

SC32WG2 N1439

Working Draft for ISO/IEC PDTR 19763-4

ISO/IEC JTC 1/SC 32 WG2

Date: 2010-05-24

ISO/IEC FCD 19763-2:2007(E)-

ISO/IEC JTC 1/SC 32/WG 2

Secretariat:

**Information technology – Metamodel framework for interoperability (MFI) --
Part-4: Structured model registry**

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.

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

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

- Forewordv
- Introduction.....vi
- 1 Scope7
- 2 Conformance.....7
 - 2.1 Overview of conformance7
 - 2.2 Degree of conformance7
 - 2.3 Levels of conformance8
 - 2.4 Obligation8
 - 2.5 Implementation conformance statement (ICS)9
 - 2.6 Roles and responsibilities for registration.....9
- 3 Normative references9
- 4 Terms, definitions and abbreviated terms10
 - 4.1 Terms and definitions.....10
 - 4.2 UML and MOF terms used in specifying the MFI encapsulation approach on model registry10
 - 4.3 General terms used in this part of ISO/IEC 1976320
 - 4.4 Abbreviation and Acronyms25
- 5 Specification of the MFI Encapsulation approach on model registry26
 - 5.1 Overview.....26
 - 5.2 Registry package (Structure of registry).....29
 - 5.3 Model mapping package35
 - 5.4 Standard formats for interchanging models.....42
- Annex A (informative) Model Classifier43
 - A.1 General.....43
- Annex B (informative) Transformation Languages46
 - B1. General.....46
 - B2. Kinds of Transformation Languages46
- Annex C (informative) MFI Registry and Model Mapping48
 - C.1 General.....48
 - C.2 Use Case48
- Annex D (informative) Use Cases of Model Transformation49
 - D1. General.....49
 - D2. Examples49
- Bibliography.....55

Table of Figures

Figure 1- MFI Metadata Architecture and artifacts for registration.....	27
Figure 2- MFI Packages and Metaclasses	28
Figure 3- Registry package in MFI Model selection	30
Figure 4- Conceptualization in registered target artefacts	32
Figure 5- Metamodel-Model Transformation Combination	39
Figure 6- Metamodel-Model Transformation Detail (Projection type II)	40
Figure 7- Model-Value Transformation Combination.....	41
Figure 8- Model-Value Transformation Detail (Projection type I).....	41
Figure C1- An Environment of MFI Registry and Model Mapping	48
Figure D1- Transformation on HL7 (1)	50
Figure D2- Transformation on HL7 (2)	50
Figure D3- Transformation on ebXML (1)	51
Figure D4- Transformation on ebXML (2)	52
Figure D5- Transformation on CWM (1).....	53
Figure C6- Transformation on CWM (2).....	53
Figure D7- Transformation on XSLT	54

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/IE 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 19763 consists of the following parts, under the general title *Information technology* — **Metamodel framework for interoperability**:

- Part 1: Reference model
- Part 2: Core model and basic mapping
- Part 3: Metamodel for ontology registration
- Part 4: Structured model registry

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. People endeavor to standardize domain specific business process models and standard modelling constructs, such as data elements, entities, and value domains. The standardization efforts tend to be focused on the contents of a model or metamodel to represent and exchange the semantics of businesses, using UML and XML.

Metamodels and models can describe business concepts and components that should be shared for interoperability between systems. For example, considering message exchange in e-commerce, a business process model about interaction among parties could be considered as a metamodel or model.

Metamodels, have been progressed through standardization activities such as United Nations Centre for Trade facilitation and Electronic Business (UN/CEFACT), Organization for the Advancement of Structured Information Standards (OASIS), Object Management Group (OMG), and Health Level Seven (HL7). However, each standards group tends to specify their metamodel scheme in their own ways. Specification of a common basis for consistent development and registration of metamodels, should help to avoid duplications and inconsistencies that are otherwise likely to occur.

Specification of a common basis for consistent registration of metamodels, should help to map between metamodels that are developed independently. 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. It should also facilitate their reuse widely across organisations. This part of the MFI standard sets forth the core model (which may appropriately be considered a "metamodel") for the unified framework.

Information technology–Metamodel framework for interoperability (MFI) –Part 4: Structured model registry

1 Scope

This part of *ISO/IEC 19763* provides the way of model registration for encapsulation approach. The metamodel of this part specifies the metamodel for registering complex metamodels and models including relationship among them.

The standardization of business concepts and business models (or metamodels) to be registered for specific business domains is out of scope in this part of *ISO/IEC 19763*.

2 Conformance

2.1 Overview of conformance

This part of *ISO/IEC 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.

MFI conformance is specified along three orthogonal viewpoints; first is a value requirement viewpoint, second is interoperability viewpoint of metadata exchange, and third is conformance between registered content such as upper model and lower model.

- 1) Conformance on Value Requirement is specified in 2.3 Table 2
- 2) Conformance on Interchange Format is specified in 2.3 Table 2
- 3) Conformance on registered contents is not specified in this standard

A conforming implementation shall:

satisfy the requirements of 5.2, 5.3, 5.4, and 5.5 ;

identify a Degree of Conformance (2.2); and

identify a Level of Conformance (2.3).

