

Committee Draft ISO/IEC CD	
Date: 2004-11-19	Reference number: ISO/JTC 1/SC 32N1186
Supersedes document SC 32N1086	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 32 Data Management and Interchange Secretariat: USA (ANSI)	Circulated to P- and O-members, and to technical committees and organizations in liaison for voting (P-members only) by: 2005-02-20 Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated.
--	---

ISO/IEC CD 19763-02:200x(E)

Title: Information technology - Framework for Metamodel interoperability Part 2: Core Model

Project: 1.32.22.01.02.00

Introductory note: The attached document is hereby submitted for a three-month letter ballot to the National Bodies of ISO/IEC JTC 1/SC 32. The ballot starts 2004-11-19.

Medium: E

No. of pages: 80

Address Reply to: Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32, Pacific Northwest National Laboratory, 13667 Legacy Circle Apt H, Herndon, VA, 20171, United States of America

Telephone: +1 202-566-2126; Facsimile; +1 202-566-1639; E-mail: MannD@battelle.org

ISO/IEC CD19763-2:2004(E)-

ISO/IEC JTC 1/SC 32/WG 2

Secretariat:

**Information Technology – Framework for Metamodel Interoperability-- Part-2
: Core Model**

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard. Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

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.ch
Web www.iso.ch

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

Foreword	vii
Introduction.....	viii
1 Scope	1
1.1 Background of MetaModel Framework	1
1.2 Objectives of MetaModel Framework	2
2 Normative references	2
3 Definitions	5
3.1 MOF Terms used in specifying the MMF metamodel	5
3.2 General Terms used in this part of ISO/IEC CD19763	15
3.3 Definitions of Metamodel Constructs.....	18
4 Structure of a MetaModel Framework	21
4.1 MDR and MOF	22
4.2 Key Concepts of MMF Core.....	23
4.3 Registered Target Structure.....	26
4.4 The Meaning of Governing.....	30
4.5 Convention for Definition of MMF Core.....	30
4.6 MMF Core Model (Framework of Registry)	32
4.7 MMF Core Model (Structure of Registered Target).....	37
4.8 MMF Core Model (Relationship of Registered Target)	44
5 Conformance.....	48
5.1 Degree of Conformance	48
5.2 Levels of Conformance	48
5.3 Obligation	48
5.4 Implementation Conformance Statement (ICS)	48
5.5 Roles and Responsibilities for Registration	48
Annex A: MDR Model (Informative)	49
Annex B: MOF Model (Informative)	51
Annex C: ModelClassifier (Informative)	54
Annex D: Level Pair (Informative).....	61
Annex E: Example (Informative).....	70
Bibliography.....	71

Table of Figures

Figure 1- MMF-Core Packages.....	22
Figure 2- Overview of the Metamodel Framework	23
Figure 3- The Relationship between Model and Metamodel	24
Figure 4- Basic Relationship of Registered Target Objects.....	28
Figure 5- Registered Target Objects for a Model Profile	28
Figure 6- Registered Target connected with Model Selection	29
Figure 7- Layered Target Objects with multi-layered registration	29
Figure 8- MMF Core (Framework of Registry).....	32
Figure 9- MMF Core (Structure of Registered Target).....	37
Figure 10- MMF Core (Relationship of Registered Target)	44
Figure 11- MDR Model (Administered Item).....	50
Figure 12- MOF Model Package	52
Figure 13- Category of Registered Target.....	54
Figure 14- MMF Core (Level Pair)	62

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 part of ISO/IEC 19763 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 19763 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 32, *Data Management and Interchange*.

ISO/IEC 19863 consists of the following parts, under the general title *Information technology — Framework for Metamodel Interoperability*:

- Part 1: Reference Model
- *Part 2: Core Model*
- Part 3: Metamodel framework for Ontology
- Part 4: Metamodel framework for Mapping

There are five Informative Annexes for this part of ISO/IEC CD19763-2

- Annex A – MDR Model
- Annex B – MOF Model
- Annex C – ModelClassifier
- Annex D – Level Pair
- Annex E – Example

Introduction

To follow the trends of Electric Commerce and internet, a lot of industrial consortia have been in charge of the standardization of domain specific business objects including business process models and software components using common modelling facilities and exchanging facilities such as UML and XML. They are endeavor to standardize domain specific business process models which represent the best practices of businesses, and standard modelling constructs such as data elements, entity profiles and value domains at each business domains.

One of the things to be mentioned today is that most of those standard efforts tend to be focused on the contents of metamodel to represent and exchange the semantics of businesses, using the UML stereotype mechanism and the XML.

The development of metamodels and UML profiles has been progressed through standardization activities such as UN/CEFACT and OASIS for UMM, ebXML, OMG for MOF, XMI, CWM, EDOC, EAI, and HL7 for HDF etc.

However, every standard group has to specify their metamodel scheme in their own ways. It is necessary to specify common bases for consistent development and registration of metamodels, duplications and inconsistencies inevitably occur.

A unified framework for classifying and registering normative model elements is needed to establish harmonization of metamodels that are developed independently and to reuse them widely across organizations.

A useful de facto standard or de jure standard developed by a standardization organization may be taken up and established as an IS of ISO/IEC/JTC1. Also it is meaningful to build a registry for metamodels based on IS or de facto standard in order to share the information about those model elements. When defining a business object model according to a metamodel and UML profile, stereotype, patterns, components, frameworks etc. are basic modeling constructs to be used as normative. The business model and information system model within an enterprise or among enterprises should be developed consistently based on those normative elements.

Information Technology—Framework for Metamodel interoperability —Part 2:Core Model

1 Scope

The primary purpose of *ISO/IEC CD19763* is to specify the *Framework for Metamodel Interoperability* (see 1.1). *ISO/IEC CD19763-2* also specifies the core model which is required to describe metamodel items, and which may be used in situations where a complete metadata/metamodel registry is appropriate.

1.1 Background of MetaModel Framework

This part of *ISO/IEC CD19763* applies to activities including:

A business object is an object that is identified in building a reusable model of business or a reusable software component that should support interoperability within an enterprise or in the trade between enterprises. To identify and define a concrete object, firstly concepts in the modeling target domain must be put in order, and secondly the meaning of business objects that are picked up from those concepts should be defined exactly with the relationships among the business objects.

If a business application were built on platform using standardized business objects that were platform neutral, it would be easy to transfer it into another platform environment.

-inter-enterprise connection

Using standardized business objects, inter-enterprise connection among customers, trading partners and business partners would be available. Also, if a compliant model that is created based on common business objects were adopted, the interoperability among the subsystems of each section of an enterprise would be maintained in the ease way.

-standardization for core models

As a standard model among enterprises or in an industry, if business objects and core models were defined independent of any particular platform, the development of components and tools conforming standard functions or interoperability would be enabled. Then, the solution business by vendors would be promoted.

-stability and reusability of a standard model

In some cases, the lifecycle of a business application may be longer than one of hardware or platform. If a business application were built based on a business object model that was independent of any particular platform, remaking the same application on a new hardware or xxx platform would be easy and flexible by generating the code of its application from the model. Much of the maintenance cost of the lifecycle could be reduced.

The terminology “business object” is used from various aspects; many kinds of business objects or models could be considered with different granularities, abstraction levels of a modeling target, purposes, intents and viewpoints. For example, modeling message exchange of e-commerce, as an upper level model a business process model such a process scenario about interaction among parties could be considered. As a lower level model, a message format and protocol could be on the modeling target. Moreover, business objects such as a component or framework might be developed and used for implementation level model.

1.2 Objectives of MetaModel Framework

A metamodel provides the definition and specification of a conceptual domain for its concept. The concrete stereotypes and patterns conforming to the metamodel correspond to its value domain and are used to build a concrete model.

Such a metamodel is a kind of standard, stereotypes and patterns conforming to its standard are also standard defined as a profile. Sharing and reusing those elements could achieve standardization of business object models. To improve sharable capability and reusability of object models, normative modeling constructs such as stereotypes and patterns must support the following features:

1. The model must consist of predefined *normative modeling constructs*, not only with modeling methods and also notations.
2. Predefined modeling constructs should include the *common atomic objects*, such as Date, Currency, and Country-code, which can be used without discussion.
3. Common aggregated objects, such as Customer, Company, or Order, which represent business entities, also should be predefined as *normative modeling constructs*, using the normative atomic objects.
4. A business concept, such as Trade, Invoice, or Settlement, which is typically represented as relationship among objects, should be defined as aggregations of the *common elementary aggregated objects* or simple objects. They also have to be predefined as *normative modeling constructs*.
5. Those aggregations that can be predefined using more basic and elementary patterns as a base, could be defined as *object patterns*.
6. Patterns can represent business concepts where they provide for aggregation of more elementary patterns. Therefore, an aggregation or composition mechanism must be provided in patterns.
7. Business rules that govern a business concept can be represented by a pattern with constraints encapsulated in it. Thus, a mechanism for constraint inheritance among patterns must be provided.

For modeling business objects, it is important to standardize conceptual models (i.e. metamodel) of each target domain. Nowadays, the active discussion about development of standards is continuing with organizations such as OMG, ebXML, UN/CEFACT, OASIS, HL7 and so on. For example, OMG has established the MOF (Meta Object Facility) and metamodel architecture, and is promoting the MDA (Model Driven Architecture).

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.

[ISO 8601:2000](#), Data elements and interchange formats – Information exchange – Representation of dates and times

[ISO/IEC 11179-1](#), Information technology – Metadata registries (MDR) - Part 1: Framework

[ISO/IEC 11179-2](#), Information technology – Metadata registries (MDR) - Part 2: Classification

[ISO/IEC 11179-3](#), Information technology – Metadata registries (MDR) - Part 3: Registry metamodel

[ISO/IEC 11179-4](#), Information technology – Metadata registries (MDR) - Part 4: Formulation of data definitions

[ISO/IEC 11179-5](#), Information technology – Metadata registries (MDR) - Part 5: Naming and identification principles

ISO/IEC 11179-6, Information technology – Metadata registries (MDR) - Part 6: Registration

ISO/IEC 11404:1996, Information technology – Programming languages, their environments and system software interfaces – Language-independent datatypes

ISO 12620:1999, Computer applications in terminology – Data categories

ISO/IEC 19501-1:2002, Information technology – Unified Modeling Language (UML) – Part 1: Specification

ISO/IEC 19502-1:200x, Information technology – Meta Object Facility (MOF): Specification

3 Definitions

For the purposes of this document, the following terms and definitions apply.

3.1 defines MOF terms, used in specifying the MMF metamodel.

3.2 lists general terms, and their definitions, used in this document that are not included in either 3.1 or 3.3

3.3 defines metamodel constructs prescribed by the metamodel itself.

3.1 MOF Terms used in specifying the MMF metamodel

3.1.1 abstraction

The essential characteristics of an entity that distinguish it from all other kinds of entities. An abstraction defines a boundary relative to the perspective of the viewer.

3.1.2 actual parameter

Synonym: *argument*.

3.1.3 aggregate [class]

A class that represents the “whole” in an aggregation (whole-part) relationship. See: *aggregation*.

3.1.4 aggregation

A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part. See: *composition*.

3.1.5 architecture

The organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

3.1.6 argument

A binding for a parameter that resolves to a run-time instance. Synonym: *actual parameter*. Contrast: *parameter*.

3.1.7 association

The semantic relationship between two or more classifiers that specifies connections among their instances.

3.1.8 association class

A model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties.

3.1.9 association end

The endpoint of an association, which connects the association to a classifier.

3.1.10 attribute

A feature within a classifier that describes a range of values that instances of the classifier may hold.

3.1.11 binary association

An association between two classes. A special case of an n-ary association.

3.1.12 binding

The creation of a model element from a template by supplying arguments for the parameters of the template.

3.1.13 cardinality

The number of elements in a set. Contrast: *multiplicity*.

3.1.14 child

In a generalization relationship, the specialization of another element, the parent. See: *subclass*, *subtype*. Contrast: *parent*.

3.1.15 call

An action state that invokes an operation on a classifier.

3.1.16 class

A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See: *interface*.

3.1.17 classifier

A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.

3.1.18 classification

The assignment of an object to a classifier. See *dynamic classification*, *multiple classification* and *static classification*.

3.1.19 class diagram

A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.

3.1.20 collaboration

The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction. See: *interaction*.

3.1.21 collaboration diagram

A diagram that shows interactions organized around the structure of a model, using either classifiers and associations or instances and links. Unlike a sequence diagram, a collaboration diagram shows the relationships among the instances. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. See: *sequence diagram*.

3.1.22 component

A physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files.

3.1.23 component diagram

A diagram that shows the organizations and dependencies among components.

3.1.24 composite [class]

A class that is related to one or more classes by a composition relationship. See: *composition*.

3.1.25 composite aggregation

Synonym: *composition*.

