Committee Draft ISO/IEC CD	
Date: **2005-06-30**	Reference number: ISO/JTC 1/SC 32N**1332**
Supersedes document SC 32N1186	

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: **2005-09-30** 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 2005-06-30.

Medium: E

No. of pages: 64

Address Reply to: Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32, Farance Inc, 360 Pelissier Lake Road, Marquette, MI 49855, United States of America

Telephone: +1 906-249-9275; E-mail: MannD@battelle.org

ISO/IEC JTC 1/SC 32 WG2

Date: 2005-06-16

ISO/IEC 2ndCD19763-2:2005(E)-

ISO/IEC JTC 1/SC 32/WG 2

Secretariat:

Information Technology – Framework for metamodel interoperability-- Part-2 : Core model

Document type: International Standard
Document subtype:
Document stage: (50) WorkingDraft
Document language: E

Contents

Table of Figures

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/IECD 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 – Target Structure

— Annex F – Examples

Introduction

Due to the spread of E-Business and E-Commerce over the Internet, the effective exchange of business transactions and other related information across countries and cultures has become a prime concern for people both inside and outside the IT industry. They 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 for each business domain.

One of the things to be mentioned today is that most of those standard efforts tend to be focused on the contents of a 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 (Organization for the Advancement of Structured Information Standards) for UMM (UN/CEFACT Modeling Methodology), ebXML, OMG (Object Management Group) for MOF (Meta Object Facility), XMI (XML Metadata Interchange), CWM (Common Warehouse Metamodel), EDOC (Enterprise Distributed Object Computing), EAI (Enterprise Application Integration), and HL7 (Health Level Seven) for HDF (Health Development Framework) 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, otherwise duplications and inconsistencies inevitably occur.

A unified framework for classifying and registering normative model elements is a vital component in any effort to establish harmonisation of metamodels that have been developed independently, and will also facilitate their reuse widely across organisations

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 modelling 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.

Those metamodels and models could describe business concepts and components that should be shared for achieving active communication over the Internet. The terminology “metamodel” is used from various aspects; many kinds of business concepts or models could be considered with different granularities, abstraction levels of a modelling target, purposes, intents and viewpoints.

For example, considering message exchange of e-commerce, a business process model about interaction among parties could be considered with metamodel or model. In this case, modelling artefacts such as a message format and protocol could be typical registered target. Moreover, concepts and their instances including vocabularies and classifications could be registered using this core model.

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.* This part of *ISO/IEC CD19763* specifies the core model which is required to register metamodel items, and which may be used in situations where a complete metamodel registry is appropriate.

The core model provides the specification of information structure for registering artefacts described with metamodel and/or model.

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.

For modelling business concepts, it is important to standardize conceptual models (i.e. metamodel) of each target domain. However, those standardizations are out of scope in this part of ISO/IEC 19763.

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 19501-1:2005, Information technology – Unified Modelling Language (UML) – Part 1: Specification

ISO/IEC 19502-1:2004, Information technology – Meta Object Facility (MOF): Specification

ISO/IEC 19503-1:2004, Information technology – XML Metadata Interchange (XMI)

ISO/IEC 8824-1:2002, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation

Editor's Note: Consider including TC 37 metamodel for terminology, MARC metadata, IEC 80045 Document Management, TC 46 Document Management, TC 211 metamodel/metadata, TC 184 Step, et

3 Definitions

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

3.1 defines UML [ISO/IEC 19501-1:2005] and 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 3.1

3.1 MOF Terms used in specifying the MMF metamodel

Editor's Note: The MOF Terminology in this clause came from the Glossary of MOF 1.4 Specification in OMG. However, the MOF PAS submission as ISO/IEC 19502 does not include the Glossary. The definitions below should be reconsidered and checked if those are appropriate or not.

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
argument
A binding for a parameter that resolves to a run-time instance.

3.1.3
aggregate class
A class that represents the "whole" in an aggregation (whole-part) relationship.

NOTE 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.

NOTE See: *composition*.

3.1.5
artefact
product
A physical piece of information that is used or produced by a software development process.

EXAMPLE Models, source files, scripts, and binary executable files. An artefact may constitute the implementation of a deployable component.

3.1.6
association
The semantic relationship between two or more classifiers that specifies connections among their instances.

3.1.7
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.

3.1.8
association end
The endpoint of an association, which connects the association to a classifier.

3.1.9
attribute
A feature within a classifier that describes a range of values that instances of the classifier may hold.

3.1.10
binding
The creation of a model element from a template by supplying arguments for the parameters of the template.

3.1.11
cardinality
The number of elements in a set.

NOTE Contrast: *multiplicity*.

3.1.12
child
In a generalization relationship, the specialization of another element, the parent.

NOTE See: *subclass*, *subtype*.

NOTE Contrast: *parent*.

3.1.13
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.

NOTE See: *interface*.

3.1.14
classifier
A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.

3.1.15
classification
The assignment of an object to a classifier.

NOTE See *dynamic classification*, *multiple classification* and *static classification*.

3.1.16
class diagram
A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.

3.1.17
component
A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (e.g., implementation classes) that reside on it, and may be implemented by one or more artefacts (e.g., binary ,executable, or script files).

NOTE Contrast *artefact.*

3.1.18
composite class
A class that is related to one or more classes by a composition relationship.

NOTE See: *composition*.

3.1.19
composition
composite aggregation
A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive.

3.1.20
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.

NOTE See: *tagged value, stereotype.*

3.1.21
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.

3.1.22
context
A view of a set of related modelling elements for a particular purpose, such as specifying an operation.

3.1.23
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.

3.1.24
dependency
A relationship between two modelling elements, in which a change to one modelling element (the independent element) will affect the other modelling element (the dependent element).

3.1.25
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.

3.1.26
domain
An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.

3.1.27
element
An atomic constituent of a model.

3.1.28
feature
A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.

3.1.29
framework
1. A stereotyped package that contains model elements which specify a reusable architecture for all or part of a system. Frameworks typically include classes, patterns or templates. When frameworks are specialized for an application domain, they are sometimes referred to as application frameworks.

NOTE See: *pattern.*

3.1.30
generalizable element
A model element that may participate in a generalization relationship. See: *generalization*.

3.1.31
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.

NOTE See: *inheritance.*

3.1.32
inheritance
The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior.

NOTE See: *generalization*.

3.1.33
instance
An entity that has unique identity, a set of operations that can be applied to it, and state that stores the effects of the operations.

NOTE See: *object*.

3.1.34
Interface
A named set of operations that characterize the behaviour of an element.

3.1.35
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.

NOTE Contrast: *partition*.

3.1.36
link
A semantic connection among a tuple of objects. An instance of an association.

NOTE See: *association.*

3.1.37
link end
An instance of an association end.

NOTE See: *association end*.

3.1.38
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.

3.1.39
metaclass
A class whose instances are classes. Metaclasses are typically used to construct metamodels.

3.1.40
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.

3.1.41
metamodel
1. A model that defines the language for expressing a model.

2. A data model that specifies one or more other data models

3.1.42
metaobject
A generic term for all metaentities in a metamodelling language.

EXAMPLE metatypes, metaclasses, metaattributes, and metaassociations.

3.1.43
method
The implementation of an operation. It specifies the algorithm or procedure associated with an operation.

3.1.44
model element
An element that is an abstraction drawn from the system being modeled.

NOTE 1 Contrast: *view element*.

NOTE 2 In the MOF specification model elements are considered to be metaobjects.

3.1.45
Meta Object Facility
MOF
A common facility to describe metamodel

3.1.46
MOF compliatnt
Described according to MOF model

3.1.47
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.

NOTE Contrast: *cardinality*.

3.1.48
name
A string used to identify a model element.

3.1.49
namespace
A part of the model in which the names may be defined and used. Within a namespace, each name has a unique meaning.

NOTE See: *name*.

3.1.50
object
An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, and behavior is represented by operations, methods, and state machines. An object is an instance of a class.

NOTE See: *class, instance*.

3.1.51
package
A general purpose mechanism for organizing elements into groups. Packages may be nested within other packages.

3.1.52
parameter
formal 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.

NOTE Contrast: *argument*.

3.1.53
parameterized element
template
The descriptor for a class with one or more unbound parameters.

3.1.54
parent
In a generalization relationship, the generalization of another element, the child.

NOTE 1 See: *subclass*, *subtype*.

NOTE 2 Contrast: *child*.

3.1.55
pattern
A template collaboration.

3.1.56
profile
A profile may specify model libraries on which it depends on the metamodel subst that it extends. A stereotyped package that contains model elements which have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints.

3.1.57
property
A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others may be user defined.

3.1.58
reference
pointer

1. A denotation of a model element.

2. A named slot within a classifier that facilitates navigation to other classifiers.

3.1.59
relationship
A semantic connection among model elements.

EXAMPLE Include associations and generalizations.

3.1.60
repository
A facility for storing object models, interfaces, and implementations.

3.1.61
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).

3.1.62
stereotype
A new type of modelling 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.

NOTE See: *constraint, tagged value.*

3.1.63
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.

3.1.64
subclass
In a generalization relationship, the specialization of another class; the superclass.

NOTE 1 See: *generalization*.

NOTE 2 Contrast: *superclass*.

3.1.65
subtype
In a generalization relationship, the specialization of another type; the supertype.

NOTE 1 See: *generalization*.

NOTE 2 Contrast: *supertype*.

3.1.66
superclass
In a generalization relationship, the generalization of another class; the subclass.

NOTE 1 See: *generalization.*

NOTE 2 Contrast: *subclass*.**3.1.67**
supertype
In a generalization relationship, the generalization of another type; the subtype.

NOTE 1 See: *generalization*.

NOTE 2 Contrast: *subtype*.

3.1.68
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.

NOTE See: *constraint, stereotype.*

3.1.69
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.

NOTE 1 See: *class, instance*.

NOTE 2 Contrast: *interface*.

3.1.70
XML Metadata Interchagne
XMI
A XML Schema generated from MOF compliant model.

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

3.2.1
Administered Item
A registry item for which administrative information is recorded in an Administration

[ISO/IEC 11179-3:2003,(3.3.1)]

3.2.2
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)]

3.2.3
conceptual data model
a data model that represents the form of data from an abstract view of the real world or the virtual world

3.2.4
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 (3.2.10) and optional (3.2.20).

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

3.2.5
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)]

3.2.6
data model
a graphical and/or lexical representation of data, specifying their properties, structure and inter-relationships

3.2.7
definition
representation of a concept by a descriptive statement, which serves to differentiate it from related concepts

[ISO 1087-1:2000 (3.3.1)]

3.2.8
designation
representation of a concept by a sign which denotes it

[ISO 1087-1:2000 (3.4.1)]

3.2.9
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 An entity exists whether data about it are available or not.

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

3.2.10
language
system of signs for communication, usually consisting of a vocabulary and rules

[ISO 5127:2001 (1.1.2.01)]

3.2.11
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 (3.2.3) and optional (3.2.20).