2.2 Degree of conformance

The degree of conforming implementation is listed in Table1.

Table 1- Implementation Conformance Degree

	Extensions	Implementation	
		Strictly conforming	Conforming
a	are not directly specified by	shall support all mandatory and optional attributes and references	shall support all mandatory and optional attributes and references

	this part of ISO/IEC 19763		
b	are specified and agreed to the other parts of ISO/IEC 19763	shall not use, test, access, or probe for any extension features nor extensions to attributes	<u>as permitted by the implementation, may use, test, access, or probe for extension features or extensions to attributes</u>
c	may serve as trial usage for future editions of this part of ISO/IEC 19763	shall not recognize, nor act on, nor allow the production of attributes that are dependent on any unspecified, undefined, or implementation-defined behavior	may recognize, act on, or allow the production of attributes that are dependent on implementation-defined behavior

NOTE A ModelConcept governs the corresponding ModelComponentSet. Conformance statements between them are needed. Each registration authority should specify conformance statements on registered contents.

NOTE The use of extensions to the metamodel or the basic attributes may cause undefined behavior. All strictly conforming implementations are also conforming implementations.

2.3 Levels of conformance

An implementation may conform to either of two levels of conformance to this standard. The distinction between "Level 1" and "Level 2" 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.

Table 2- Level of Conformance

	Conformance View	Level of Conformance	
		Level 1	Level 2
1	Value Requirements	Only those metadata elements, attributes and references specified in Clause 5 are supported and used. No extensions are permitted.	All metadata elements, attributes and references specified in Clause 5 and <u>Annex A</u> are supported and may be used. Extensions are permitted.
2	Interchange Format	Only XML format is supported and used	Not specified.

NOTE A Level 1 implementation may be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 19763. A Level 2 conforming implementation may be more useful, but may be less interoperable with respect to this part of ISO/IEC 19763.

NOTE All format including bindings of ISO/IEC 20944 may be supported and be used. An implementation may claim conformance to both ISO/IEC19763 and parts of ISO/IEC 20944 if appropriate.

2.4 Obligation

One of two obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. Attributes and references specified in this part of ISO/IEC 19763 are stated to be Mandatory or Optional.

For the purpose of conformance:

a) Mandatory attributes and references shall exist, and shall conform to the provisions of this part of ISO/IEC 19763.

b) Optional attributes and references are not required to exist, but if they do exist they shall conform to the provisions of this part of ISO/IEC 19763.

Such obligation is enforced if and only if the Registration Status of the associated metadata items is Recorded or higher.

2.5 Implementation conformance statement (ICS)

An implementation-claiming conformance to this part of ISO/IEC 19763 shall include an implementation Conformance Statement stating:

- a) whether it conforms or strictly conforms (2.2);
- b) whether conformance is to Level 1, Level 2 (2.3) or both;
- c) what extensions are supported or used.

2.6 Roles and responsibilities for registration

Conformance needs to be considered in the context of the roles and responsibilities of registration authorities, as covered by ISO/IEC 11179-6: Registration of data elements. In order to exchange registry contents among MFI registries, conformance of systems may be extended concerning formalisation of procedures, agreement of roles and responsibilities between parties, and guidelines addressing use of software products and conversions from other systems. The formalisation of these aspects shall be consistent with the conformance requirements in the above Clauses, and roles of registration authorities as set out in ISO/IEC 11179-6.

3 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/IEC 11179-3:2003, Information technology – Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes

ISO/IEC 11179-6:2005, Information technology – Metadata registries (MDR) - Part 6: Registration

ISO/IEC 19501:2005, Information technology – Open Distributed Processing – Unified Modelling Language (UML)

ISO/IEC 19502:2005, Information technology – Meta Object Facility (MOF)

ISO/IEC 19503:2005, Information technology – XML Metadata Interchange (XMI)

4 Terms, definitions and abbreviated terms

4.1 Terms and definitions

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

NOTE 4.2 defines UML [ISO/IEC 19501:2005] and MOF [ISO/IEC 19502:2005] terms, used in specifying the MFI model. 4.3 defines terms, and their definitions, used in this document that are not included in 4.2

4.2 UML and MOF terms used in specifying the MFI encapsulation approach on model registry

4.2.1

abstraction

essential **characteristics** of an **entity** that distinguish it from all other kinds of **entities**

NOTE 1 An **abstraction** defines a boundary relative to the perspective of the viewer.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.2

abstract syntax

⟨MOF modelling⟩ **syntax** presented in a UML class diagram showing the **metaclasses** defining the constructs and their **relationships**

NOTE See ISO/IEC 19501:2005, 4.3.2.1.

4.2.3

artefact

physical piece of information that is used or produced by a software development process

NOTE 1 An **artefact** may constitute the implementation of a deployable **component**.