3.1.26 composition

A form of aggregation association with strong ownership and coincident lifetime as part of the whole. Parts with non-fixed multiplicity may be created after the composite itself, but once created they live and die with it (i.e., they share lifetimes). Such parts can also be explicitly removed before the death of the composite. Composition may be recursive. Synonym: *composite aggregation*.

3.1.27 concrete class

A class that can be directly instantiated. Contrast: *abstract class*.

3.1.28 constraint

A semantic condition or restriction. Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML. See: *tagged value*, *stereotype*.

3.1.29 container

1. An instance that exists to contain other instances, and that provides operations to access or iterate over its contents. (for example, arrays, lists, sets). 2. A component that exists to contain other components.

3.1.30 containment hierarchy

A namespace hierarchy consisting of model elements, and the containment relationships that exist between them. A containment hierarchy forms a graph.

3.1.31 context

A view of a set of related modeling elements for a particular purpose, such as specifying an operation.

3.1.32 datatype

A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations.

3.1.33 defining model [MOF]

The model on which a repository is based. Any number of repositories can have the same defining model.

3.1.34 dependency

A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

3.1.35 derived element

A model element that can be computed from another element, but that is shown for clarity or that is included for design purposes even though it adds no semantic information.

3.1.36 diagram

A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). UML supports the following diagrams: class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, and deployment diagram.

3.1.37 domain

An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.

3.1.38 element

An atomic constituent of a model.

3.1.39 export

In the context of packages, to make an element visible outside its enclosing namespace. See: *visibility*. Contrast: *import*.

3.1.40 facade

A stereotyped package containing only references to model elements owned by another package. It is used to provide a 'public view' of some of the contents of a package.

3.1.41 feature

A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.

3.1.42 formal parameter

Synonym: *parameter*.

3.1.43 framework

1. A stereotyped package consisting mainly of patterns. See: *pattern*. 2. An architectural pattern that provides an extensible template for applications within a specific domain.

3.1.44 generalizable element

A model element that may participate in a generalization relationship. See: *generalization*.

3.1.45 generalization

A taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed. See: *inheritance*.

3.1.46 import

In the context of packages, a dependency that shows the packages whose classes may be referenced within a given package (including packages recursively embedded within it). Contrast: *export*.

3.1.47 inheritance

The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior. See *generalization*.

3.1.48 instance

An entity to which a set of operations can be applied and which has a state that stores the effects of the operations. See: *object*.

3.1.49 interaction

A specification of how stimuli are sent between instances to perform a specific task. The interaction is defined in the context of a collaboration. See *collaboration*.

3.1.50 interaction diagram

A generic term that applies to several types of diagrams that emphasize object interactions. These include collaboration diagrams and sequence diagrams.

3.1.51 interface

A named set of operations that characterize the behavior of an element.

3.1.52 interface inheritance

The inheritance of the interface of a more specific element. Does not include inheritance of the implementation. Contrast: *implementation inheritance*.

3.1.53 layer

The organization of classifiers or packages at the same level of abstraction. A layer represents a horizontal slice through an architecture, whereas a partition represents a vertical slice. Contrast: *partition*.

3.1.54 link

A semantic connection among a tuple of objects. An instance of an association. See: *association*.

3.1.55 link end

An instance of an association end. See: *association end*.

3.1.56 message

A specification of the conveyance of information from one instance to another, with the expectation that activity will ensue. A message may specify the raising of a signal or the call of an operation.

3.1.57 metaclass

A class whose instances are classes. Metaclasses are typically used to construct metamodels.

3.1.58 meta-metamodel

A model that defines the language for expressing a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

3.1.59 metamodel

A model that defines the language for expressing a model.

3.1.60 metaobject

A generic term for all metaentities in a metamodeling language. For example, metatypes, metaclasses, metaattributes, and metaassociations.

3.1.61 method

The implementation of an operation. It specifies the algorithm or procedure associated with an operation.

3.1.62 Model [MOF]

An abstraction of a physical system, with a certain purpose. See: *physical system*.

Usage note: In the context of the MOF specification, which describes a meta-metamodel, for brevity the meta-metamodel is frequently to as simply the model.

3.1.63 model aspect

A dimension of modeling that emphasizes particular qualities of the metamodel. For example, the structural model aspect emphasizes the structural qualities of the metamodel.

3.1.64 model element [MOF]

An element that is an abstraction drawn from the system being modeled. Contrast: *view element*.

In the MOF specification model elements are considered to be metaobjects.

3.1.65 module

A software unit of storage and manipulation. Modules include source code modules, binary code modules, and executable code modules. See: *component*.

3.1.66 multiple classification

A semantic variation of generalization in which an object may belong directly to more than one classifier. See: *static classification*, *dynamic classification*.

3.1.67 multiple inheritance

A semantic variation of generalization in which a type may have more than one supertype. Contrast: *single inheritance*.

3.1.68 multiplicity

A specification of the range of allowable cardinalities that a set may assume. Multiplicity specifications may be given for roles within associations, parts within composites, repetitions, and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the nonnegative integers. Contrast: *cardinality*.

3.1.69 multi-valued [MOF]

A model element with multiplicity defined whose Multiplicity Type:: upper attribute is set to a number greater than one. The term multi-valued does not pertain to the number of values held by an attribute, parameter, etc. at any point in time. Contrast: *single-valued*.

3.1.70 n-ary association

An association among three or more classes. Each instance of the association is an n-tuple of values from the respective classes. Contrast: *binary association*.

3.1.71 name

A string used to identify a model element.

3.1.72 namespace

A part of the model in which the names may be defined and used. Within a namespace, each name has a unique meaning. See: *name*.

3.1.73 object

An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class. See: *class*, *instance*.

3.1.74 object diagram

A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram. See: *class diagram*, *collaboration diagram*.

3.1.75 package

A general purpose mechanism for organizing elements into groups. Packages may be nested within other packages.

3.1.76 parameter

The specification of a variable that can be changed, passed, or returned. A parameter may include a name, type, and direction. Parameters are used for operations, messages, and events. Synonyms: *formal parameter*. Contrast: *argument*.

3.1.77 parameterized element

The descriptor for a class with one or more unbound parameters. Synonym: *template*.

3.1.78 parent

In a generalization relationship, the generalization of another element, the child. See: *subclass*, *subtype*. Contrast: *child*.

3.1.79 participate

The connection of a model element to a relationship or to a reified relationship. For example, a class participates in an association, an actor participates in a use case.

3.1.80 partition

1. activity graphs: A portion of an activity graphs that organizes the responsibilities for actions. See: *swimlane*. 2. architecture: A set of related classifiers or packages at the same level of abstraction or across layers in a layered architecture. A partition represents a vertical slice through an architecture, whereas a layer represents a horizontal slice. Contrast: *layer*.

3.1.81 pattern

A template collaboration.

3.1.82 property

A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are physical system can be described by one or more models, possibly from different viewpoints. Contrast: *system*.

3.1.83 reference

1. A denotation of a model element. 2. A named slot within a classifier that facilitates navigation to other classifiers. Synonym: *pointer*.

3.1.84 relationship

A semantic connection among model elements. Examples of relationships include associations and generalizations.

3.1.85 repository

A facility for storing object models, interfaces, and implementations.

3.1.86 role

The named specific behavior of an entity participating in a particular context. A role may be static (e.g., an association end) or dynamic (e.g., a collaboration role).

3.1.87 schema [MOF]

In the context of the MOF, a schema is analogous to a package which is a container of model elements. Schema corresponds to an MOF package. Contrast: *metamodel*, *package*.

3.1.88 single inheritance

A semantic variation of generalization in which a type may have only one supertype. Synonym: *multiple inheritance* [OMA]. Contrast: *multiple inheritance*.

3.1.89 stereotype

A new type of modeling element that extends the semantics of the metamodel. Stereotypes must be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML. See: *constraint*, *tagged value*.

3.1.90 string

A sequence of text characters. The details of string representation depend on implementation, and may include character sets that support international characters and graphics.

3.1.91 subclass

In a generalization relationship, the specialization of another class; the superclass. See: *generalization*. Contrast: *superclass*.

3.1.92 subsystem

A grouping of model elements that represents a behavioral unit in a physical system. A subsystem offers interfaces and has operations. In addition, the model elements of a subsystem can be partitioned into specification and realization elements. See *package*. See: *physical system*.

3.1.93 subtype

In a generalization relationship, the specialization of another type; the supertype. See: *generalization*. Contrast: *supertype*.

3.1.94 superclass

In a generalization relationship, the generalization of another class; the subclass. See: *generalization*. Contrast: *subclass*. **supertype**

In a generalization relationship, the generalization of another type; the subtype. See: *generalization*. Contrast: *subtype*.

3.1.96 system

A top-level subsystem in a model. Contrast: *physical system*.

3.1.97 tagged value

The explicit definition of a property as a name-value pair. In a tagged value, the name is referred as the tag. Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML. See: *constraint*, *stereotype*.

3.1.98 template

Synonym: *parameterized element*.

3.1.99 top level

A stereotype of package denoting the top-most package in a containment hierarchy. The *topLevel* stereotype defines the outer limit for looking up names, as namespaces “see” outwards. For example, *opLevel* subsystem represents the top of the subsystem containment hierarchy.

3.1.100 type

A stereotype of class that is used to specify a domain of instances (objects) together with the operations applicable to the objects. A type may not contain any methods. See: *class*, *instance*. Contrast: *interface*.

3.2 General Terms used in this part of ISO/IEC CD19763**3.2.1 characteristic**

abstraction of a property of an object or of a set of objects

NOTE Characteristics are used for describing concepts.

[ISO 1087-1:2000 (3.2.4)]

3.2.2 conceptual data model

a data model that represents an abstract view of the real world

3.2.3 conditional

required under certain specified conditions

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also mandatory (3.2.17) and optional (3.2.28).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.4 data

a re-interpretable representation of information in a formalized manner suitable for communication, interpretation or processing

NOTE Data can be processed by human or automatic means.

[ISO/IEC 2382-1:1998 (01.01.02)]

3.2.5 data model

a graphical and/or lexical representation of data, specifying their properties, structure and inter-relationships

3.2.6 definition

representation of a concept by a descriptive statement which serves to differentiate it from related concepts

[ISO 1087-1:2000 (3.3.1)]

3.2.7 designation

representation of a concept by a sign which denotes it

[ISO 1087-1:2000 (3.4.1)]

3.2.8 entity

any concrete or abstract thing that exists, did exist, or might exist, including associations among these things

EXAMPLE A person, object, event, idea, process, etc...

NOTE Please observe that an entity exists whether data about it are available or not.

[ISO/IEC 2382-17:1999 (17.02.05)].

3.2.9 language

system of signs for communication, usually consisting of a vocabulary and rules

[ISO 5127:2001 (1.1.2.01)]

3.2.10 mandatory

always required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also conditional (3.2.9) and optional (3.2.28).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.11 metadata

data that defines and describes other data

3.2.12 metadata item

an instance of a metadata object

NOTE 1 In all parts of ISO/IEC 19763, this term is applied only to instances of metadata objects described by the metamodel in Clause 4 of ISO/IEC 19763-2. Examples include instances of Model Concept, Model Domain, Model Instances etc.

NOTE 2 A metadata item has associated attributes, as appropriate for the metadata object it instantiates.

3.2.13 metadata object

an object type defined by a metamodel

NOTE 1 In all parts of ISO/IEC 19763, this term is applied only to instances of metadata objects described by the metamodel in Clause 4 of ISO/IEC 19763-2. Examples include instances of Model Concept, Model Domain, Model Instances etc.

3.2.14 metadata register

the information store or database maintained by a Metadata Registry

3.2.15 Metadata Registry MDR

an information system for registering metadata

NOTE The associated information store or database is known as a metadata register.

3.2.16 metadata set

any collection of metadata

3.2.17 metamodel

a data model that specifies one or more other data models

3.2.18 metamodel construct

a unit of notation for modelling

NOTE The metamodel constructs used in ISO/IEC 19763-2 are defined in 3.3.

3.2.19 name

the designation of an object by a linguistic expression

NOTE See also name (of Administered Item) (3.3.83)

3.2.20 object

anything perceivable or conceivable

NOTE Objects may also be material (e.g. an engine, a sheet of paper, a diamond), immaterial (e.g. a conversion ratio, a project plan) or imagined (e.g. a unicorn).

[Adapted from ISO 1087-1:2000 (3.1.1)]

3.2.21 optional

permitted but not required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also conditional (3.2.9) and mandatory (3.2.17).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

3.2.22 registry item

a metadata item recorded in a Metadata Registry

3.2.23 registry metamodel

a metamodel specifying a Metadata Registry

3.2.24 Concept

unit of knowledge created by a unique combination of characteristics

NOTE Metamodel construct is: Class.