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

3.2.12
metadata
data that defines and describes other data

3.2.13
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 Profile, Model Instances etc.

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

3.2.14
metadata object
an object type defined by a metamodel

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

3.2.15
metadata register
the information store or database maintained by a Metadata Registry

3.2.16
Metadata Registry
MDR
an information system for registering metadata

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

3.2.17
metamodel construct
a unit of notation for modelling

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

3.2.18
Metamodel Framework
MMF
a framework for registering artefacts based on metamodel and model

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 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 (3.2.3) and mandatory (3.2.10).

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 or specified by a unique combination of characteristics

[ISO 1087-1:2000 (3.2.1)]

3.2.25
Uniform Resource Identifier
URI
a simple and extensible means for identifying a resource.

[IETF RFC 2396]

3.2.26
Universal Unique IDentifier
UUID
These are 128 bit numbers which is guaranteed to be unique. The mechanism used to guarantee that UUIDs are Unique is through combinations of hardware addresses, time stamps and random seeds.

3.2.26
Object Identifier
OID
A value (distinguishable from all other such values) which is associated with an object.

[ISO/IEC 8824-1:2002 3.6.47]

3.2.27
XML Schema
[W3C]

4 Structure of a MetaModel Framework (MMF)

4.1 Overview of MMF Core Model

This part of ISO/IEC 19763 uses a metamodel to describe the structure of an MMF's metadata register. 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, the implementation should be strictly derived from the MMF's models to establish the common management of metamodels and their derived models.

In this standard, concerning metamodel for the upper layer and model for the lower 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.

For descriptive purposes, the model specifying MMF is organized into five following packages:

-MDR Package (see Annex A)

-MOF Package (see Annex B)

-MMF Core Package (see 4.3, 4.4, 4.5, Annex C, and D)

-MMF Ontology Package (see ISO/IEC 19763-3)

-MMF Model Mapping Package (see ISO/IEC 19763-4)

The division of the model into packages is for descriptive purposes only and has no other significance. Figure 1 shows the MMF Core Package that consists of five packages such as Target, Registry, Relationship, LevelPair and ModelClassifier. The packages of MOF and MDR, shown with non-shaded package in Figure 1, have been specified outside of this standard (see 2).

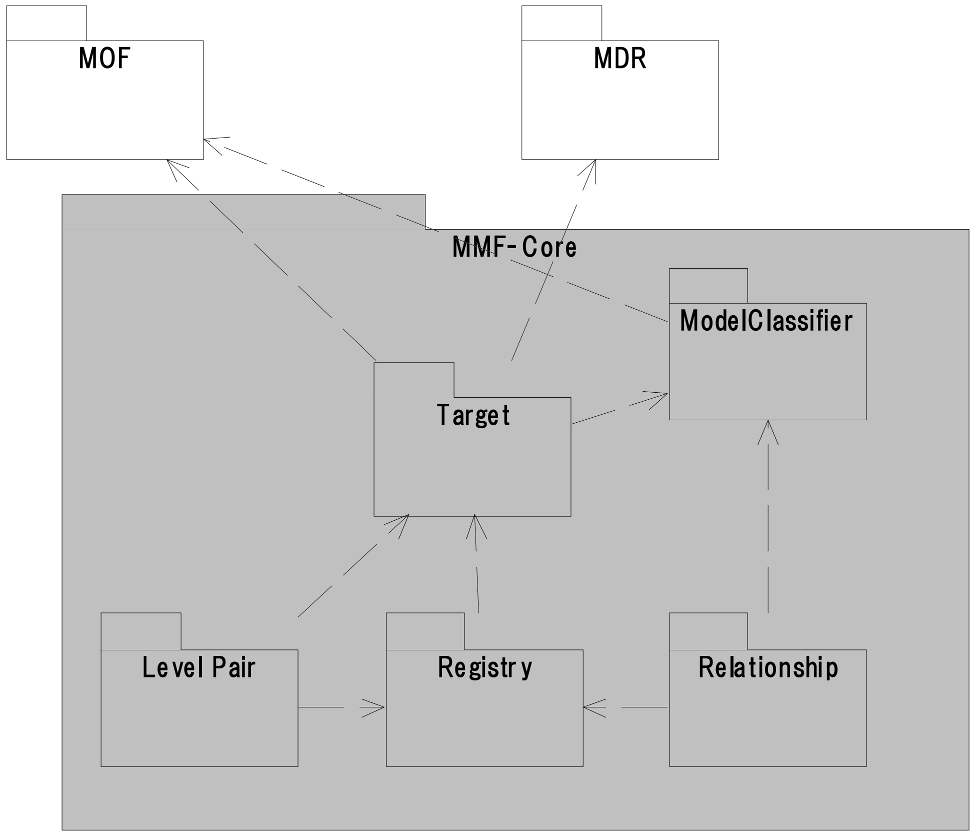

Figure 1- MMF-Core Packages

4.2 Convention for Definition of MMF Core

The MMF core registry metamodel is specified using Administered Items as defined in the Metadata Registry (MDR), and conforming Meta Object Facility (MOF) standards. This standard uses the term "metamodel construct" for the model constructs it uses, but "metadata objects" for the model constructs it specifies in the MOF. The metamodel constructs used include classes, relationships, association classes, attributes and references. These terms are defined in 3.1, and their use is described in Annex B. The specified metadata objects are defined in this Clause.

The MMF core metamodel is shown as a series of UML Class diagrams, with each Class being described as follows

(1) Superclasses

<immediate inherited classes>

(2) Attributes

attribute name : Datatype and Cardinality

<content and purpose>

(3) References

reference name : Datatype and Cardinality

<content and purpose>

(4) Constraints

<constraints if necessary, written in natural language>

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

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

-Registry Package (normative; see Figure 2 and 4.3)

-Target Package (normative; see Figure 3 and 4.4)

-Relationship Package (normative; see Figure 4 and 4.5)

-ModelClassifier Package (informative; see Annex C)

-LevelPair Package (informative; see Annex D)

These packages include some classes that are subclasses of Administered Item, as follows:

-ModelSign (see 4.3.1)

-ModelConcept (see 4.3.2)

-ModelInstances (see 4.3.3)

-ModelSelection (see 4.3.4)

-ModelDomainProfile (see 4.4.1)

-ModelComponent (see 4.4.6)

An instance of Administered Items should be globally identified. In this part of ISO/IEC 19763, a pointer to those memaobjects should be unique using such as OID or UUID.

The division of the model into packages 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.