EXAMPLE Models, source files, scripts, and binary executable files.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.4

association

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.5

association end

endpoint of an **association**, which connects the **association** to a **classifier**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.6

attribute

⟨MOF modelling⟩ **feature** within a **classifier** that describes a range of values that **instances** of the **classifier** may hold

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.7**attribute**

⟨metamodel⟩ **characteristic** of an **object** or **entity**

[ISO/IEC 11179-3:2003 (3.1.3)]

4.2.8**binding**

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.9**cardinality**

number of **elements** in a set

cf. *multiplicity*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.10**class**

description of a set of **objects** that share the same **attributes**, **operations**, methods, **relationships**, and semantics

NOTE 1 A class may use a set of interfaces to specify collections of **operations** it provides to its environment.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.11**classifier**

mechanism that describes behavioral and structural **features**

NOTE 1 **Classifiers** include interfaces, **classes**, **datatypes**, and **components**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.12**classification**

assignment of an **object** to a **classifier**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.13**class diagram**

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.14**collaboration diagram**

diagram that shows interactions organized around the structure of a model, using either **classifiers** and **associations** or **instances** and **links**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.15 component

modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces

NOTE 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).

cf. *artefact*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.16 composition

form of **aggregation** that 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

NOTE 1 **Composition** may be recursive

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.17 constraint

semantic condition or restriction, certain **constraints** are predefined in the UML, others may be user defined.

cf. *tagged value, stereotype*.

NOTE 1 **Constraints** are one of three extensibility mechanisms in UML

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.18 container

⟨UML modelling data⟩ **instance** that exists to contain other **instances**, and that provides **operations** to access or iterate over its contents.

EXAMPLE arrays, lists, sets.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.19 container

⟨UML modelling component⟩ **component** that exists to contain other **components**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.20 context

view of a set of related modelling **elements** for a particular purpose, such as specifying an **operation**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.21 datatype

descriptor of a set of values that lack identity and whose **operations** do not have side effects

NOTE 1 **Datatypes** include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.22

dependency

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)

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.23

diagram

graphical presentation of a collection of **model elements**, most often rendered as a connected graph of arcs (**relationships**) and vertices (other model elements)

NOTE 1 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.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.24

domain

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.25

element

atomic constituent of a **model**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.26

feature

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.27

framework

⟨MOF modelling⟩ stereotyped **package** that contains **model elements**, which specify a reusable architecture for all or part of a system

cf. *pattern*.

NOTE 1 **Frameworks** typically include **classes**, **patterns** or **templates**. When **frameworks** are specialized for an application **domain**, they are sometimes referred to as application **frameworks**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.28**generalizable element**

model element that may participate in a **generalization relationship**

cf. generalization.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.29**generalization**

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

cf. inheritance.

NOTE 1 An instance of the more specific **element** may be used where the more general **element** is allowed.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.30**inheritance**

mechanism by which more specific **elements** incorporate structure and behavior of more general **elements** related by behavior

cf. generalization.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.31**instance**

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

cf. object.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.32**layer**

organization of **classifiers** or **packages** at the same level of **abstraction**, a **layer** represents a horizontal slice through architecture, whereas a **partition** represents a vertical slice

cf. partition.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.33**link**

semantic connection among a tuple of **objects**, an **instance** of an **association**

cf. association.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.34**link end**

instance of an **association end**

cf. association end.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.35**metaclass**

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

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.36**meta-metamodel**

model that defines the language for expressing a **metamodel**

NOTE 1 The **relationship** between a **meta-metamodel** and a **metamodel** is analogous to the **relationship** between a **metamodel** and a **model**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.37**metamodel**

⟨MOF modelling⟩ **model** that defines the language for expressing a **model**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.38**metaobject**

generic term for all metaentities in a metamodelling language

EXAMPLE Metatypes, metaclasses, metaattributes, and metaassociations.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.39**model**

abstraction of a physical system with a certain purpose

NOTE 1 In the context of the MOF specification, which describes a **meta-metamodel**, for brevity the **meta-metamodel** is frequently referred to as simply the **model**.

NOTE 2 In the context of the MOF modelling, which describes a **metamodel**, for brevity the **metamodel** is frequently referred to as simply the **model**.

NOTE 3 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.40**model element**

element that is an **abstraction** drawn from the system being modeled

cf. view element.

NOTE 1 In the MOF specification **model elements** are considered to be **metaobjects**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.41

Meta Object Facility

MOF

common facility to describe a **metamodel**

NOTE Adapted from ISO/IEC 19502:2005, 6.1.

4.2.42

MOF compliant

MOF based

described according to MOF **model**

NOTE Adapted from ISO/IEC 19502:2005, Annex A.

4.2.43

multiplicity

specification of the range of allowable cardinalities that a set may assume

cf. cardinality.

NOTE 1 **Multiplicity** specifications may be given for roles within **associations**, parts within composites, repetitions, and other purposes.

NOTE 2 Essentially a **multiplicity** is a (possibly infinite) subset of the nonnegative integers.

NOTE 3 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.44

name (of model element)

⟨MOF modelling⟩ string used to identify a **model element**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.45

namespace

⟨MOF modelling⟩ part of the **model** in which the **names** may be defined and used

cf. name.

