.4/XMI1.1/Model1.4/CorbaIdlTypes.xml

This document (ptc/2001-10-07) is *normative*. It expresses the MOF 1.4 CorbaIdlTypes package using the XMI 1.1 for MOF Model 1.4 interchange format. The XMI document was produced by serializing a hard-coded representation of the package using an automatically generated XMI serializer.

MOF1.4/XMI1.1/Model.dtd

This document (ptc/2001-08-09) is *normative*. It is the standard DTD for XMI 1.1 interchange of MOF 1.4 metamodels.

5 MOF Usage Scenarios

5.1 Overview

The MOF is intended to support a wide range of usage patterns and applications. To understand the possible usage patterns for the MOF, the first thing one needs to understand is the two distinct viewpoints for the MOF:

1. Modeling viewpoint: The designer's viewpoint, looking "down" the meta levels. From the modeling viewpoint, the MOF is used to define an information model for a particular domain of interest. This definition is then used to drive subsequent software design and/or implementation steps for software connected with the information model.

2. Data viewpoint: The programmer's viewpoint, looking at the current meta-level, and possibly looking up at the higher meta-levels. From the data viewpoint, the MOF (or more accurately, a product of the MOF) is used to apply the OMA-based distributed computing paradigm to manage information corresponding to a given information model. In this mode, it is possible for a CORBA client to obtain the information model descriptions and to use them to support reflection.

The second thing one needs to realize is that this MOF specification is intended to provide an open-ended information modeling capability. The specification defines a core MOF model that includes a relatively small, though not minimal, set of constructs for object-oriented information modeling. The MOF model can be extended by inheritance and composition to define a richer information model that supports additional constructs. Alternatively, the MOF model can be used as a model for defining information models. This feature allows the designer to define information models that differ from the philosophy or details of the MOF model. In this context, the MOF Model is referred to as a meta-metamodel because it is being used to define metamodels such as the UML.

Finally, one needs to understand the purpose and the limitations of the MOF model to the CORBA IDL mapping defined by this specification. The prime purpose of the mapping is to define CORBA interfaces for information models defined in terms of the MOF model[1] using standard interfaces and interoperable semantics. These interfaces allow a client to create, access, and update information described by the model, with the expectation that the information will be managed in a way that maintains the structural and logical consistency constraints specified in the information model definition.

While we anticipate that some vendors will supply tools (for example, IDL generators, server generators, and so on) to support the development of software conforming to the mapping, provision of these tools is not a requirement of this specification. The second limitation is that the mapping is only intended to support the MOF model itself; that is, it does not support extensions to the metamodel or to other unconnected information models. Furthermore, since the IDL mapping is not itself modeled in the MOF, there can be no standardized support for extending the mapping or defining new mappings. Finally, the IDL mapping in this specification supports only CORBA IDL. Mappings from the MOF model to other interface definition languages are certainly feasible, as are direct mappings to programming languages or data definition languages. However, these mappings are beyond the scope of the first version of the MOF specification.

1. Both extensions to the MOF meta-model that are expressible in the meta-model itself, and unconnected information models expressed using the MOF meta-model.

5.2 Software Development Scenarios

Initially, one of the most likely applications of the MOF will be to support the development of distributed object-oriented software from high-level models. Such a software development system would typically consist of a repository service for storing the computer representations of models and a collection of associated tools. The latter would allow the programmers and designers to input the models, and would assist in the process of translating these models into software implementations.

In the simple case, the repository service could be an implementation of the MOF model interfaces. This service would be accompanied by tools (for example, compilers or graphical editors) that allow the designer to input information models using a human readable notation for the MOF model. Assuming that the target for software development is CORBA based, the system would include an IDL generator that implements the standard MOF model-to-CORBA IDL mapping.

The usage scenario for this repository service would be along the following lines:

1. The programmer uses the input tools provided by the system to define an object-oriented information model using the notation provided.
2. When the design is complete, the programmer runs the IDL generator to translate the model into CORBA IDL.
3. The programmer examines the IDL, repeating steps 1 and 2 to refine the model as required.
4. The programmer then implements the generated IDL to produce a target object server, and implement the applications that use the object server.