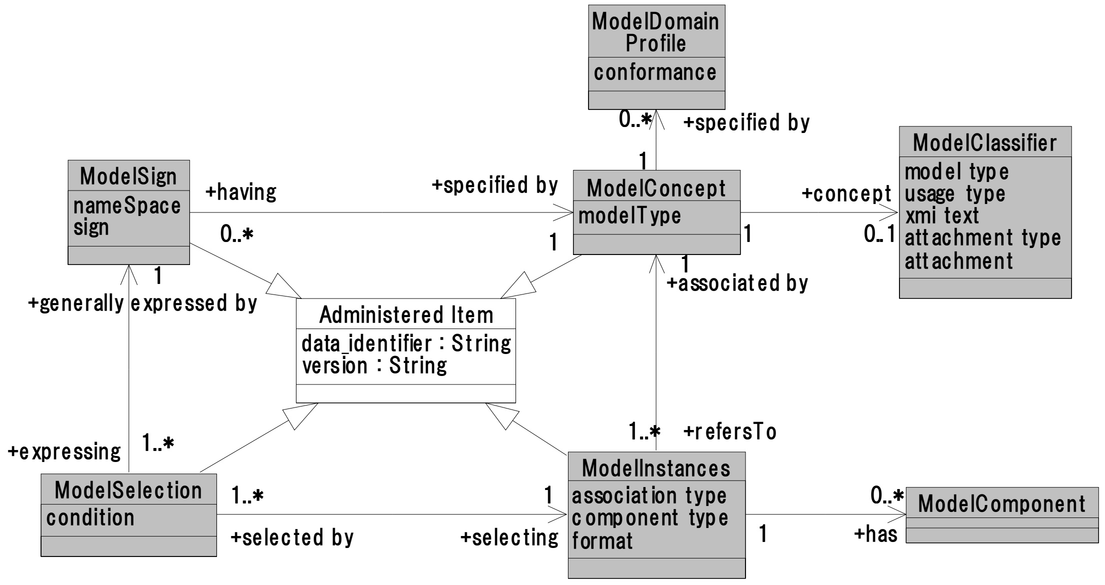

Figure 2- Registry Package in MMF Core

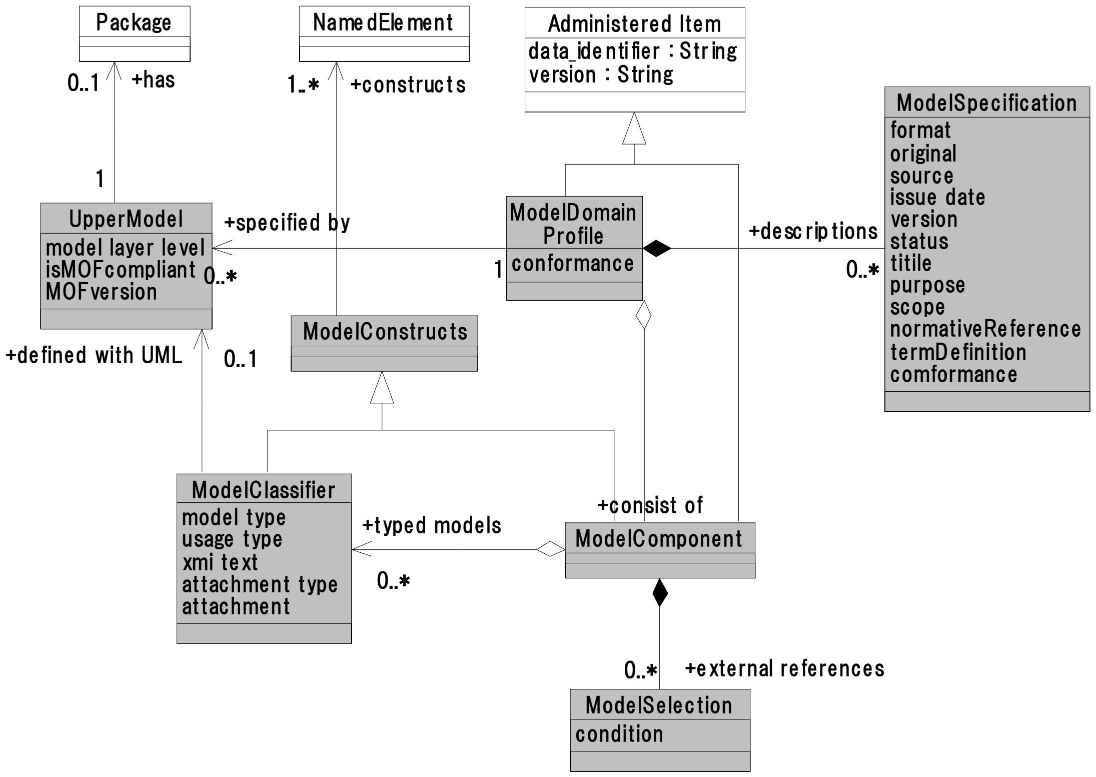

Figure 3- Target Package in MMF Core

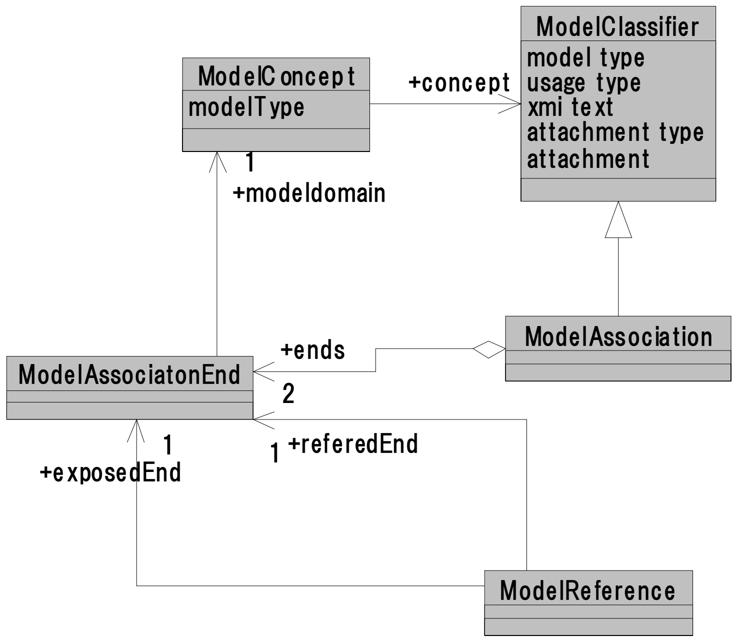

Figure 4- Relationship Package in MMF Core

4.3 Registry Package (Structure of Registry)

4.3.1 ModelSign

A sign is a symbol designating a named element in a namespace. A namespace is a collection of signs. It provides a name designating the concept. Synonyms may be registered for a concept.

In the core model, ModelSign is a metaclass whose instances each identify, within a namespace, a named element (package, class, association, etc) within a model to distinguish a thing of interest. It plays a role of container containing index for a model concept. A ModelSign is managed as an MDR Administered Item. The meaning of the sign of the ModelSign is specified by the associated ModelConcept.

A ModelSign, including its item name inherited from its superclass Administrated Item, should be registered as a multi-lingual item. This provides the multi-lingual indexes for a model concept.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

namespace : string [1..1]

Provides a string identifying the namespace where the sign is uniquely defined.

sign : string [1..1]

Provides a string identifying the named element invoking the concept defined by the ModelSign.

(3) References

specified by : ModelConcept [1..1]

Provides a pointer to the ModelConcept by which the meaning of this ModelSign is specified.

(4) Constraints

An instance of the ModelSign has a sign for the named element, which should be unique in the namespace

The combination of namespace and sign is to be unique.

The name of the namespace is to follow the rules of the particular environment (MOF, XML, etc) from which the namespace originates.

4.3.2 ModelConcept

Concept specifies meaning of the particular sign. It is generally expressed with appropriate signs. The model classifier, specified in the designating model domain profiles, defines the concept invoked by a sign.

In the core model, ModelConcept is a metaclass whose instances each identify a model classifier specified within ModelDomainProfiles. The ModelConcept provides a meaning of the concept that is defined by a ModelClassifier in the context of the ModelDomainProfile. The ModelClassfier may be described in an UpperModel of the ModelDomainProfile. A ModelClassifier plays a role of typed model, and ModelDomainProfiles play a role of specifying particular domain knowledge. A ModelConcept may be managed with a MDR Administered Item.

A ModelConcept, including its administrated item, should be registered as a multi-lingual item. This provides the multi-lingual name for a model domain profile.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

model type : ModelClassifier Code [1..1]

Identifies a code of kind of ModelClassifier.

(3) References

specified by : ModelDomainProfile [0..*]

Provides pointers to ModelDomainProfile profiles

concept : ModelClassifier [0..1]

Provides a pointer for Identifing to ModelClassifier

(4) Constraints

A ModelClassifier and its code system should be specified in particular ModelDomainProfile profiles.

4.3.3 ModelInstances

Instances are a set of referents of the concept designated by a sign. It consists of a set of model components.

In the core model, ModelInstances is a metaclass designating sets of values of ModelClassifier specified by a ModelConcept. ModelInstances consists of ModelComponensts as referents of ModelClassifier. It plays a role of container of registered ModelComponents. It has also an association “associated by” with the ModelConcept, and may be managed with a MDR Administered Item.

A ModelInstances, including administrated item, should be registered as multi-lingual item. This provides the multi-lingual name for a model instances.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

association type : type code [1..1]

Declares a code of kind of Association

The ModelSign and ModelConcept refer to the ModelInstances. Also, a ModelComponent in the ModelInstances has associations with the ModelClassifier and UpperModels in the ModelDomainProfile. There are four association types as follows:

a) Type-Instance

“Type-Instance” is an association type between a class and its object. Also, the class diagram in a model package and its object diagram may be included.

b) Super-Sub

“Super-Sub” is an association type between a super class and its inherited sub classes. Also a model package and its sub packages may be included.

c) Base-Variant

“Base-Variant” is an association type between a base model and its variant models that are created by modifying the base model according to the permitted operation. There are operations such as renaming, specifying, refining, substituting, extending and merging. See Table 5 in Annex F.

In the association type “Base-Variant” many operations above are performed on a base model partially and many times. Eventually, the lower model will be derived from the upper base model. The detail specification on specifying operations should be provided as a ModelSpecification for each registering target.

d) Abstract Syntax-Expression

“Abstract Syntax-Expression” is an association type between an upper metamodel and a lower model. In this case, the upper model in a ModelDomainProfile provides a metamodel. The lower model must be described according to the abstract syntax. Usually, stereotypes of UML are defined by such metamodels as a UML profile. The lower model will be drawn using those stereotypes.

component type : type code [1..1]

Declares a code of kind of ModelClassifier

format : string [1..1]

Identifies a file format allowed as a ModelComponent

(3) References

associated by : ModelConcept [0..1]

Provides a pointer to ModelConcept

has : ModelComponent [0..*]

Provides pointers to ModelComponent

(4) Constraints

The ModelInstances should be a set of ModelComponents derived from its ModelClassifier and associated by its ModelConcept.

The model elements such as pattern (template), stereotype (tagged value), coded value and constraint could be derived from upper model (metamodel).

A ModelClassifier and its code system should be specified in particular ModelDomainProfile profiles.

4.3.4 ModelSelection

Selection provides a mechanism that is combining a sign and its instances.
In the core model, ModelSelection is a metaclass whose instance keeps a selection from ModelInstances that are generally expressed with a corresponding sign of a ModelSign. It plays a role of connecting a ModelSign. A ModelInstances may be used in ModelComponent in order to assemble ModelConstructs and may be managed with a MDR Administered Item. A ModelSelection may have a condition concerning range of selected Model Instances. Also, ModelComponents may include ModelSelections that have a link to external reference.

A ModelSelection, including administered item, should be registered as multi-lingual measure. This provides the multi-lingual name for a model selection.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

condition : string [1..1]

Specifies a filtering for Model instances

(3) References

generally expressed by : ModelSign [1..1]

Provides a pointer to ModelSign

selecting : ModelInstances [0..1]

4.3.5 Provieds a pointer to ModelInstances

4.4 Target Package (Structure of Registered Target)

4.4.1 ModelDomainProfile

ModelDomainProfile is a metaclass designating a standard or profile composing of the model elements such as UpperModel, ModelSpecification, and ModelComponent. It plays a role of holding artefacts concerning a standard or its related profile. UpperModel may include a machine-readable model (or metamodel). ModelSpecififcatoin may have human-readable documents and materials ModelConstruct may have named model elements such as stereotypes, tagged values, patterns and so on. ModelComponent may be assembled with ModelSelections and defined as a ModelClassifier. A ModelDomainProfile may be managed with a MDR Administered Item.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

name: string [1..1]

Provides an identifier of the model profile

conformance : string [0..*]

Declares the level of conformance defined in ModelSpecification

(3) References

descriptions : ModelSpecification [1..*]

Provides specification of the standard or profile.

specified by : UpperModel [0..*]

Provides the model or metamodel

consist of : ModelComponent [0..*]

Provides pointers to ModelComponent

(4) Constraints

A ModelClassifier may be corresponding to a particular sign in ModelSign.

4.4.2 ModelSpecification

ModelSpecification is a metaclass designating a document of standard or profile. It may play a role of holding human-readable documents concerning a standard or its related profile. A ModelSpecification may include UpperModel, ModelClassifier and Model Constructs as a document if necessary.

(1) Attributes

format : string [1..1]

Provides a document file type. For example, ppt, pdf, http.

original : URI[1..1]

Declares the original material.

source : string [1..1]

Provides a name of standard development organization (SDO).

issue date : date [1..1]

Identifies a published date

version : number[1..1]

Identifies a version number of the standard or profile

status : string [1..1]

Provides the stage in standardization process

title : string[1..1]

Provides a title of the standard or profile.

purpose : string [1..1]

Describes objectives of the document

scope : string [1..1]

Describes applied area and domain

normativeReference : string [0..*]

Declares normative references

termDefinitions : string [1..*]

Include the terminology definitions

conformance : string [0..*]

Declares conformance level conditions

4.4.3 ModelConstructs

ModelConstructs is a metaclass representing a set of NamedElements in UML. It plays a role of composing of a NamedElement. A ModelConstructs may have named elements such as pattern (template), stereotype (tagged value), coded value and constraint. ModelConstructs should be standardized as common parts for modelling if necessary.

ModelConstructs has an association with NamedElemens from MOF.

(1) Superclasses

none

(2) References

constructs : NamedElement [0..*]

Provides a pointer to Named Element derived from MOF

4.4.4 ModelClassifier

ModelClassifier is a mataclass identifying and defining modelling unit and typed model as a concept. It may be used by a ModelConcept to designate a typed model with association link “concept”. A ModelDomainProfile may have a set of ModelClassifiers as ModelConstruct. A ModelClassifier may have some ModelAssociations and ModelReferences.

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

(1) Superclasses

ModelConstructs

(2) Attributes

model type : type code [1..1]

Specifies a type of ModelClassifier

usage type : type code [1..1]

Provides a purpose of usage.

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 : URI [0..1]

Provides the content with various expressions

(3) References

defined with UML : UpperModel[0..1]

Provides a pointer to Upper Model Package.