NOTE 1 Within a **namespace**, each name has a unique meaning.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.46

navigable

ability for objects of a Class at one end of an Association to retrieve Objects from the other end.

NOTE **Associations** need not be navigable.

4.2.47

object

⟨MOF modelling⟩ entity with a well-defined boundary and identity that encapsulates state and behaviour

cf. class, instance.

NOTE 1 State is represented by **attributes** and **relationships**, and behavior is represented by **operations**, **methods**, and state machines.

NOTE 2 An object is an **instance** of a **class**.

NOTE 3 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.48

operation

service that can be requested from an object to effect behavior

NOTE 1 An **operation** has a signature, which may restrict the **actual parameters** that are possible.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.49

package

general-purpose mechanism for organizing **elements** into groups

NOTE 1 **Packages** may be nested within other **packages**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.50

parameterized element

template

descriptor for a **class** with one or more unbound **parameters**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.51

partition

set of related **classifiers** or **packages** at the same level of **abstraction** or across **layers** in a layered architecture

cf. *layer*.

NOTE 1 A partition represents a vertical slice through architecture, whereas a **layer** represents a horizontal slice.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.52

pattern

template collaboration

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.53

profile

profile may specify **model** libraries on which it depends and the **metamodel** subset that it extends

NOTE 1 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.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.54**property**

⟨MOF modelling⟩ named value denoting a **characteristic** of an **element**, a **property** has semantic impact

NOTE 1 Certain **properties** are predefined in the UML; others may be user defined.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.55**reference****pointer**

⟨MOF modelling⟩ named slot within a **classifier** that facilitates navigation to other **classifiers**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.56**relationship**

semantic connection among **model elements**

EXAMPLE Include associations and generalizations

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.57**repository**

facility for storing **object models**, **interfaces**, and implementations

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.58**role**

named specific behavior of an entity participating in a particular context

NOTE 1 A role may be static (e.g., an **association end**) or dynamic (e.g., a collaboration role).

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.59**stereotype**

new type of **modelling element** that extends the semantics of the **metamodel**

cf. constraint, tagged value.

NOTE 1 **Stereotypes** must be based on certain existing **types** or **classes** in the **metamodel**.

NOTE 2 **Stereotypes** may extend the semantics, but not the structure of pre-existing types and classes.

NOTE 3 Certain **stereotypes** are predefined in the UML, others may be user defined.

NOTE 4 **Stereotypes** are one of three extensibility mechanisms in UML.

NOTE 5 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.60**subclass**

specialization of another **class**; the **superclass** in a **generalization relationship**

cf. *generalization*.

cf. *superclass*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.61

subtype

specialization of another type; the **supertype** in a **generalization relationship**

cf. *generalization*.

cf. Contrast: *supertype*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.62

superclass

generalization of another **class**; the **subclass** in a **generalization relationship**

cf. *generalization*.

cf. *subclass*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.63

supertype

generalization of another **type**; the **subtype** in a **generalization relationship**

cf. *generalization*.

cf. Contrast: *subtype*.

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.64

tagged value

explicit definition of a **property** as a name-value pair, in a **tagged value**, the name is referred to as the tag

cf. *constraint*, *stereotype*.

NOTE 1 Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.65

type

stereotype of **class** that is used to specify a domain of **instances (objects)** together with the **operations** applicable to the **objects**

cf. *class*, *instance*.

NOTE 1 A **type** may not contain any **methods**.

NOTE 2 Adapted from ISO/IEC 19501:2005, Glossary.

4.2.66

view

projection of a **model**, which is seen from a given perspective or vantage point and omits **entities** that are not relevant to this perspective

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.67

view element

textual and/or graphical projection of a collection of **model elements**

NOTE Adapted from ISO/IEC 19501:2005, Glossary.

4.2.68

XML Metadata Interchange

XMI

XML Schema generated from MOF compliant model

NOTE Adapted from ISO/IEC 19503:2005, 1.

4.3 General terms used in this part of ISO/IEC 19763

4.3.1

Administered Item

registry item for which administrative information is recorded in an Administration Record

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

4.3.2

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)]

4.3.3

concept

unit of knowledge created or specified by a unique combination of **characteristics**

[ISO 1087-1:2000 (3.2.1)]

4.3.4

conceptual data model

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

[ISO/IEC 11179-3:2003 (3.2.8)]

4.3.5

data

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)]

4.3.6

data model

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

[ISO/IEC 11179-3:2003 (3.2.11)]

4.3.7

definition

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

[ISO 1087-1:2000 (3.3.1)]

4.3.8

designation

representation of a **concept** by a sign that denotes it

[ISO 1087-1:2000 (3.4.1)]

4.3.9

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)].

4.3.10

language

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

[ISO 5127:2001 (1.1.2.01)]

4.3.11

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 **optional** (4.3.36).

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

[ISO/IEC 11179-3:2003 (3.2.17)]

4.3.12

metadata

data that defines and describes other **data**

[ISO/IEC 11179-3:2003 (3.2.18)]

4.3.13
metadata item
 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 5 of ISO/IEC 19763-2.