[ISO 1087-1:2000 (3.2.1)]

3.3 Definitions of Metamodel Constructs

This subclause defines the metamodel constructs used in specifying the MMF metamodel in Clause 4, Annex C, and Annex D.

3.3.1 ModelConcept

a metaclass identifying a namespace and a named element of models to distinct an interest thing.

3.3.2 ModelDomain

a metaclass identifying a model classifier and model profiles to define the concept invoked by a sign of a ModelConcept.

3.3.3 ModelInstance

a metaclass designating values of ModelDomain to point out referents of ModelComponent referred by a concept of ModelClassifier invoked by a sign of ModelConcept.

3.3.4 ModelSelection

a metaclass designating a selection from a ModelInstance that is generally expressed with a corresponding sign of a ModelConcept.

3.3.5 ModelProfile

a metaclass designating a standard or profile composing of the model elements such as ModelUpper, ModelSpecification, ModelConstruct, and ModelComponent.

3.3.6 ModelSpecification

a metaclass designating a document of standard or profile.

3.3.7 ModelConstructs

a metaclass representing a NamedElement in UML.

3.3.8 ModelClassifier

a metaclass identifying and defining modelling unit and typed model as a concept.

NOTE A ModelClassifier may be used by a ModelDomain to designate a typed model with association link "concept".

3.3.9 UpperModel

a metaclass identifying model or metamodel at UpperLayer.

NOTE 1 A UpperModel may be defined from various ModelView.

NOTE 2 The Links among UpperModels may be connected according to associations and references.

3.3.10 ModelComponent

a metaclass designating a specialized ModelConstructs that may be assembled with a set of ModelClassifier or ModelSelection.

3.3.11 ModelAssociation

a metaclass specifying an association among ModelClassifiers.

3.3.12 ModelAssociationEnd

a metaclass designating the two ends of a ModelAssociation.

3.3.13 ModelReference

a metaclass defining a ModelClassifier's information of, and access to, links and their instances defined by a ModelAssociation.

3.3.14 Stereotype

one of ModelClassifiers, is a metaclass designating stereotyped model elements.

NOTE 1 The stereotype is defined and declared in the metamodel to extend and restrict the meaning of existing model elements.

NOTE 2 The instance of Stereotype is an object declared as a specific stereotype.

3.3.15 CodedValue

a metaclass designating coded values as a ModelConstructs.

NOTE A CodedValue can be specified against the datatype having coded values for the ModelDomain and ModelInstances.

3.3.16 Pattern

a metaclass designating a pattern mechanism.

NOTE It is a facility to define patterns as a model element and apply their patterns.

3.3.17 Comunication

a metaclass designating collaboration modelling using the pattern mechanism.

3.3.18 Component

a metaclass designating component modelling using the pattern mechanism.

3.3.19 Framework

a metaclass designating framework modelling using the pattern mechanism.

3.3.20 UpperLayer

a metaclass designating a metamodel layer pointed by a ModelLevel.

NOTE An UpperLayer has a ModelLayer according to their LevelPair.

3.3.21 UpperModelElement

a metaclass designating composite elements of a metamodel.

NOTE 1 A metamodel elements forms aggregation of ModelSelections.

NOTE 2 Each lower ModelSelection is an instance of the upper metamodel elements.

NOTE 3 A metamodel also consists of aggregation of metamodel elements.

3.3.22 LowerLayer

a metaclass designating a model level layer in the metadata architecture.

NOTE A LowerLayer has links to a ModelContext of modelling target and a ModelConcept of model elements namespace.

3.3.23 LowerModel

a metaclass identifying model or metamodel at LowerLayer

NOTE 1 A LowerModel may be defined from various ModelViews.

NOTE 2 Links among LowerModels may be connected according to associations and references.

3.3.24 LowerModelElement

a metaclass designating composite elements of a metamodel.

3.3.25 ModelView

a metaclass designating modelling viewpoints.

NOTE 1 Modelview may be classified and identified with the specific ontology.

NOTE 2 A kind of ontology should be registered with MMF Ontology Registry.

3.3.26 ModelLevel

a metaclass designating a meta level in the metadata architecture.

NOTE MetaLevel has links to a ModelContext of modelling target and a ModelConcept of model elements namespace.

3.3.27 Context

a metaclass specifying a context of modelling target.

NOTE 1 ModelContext may be classified and identified with the specific ontology.

NOTE 2 A kind of ontology should be registered with MMF Ontology Registry.

3.3.28 Category

a metaclass specifying a category of modelling target.

NOTE 1 ModelCategory may be classified and identified with the specific ontology.

NOTE 2 A kind of ontology should be registered with MMF Ontology Registry.

4 Structure of a MetaModel Framework

A MMF core metamodel is a model that describes a metamodel framework. A metamodel framework provides a mechanism for understanding the precise structure and components of the specified models, which are needed for the successful sharing of the metamodels by users and/or software facilities.

This part of ISO/IEC 19763 uses a metamodel to describe the structure of a MMF's Metadata Registry as an information model. The MMF's registry metamodel is specified as a conceptual and abstract data model, i.e. one that describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information. Any implementation model will not be mentioned in this part of ISO/IEC 19763. However, it should be also governed by the MMF's metamodel to establish the common way of managing metamodels and its derived models.

For descriptive purposes, the metamodel is organized into five functional packages (see Figure 1):

- MDR Model (see Annex A)
- MOF Model (see Annex B)
- MMF Core (see 4.6, 4.7, 4.8 and Annex C, and D)
- MMF Ontology (see ISO/IEC 19763-3)
- MMF Model Mapping (see ISO/IEC 19763-4)

The division of the model into regions is for descriptive purposes only and has no other significance.

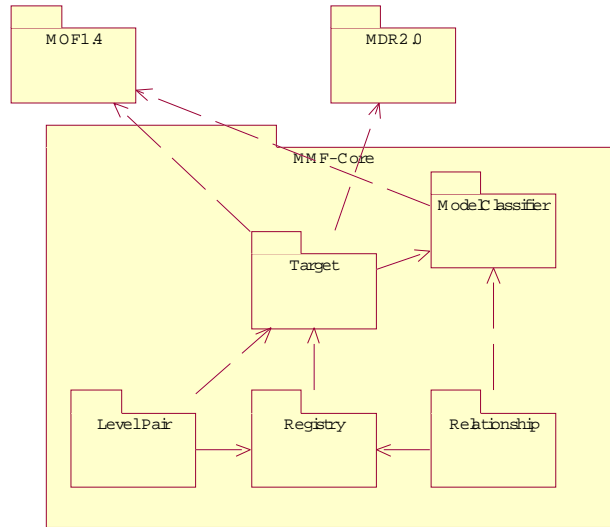


Figure 1- MMF-Core Packages

4.1 MDR and MOF

The MMF core registry metamodel is specified using Metadata Registry (MDR) and Meta Object Facility (MOF). This document uses the term "metamodel construct" for the model constructs it uses, but "metadata objects" for the model constructs it specifies in MOF. The metamodel constructs used are: classes, relationships, association classes, attributes, reference and so on. These terms are defined in 3.1, and their use is described in Annex B. The specified metadata objects are defined in 3.3, and as the main subject of this Clause.

However, there are certain parallels between the two models. For example, the "Object Class" specified in the model is equivalent to the metamodel construct "class" used to specify the model, and the "Property" specified in the model is equivalent to the metamodel construct "attribute" used to specify the model. The different terms are used to make it clear which model is being referred to, not because they represent different concepts. One term that this document uses at both levels is "datatype", but the level to which it applies should be apparent from the context in which it is used.

In this standard, the features of MOF are preserved and the following facilities are enhanced

1. Specifying relationships among MOF-based metamodels (of each standard) or between M2 level and M1level
2. Classifying and registering (based on ISO/IEC 11179-3:2003.) namespaces of each metamodel and UML profile, also patterns as a model instance
3. Defining mapping a part of metamodel at M2 level or model or data at M1 level among MOF-based metamodels

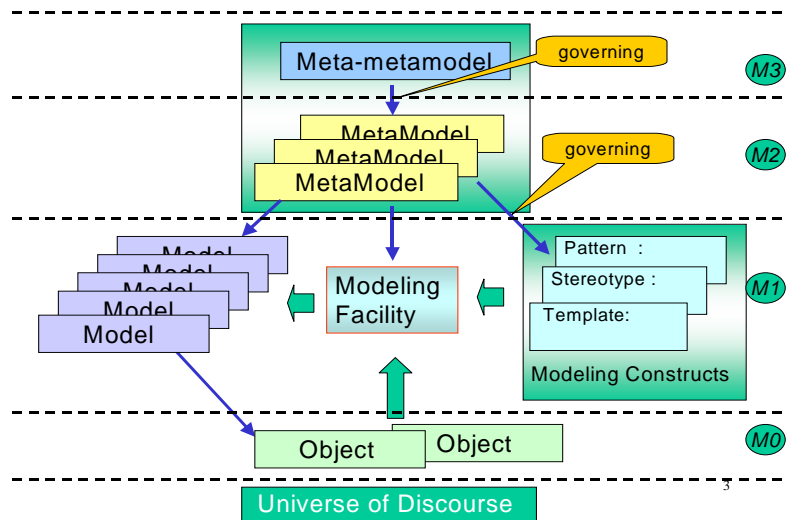


Figure 2- Overview of the Metamodel Framework

4.2 Key Concepts of MMF Core

This standard defines the relationship between metamodel and model, the framework of registering, sharing and reusing normative modeling constructs. In the layer M2 of the metamodel architecture, standards related to business objects modeling, which are developed by global standardization organizations, are registered with those namespace including defined concepts. In addition, the model instances conforming those global standard, for instance concrete stereotypes or patterns, also are registered. Users of the registry, such as modelers, pick up and use some stereotypes and patterns that are appropriate to build the own business object model in the localized standard layer. The localized standard layer has similar structure to the global standard layer, consisting of named element and namespace (ModelConcept), model profile and model classifier (ModelDomain), model component (ModelInstance) and selected model element (ModelSelection).

As shown in figure 3, the abstraction against target to be modeled in universe of discourse is formed as a sign in our mind and concepts (to be metamodeled) are evoked. The specification of those concepts is defined through matamodels from various scopes, purposes and viewpoints. Then instances of the model governed by the metamodels can be referred to as referents. The sign stands for those referents. (See “the meaning triangle” [Ogden, Richards, 1923])

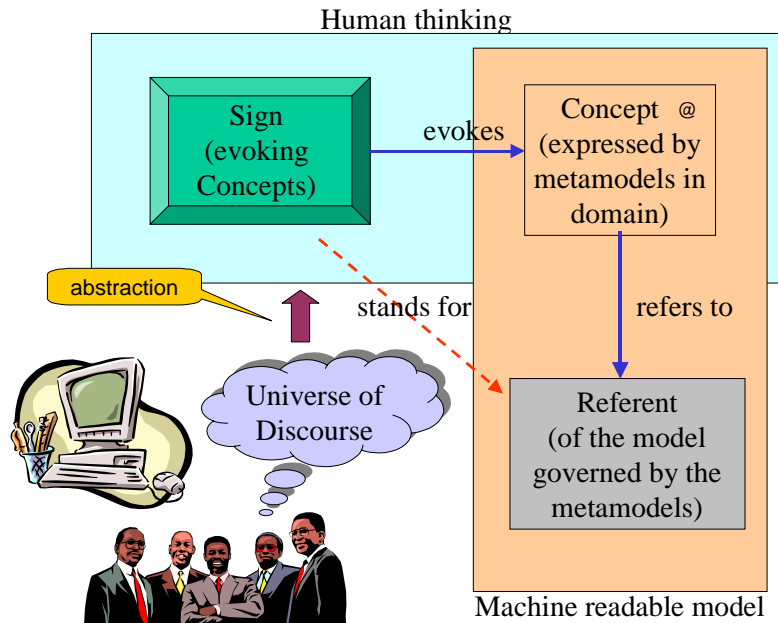


Figure 3- The Relationship between Model and Metamodel

Table 1- Key Concepts of MMF

Key Concepts	Role and Usage	Examples
ModelConcept <Sign>	Sign is a symbol designating a named element in a namespace. Namespace is a collection of signs	Stereotype name, pattern name, component name, etc.
ModelDomain <Concept>	Concept specifies meaning of sign. Concept is generally expressed with sign.	Metamodel, profile, standard etc.
ModelInstances <Referent>	Referents are a set of instance of the Concept designated by a sign.	Pattern, stereotyped element, etc.
ModelSelection	Concrete Model Element selected from specific Model Instance	Choice from various model elements

Various models, for each modeled target and business domain, would be developed. In those cases, it is an important issue which stereotypes or patterns should be used as modeling constructs. The specification of those modeling constructs may be provided with metamodels.