(4) Constraints

A ModelClassifire is composed of a package that may be also a composition of ModelConstructs. One unit of model consists of one package.

Packages are also used as organizational constructs in modelling. Nesting, importation, and generalization are used to manage the complexity of models.

A ModelClassifier and its code system should be specified in particular ModelDomainProfile profiles.

4.4.5 UpperModel

UpperModel is a metaclass identifying model or metamodel at UpperLayrer. An UpperModel may be defined from various ModelView. The Links among UpperModels may be connected according to associations and references.

UpperModel is corresponding to a metamodel (M2) or model (M1) 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.

(1) Superclasses

None

(2) Attributes

model layer level : string [1..1]

Provides the layer level of model classifier such as M2 and M1.

has : Package [0..1]

Provides a MOF compliant model.

isMOFcompliant : boolean [1..1]

Declares whether the metamodel is MOF compliant or not.

MOFversion : string [1..1]

Provides the version number about MOF

(3) Constraints

An UpperModel refers to corresponding LowerModel.

LowerModel should be described with conforming to abstract syntax on upper metamodel under the constraints.

4.4.6 ModelComponent

The ModelComponent is a unit of target modelling artefacts.

ModelComponent is a metaclass designating a specialized ModelConstructs that may be assembled with a set of ModelClassifiers or ModelSelections. A ModelComponent may play a role of ModelClassifier as a typed model. A ModelComponent may have ModelSelections that plays a role of connecting external elements with like “plug and play” manner.

(1) Superclasses

ModelConstructs and Administered Item (from MDR)

(2) References

typed models : ModelClassifier [1..*]

Provide pointers to ModelClassifier for specifying a model type or model types.

external references : ModelSelection [0..*]

Provide pointers to ModelSlection for specifiing registered ModelComonents through ModelSelection

(3) Constraints

4.4.7 ModelSelection should be used limited to appropriate for the model type of a ModelComponent in the ModelInstances that is pointed by the ModelSelection.

4.5 Relationship Package (Relationship of Registered Target)

4.5.1 ModelAssociation

ModelAssociation is a metaclass specifying an association among ModelClassifiers. 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.

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.

(1) Superclasses

ModelClassifier

(2) References

ends : ModelAssociationEnd [2..*]

Provides a link of ModelClassifiers.

(3) Constraints

The definition of ModelAssociation needs to specify two ModelAssociationEnds.

If a ModelAssociation is directional, the name such as suggesting its direction should be used.

Even if there are no direct ModelAssociation, indirect one derived from given metamodels may exist.

The mutual reference among ModelClassifiers may be represented with ModelReference.

4.5.2 ModelAssociationEnd

ModelAssociationEnd is a metaclass designating two end of a ModelAssociation. Each ModelAssociationEnd defines a role of a ModelClassifier participant in the ModelAssociation and constraints on sets of the ModelClassifiers

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

(1) Superclasses

None

(2) References

modeldomain : ModelConcept [1..1]

Declares ModelConcept in which it is defined

(3) Constraints

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. An instance of a ModelAssociationEnd provides the information about ModelReference related to the referenced ModelClassifier instance.

4.5.3 ModelReference

ModelReference is a metaclass defining a ModelClassifier’s information of, and access to, links and their instances defined by a ModelAssociation. Although a ModelReference derives much of its state from a corresponding ModelAssociationEnd, it provides additional information. A ModelReference is defined against a ModelClassifier. An instance of a ModelReference may hold one or more links toward instances of corresponding ModelInstances. A ModelReference has a roll defined by ModelAssociationEnd.

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.

(1) Superclasses

None

(2) References

exposedEnd : ModelAssociationEnd [1..1]

The exposedEnd of a ModelReference is the ModelAssociationEnd representing the end of the ModelReference's owning ModelClassifier within the defining ModelAssociation.

referedEnd : ModelAssociatonEnd [1..1]

The referencedEnd of a ModelReference is the end representing the set of LinkEnds of principle interest to the ModelReference. The ModelReference provides access to the instances of that ModelAssociationEnd’s ModelClassifier, which are participants in that ModelAssociationEnd’s ModelAssociation, connected through that ModelAssociationEnd’s LinkEnds. In addition, the ModelReference derives the majority of its state information - multiplicity, etc., from that ModelReference.

(3) Constraints

The multiplicity for a ModelReference must be the same as the multiplicity for the referenced ModelAssociationEnd.

ModelClassifier scoped ModelReferences are not meaningful in the current M1 level computational model.

The type attribute of a ModelReference and its referenced ModelAssociationEnd must be the same.

A ModelReference is only allowed for a navigable ModelAssociationEnd

The containing Class for a ModelReference must be equal to or a subtype of the type of the ModelReference’s exposed ModelAssociationEnd.

The referenced ModelAssociationEnd for a ModelReference must be visible from the ModelReference.

5 Conformance

This part of ISO/IEC CD 19763 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 4.2, 4.3 and 4.4. Conformance claims shall specify a Degree and a Level of Conformance, as described below.

5.1 Degree of Conformance

MMF Core conformance is specified along three orthogonal viewpoints; first is a value requirement viewpoint, second is interoperability viewpoint of metadata format, and third is conformance between registered content such as Uppermodel and Lowermodel.

5.2 Levels of Conformance

MMF Core provides two level of conformance. Level one conformance must satisfy value requirements. Level two conformance must support the XMI format in addition to Level one. Concerning conformance on registered content, it should provide the conformance statements for each object.

5.3 Obligation

(1) Value requirement viewpoint
One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher. Table1, 2 and 3 show the requirement about attributes and references for each metaclass. “M” means that the value is Mandatory and “O” does that it is Optional.

Table 1- Requirement for the Registry Package

Class	Attribute/Reference		Class	Attribute/Reference	
ModelSign	*namespace*	M	ModelSelection	*condition*	O
	sign	M		*generally expressed by*	M
	specified by	M		*selecting*	M
Modelconcept	*model type*	M	ModelDomainProfile	*name*	M
	specified by	M		*descriptions*	O
	concept	M		*conformance*	O
ModelInstances	*association type*	M		*specified by*	O
	component type	M		*consist of*	O
	associated by	M			

	has	M			

Table 2- Requirement for the Target Package

Class	Attribute/Reference		Class	Attribute/Reference	
ModelSpecification	*format*	M	ModelConstructs	*constructs*	O
	original	M	ModelClassifier	*model type*	O
	source	O		*usage type*	O
	Issue date	O		*defined with UML*	O
	version	O		*xmi text*	O
	status	O		*attachment type*	O
	title	M		*attachment*	O
	purpose	O	UpperModel	*model layer level*	M
	scope	O		*has*	M
	normativeReference	O		*isMOFcompliant*	M
	termDefinitions	O		*MOFversion*	O
	conformance	O	ModelComponent	*type models*	O
				external references	O

Table 3- Requirement for Relationship Package

Class	Attribute/Reference		Class	Attribute/Reference	
ModelAssociation	*ends*	M	ModelReference	*exposeEnd*	M
ModelAssociationEnd	*modeldomain*	M		*referedEnd*	M

(2) Interoperability of metadata format
In this part of ISO/IEC 19763, no concrete metadata format is specified. However, the XMI schema for MMF core model is a primary metadata format for interoperability. Another metadata format may be allowed if the appropriate XML schema is supplied as a profile.

(3) Interoperability about registered content

A metamodel provides the specification of a model domain. The ModelClassifier conforming to the upper metamodel should be used to build a concrete lower model.

As such a metamodel is a kind of standard, stereotypes and patterns conforming to the standard should be also specialized as a profile of the standard. Sharing and reusing those elements could achieve standardization of business object models.

5.4 Implementation Conformance Statement (ICS)

The distinction between "Level one" and "Level two" implementations is necessary to address the simultaneous needs for interoperability and extensions. This part of ISO/IEC 19763 describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions, and industries, and:

a) are not directly specified by this part of ISO/IEC 19763,

b) are specified and agreed to the other parts of ISO/IEC 19763, and

c) may serve as trial usage for future editions of this part of ISO/IEC 19763.

A level one implementation may be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 19763. A level two conforming implementation may be more useful, but may be less interoperable with respect to this part of ISO/IEC 19763.

About the content conformance, registered objects should declare a conformance specification in the ModelDomainProfile or ModelSpecification. It may say how much and what parts of metamodels each model instance accords with or not. The comformance level of a registered object should be identified for building a reusable model of business or a reusable software component to keep interoperability within an enterprise or in trade between enterprises.

5.5 Roles and Responsibilities for Registration

This standard defines the relationship between metamodel and model, the framework of registering, sharing and reusing normative modelling constructs. In the layer M2 of the metamodel architecture, standards related to business concepts modelling, 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 modellers, 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 (ModelSign), model domain (ModelDomainProfile) and model classifier (ModelConcept), model component (ModelInsatnce) and selected model element (ModelSelection).

Table 4 shows an example of different registers and users against the registry conforming this standard. Supposed organizations and users who register metamodels or patterns for the upper layer and modelling artefacts or products for the lower layer respectively, are listed.

Table 4- Typical Developer/Register and User (Informative)

Setting Layer	Metamodel Elements	Developer/Register	User
Global Standard	**ModelSign**	Standardization organization	Standard maker and developer of patterns and stereotypes
	ModelConcept and ModelDomainProfile (upper model)	Standardization organization	Standard maker and developer of patterns and stereotypes
	ModelInstances (lower model)	Standardization organization and developer of patterns and stereotypes	Industry standard maker
	ModelSelection	Organization for developing industry standard model	Developer of modelling artefacts conforming standard model
Localized Standard	**ModelSign**	Organization for developing industry standard model	Developer of modelling artefacts conforming standard model
	ModelConcept and ModelDomainProfile (upper model)	Organization for developing industry standard model	Developer of modelling artefacts conforming standard model
	ModelInstances (lower model)	Enterprise of developing products conforming standard model.	Vendor of modelling artefacts or products
	ModelSelection	(Not public. Should be managed by within each enterprise)	(Within each enterprise)

Annex A: MDR Model (Informative)

The metamodel for a metadata registry is specified in ISO/IEC 11179-3 Metadata Registries (MDR). Data modelling 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 modelling.

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 ModelSign, ModelConcept (including ModelDomainProfile), ModelInstance and ModelSelection described in 4.3.

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 5.