EXAMPLE **Instances** of Model Concept, Model Domain Profile and Model Component Set etc.

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

[ISO/IEC 11179-3:2003 (3.2.19)]

4.3.14
metadata object
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 5 of ISO/IEC 19763-2.

EXAMPLE Model Concept, Model Domain Profile and ModelComponentSet etc.

[ISO/IEC 11179-3:2003 (3.2.20)]

4.3.15
metadata register
 information store or database maintained by a Metadata Registry

[ISO/IEC 11179-3:2003 (3.2.21)]

4.3.16
Metadata Registry
MDR
 information system for registering **metadata**

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

[ISO/IEC 11179-3:2003 (3.2.22)]

4.3.17
Model Association
ModelAssociation
 metaclass for specifying the association among Model Classifiers

4.3.18
Model Association End
ModelAssociationEnd
 metaclass for specifying the end of a Model Association

4.3.19
Model By MOF
ModelByMOF
 metaclass for specifying a model or metamodel described by MOF

4.3.20**Model Classifier****ModelClassifier**

metaclass for classifying a Model Concept

4.3.21**Model Component****ModelComponent**

metaclass for specifying a registered element

4.3.22**Model Component Set****ModelComponentSet**

metaclass for specifying the set of registered Model Components

4.3.23**Model Concept****ModelConcept**

metaclass for specifying the meaning of a particular concept

4.3.24**Model Domain Profile****ModelDomainProfile**

metaclass for specifying the context in which a Model Concept is defined

4.3.25**Model Instances****ModelInstances**

metaclass for specifying the set of registered Model Component Sets

4.3.26**Model Reference****ModelReference**

metaclass for specifying the information about a Model Association

4.3.27**Model Selection****ModelSelection**

metaclass for specifying the selected set of registered elements

4.3.28**Model Sign****ModelSign**

metaclass for designating a name in a namespace

4.3.29**Model Specification****ModelSpecification**

metaclass for specifying the document of a domain specification

4.3.30**modelling construct**

unit of notation for modelling, such as named elements in UML

4.3.31**Metamodel framework for interoperability****MFI**

framework for registering **artefacts** that are based on **metamodel** and **model**

4.3.32**name (of object)**

⟨metamodel⟩ **designation** of an **object** by a linguistic expression

NOTE See also **name (of Administered Item)** (4.3.33)

[ISO/IEC 11179-3:2003 (3.2.26)]

4.3.33**name (of Administered item)**

⟨administered item⟩ **name** by which an **Administered Item** is designated within a specific Context

NOTE 1 **Metamodel** construct is: Attribute of Designation.

NOTE 2 See also **name (of Object)** (4.3.32).

[ISO/IEC 11179-3:2003 (3.3.83)]

4.3.34**object**

⟨metamodel⟩ anything perceivable or conceivable

NOTE 1 **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).

NOTE 2 Adapted from ISO 1087-1:2000, 3.1.1.

4.3.35**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 **mandatory** (4.3.11).

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

[ISO/IEC 11179-3:2003 (3.2.28)]

4.3.36**Registration Authority**

Organization responsible for maintaining a register

[ISO/IEC 11179-3:2003 (3.3.121)]

4.3.37**registry item**

metadata item recorded in a Metadata Registry

[ISO/IEC 11179-3:2003 (3.2.29)]

4.3.38

registry metamodel

metamodel specifying a Metadata Registry

[ISO/IEC 11179-3:2003 (3.2.30)]

4.3.39

sign

designation of a defined **concept** in a special language or symbol

4.3.40

Uniform Resource Identifier

URI

formatted string that serves as an identifier for a resource, typically on the Internet

NOTE 1 The syntax is designed to meet the recommendations laid out in "Functional Recommendations for Internet Resource Locators" [RFC1736] and "Functional Requirements for Uniform Resource Names" [RFC1737].

NOTE 2 Adapted from IETF RFC 2396.

4.3.41

XML Schema

schema definition language for XML (Extensible Markup Language)

4.4 Abbreviation and Acronyms

DI Data Identifier

HL7 Health Level Seven

IRDI International Registration Data Identifier

MDR Metadata Registry

MFI Metamodel framework for interoperability

MOF Meta Object Facility

OASIS Organization for the Advancement of Structured Information Standards

OMG Object Management Group

QVT query, view and transformation

RAI Registration Authority Identifier

[UML](#) Unified Modelling Language

[UN/CEFACT](#) United Nations Centre for Trade Facilitation and electronic Business

URI Uniform Resource Identifier

VI Version Identifier

XML Extensible Markup Language

5 Specification of the MFI Structured model registry

5.1 Overview

This chapter describes the model that specifies the MFI (ISO/IEC 19763) encapsulation approach on model registry. The MFI encapsulation approach metamodel provides a set of modelling elements, including the rules for their use, with which to register models.

5.1.1 MFI Part 4 conceptual overview

5.1.1.1 Sharing metamodels and models

The MFI provides an infrastructure for sharing information that is required for establishing cooperation between companies in e-business and e-commerce. MFI facilitates sharing metamodels and models, which have been developed independently. The MFI utilizes model elements from ISO/IEC 19502 – Meta Object Facility (MOF). The MOF provides a set of modelling elements and the rules for their use to support development of metamodels.