For example, names of business processes and those metamodels are registered in “model concept” and “model domain” of the M2 layer respectively. The business process models defined using normative modeling constructs at the M2 layer are registered in “referent” of the other M1 layer. In “referents” of the M1 layer, concrete models or products, satisfying the specification of those metamodels, are registered by each developer or vendor that develops and ships them.

Also, the relationship among model classifiers needs to be described definitely. In this standard, the facility to define reference of any metamodel over different standards or mapping for data exchange should be provided. Moreover, in the case of registering the model instance, it is necessary to describe the conformance level such as how much and what parts of metamodels each model instance accords with or not. This standard provides the framework for defining those things.

Table2 shows that there are different registers and users against the registry conforming this standard. Supposed organizations and users who register metamodels or patterns for the M2 layer and models or products for the M1 layer respectively, are listed.

In this standard, concerning metamodel for the M2 layer and model for the M1 layer, as described above, the purpose is specifying the framework for classifying namespace and defining the relationship and meaning among metametamodel elements or metamodel layers. The M3 layer of MOF might be enhanced to provide the facility for treating those concepts.

Table 2- Typical Developer/Register and User

Setting Layer	Metamodel Elements	Developer/Register	User
Global Standard	ModelConcept	Standardization organization	Standard maker and developer of patterns and stereotypes
	ModelDomain (M2)	Standardization organization	Standard maker and developer of patterns and stereotypes
	ModelInstance (M1)	Standardization organization and developer of patterns and stereotypes	Industry standard maker
	ModelSelection	Organization for developing industry standard model	Developer of products conforming standard model
Localized Standard	ModelConcept	Organization for developing industry standard model	Developer of products conforming standard model
	ModelDomain (M2)	Organization for developing industry standard model	Developer of products conforming standard model
	ModelInstance (M1)	Enterprise of developing products conforming standard model.	Vendor of products
	ModelSelection	(not public. Should be managed by within each enterprise)	(within each enterprise)

4.3 Registered Target Structure

In ModelConcept, a namespace and a sign are registered. In ModelDomain, the ModelProfile and the ModelClassifier corresponding to the sign are registered. In the ModelProfile, UpperModels ModelSpecifications and ModelComponents that specify the meaning of the ModelClassifier are included. In the ModelInstance, ModelComponents that are governed by corresponding the ModelClassifier on the ModelDomain should be registered.

There are three patterns of “governing” as mentioned in 4.4.

Figure 4 shows the basic framework of registered target objects.

Actually, a ModelProfile registered in a ModelDomain may include a set of ModelConcepts defined within the ModelProfile. If necessary, corresponding ModelClassifiers and ModelInstances for each sign may be registered.

Figure 5 shows the like actual example. In a ModelComponent, which is pointed by corresponding the ModelDomain and ModelInstances, may include ModelClassifiers and ModelSelections

ModelSelection has a role to select some appropriate values from the other ModelConcept and the ModelInstances. It suggests that alternative value can be used to composite own models if possible. Using ModelSlection, the multi-component structure can be expressed as a registered target object.

Figure 6 shows an example of registered target objects connected with ModelSelections.

LowerModel, which is governed by UpperModel, may be registered itself as another ModelConcept (sign) in UpperModel layer.

Figure 7 shows an example of layered structure such as LowerModel becomes UpperModel.

In summary, registered target objects may be built with dependency between each other and more complex structure can be expressed as registered target objects.

The structure is recognized as from the following three viewpoints

(1) Multi-forest structure

Various families of standard will be developed in different organization independently. In registry, related standard should be linked meaningfully beyond relationship of among standard development organizations. Such registered target objects must form forest structures of related standard models. If necessary, harmonization activities should be done and develop some profiles to achieve the interoperability among those different standards.

(2) Multi-layer structure

In the forest of a group of standard, terminology and concept should be controlled and maintained systematically. Also, the relationship of such as upper and lower should be clear. In registry, upper may be registered as ModelClassifier and lower may be done as ModelComponents. Such relationship forms multi-layer structure of target objects.

(3) Multi-component structure

ModelComponent can include another component as external reference objects. Such relationship forms multi-component structure. Also, a ModelClassifier may consist of elements including external objects that are selected from in another registered Model Instances.

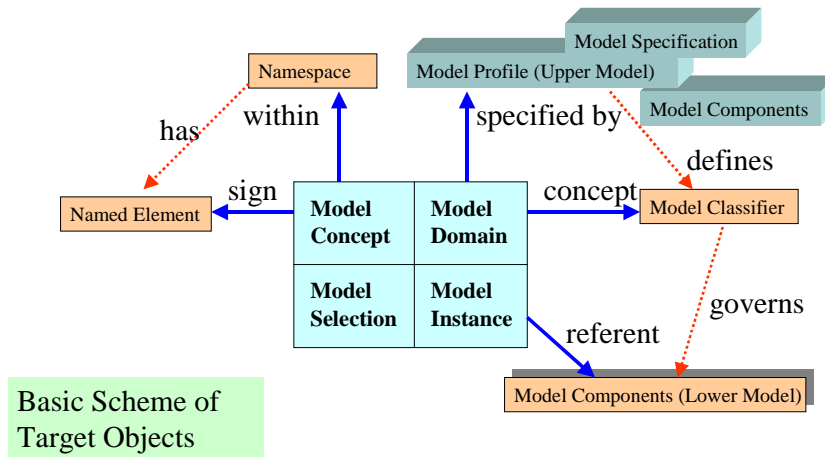


Figure 4- Basic Relationship of Registered Target Objects

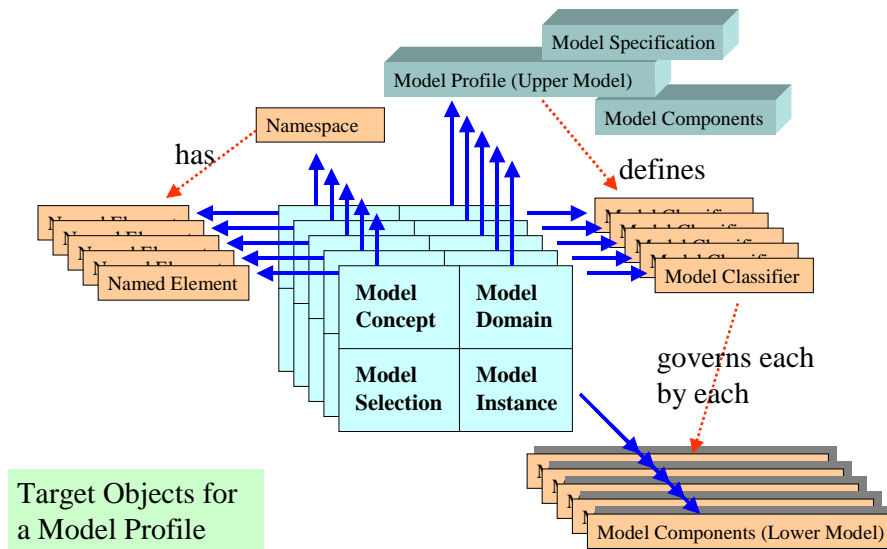
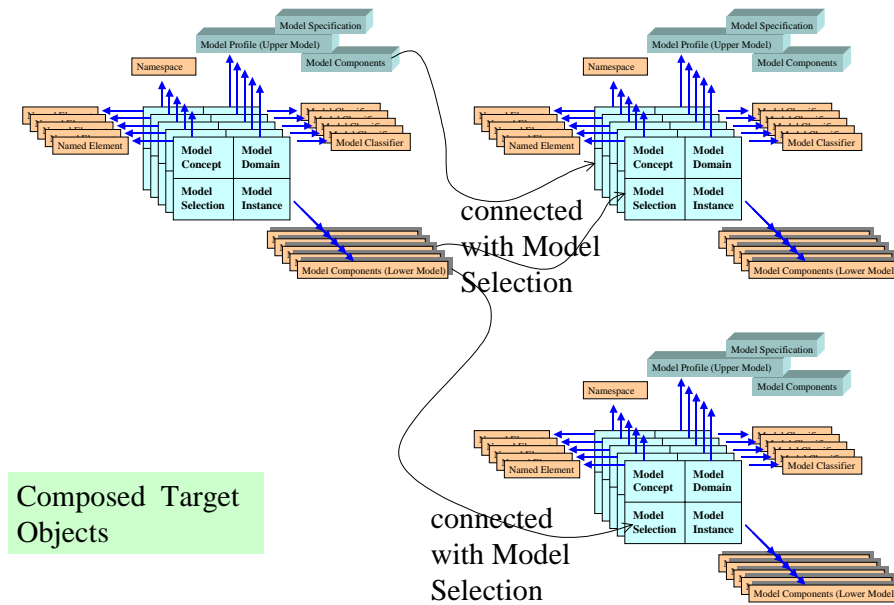
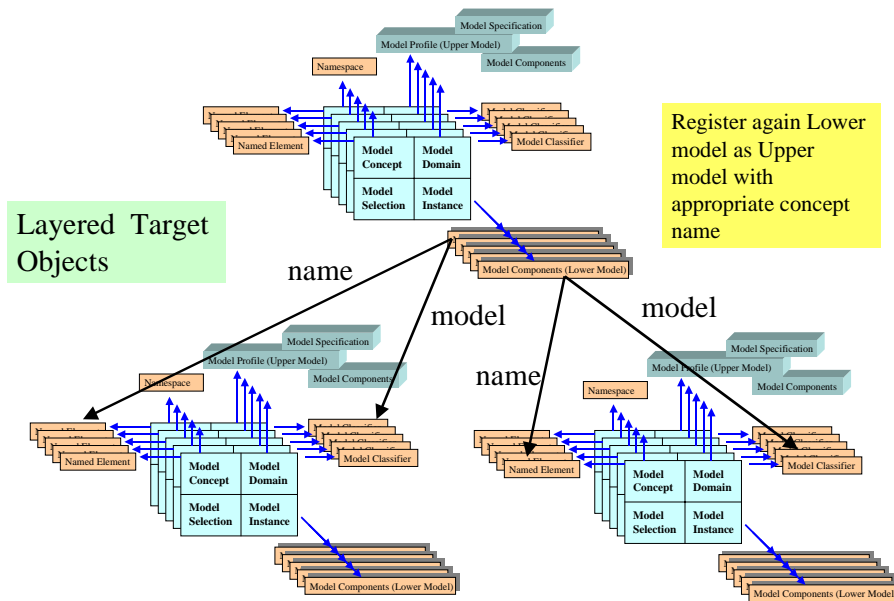


Figure 5- Registered Target Objects for a Model Profile



Composed Target Objects

Figure 6- Registered Target connected with Model Selection



Layered Target Objects

Figure 7- Layered Target Objects with multi-layered registration

4.4 The Meaning of Governing

Upper and Lower		Description
Metamodel governs Model		
Metamodel (upper)	Define the abstract syntax for expressing the lower governing model Described using MOF, the other notation may be allowed as informative	
Model (lower)	Should be expressed according to concrete syntax governed by upper metamodel Described using UML, the other notation may be allowed as informative	
Model governs Model		
Model (upper)	Has role as a source model for derived lower models Described using UML, the other notation may be allowed as informative	
Model (lower)	Should be derived from based on governed upper model Described using UML, the other notation may be allowed as informative	
Model governs Element		
Model (upper)	Has role as a source model for corresponding value sets Described using UML, the other notation may be allowed as informative	
Element (lower)	Should be enumerated for upper model governing values Described using UML, the other notation may be allowed as informative	

4.5 Convention for Definition of MMF Core

The metamodel of MMF core is depicted with Class diagram and described using the following form.

<ClassName>	Super class	<immediate inherited classes>
	<purpose and role of defined Class>	
Attributes or Reference	Datatype and Cardinality	Description
<attributes and reference including derived ones>	<datatype and occurrences>	<content and purpose>

Constraints

<constrains if necessary, written by natural language or OCL>

The model shows constraints cardinality on minimum and maximum occurrences of attributes. The constraints on maximum occurrences are to be enforced at all times. The constraints on minimum occurrences are to be enforced when the registration status for the metadata item is "recorded" or higher. In other words, a registration status of "recorded" indicates that all mandatory attributes have been documented.

For descriptive purposes, the MMF core metamodel is organized into three functional packages:

- MMF Core Model (Framework of Registry; see 4.6):
- MMF Core Model (Structure of Registered Target; see 4.7)
- MMF Core Model (Relationship of Registered Target; see 4.8)

Descriptions of specific types of Administered Item:

- ModelConcept (see 4.6.1)
- ModelDomain (see 4.6.2)
- ModelInstance (see 4.6.3)
- ModelSelection (see 4.6.4)
- ModelProfile (see 4.7.1)
- ModelComponent (see 4.7.6)

The division of the model into regions is for descriptive purposes only and has no other significance.