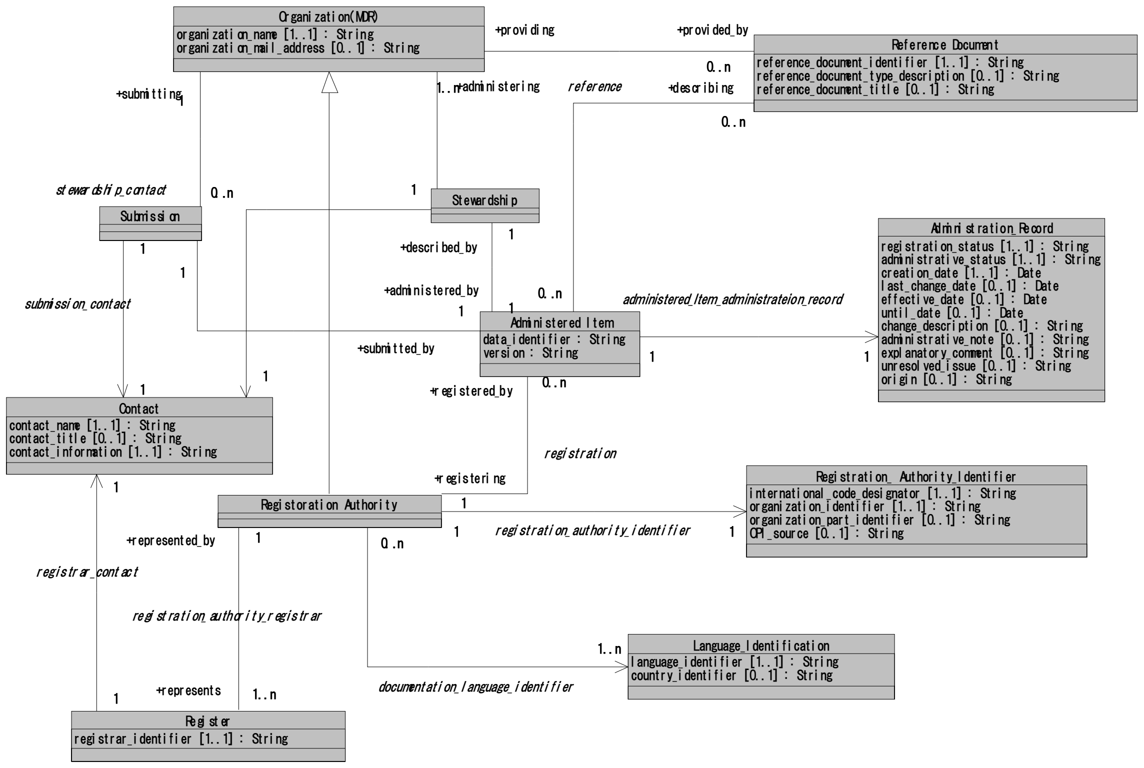
Organization(MDR)
organization_name [1..1] : String
organization_mail_address [0..1] : String
+providing
+provided_by
Reference Document
reference_document_identifier [1..1] : String
reference_document_type_description [0..1] : String
reference_document_title [0..1] : String
0..n
+submitting
1
1..n
+administering
reference
+describing
0..n
stewardship_contact
0..n
1
Stewardship
Submission
Administration_Record
registration_status [1..1] : String
administrative_status [1..1] : String
creation_date [1..1] : Date
last_change_date [0..1] : Date
effective_date [0..1] : Date
until_date [0..1] : Date
change_description [0..1] : String
administrative_note [0..1] : String
explanatory_comment [0..1] : String
unresolved_issue [0..1] : String
origin [0..1] : String
+described_by
+administered_by
submission_contact
0..n
administered_Item_administrateion_record
Administered Item
data_identifier : String
version : String
+submitted_by
+registered_by
0..n
Contact
contact_name [1..1] : String
contact_title [0..1] : String
contact_information [1..1] : String
registration
+registering
Registration_ Authority_Identifier
international_code_designator [1..1] : String
organization_identifier [1..1] : String
organization_part_identifier [0..1] : String
OPI_source [0..1] : String
Registoration Authority
+represented_by
0..n
registration_authority_identifier
registrar_contact
registration_authority_registrar
Language_Identification
language_identifier [1..1] : String
country_identifier [0..1] : String
1..n
documentation_language_identifier
+represents
1..n
Register
registrar_identifier [1..1] : String

Figure 5- 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 6 shows the overview of MOF Model Package. The operations of the metaclasses are not shown in this diagram. In this standard, Package, ModelElement and Namespace are used as external reference elements. NamedElement inherited from Namespace is defined as an external reference element.

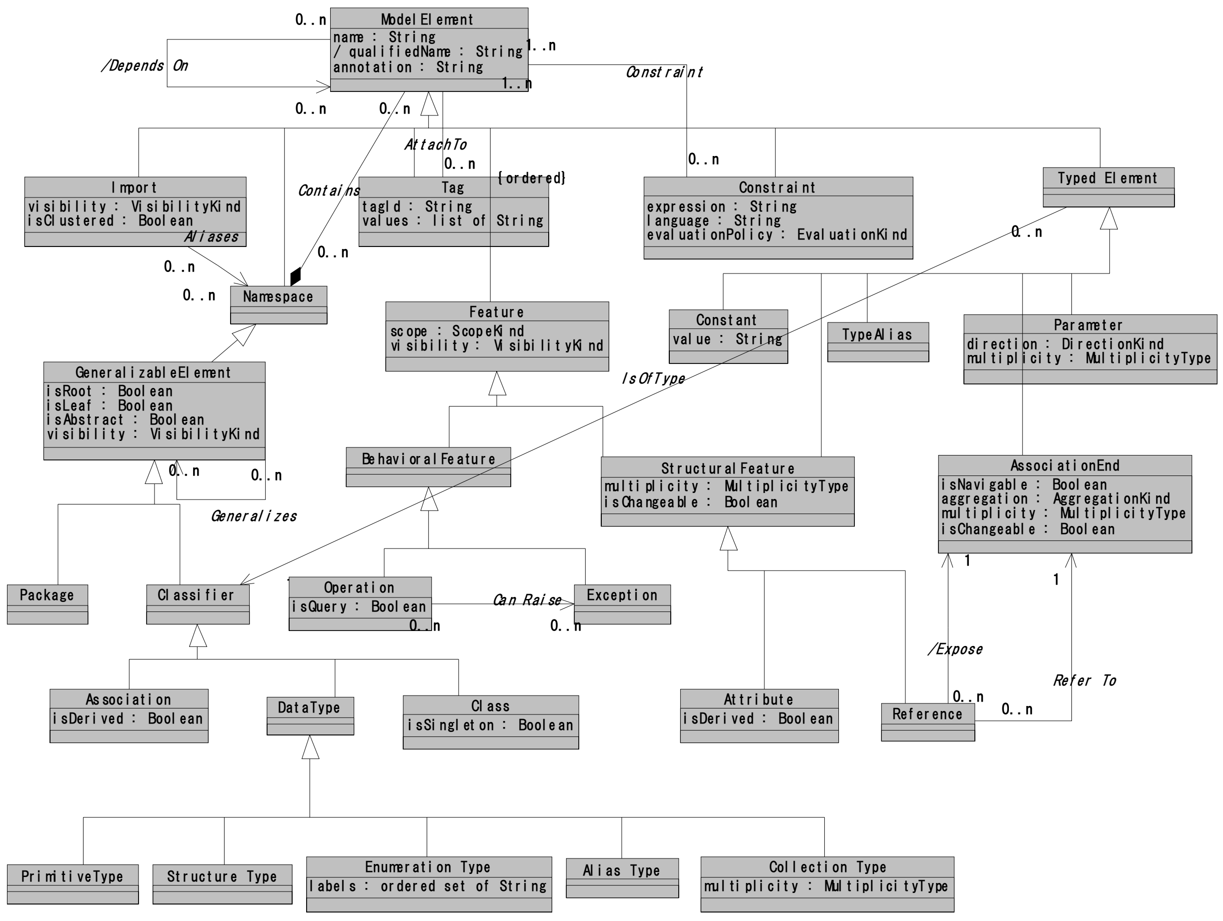

Figure 6- MOF Model Package

B.1 NamedElement

NamedElement is a metaclass identifying Model Element with Namespace.

(1) Superclasses

Namespace (from MOF)

(2) Attributes

name[MOF] : string [1..1]

Provides a meta-modeller 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.

(3) Constraints

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.

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.

Since the Contains Association is a Composite Association, any ModelElement can have at most one container, and the containment graph is strictly tree shaped.

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.

Annex C: ModelClassifier (Informative)

This annex of this document specifies the subclasses of ModelClassifier as examples.

A metamodel provides the specification of a model concept domain. The ModelClassifier including concrete stereotypes and patterns conforming to the metamodel are used to build a concrete model.

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

To improve shareability and reusability of object models, normative modelling constructs such as stereotypes and patterns must support the following features:

1. The model must consist of predefined *normative modelling constructs*, not only with modelling methods and also notations.
2. Predefined modelling 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 modelling 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 modelling 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.

Figure 7 shows the overview of Classifier Package. Some of metaclasses in this diagram are described as below. However, more precise specification should be defined for each registered objects as a profile.

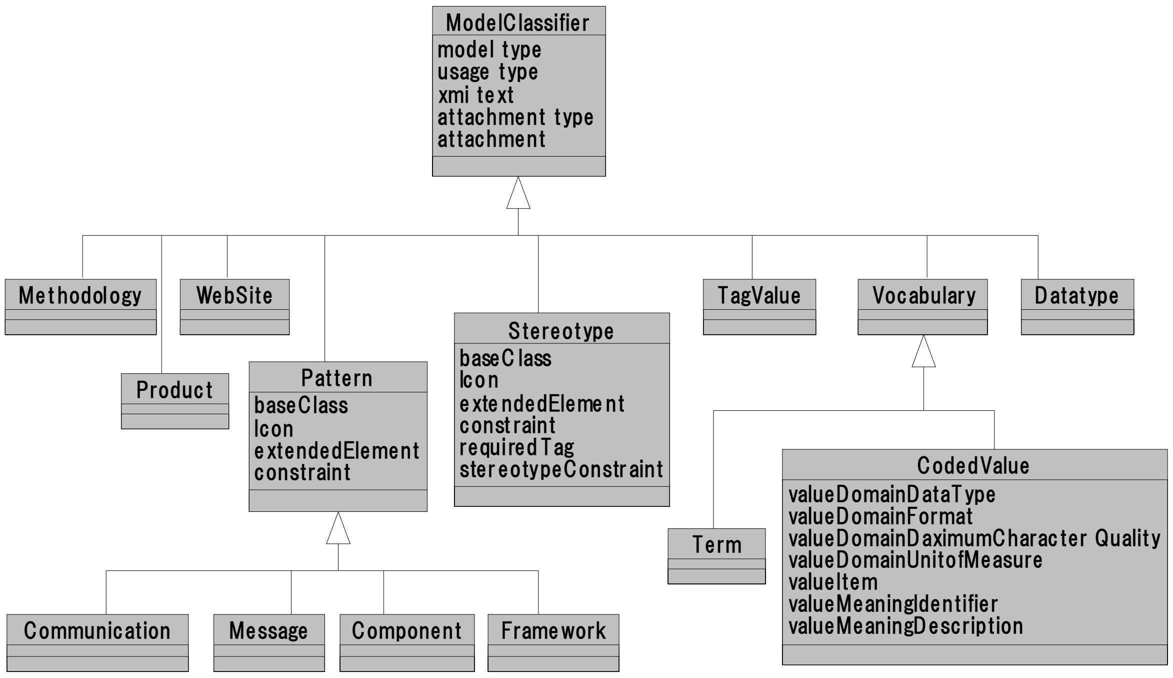

Figure 7- ModelClassifier Package

C.1 Stereotype

Stereotype, which is one of ModelClassifiers, is a metaclass designating stereotyped model elements. The stereotype is defined and declared in the metamodel to extend and restrict the meaning of existing model elements. The instance of Stereotype is an object declared as a specific stereotype.