5.1.1.2 MOF four layer metadata architecture

The MOF metadata architecture is based on the traditional four layer metadata architecture as described in ISO/IEC 19502:2005, which in turn was based on ISO 10027:1990. The layers can be described as:

- 1) The information (object) layer (M0) consists of the data that you want to describe.
- 2) The model layer (M1) is comprised of the metadata that describes data in the information layer. Metadata is informally aggregated as models.
- 3) The metamodel layer (M2) is comprised of the descriptions (i.e., meta-metadata) that define the structure and semantics of metadata. Meta-metadata is informally aggregated as metamodels. A metamodel is an “abstract language” for describing different kinds of data; that is, a language without a concrete syntax or notation.
- 4) The meta-metamodel layer (M3) is comprised of the description of the structure and semantics of meta-metadata. In other words, it is the “abstract language” for defining different kinds of metadata.

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

UML is a general-purpose modelling language and is independent from an object domain or implementation environment. A stereotype, a tag value and constraints may be defined as a profile to extend the UML language. A UML profile may be specified based on an existing metamodel by adding a new model element to that metamodel. In a domain model, described by using a UML profile, a stereotype to a model element should be attached to express the meaning.

Figure 1 shows that a model of a domain is expressed with a modelling construct (a target artefact for registration) based on the typical MOF metadata architecture. Any other metamodels described either using MOF or not using MOF can be placed independently within the MOF architecture. The main purpose of ISO/IEC 19763 is to achieve the sharing of common and useful modelling artefacts.

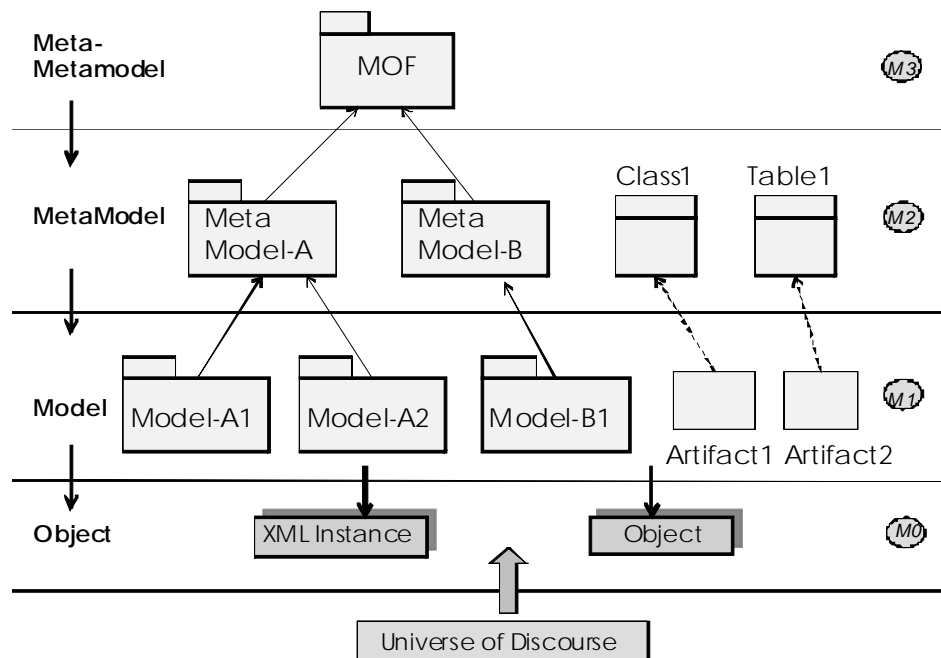


Figure 1- MFI Metadata Architecture and artifacts for registration

5.1.1.3 MFI specification at glance

This part of MFI uses a metamodel to describe the structure of an MFI's metadata registry. The MFI's registry metamodel is specified as a conceptual model, i.e. one that describes how relevant information is structured in the natural world. An implementation model is not specified in this part of ISO/IEC 19763. However, the implementation should be strictly derived from the MFI's metamodel to establish the common management of metamodels and their derived models.

In this standard, the purpose of specifying the framework according to the MOF Metadata Architecture layers is to help define the relationships and meaning among metamodel elements. The M3 layer of MOF metadata architecture is enhanced to provide the facility for registering those elements.

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

- MFI Part 2 Core and model mapping (see ISO/IEC 19763-2)
- MFI Part 3 Ontology registration (see ISO/IEC 19763-3)
- MFI Part 4 Encapsulation approach on model registry (see ISO/IEC 19763-4)

Figure 2 shows that the MFI Encapsulation approach package consists of three packages. The package and classes of MOFQVT and MDR, shown, as non-shaded packages and classes in Figure 2, 3, 4, 5, 6 and 8 are based on specifications of outside of this standard (see ISO/IEC 11179 and ISO/IEC 19502). The MFI part 2 model is also located within the MOF architecture as a metamodel conforming to MOF.