NOTE If any discrepancy exists in Clause 4 between the figures and the text, the text shall take precedence.

4.6 MMF Core Model (Framework of Registry)

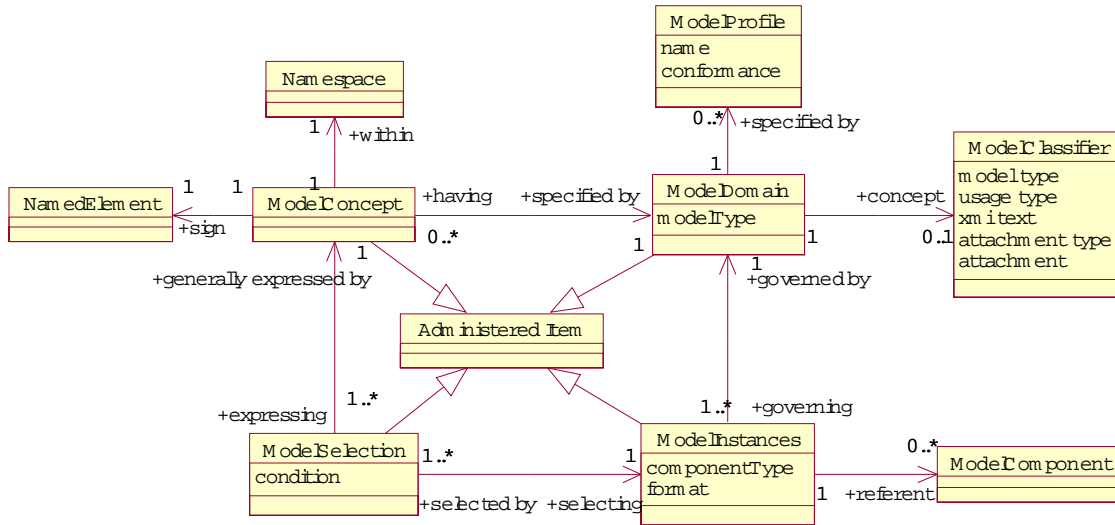


Figure 8- MMF Core (Framework of Registry)

4.6.1 ModelConcept

ModelConcept	Super Class	Administered Item (from MDR)
	<p>ModelConcept is a metaclass identifying a namespace and a named element of models to distinct an interest thing.</p> <p>ModelConcept plays a role of container containing ModelConcept instances.</p> <p>A ModelConcept may be managed with a MDR Administered Item.</p> <p>The meaning of the sign of the ModelConcept is defined with the specification of a ModelDomain.</p> <p>The NamedElement and the Namespece is defined in MOF.</p> <p>multi-lingual measure</p> <p>A ModelConcept including its administrated item and item name should be registered with multi-lingual measure. That is, a model concept item could be designated with the specific registered language.</p>	
Attributes or Reference	Datatype and Cardinality	Description
within	string	Provides a string of namespace where sign is uniquely defined.
sign	string	Identifies a string of named element invoking a concept defined in a Model Domain
specified by	ModelDomain [1..1]	Provides an OID to a ModelDomain.
<p>Constraints</p> <p>An instance of the ModelConcept has a sign of the named element that should be specified in the namespace.</p>		

4.6.2 ModelDomain

ModelDomain	Super Class	Administered Item (from MDR)
	<p>ModelDomain is a metaclass identifying a model classifier and model profiles to define the concept invoked by a sign of a ModelConcept.</p> <p>ModelDomain is an abstraction of ModelClassifier specified in the designating ModelProfiles. A ModelClassifier plays a role of typed model and ModelProfiles play a role of specifying a particular Domain knowledge.</p> <p>A ModelDomain may be managed with a MDR Administered Item.</p> <p>The meaning of the sign of a ModelConcept is defined with the specification of a ModelDomain.</p> <p>multi-lingual measure</p> <p>A ModelDomain including its administrated item should be registered with the multi-lingual measure. That is a model domain name could be designated with the specific registered language.</p>	
Attributes or Reference	Datatype and Cardinality	Description
specified by	ModelProfile [0..*]	Provides OIDs to ModelProfiles
concept	ModelClassifier [0..1]	Identifies an OID to ModelClassifier
model type	ModelConstruct Code	Identifies a code of kind of ModelClassifier.
Constraints		
A ModelClassifier should be specified in particular ModelProfiles.		

4.6.3 ModelInstances

ModelInstances	Super Class	Administered Item (from MDR)
	<p>ModelInstance is a metaclass designating values of ModelDomain to point out referents of ModelComponent referred by a concept of ModelClassifier invoked by a sign of ModelConcept.</p> <p>A ModelInstances plays a role of container of registered ModelComponents. The ModelInstance has an association “governed by” with the ModelDomain.</p> <p>A ModelInstances may be managed with a MDR Administered Item.</p> <p>The meaning of the sign of a ModelConcept is defined with the specification of a ModelDomain.</p> <p>multi-lingual measure</p> <p>A ModelInstances including administrated item should be registered with the multi-lingual measure. That is a model instance name could be designated with the specific registered language.</p>	
Attributes or Reference	Datatype and Cardinality	Description
governed by	ModelDomain [0..1]	Provides a OID to a ModelDomain
referents	ModelComponent [0..*]	Provides OIDs to a ModelComponent
component type	type code	Declares a code of kind of ModelClassifier.
format	string	Identifies a file format allowed as a ModelComponent
Constraints		
<p>The ModelInstances should be a set of ModelComponent derived from its ModelClassifier and governed by its ModelDomain.</p> <p>The model elements such as pattern (template), stereotype (tagged value), coded value and constraint could be derived from upper model (metamodel).</p>		

4.6.4 ModelSelection

ModelSelection	Supper Class	Administered Item (from MDR)
	<p>ModelSelection is a metaclass designating a selection from a ModelInstance that is generally expressed with a corresponding sign of a ModelConcept.</p> <p>A ModelSelection plays a role of connecting a ModelConcept and a ModelInstances. A ModelSelection may be used in ModelComponent in order to assemble ModelConstructs.</p> <p>A ModelInstances may be managed with a MDR Administered Item.</p> <p>A ModelSelection may have a condition concerning range of selected Model Instances.</p> <p>multi-lingual measure</p> <p>A ModelSelection including administration item should be registered with multi-lingual measure. That is a model selection name could be designated with the specific registered language.</p>	
Attributes or Reference	Datatype and Cardinality	Description
generally expressed by	ModelConcept [1..1]	Points a sign in a Model Concept.
selecting	ModelInstances [0..1]	Points a Model Instances from which interested ones will be chosen
condition	string	Specifies a filtering for Model instances
Constraints		

4.7 MMF Core Model (Structure of Registered Target)

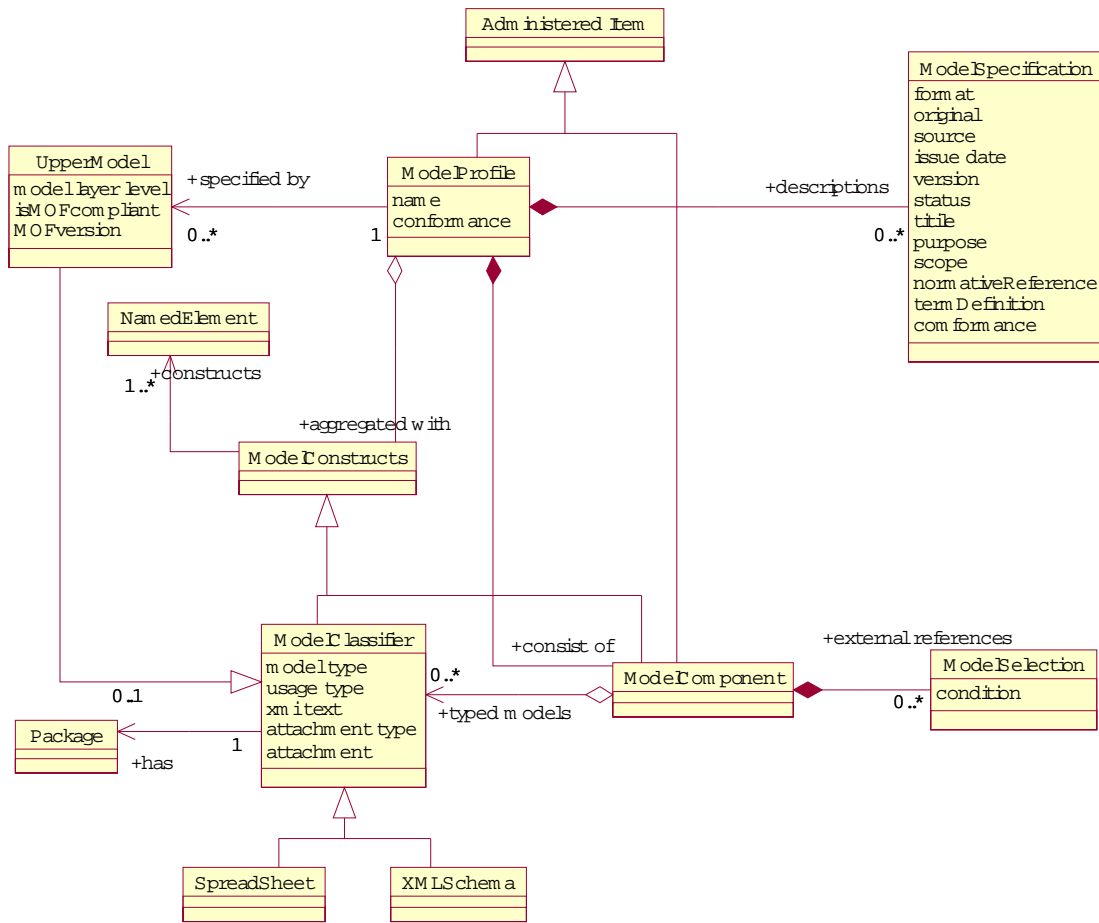


Figure 9- MMF Core (Structure of Registered Target)

4.7.1 ModelProfile

ModelProfile	Supper Class	Administered Item (from MDR)
	<p>ModelProfile is a metaclass designating a standard or profile composing of the model elements such as UpperModel, ModelSpecification, ModelConstruct, and ModelComponent.</p> <p>A ModelProfile plays a role of holding artifacts concerning a standard or its related profile.</p> <p>UpperModel may include a machine-readable model (or metamodel).</p> <p>ModelSpecifcatoin may have human-readable documents and materials.</p> <p>ModelConstruct may have namded model elements such as stereotypes, tagged values, patterns and so on.</p> <p>ModelComponent may be assembled with ModelSelections and defined as a ModelClassifier.</p> <p>A ModelProfile may be managed with a MDR Administered Item.</p>	
Attributes or Reference	Datatype and Cardinality	Description
name	string	Provides an identifier of the model profile
descriptions	ModelSpecification [1..*]	Provides specification of the standard or profile.
comformance	string	Declares the level of comformance defined in ModelSpecification
specified by	UpperModel [0..*]	Provides the model or metamodel
aggregated with	ModelConstructs [0..*]	Declares model constructs such as stereotype, pattern, framework and so on.
consist of	ModelComponent [0..*]	Declares the external reference elements and ModelClassifiers.
Constraints		
A ModelClassifier may be corresponding to a particular sign in ModelConcept.		