The functionality of the development suite described above can be expanded in a variety of ways. We can:

- Add generator tools to automatically produce the skeleton of an object server corresponding to the generated IDL. Depending on the sophistication of the tool, this skeleton might include code for the query and update operations prescribed by the IDL mapping, and code to check the constraints on the information model.
- Add generator tools to produce automatically stereotypical applications such as scripting tools and GUI-based browsers.
- Extend the repository service to store the specifications and/or implementation code for target server and application functionality that cannot be expressed in the MOF model.

While the MOF model is a powerful modeling language for expressing a range of information models, it is not intended to be the ultimate modeling language. Instead, one intended use of the MOF is as a tool for designing and implementing more sophisticated modeling systems. The following example illustrates how the MOF might be used to construct a software development system centered around a hypothetical “Universal Design Language” (UDL).

Many parallels can be drawn between the hypothetical UDL discussed below and the draft OA&DF UML proposal in that UML is designed to be a general purpose modeling language for visualizing, designing, and developing component software. The UDL can be thought of as an extension, as well as a refinement, of many of the concepts in the UML. The extensions are mainly in the area of providing sufficient detail to complete the implementation framework technologies and defining additional meta models that address various technology domains such as database management, transaction processing, etc.

The developer of a software development system based on UDL might start by using a MOF Model notation to define a meta-model for UDL. Conceivably, the UDL metamodel could reuse part or all of the MOF Model, though this is not necessarily a good idea[2]. The developer could then use a simple MOF-based development system (along the lines described above) to translate the UDL metamodel into CORBA IDL for a UDL repository, and to provide hand-written or generated software that implements the UDL repository and suitable UDL model input tools.

The hypothetical UDL development system cannot be considered complete without some level of support for the process of creating working code that implements systems described by the UDL models. Depending on the nature of the UDL, this process might involve a number of steps in which the conceptual design is transformed into more concrete designs and, finally, into program source code. A UDL development system might provide a range of tools to assist the target system designer or programmer. These tools would need to be supported by repository functions to store extra design and implementation information, along with information such as version histories, project schedules, and so on, that form the basis of a mature software development process.

In practice, a software development system implemented along these lines would have difficulty meeting the needs of the marketplace. A typical software engineering “shop” will have requirements on both the technical and the process aspects of software engineering that cannot be met by a “one-size-fits-all” development system. The current trend in software development systems is for Universal Repository systems; that is, for highly flexible systems that can be tailored and extended on the fly.

A MOF-based universal repository system would be based around the core of the MOF Model, and a suite of tools for developing target metamodels (for example, the UDL) and their supporting tools. Many of the tools in the universal repository could be reflective; that is, the tools could make use of information from higher meta-levels to allow them to operate across a range of model types. Functionality, such as persistence, replication, version control, and access control would need to be supported uniformly across the entire repository framework.

5.3 Type Management Scenarios

A second area where early use of the MOF is likely is in the representation and management of the various kinds of type information used by the expanding array of CORBA infrastructure services.

The CORBA Interface Repository (IR) is the most central type-related service in CORBA. The IR serves as a central repository for interface type definitions in a CORBA-based system. The current IR essentially provides access to interface definitions that conform to the implied information model of CORBA IDL. While the IR interfaces are tuned fairly well to read-only access, there is no standard update interface and no way to augment the interface definitions in the IR with other relevant information, such as behavioral semantics.

Given a simple MOF-based development environment (as described above), it would be easy to describe the implied information model for CORBA IDL using a notation for the MOF Model. The resulting CORBA IDL model could then be translated into the IDL for a MOF-based replacement for the CORBA IR. While this replacement IR would not be upwards compatible with the existing IR, the fact that it was MOF-based would provide a number of advantages. The MOF-based IR would:

- Support update interfaces.
- Be extensible in the sense that it would be feasible to extend the CORBA IDL model specification by (MOF Model) composition and inheritance. This ability would help smooth the path for future extensions to the CORBA object model.
- Make it easier to federate multiple IR instances and to represent associations between CORBA interface types and other kinds of type information.
- Automatically include links to its own meta-information definition expressed using MOF meta-objects.

---

2. The MOF meta-model has specific requirements (e.g., model simplicity and support for automatic IDL generation) that are not generally applicable. As a consequence, it is unreasonable to expect the MOF metamodel design to be suitable for all kinds of object modeling.