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.

(1) Superclasses

ModelClassifier

(2) Attributes

baseClass[UML] : Class Name [1..1]

Icon[UML] : Geometry [0..1]

extendedElement[UML] : ModelElementName [0..*]

constraint[UML] : string [0..1]

requiredTag[UML] : string [0..*]

stereotypeConstraint[UML] : string [0..1]

C.2 CodedValue

The CodedValue is a metaclass designating coded values as a ModelConstructs. A CodedValue can be specified against the datatype having coded values for the ModelDomainProfile and ModelInstances.

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

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 ModelDomainProfile and their coded values and meanings may be registered in the MDR framework including permissible values and value meanings.

(1) Superclasses

ModelClassifier

(2) Attributes

valueDomainDataType : datatype [1..1]

valueDomainFormat : string [0..*]

valueDomainMaximumCharacter : Quality integer [0..1]

valueDomainUnitofMeasure : Unit of measure [0..1]

valueItem : string [1..1]

valueMeaningIdentifier : string [1..1]

valueMeaningDescription : string [0..1]

C.3 Pattern

Pattern is a metaclass designating a pattern mechanism. It is a facility to define patterns as a model element and apply their patterns. 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.

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.

The expression derived from the abstract syntax of a metamodel may be defined as a pattern.

ModelPattern inherits indirectly Package and Classifier from MOF.

(1) Superclasses

ModelClassifier

(2) Attributes

baseClass[UML] : Class Name [1..1]

Icon[UML] : Geometry [0..1]

extendedElement[UML] : ModelElementName [0..*]

constraint[UML] : string [0..1]

requiredTag[UML] : string[0..*]

stereotypeConstraint[UML] : string [0..1]

C.4 Communication

Communication is a metaclass designating collaboration modelling using the pattern mechanism. Communication provides a facility to build composed and nested collaborations based on Pattern. Each part of layered collaborations may be standardized if necessary.

In MOF, there is no metaclass corresponding to “Communication”. However, the metaclass “Communication” inherits indirectly the metaclass “Package” from MOF.

(1) Superclasses

ModelClassifier

(2) Attributes

targetSystemCollaboration : string [1..1]

C.5 Component

Component is a metaclass designating component modelling using the pattern mechanism.

ModelComponent provides a facility to build composed and nested components based on Pattern. Each part of layered components may be standardized if necessary. Actually, the operations attached to the pattern may be connected in assembling the components.

In MOF, there is no metaclass corresponding to “Component”. However, the metaclass “Component” inherits indirectly the metaclass “Package” from MOF.

(1) Superclasses

ModelClassifier

(2) Attributes

targetSystemComponent : **string [1..1]**

C.6 Framework

Framework is a metaclass designating framework modelling using the pattern mechanism.

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.

In MOF, there is no metaclass corresponding to “ModelFramework”. However, the metaclass “ModelFramework” inherits indirectly the metaclass “Package” from MOF.

(1) Superclasses

ModelClassifier

(2) Attributes

targetSystemFramework : **String[1..1]**

Annex D: Level Pair (Informative)

This annex 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)

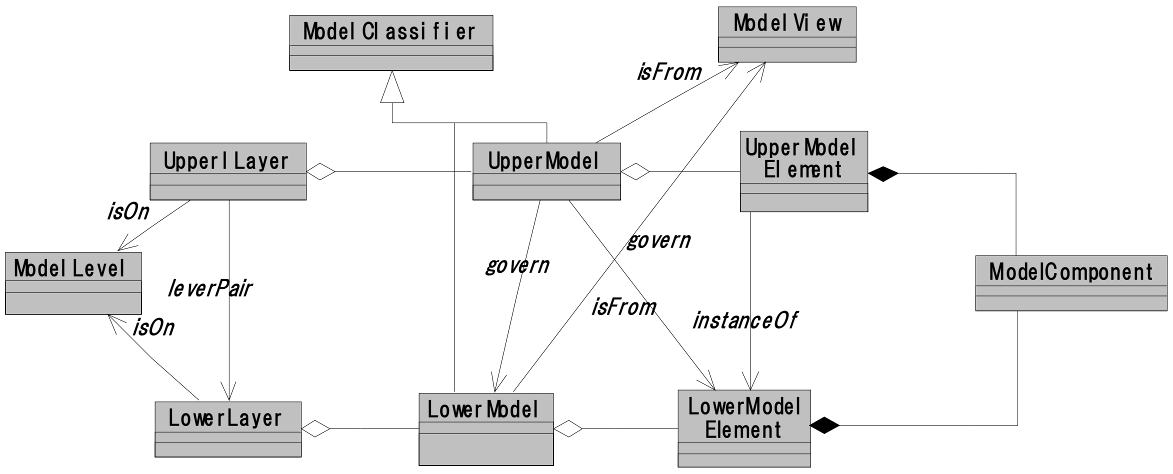

Figure 8- Level Pair Package

D.1 UpperLayer

UpperlLayer 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.

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.

(1) Superclasses

None

(2) Attributes

isOn ModelLevel :

levelPair LowerLayer :

D.2 UpperModelElement

The UpperModelElement is a metaclass designating composite elements of an UpperModel. An UpperModelElement forms assembling ModelSelections. An UpperModel also forms aggregation of UpperModelElement.

The UpperModelElement consists of model elements by ModelComponent and metamodel constructs in MOF.

(1) Superclasses None

(2) Attributes

(3) Constraints

A LowerModelElement should be an instance of the corresponding UpperModelElement.

D.3 LowerLayer

LowerLayer is a metaclass designating a model level layer in the metadata architecture. A LowerLayer has links to a ModelContext of modelling target and a ModelConcept of model elements namespace.

LowerLayer is corresponding to meta level layer in MOF metadata architecture.

(1) Superclasses None

(2) Attributes

metaLevelIdentifier : String [1..1]

basedOn : Model Context [1..1]

isOn : Model Concept [1..1]

D.4 LowerModel

LowerModel is a metaclass identifying model or metamodel at LowerrLayer A LowerModel may be defiened from various ModelViews. Links among LowerModels may be connected according to associations and references.

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.

(1) Superclasses

ModelClassifier

(2) Attributes

ModelAssociation : [0..*]

(3) Constraints

A LowerModel should be associated by corresponding UpperModel.

LowerModel should be described with conforming to abstract syntax defined by UpperModel under additional constraints.

D.5 LowerModelElement

The LowerModelElement is a metaclass designating composite elements of ModelSelection. LowerModelElement plays a role of assembling ModelSelections.

The LowerModelElement consists of ModelSelections and metamodel constructs in MOF.

(1) Superclasses None

(2) Attributes

(3) Constraints

LowerModelElement should be an instance of corresponding UpperModelElement.

D.6 ModelView

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.

(1) Superclasses None

(2) Attributes

viewPoint : string

scope : String [0..1]

purpose : Sring [0..1]

D.7 ModelLevel

MetaLevel is a metaclass designating a meta level in the metadata architecture. MetaLevel has links to a ModelContext of modelling target and a ModelConcept of model elements namespace.

The metaclasss “MetaLevel” is corresponding to meta level layer in MOF metadata architecture.

(1) Superclasses None

(2) Attributes

metaLevelIdentifier : string [1..1]

(3) Referencs

basedOn: Model Context

isOn : Model Concept

Annex E: Target Structure (Informative)

In the MMF registry, a concept will be registered with a name and the definition of its meaning. Each component of the instances also can be registered as a lower (narrower) concept.

Figure 9 shows the basic framework of registered target objects. In the definition of the meaning, already registered concepts and those instances may be used by linking to a ModelSelection. In addition, ModelComponent may include external references of ModelSelections. The associations among model classifier and lower model could be specified as an association type. Also, the relationship among model classifiers needs to be described definitely. Actually, a ModelDomainProfile may include a set of concepts defined within the ModelConcept. If necessary, corresponding ModelSign, ModelClassifier and ModelInstances for each concept may be registered.

Figure 10 shows the like actual example about “Vehicle”. In a component (of ModelComponent), which is pointed by corresponding the particular concept (of ModelConcept) and instance (of ModelInstances), may include particular classifiers (of ModelClassifier) and selections (of ModelSelection). A ModelSelection has a role to select some appropriate pair of values of sign (of ModelSign) and the instance (of ModelInstances). It suggests that alternative value can be used to compose own models if possible. Using ModelSlection, the multi-component structure can be expressed as a registered target object.

Figure 11 shows an example of registered target objects connected with ModelSelections. LowerModel, which is associated with UpperModel, may be registered itself as another ModelSign in UpperModel Layer.

Figure 12 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 concepts 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 ModelInstances.

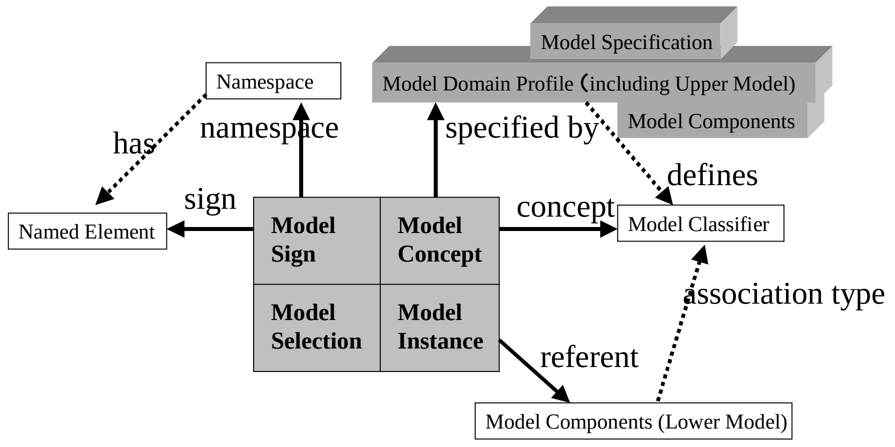

Figure 9- Basic Relationship of Registered Target Objects

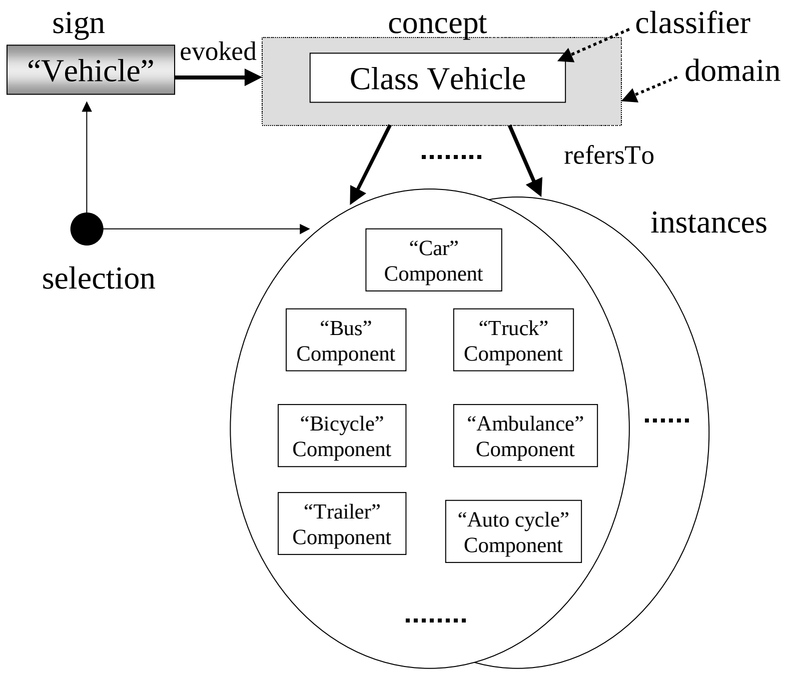

Figure 10- Registered Target Objects about "Vehicle"