4.7.2 ModelSpecification

ModelSpecification	Supper Class	None
	<p>ModelSpecification is a metaclass designating a document of standard or profile.</p> <p>A ModelSpecificaton may play a role of holding human-readable documents concerning a standard or its related profile.</p> <p>A ModelSpecification may include UpperModel, ModelClassifier and Model Constructs as a document if necessary.</p>	
Attributes or Reference	Datatype and Cardinality	Description
format	string	Provides a document file type. For example, ppt, pdf, http.
original	file	Declares the original material.
source	string	Provides a name of standard development organization (SDO).
issue date	date	Identifies a published date
version	number	Identifies a version number of the standard or profile
status	string	Provides the stage in standardization process
title	string	Provides a title of the standard or profile.
purpose	string	Describes objectives of the document
scope	string	Describes applied area and domain
normativeReference	string [0..*]	Declares normative references
termDefinitions	string [1..*]	Include the terminology definitions
conformance	string	Declares conformance level conditions
Constraints		

4.7.3 ModelConstructs

ModelConstructs	Supper Class	None
<p>ModelConstructs is a metaclass representing a NamedElement in UML.</p> <p>ModelConstructs plays a role of a composing of a NamedElement.</p> <p>A ModelConstructs may have named elements such as pattern (template), stereotype (tagged value), coded value and constraint.</p> <p>ModelConstructs should be standardized as common parts for modelling if necessary.</p> <p>MOF relationship</p> <p>ModelConstructs has an association with NamedElemens from MOF.</p>		
Attributes or Reference	Datatype and Cardinality	Description
constructs	NamedElement [0..*]	Named Element in MOF
Constraints		

4.7.4 ModelClassifier

ModelClassifier	Supper Class	ModelConstructs
	<p>ModelClassifier is a metaclass identifying and defining modelling unit and typed model as a concept.</p> <p>A ModelClassifier may be used by a ModelDomain to designate a typed model with association link “concept”.</p> <p>A ModelProfile may have a set of ModelClassifiers as ModelConstruct.</p> <p>A ModelClassifier may have some ModelAssociations and ModelReferences.</p> <p>MOF relationship</p> <p>ModelClassifier has an association with Package from MOF. A Package is a container for a collection of related ModelElements that form a logical meta-model.</p>	
Attributes or Reference	Datatype and Cardinality	Description
model type	type code	Specifies a type of ModelClassifier.
usage type	type code	Provides a purpose of usage.
has	Package[0..1]	Provides a MOF compliant model.
xmi text	String[0..1]	Provides a XML schema of XMI format
attachment type	string[0..1]	Provides a file type of attachment format. For example, ppt, pdf, http, and xls.
attachment	File or URI	Provides the content with various expressions
<p>Constraints</p> <p>A ModelClassifier is composed of a package that may be also a composition of ModelConstructs. One unit of model consists of one package.</p> <p>Packages are also uses as organizational constructs in modelling. Nesting, importation, and generalization are used to manage the complexity of models.</p>		

4.7.5 UpperModel

UpperModel	Supper Class	ModelClassifier
	<p>UpperModel is a metaclass identifying model or metamodel at UpperLayrer.</p> <p>A UpperModel may be defined from various ModelView.</p> <p>The Links among UpperModels may be connected according to associations and references.</p> <p>MOF relationship</p> <p>UpperModle is corresponding to a metamodel at the meta level layer in the MOF metadata architecture. MOF has the facility to handle the reflective operations. In MMF, a metamodel is expanded into a set of meta objects, and the operations at the lower model layer are allowed to handle the information about the upper metamodel.</p>	
Attributes or Reference	Datatype and Cardinality	Description
model layer level	string	Provides the layer level of model classifier such as M1 or M2.
isMOFcompliant	boolean	Declares whether the metamodel is MOF compliant or not.
MOFversion	string	Provides the version number about MOF.
<p>Constraints</p> <p>A UpperModel governs corresponding LowerModel.</p> <p>LowerModel should be described with conforming to abstract syntax on upper metamodel under the constraints.</p>		

4.7.6 ModelComponent

ModelComponent	Supper Class	ModelConstructs and Administered Item (from MDR)
	<p>ModelComponent is a metaclass designating a specialized ModelConstructs that may be assembled with a set of ModelClassifier or ModelSelection.</p> <p>A ModelComponent may play a role of ModelClassifier as a typed model.</p> <p>A ModelComponent may have ModelSelections that plays a role of connecting external elements with like “plug and play” manner.</p>	
Attributes or Reference	Datatype and Cardinality	Description
typed models	ModelClassifier [1..*]	Specifies a model type or model types.
external references	ModelSelection [0..*]	Specifies registered ModelComonents through ModelSelection
<p>Constraints</p> <p>ModelSelection should be used limited to appropriate for the model type of a ModelComponent in the ModelInstances that is pointed by the ModelSelection.</p>		

4.8 MMF Core Model (Relationship of Registered Target)

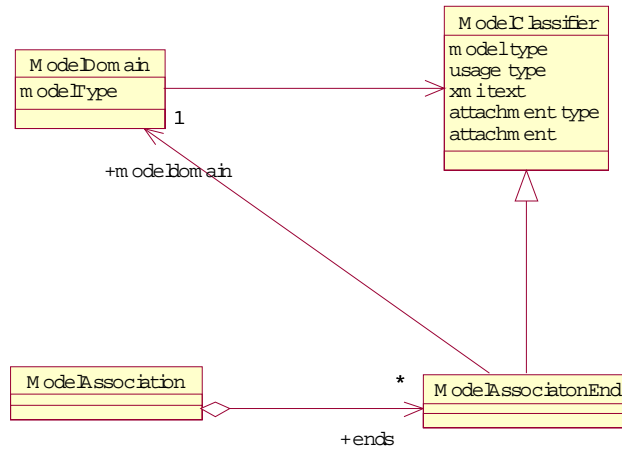


Figure 10- MMF Core (Relationship of Registered Target)

4.8.1 ModelAssociation

ModelAssociation	Supper Class	ModelElement (from MOF)
	<p>ModelAssociation is a metaclass specifying an association among ModelClassifiers.</p> <p>A ModelAssociation defines a classification for a set of links on ModelClassifier. A link, which is an instance of ModelAssociation, is a connection between object instances of a ModelClassifier.</p> <p>MOF relationship</p> <p>In MOF, Association among Classifiers (which do not include “Package”) can be provided. AssociationEnd and Reference can be described as a relationship among those Classifiers. A link to two participant objects is limited in only binary relationship. In 19763, “ModelAssociation”, similar to the “Association” in MOF, is an extension for ModelClassifier.</p>	
Attributes or Reference	Datatype and cardinality	Description
ends	ModelAssociation End[2..*]	Provides a link of ModelClassifiers.
<p>Constraints</p> <p>The definition of ModelAssociation needs to specify two ModelAssociationEnds.</p> <p>If a ModelAssociation is directional, the name such as suggesting its direction should be used.</p> <p>Even if there are no direct ModelAssociation, indirect one derived from given metamodels may exist.</p> <p>The mutual reference among ModelClassifiers may be represented with ModelReference.</p>		

4.8.2 ModelAssociationEnd

ModelAssociationEnd	Supper Class	ModelClassifier
	<p>ModelAssociationEnd is a metaclass designating two end of a ModelAssociation.</p> <p>Each ModelAssociationEnd defines a role of a ModelClassifier participant in the ModelAssociation and constraints on sets of the ModelClassifier.</p> <p>MOF relationship</p> <p>In MOF, Association among Classifiers (which do not include “Package”) can be provided. AssociationEnd and Reference can be described as a relationship among those Classifiers. In MMF, “ModelAssociationEnd”, similar to the “AssociationEnd” in MOF, is an extension for ModelClassifier.</p>	
Attributes or Reference	Datatype and cardinality	Description
modeldomain	ModelDomain [1..1]	Declares ModelDomain in which it is defined.
<p>Constraints</p> <p>An instance of a ModelAssociationEnd is a LinkEnd, which defines a relationship between a link, in instance of a ModelAssociation, and an instance of the ModelAssociationEnd's ModelClassifier, provided in its type attribute.</p> <p>An instance of a ModelAssociationEnd provides the information about ModelReference related to the referenced ModelClassifier instance.</p>		

4.8.3 ModelReference

ModelReference	Supper Class	Reference (from MOF).
<p>ModelReference is a metaclass defining a ModelClassifier's information of, and access to, links and their instances defined by a ModelAssociation.</p> <p>Although a ModelReference derives much of its state from a corresponding ModelAssociationEnd, it provides additional information.</p> <p>A ModelReference is defined against a ModelClassifier.</p> <p>An instance of a ModelReference may hold one or more links toward instances of corresponding ModelInstances.</p> <p>A ModelReference has a roll defined by ModelAssociationEnd.</p> <p>MOF relationship</p> <p>In MOF, Association among Classifiers (which do not include "Package") can be provided. AssociationEnd and Reference can be described as a relationship among those Classifiers. In MMF, "ModelReference", similar to the "Reference" in MOF, is an extension for ModelClassifier.</p>		
Attributes or Reference	Datatype and cardinality	Description
name	String	Item name with multi-lingual measure
Constraints		

5 Conformance

This part of ISO/IEC CD 19673 prescribes a conceptual metamodel, not a physical implementation. Therefore, the metamodel need not be physically implemented exactly as specified. However, it must be possible to unambiguously map between the implementation and the metamodel in both directions.

Conformance may be claimed to either the conceptual model; see 5.2. Conformance claims shall specify a Degree and a Level of Conformance, as described below.

5.1 Degree of Conformance

<TBD>

5.2 Levels of Conformance

<TBD>

5.3 Obligation

<TBD>

5.4 Implementation Conformance Statement (ICS)

<TBD>

5.5 Roles and Responsibilities for Registration

<TBD>

Annex A: MDR Model (Informative)

The metamodel for a metadata registry is specified in ISO/IEC 11179-3 Metadata Registries (MDR). Data modeling is founded on the theory that all data describes properties (attributes) of objects in the natural world, namely Universe of Discourse (UoD). Data represents the properties of these things. The basic units of data are data elements. This metamodel uses many of the same conceptual data structures used in data modeling.

The more important key components of the metadata registry are as follows.

Data Element Concept: An idea that can be represented in the form of a data element, described independently of any particular representation.

Conceptual Domain: A set of possible value meanings of a data element expressed without representation.

Value Domain: A set of permissible values.

Data Element: A unit of data for which the definition, identification, representation and Permissible Values are specified by means of a set of attributes

The specification in ISO/IEC 11179-3 Metadata Registries (MDR) doesn't expect that the metamodel will completely accommodate all users. Particular sectors, such as registering metamodel and model, require metadata attributes not addressed in the standard. Such extensions shall be considered conformant if they do not violate any of the rules inherent in the structure and content as specified by the metamodel in the standard. Classes, relationships, and attributes may be added to this conceptual data model. Implementers of the standard may include extensions as part of an implementation, and/or they may provide facilities to allow a registry user to define their own extensions.

This standard is developed based on ISO/IEC 11179-3. However, in this ISO/IEC 19763 registered targets are extended to register complex models and objects. Then, the notion of four key components has been redefined as ModelConcept, ModelDomain, ModelInstance and ModelSelection described in 4.2.

About administration process and procedure, this standard is conforming to the ISO/IEC 11179 family. Target objects have an Administered Item specified in ISO/IEC 11179-3 shown in Figure 11.

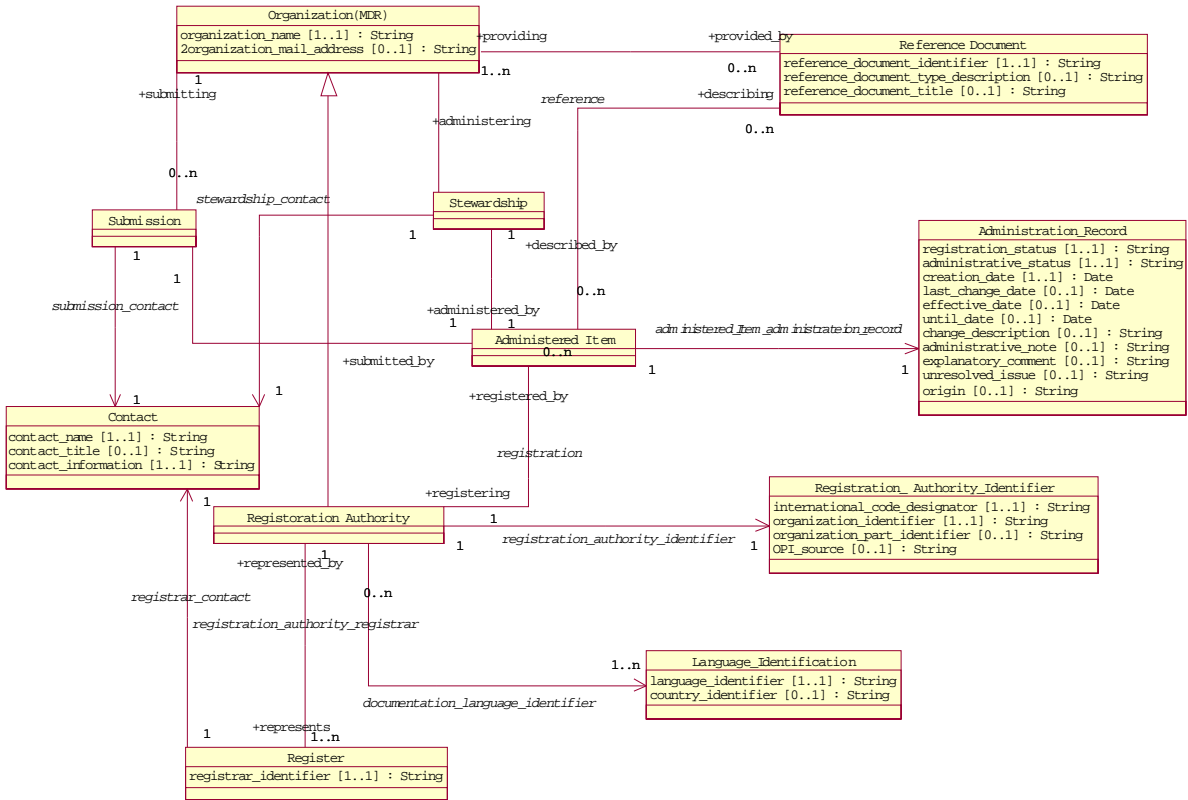


Figure 11- MDR Model (Administered Item)

Annex B: MOF Model (Informative)

The MOF is an adopted technology by OMG for defining metadata and representing it as CORBA objects. Metadata is a data describing data or information, and can accordingly be described by other metadata. In MOF terminology, metadata that describes metadata is called meta-metadata, and a model that consists of meta-metadata is called a metamodel.