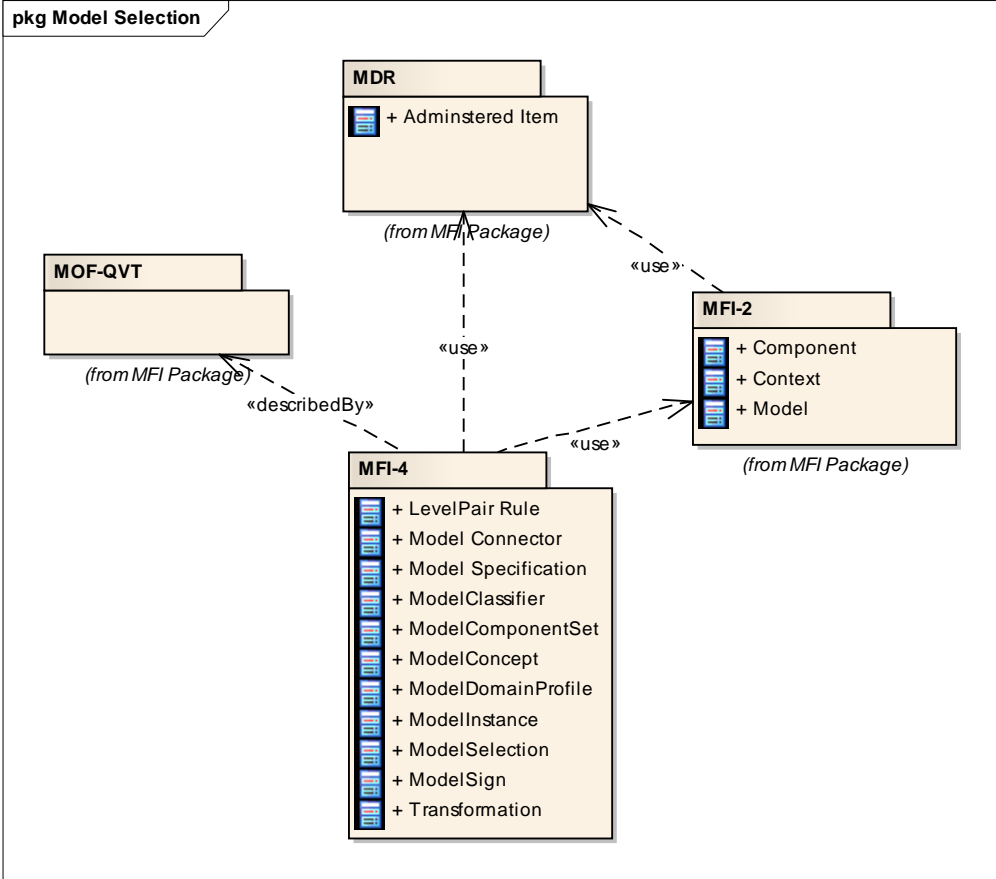


Figure 2- MFI Packages and Metaclasses

For descriptive purposes, the MFI encapsulation approach model is organized into two functional packages:

- Registry Package (normative; see Figure 6 and 5.4)
- ModelClassifier (informative; see Annex A)

5.1.2 Convention for Definition of MFI encapsulation approach metamodel

The MFI extendable registry model is specified using the MOF and Administered Items as defined in the MDR. The MOF metamodel constructs used include classes, relationships, association classes, attributes and references. These terms are defined in 4.2, and their model is described in Annex B.

The MFI core model is shown as a series of UML Class diagrams, with each Class being described as follows;

(1) Superclasses

immediate inherited classes

(2) Attributes

n. attribute name: datatype and multiplicity

-Use: *Mandatory* or *Optional* condition for attribute

-Description: description for content and purpose of attribute

(3) References

n. reference name : Class name and multiplicity

-Use: *Mandatory* or *Optional* condition for reference

-Description: description for content and purpose of reference

(4) Constraints

- reference name: *Class name and multiplicity* if exposed reference exists

- binding constraint: description about enforcement derived from a reference

-constraints specified if necessary, in natural language

The model shows minimum and maximum cardinality 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 as specified in ISO/IEC 11179-6. "Recorded" indicates that all mandatory attributes and references have been documented.

5.1.3 MFI Model selection

Figure 3 illustrates a high level overview of MFI core model. This part of ISO/IEC 19763 specifies the following types of Administered Items. The Administered Items shown in the figure are described in more detail later in this Clause.

-ModelInstances (see 5.4.2)

-ModelConcept (see 5.4.3)

-ModelSign (see 5.4.4)

-ModelSelection (see 5.4.5)

An instance of an Administered Item shall be globally identified as specified in ISO/IEC 11179-3 and ISO/IEC 11179-6. In this part of ISO/IEC 19763, identifiers for metaobjects shall be unique.

NOTE 1 The division of the model into packages is for descriptive purposes only and has no other significance. If any discrepancy exists in Clause 5 between the figures and the text, the text shall take precedence.

NOTE 2 The model diagrams in each of the packages only portray the relevant associations as required for that particular package.