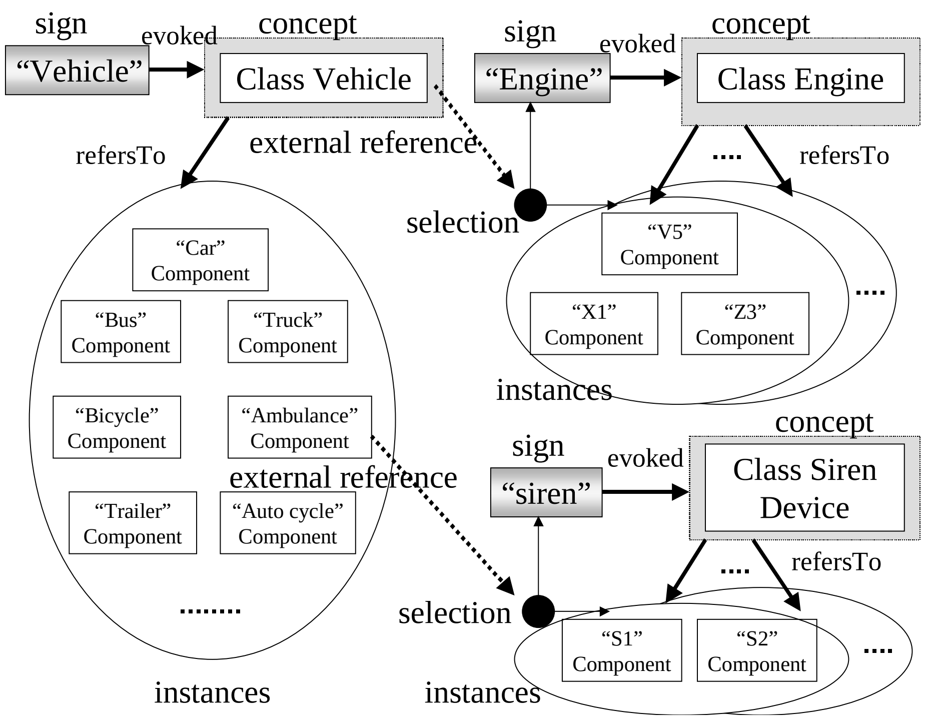

Figure 11- Registered Target connected with Selection

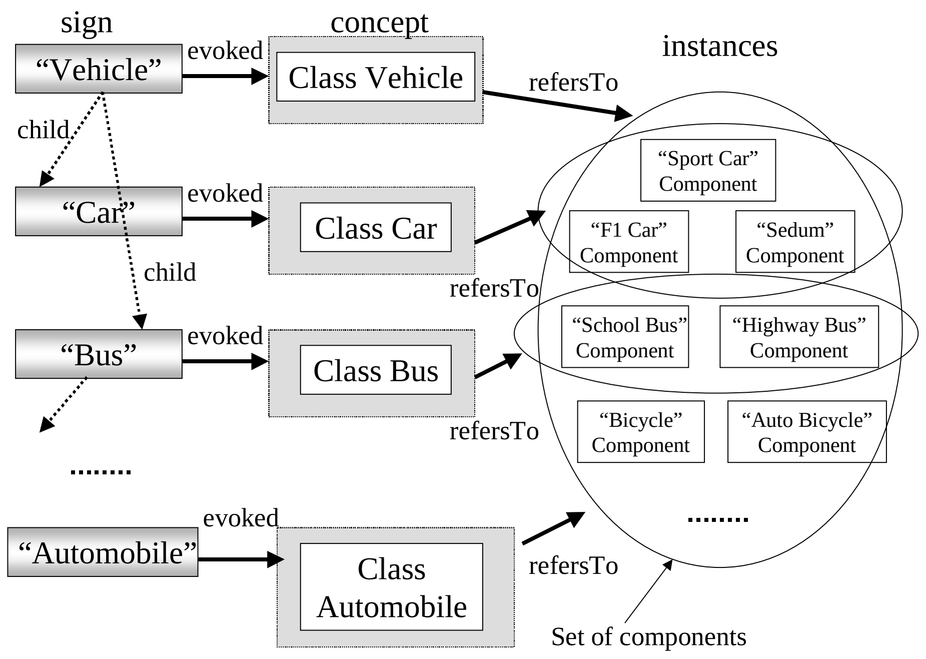

Figure 12- Layered Target Objects with muti-layered registration

Annex F: Examples (Informative)

This annex of this document provides some more examples of the registered target objects such as “Country”, “Asian Country”, “Country Group” and “Political Boundary Model”.

Examples of association types are shown as follows:

-Example1: Association Type “Type-Instance” (see Figure 15)

-Example2: Association Type “Type-Instance” (see Figure 16)

-Example3: Association Type “Super-Sub”(see Figure 17)

-Example4: Association Type “Base-Variant” (see Figure 18, 19)

-Example5: Association Type “Abstract Syntax-Expression”(see Figure 20,21)

Also, Table 3 shows operations on Association type “Base-Variant”

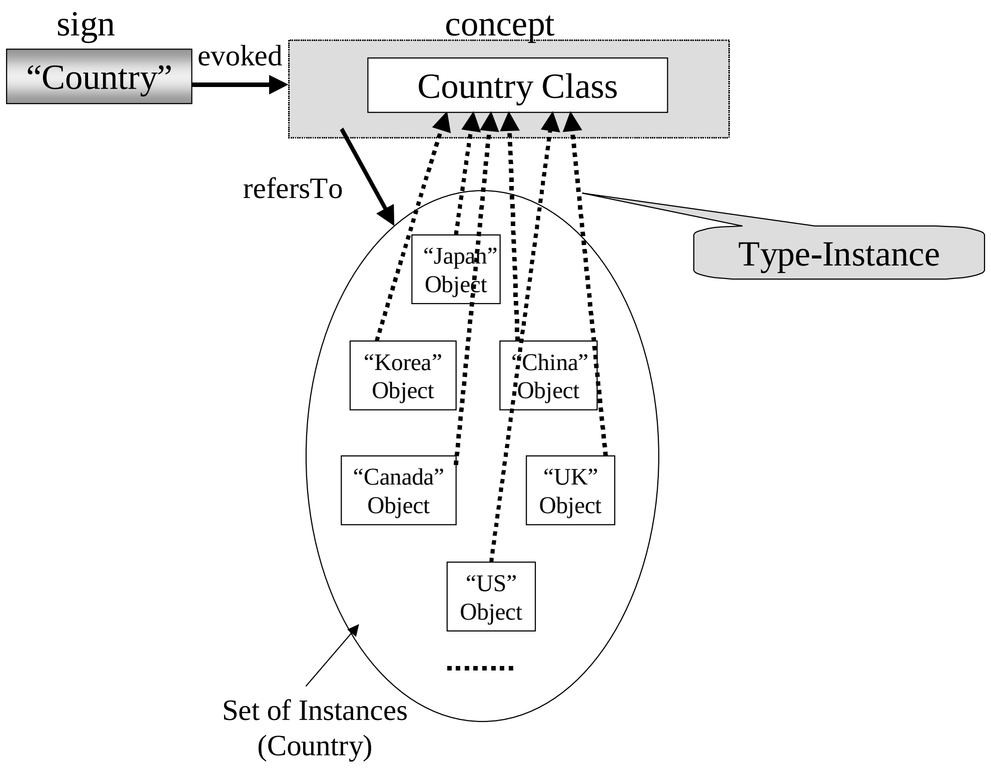

Figure 13- Example 1: Association Type “Type-Instance” (1)

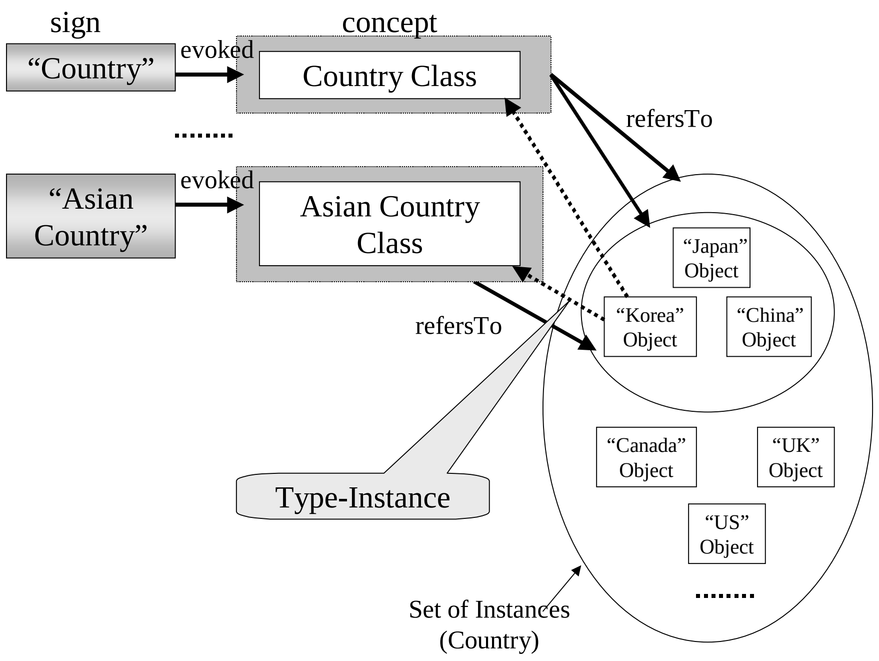

Figure 14- Example 2: Association Type "Type-Instance" (2)