One kind of metamodel plays a central role in the MOF. A MOF metamodel defines the abstract syntax of the metadata in the MOF representation of a model. Since there are many kinds of metadata in a typical system, the MOF framework needs to support many different MOF metamodels. The MOF integrates these metamodels by defining a common abstract syntax for defining metamodels. This abstract syntax is allied the MOF Model and is a model for metamodels; i.e. a meta-metamodel. The MOF metadata framework is typically depicted as a four-layer architecture (M0, M1, M2, M3). There is the following features of MOF-based metamodel and UML profile;

1. Using a restricted subset of the UML notation
2. Providing common style of describing metamodel
3. A metamodel at M2 level is as an abstract syntax of models at M1 level
4. A model at M1 level is an expression of the metamodel
5. UML profile (stereotype, tagged values etc.) is as additional constraints for metamodel
6. Mapping between the metamodel and the UML profile is needed as a basis for the development of tools
7. Enable exchanging metamodel/model/data between tools

Figure 12 shows the overview of MOF Model Package. The operations of the metaclasses are not shown in this diagram. In this standard, Package, ModelElement, Reference and Namespace are used as external reference elements. NamedElement inherited from Namespace is defined as an external reference element.

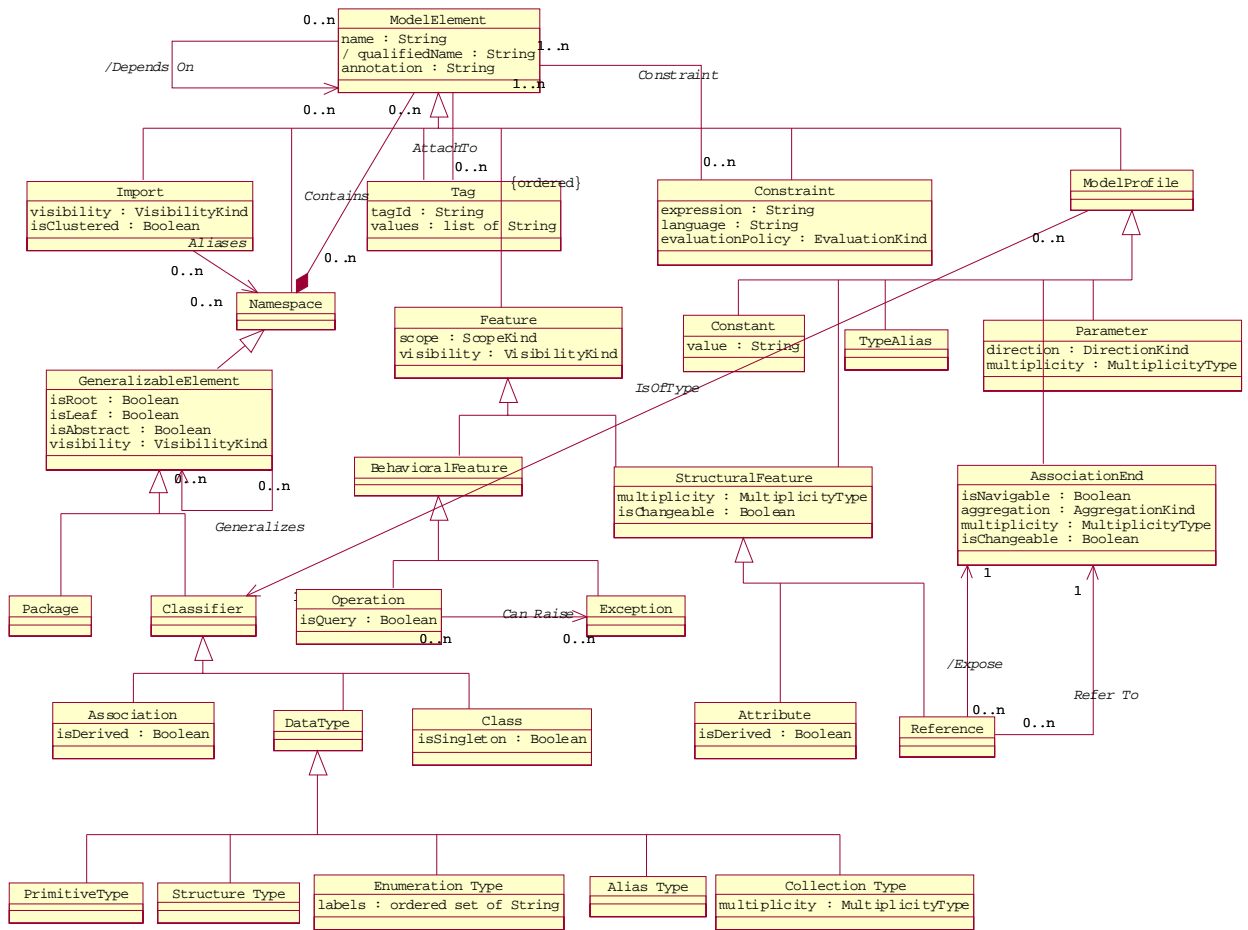


Figure 12- MOF Model Package

NamedElement	Supper Class	Namespace (from MOF)
	NamedElement is a metaclass corresponding to Namespace in MOF.	
Attributes or Reference	Datatype and Cardinality	Description
name[MOF]	string	Provides a meta-modeler supplied name that uniquely identifies the ModelElement in the context of the ModelElement's containing Namespace.
qualifiedName[MOF]	String [0..1]	Provides a unique name for the ModelElement within the context of its outermost containing Package.
annotation[MOF]	String [0..1]	Provides an informal description of the ModelElement.
container[MOF]	Namespace [0..1]	Identifies the Namespace that contains the ModelElement.
requiredElement[MOF]	provider [0..*]	Identifies the ModelElements on whose definition the definition of this ModelElement depends.
constraints[MOF]	constrainedElements[0..*]	Identifies the set of Constraints that apply to the ModelElement. A Constraint applies to all instances of the ModelElement and its sub-Classes.
contains[MOF]	container [0..*]	A meta-model is defined through a composition of ModelElements.
<p>Constraints</p> <p>When choosing a ModelElement's name, the meta-modeler should consider the rules for translating names into identifiers in the relevant mappings. To minimize portability problems, use names that start with an ASCII letter, and consist of ASCII letters and digits, space and underscore. Avoid names where capitalization, spaces, or underscores are significant.</p> <p>The qualifiedName is a list of String values consisting of the names of the ModelElement, its container, its container's container and so on until a non-contained element is reached. The first member of the list is the name of the non-contained element.</p> <p>Since the Contains Association is a Composite Association, any ModelElement can have at most one container, and the containment graph is strictly tree shaped.</p> <p>A Namespace defines a ModelElement that composes other ModelElements. Since Namespace has several subclasses, there is a sizable combinatorial set of potential Namespace-ModelElement pairings.</p>		

Annex C: ModelClassifier (Informative)

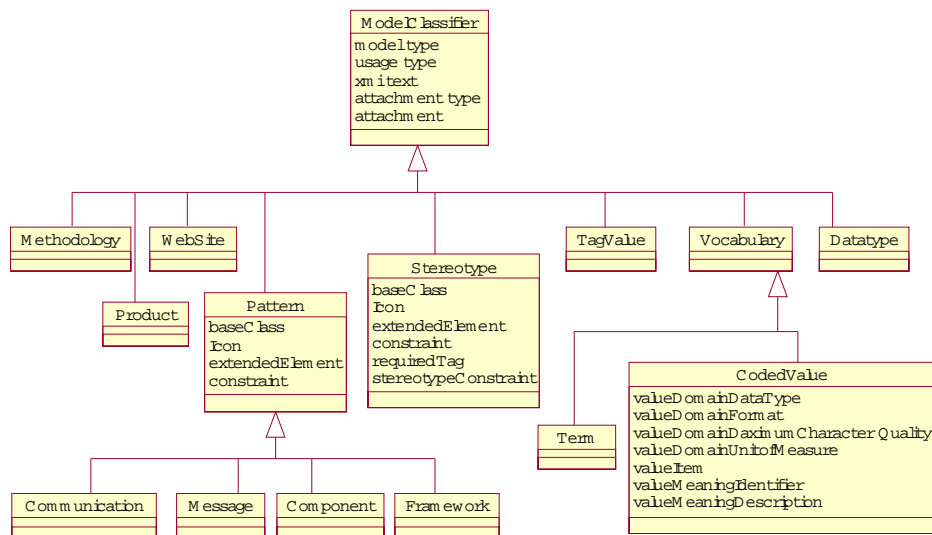


Figure 13- Category of Registered Target

C.1 Stereotype

Stereotype	Supper Class	ModelClassifier
	<p>Stereotype, which is one of ModelClassifiers, is a metaclass designating stereotyped model elements.</p> <p>The stereotype is defined and declared in the metamodel to extend and restrict the meaning of existing model elements.</p> <p>The instance of Stereotype is an object declared as a specific stereotype.</p> <p>MOF relationship</p> <p>In MOF, there is no metaclass corresponding “Stereotype”. On one hand, UML metamodel has a specification concerning “Stereotype” as an extension mechanism. The stereotype is one of three extension mechanisms in UML. The meaning of each stereotype is specified with the profiles and metamodels. It performs like an instance of virtual metamodel constructs. An instance has the same structure (e.g. attribute, association, and operation) as the non-stereotyped model element. The stereotype can specify additional constraints and required tagged value to be applied for instances. The stereotype may be also used for indicating the difference of meaning and usage to two same structured model elements.</p>	
Attributes or Reference	Datatype and Cardinality	Description
baseClass[UML]	Class Name	
Icon[UML]	Geometry [0..1]	
extendedElement[UML]	ModelElementName [0..*]	
constraint[UML]	String[0..1]	
requiredTag[UML]	String [0..*]	
stereotypeConstraint[UML]	String [0..1]	
Constraints		

C.2 CodedValue

CodedValue	Supper Class	ModelClassifier
	<p>The CodedValue is a metaclass designating coded values as a ModelConstructs. A CodedValue can be specified against the datatype having coded values for the ModelDomain and ModelInstances.</p> <p>MOF relationship</p> <p>In MOF, there is no metaclass corresponding “CodedValue”. The metamodel “CodedValue” inherits the mataclass “Classifier” from MOF via ModelConstructs. Coded values may be specified for an enumerated datatype of model elements</p> <p>MDR relationship</p> <p>In MDR, each value and its meaning for enumerated datatype may be defined and registered as a ValueDomain. A correspondence between VauleDomain and CodedValue may be specified with a link of association “referents” based on ModelDomain and their coded values and meanings may be registered in the MDR framework including permissible values and value meanings.</p>	
Attributes or Reference	Datatype and cardinality	Description
valueDomainDataT ype	datatype	
valueDomainFormat	String [0..*]	
valueDomainDaxim umCharacter Quality	integer [0..1]	
valueDomainUnitof Measure	Unit of measure [0..1]	
valueItem	String	
valueMeaningIdentif ier	String	
valueMeaningDescr iption	String [0..1]	
Constraints		

C.3 Pattern

Pattern	Supper Class	ModelClassifier
	<p>Pattern is a metaclass designating a pattern mechanism. It is a facility to define patterns as a model element and apply thier patterns.</p> <p>The pattern is a kind of model construct elements that is a reusable piece of model and generally applicable to the similar model. It can be treated as a type (or template). The metameta model elements such Collaboration, Component, Framework are subclasses of the pattern.</p> <p>A pattern is defined with a package and parameterised collaboration diagram. In the model elements appeared on the pattern definition, class names, attributes, association names and associatonEnds and operation names can be specified as a formal parameter. The nested pattern definition should be allowed. The applying the pattern with actual parameters performs to get newly created collaboration diagram unfolded from the pattern. In unfolding, the function of renaming and filtering, which modify and hidden the names, may be specified if necessary.</p> <p>The expression derived from the abstract syntax of a metamodel may be defined as a pattern.</p> <p>MOF relationship</p> <p>ModelPattern inherits indirectly Package and Classifier from MOF.</p>	
Attributes or Reference	Datatype and Cardinality	Description
baseClass[UML]	Class Name	
Icon[UML]	Geometry[0..1]	
extendedElement[UML]	ModelElementName [0..*]	
constraint[UML]	String [0..1]	
requiredTag[UML]	String[0..*]	
stereotypeConstraint[UML]	String [0..1]	
Constraints		

C.4 Communication

<p>Communication</p>	<p>Supper Class</p>	<p>ModelClassifier</p>
	<p>Communication is a metaclass designating collaboration modelling using the pattern mechanism.</p> <p>Communication provides a facility to build composed and nested collaborations based on Pattern. Each part of layered collaborations may be standardized if necessary.</p> <p>MOF relationship</p> <p>In MOF, there is no metaclass corresponding to “Communication”. However, the metaclass “Communication” inherits indirectly the metaclass “Package” from MOF.</p>	
<p>Attributes or Reference</p>	<p>Datatype and Cardinality</p>	<p>Description</p>
<p>targetSystemCollaboration</p>	<p>String</p>	
<p>Constraints</p>		

C.5 Component

Component	Supper Class	ModelClassifier
	<p>Component is a metaclass designating component modelling using the pattern mechanism.</p> <p>ModelComponent provides a facility to build composed and nested components based on Pattern. Each part of layered components may be standarized if necessary. Actually, the operations attached to the pattern may be connected in assembling the components.</p> <p>MOF relationship</p> <p>In MOF, there is no metaclass corresponding to “Component”. However, the metaclass “Component” inherits indirectly the metaclass “Package” from MOF.</p>	
Attributes or Reference	Datatype and Cardinality	Description
targetSystemComponent	String	
Constraints		

C.6 Framework

Framework	Supper Class	ModelClassifier
	<p>Framework is a metaclass designating framework modelling using the pattern mechanism.</p> <p>Framework provides a facility to build composed and nested frameworks based on Pattern. Each part of layered frameworks may be standardized if necessary. Actually, the operations attached to the pattern may be connected in assembling the components and frameworks.</p> <p>MOF relationship</p> <p>In MOF, there is no metaclass corresponding to “ModelFramework”. However, the metaclass “ModelFramework” inherits indirectly the metaclass “Package” from MOF.</p>	
Attributes or Reference	Datatype and Cardinality	Description
targetSystemFramework	String	
Constraints		

Annex D: Level Pair (Informative)

This clause of this document specifies the Level Pair of metadata objects that form the structure of a MMF's metadata registry. A MMF's metadata registry will be populated with instances of these metadata objects (metadata items), which in turn define Layer, View, Level, Context and Category, e.g. in an application domain. In other words, instances of metadata specify Level Pairs of application level data. In turn, the level specific data of application will be populated by the real world data as instances of those defined Layer, View, Context, Category and so on.

NOTE ISO/IEC 10027:1990 IRDS Framework explains the concepts of different levels of modelling.

Descriptions of specific types of optional Administered Item:

- UpperLayer (see D.1)
- UpperModelElement (see D.2)
- LowerLayer (see D.3)
- LowerModel (see D.4)
- LowerModelElement (see D.5)
- ModelView (see D.6)
- ModelLevel (see D.7)
- ModelContext (see D.8)
- ModelCategory (see D.9)

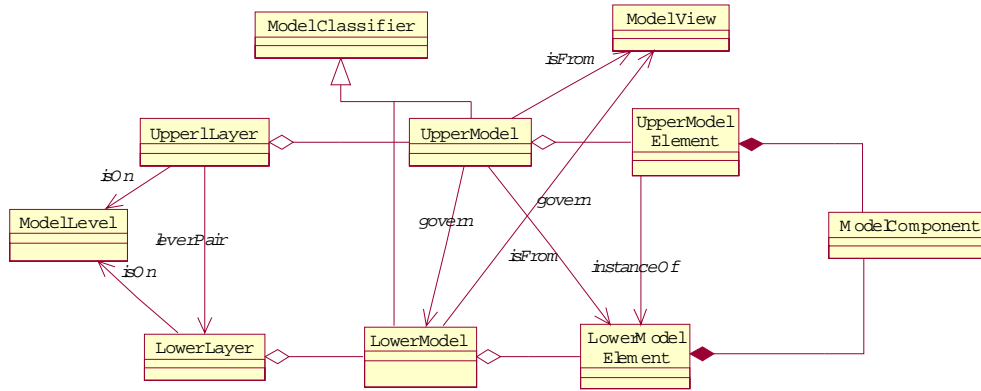


Figure 14- MMF Core (Level Pair)

D.1 UpperLayer

UpperLayer	Supper Class	None
	UpperLayer is a metaclass designating a metamodel layer pointed by a ModelLevel. An UpperLayer has a ModelLayer according to their LevelPair. The UpperLayer forms an aggregation of metamodels belonging to the meta level. MOF relationship UpperLayer is corresponding to meta level layer in MOF metadata architecture. If M3 level is metamodel layer then M2 level is model layer, and similarly if M2 is metamodel then M1 is model.	
Attributes or Reference	Datatype and cardinality	Description

isOn	ModelLevel	
levelPair	LowerLayer	
Constraints		

D.2 UpperModelElement

UpperModelElement	Supper Class	None
	<p>The UpperModelElement is a metaclass designating composite elements of a UpperModel.</p> <p>An UpperModelElement forms assembling ModelSelections.</p> <p>An UpperModel also forms aggregation of UpperModelElement.</p> <p>MOF relationship</p> <p>The UpperModelElement consists of model elements by ModelComponent and metamodel constructs in MOF.</p>	
Attributes or Reference	Datatype and cardinality	Description
Constraints		
A LowerModelElement should be an instance of the corresponding UpperModelElement.		

D.3 LowerLayer

LowerLayer	Supper Class	None
	<p>LowerLayer is a metaclass designating a model level layer in the metadata architecture.</p> <p>A LowerLayer has links to a ModelContext of modelling targete and a ModelConcept of model elements namespace.</p> <p>MOF relationship</p> <p>LowerLayer is corresponding to meta level layer in MOF metadata architecture.</p>	
Attributes or Reference	Datatype and Cardinality	Description
metaLevelIdentifier	String	
basedOn	Model Context	
isOn	Model Concept	
Constraints		

D.4 LowerModel

LowerModel	Supper Class	ModelClassifier
	<p>LowerModel is a metaclass identifying model or metamodel at LowerrLayer</p> <p>A LowerModel may be defiened from various ModelViews.</p> <p>Links among LowerModels may be connected according to associations and references.</p> <p>MOF relationship</p> <p>LowerModel is corresponding to a model at the model level layer in the MOF metadata architecture. MOF has the facility to handle the reflective operations. In MMF, a metamodel is expanded into a set of meta objects, and the operations at the lower model layer are allowed to handle the information about the upper metamodel.</p>	
Attributes or Reference	Datatype and cardinality	Description
modelAssociation	0..*	
<p>Constraints</p> <p>A LowerModel should be governed by corresponding UpperModel.</p> <p>LowerModel should be described with conforming to abstract syntax defined by UpperModel under additional constraints.</p>		

D.5 LowerModelElement

LowerModelElement	Supper Class	None
	<p>The LowerModelElement is a metaclass designating composite elements of ModelSelection.</p> <p>LowerModelElement plays a role of assembling ModelSelections.</p> <p>MOF relationship</p> <p>The LowerModelElement consists of ModelSelections and metamodel constructs in MOF.</p>	
Attributes or Reference	Datatype and cardinality	Description
Constraints		
LowerModelElement should be an instance of corresponding UpperModelElement.		

D.6 ModelView

ModelView	Supper Class	None
	The ModelView is a metaclass designating modelling viewpoints. The Modelview may be classified and identified with the specific ontology. A kind of ontology should be registered with a classification schema in MDR.	
Attributes or Reference	Datatype and cardinality	Description
viewPoint	string	
scope	String [0..1]	
purpose	Sring [0..1]	
Constraints		

D.7 ModelLevel

ModelLevel	Supper Class	None
	<p>MetaLevel is a metaclass designating a meta level in the metadata architecture.</p> <p>MetaLevel has links to a ModelContext of modelling target and a ModelConcept of model elements namespace.</p> <p>MOF relationship</p> <p>The metaclasss “MetaLevel” is corresponding to meta level layer in MOF metadata architecture.</p>	
Attributes or Reference	Datatype and cardinality	Description
metaLevelIdentifier	String	
basedOn[Reference]	Model Context	
isOn[Reference]	Model Concept	
Constraints		

Annex E: Example (Informative)

<TBD>

Bibliography

- [1] ISO/IEC TR 9007:1987, *Information processing systems – Concepts and terminology for the conceptual schema and the information base*

TR 9007 provides information on conceptual modelling.

- [2] ISO/IEC 10027:1990, *Information technology – Information Resource Dictionary System (IRDS) Framework*

ISO/IEC 10027 describes the concept of levels of modelling.

- [3] ISO/IEC TR 15452:2000, *Information technology – Specification of data value domains*

TR 15452 describes the specification of value domains. It is expected to be replaced by ISO/IEC TR 20943-3.

- [4] ISO/IEC TR 20943-1:200n (to be published), *Information technology – Achieving metadata registry content consistency – Part 1:Data elements*

TR 20943-1, which is under development at the time of publication of ISO/IEC 11179-3, will provide guidelines for recording data elements in a 11179-3 metadata registry.

- [5] ISO/IEC TR 20943-3:200n (to be published), *Information technology – Achieving metadata registry content consistency – Part 3:Value domains*

TR 20943-3, which is under development at the time of publication of ISO/IEC 11179-3, will provide guidelines for recording value domains in a 11179-3 metadata registry.

- [IS11179] ISO/IEC FCD 11179-3 Information Technology-Data Management and Interchange- Metadata Registry(MdR)- Part 3: Registry Metamodel (MdR3)

- [TR15452] ISO/IEC TR 15452:2000, Information technology – Specification of data value domains

- [ISO 9735 (TC 154)]

- [MOF] Meta Object Facility (MOF) Specification, Needham: OMG, 2000, formal/00-04-03

- [CWM] Common Warehouse Metamodel (CWM) Specification: OMG, 2000, ad/2000-01-01, ad/2000-01-02, ad/2000-01-03, ad/2000-01-11

- [MDA] Policies and Procedures for MDA: OMG, 2001, pp/2001-09-01

- [EDOC] UML Profile for EDOC submission: OMG, 2001, ad/2001-06-09

- [EAI] UML Profile for EAI submission: OMG, 2001, ad/2001-08-02

- [bpWS] ebXML Business Process Analysis Worksheets and Guidelines. V1.0, May 11 2001. ebXML Business Process Project Team.

- [ebBPBIAO] ebXML Business Process and Business Information Analysis Overview . Version 1.0 May 11 2001. ebXML Business Process Project Team.

ISO/IEC CD19763-2:2004(E)

- [ebBPSS] ebXML Business Process Specification Schema. Version 1.0 May 11 2001. Context/*Meta Model* Group of the CC/BP Joint Delivery Team.
- [ebPROC] ebXML Catalog of Common Business Processes. Version 1.0, May 11 2001. ebXML CC/BP Analysis Team.
- [bpPATT] ebXML Business Process and Simple Negotiation Patterns. Version 1.0, May 11 2001. ebXML Business Process Project Team.
- [ISO14662] Information Technologies - Open-EDI Reference Model. ISO/IEC 14662:1997(E). International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).
- [ebCCD&A] ebXML Methodology for the Discovery and Analysis of Core Components. V1.0, May 11 2001. ebXML Core Components Project Team.
- [ebCNTXT] The role of context in the re-usability of Core Components and Business Processes. Version 1.0, May 11 2001. ebXML Core Components Project Team.
- [ebGLOSS] ebXML. TA Glossary. Version 1.0, May 11 2001 . Technical Architecture Project Team.
- [ebTA] ebXML Technical Architecture Specification. Version 1.0.4. 16 February 2001. ebXML Technical Architecture Project Team.
- [ebCCDOC] ebXML specification for the application of XML based assembly and context rules. Version 1.0, 11 May 2001. ebXML Core Components.
- [ebCNTXT] *ebXML Concept - Context and Re-Usability of Core Components*. Version 1.01. February 16, 2001. ebXML Core Components Project Team.
- [ebRIM] *ebXML Registry Information Model*. Version 0.56. Working Draft. 2/28/2001. ebXML Registry Project Team.
- [ebRS] *ebXML Registry Services*. Version 0.85. Working Draft. 2/28/2001. ebXML Registry Project Team.
- [ebTA] *ebXML Technical Architecture Specification*. Version 1.0. 4 January 2001. ebXML Technical Architecture Project Team.
- [bpOVER] *Business Process and Business Information Analysis Overview*. Version 1.0. Date 11 May 2001. ebXML Business Process Project Team
- [bpPROC] *ebXML Catalog of Common Business Processes*. Version 1.0. Date May 11, 2001. ebXML Business Process Project Team
- [UMM] UN/CEFACT Modeling Methodology. CEFACT/TMWG/N090R9. February 2001. UN/CEFACT Technical Modeling Working Group.