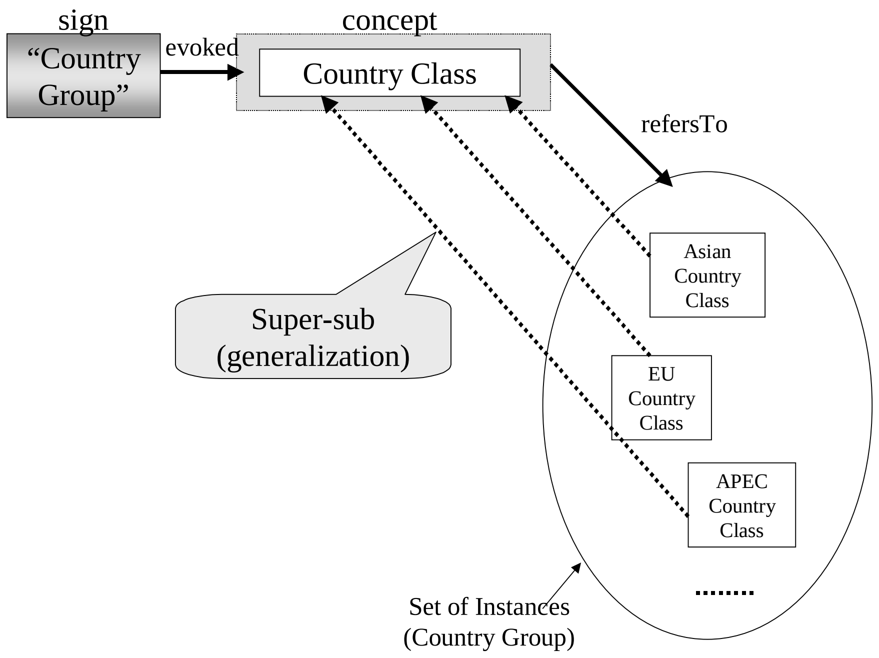

Figure 15- Example 3: Association Type "Super-Sub"

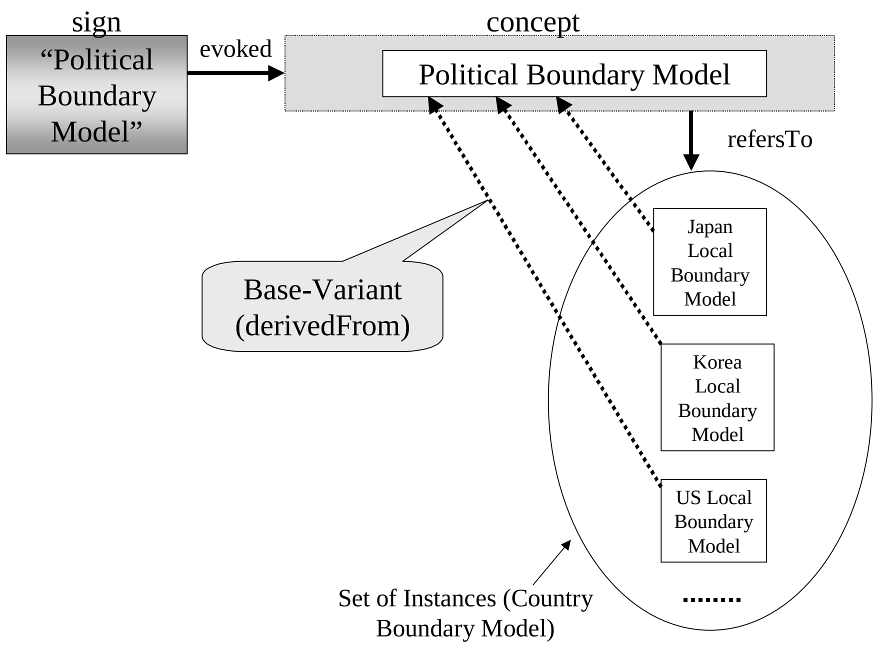
sign
"Political Boundary Model"
evoked
concept
Political Boundary Model
refersTo
Japan Local Boundary Model
Korea Local Boundary Model
US Local Boundary Model
....
Base-Variant (derivedFrom)
Set of Instances (Country Boundary Model)

Figure 16- Example 4: Association Type "Base-Variant" (1)

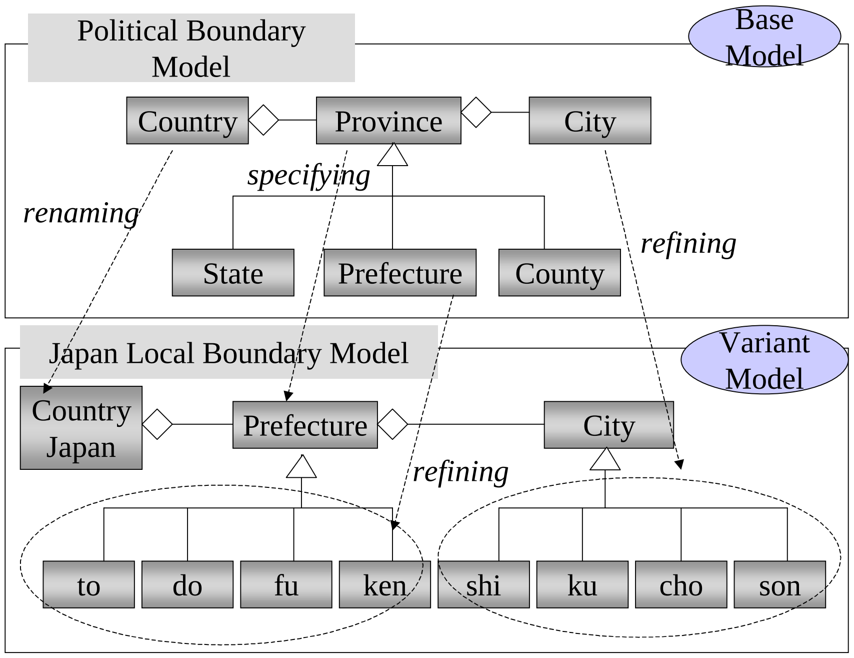

Figure 17- Example 4: Association Type "Base-Variant" (2)

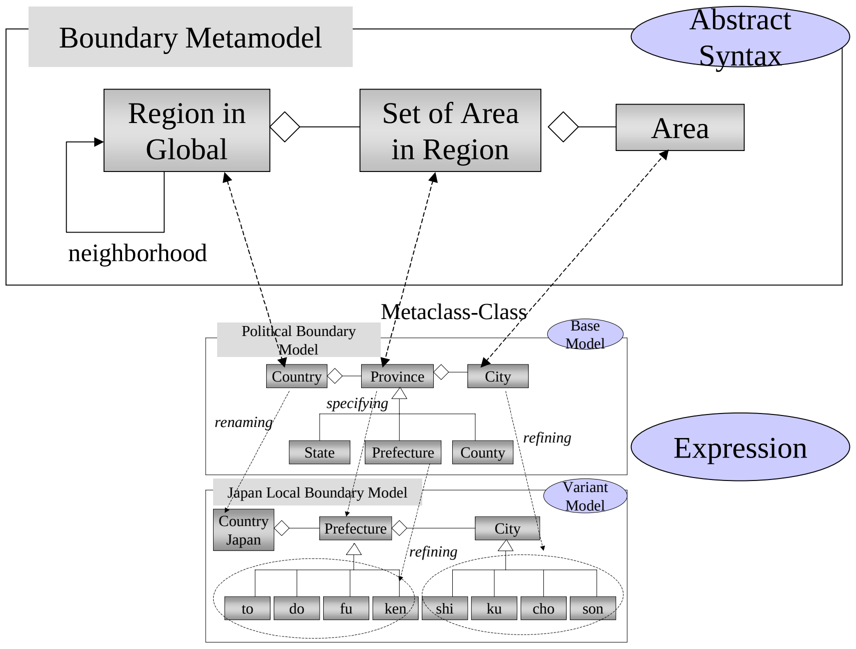

Figure 18- Example 5: Association Type "Abstract Syntax-Expression" (1)

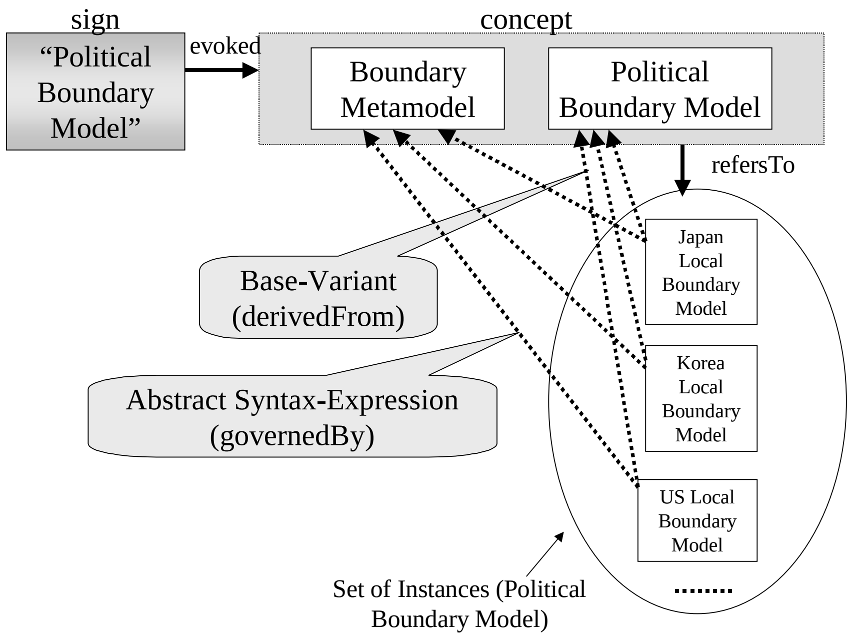
sign
"Political Boundary Model"
evoked
concept
Boundary Metamodel
Political Boundary Model
refersTo
Base-Variant (derivedFrom)
Abstract Syntax-Expression (governedBy)
Japan Local Boundary Model
Korea Local Boundary Model
US Local Boundary Model
....
Set of Instances (Political Boundary Model)

Figure 19- Example 5: Association Type "Abstract Syntax-Expression" (2)

Table 5- Operations on Association type "Base-Variant"

Operation	Description
Renaming	Change the name of such as class and attribute into the appropriate name in the context.
Specifying	Specify a model element of the base model within permissible range. For instance, -Fix the cardinality of an association end or limit the range. -Select a code set for an attribute or indicate permissible values. -Select effective attributes and remove unused ones. -Select and fix a subclass in a class hierarchy -Remove unused associations
Refining	Refining the model elements in a base model. For instance, -Add a subclasses that is enhanced with attributes and operations -Add a constraint that is applied in the particular context
Substituting	Substituting the model component in a base model. For instance, -Replace a universal component into the component used in a particular local region. -Replace the old version of a component into the new one.
Extending	Add a new class or association to a base model.
Merging	Combine and assemble base models into a new model.

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*

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