5.2 Registry package (Structure of registry)

Figure 3 shows the Registry package of the MFI core metamodel.

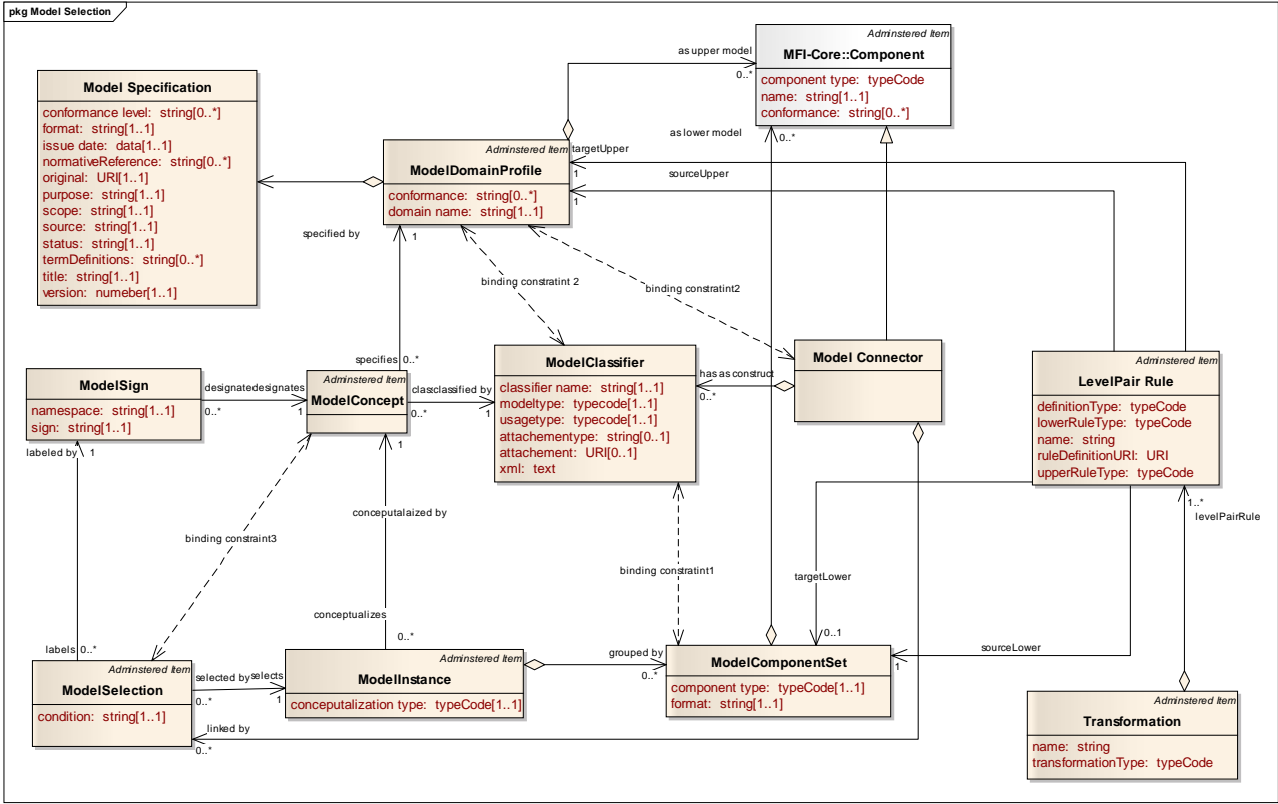


Figure 3- Registry package in MFI Structured model registry

5.2.1 ModelInstances

In the MFI model, ModelInstances is a metaclass that shall hold a conceptualization type. A ModelInstances shall be conceptualized by a ModelConcept and shall be grouped by ModelComponentSets.

A ModelInstances is a referent of ModelConcept designated by a ModelSign and classified by a ModelClassifier. A conceptualization type specifies the association between a ModelClassifier and ModelComponents in the ModelComponentSets of ModelInstances. A ModelInstances is an Administered Item.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

1. *conceptualization type*: typeCode[1..1]

-Use: Mandatory

-Description: A conceptualization type defines the type of association that exists between the contained ModelComponents of ModelComponentSet grouping ModelInstance and the ModelClassifier represented by the “conceptualized by” ModelConcept.

NOTE There are four pre-defined conceptualization types shown in Table 3 and Figure 7, but extensions are allowed (see also Level Pair in Annex D).

(3) References

1. grouped by: *ModelComponentSet[0..*]*

-Use: Mandatory

-Description: The ModelInstances that is grouped by ModelComponentSet.

2. conceptualized by: *ModelConcept[1..1]*

-Use: Mandatory

-Description: The ModelConcept that conceptualizes the ModelInstances.

(4) Constraints

-selected by: *ModelSelection[0..*]* is exposed non-navigable reference

-binding constraint 1: A ModelInstances may be conceptualized independently by a ModelConcept. The ModelComponent in the ModelComponentSets of a ModelInstances shall be enforced to be classified by the ModelClassifier.

Table 3- Code List of Conceptualization Types

Code	Name	Description
T-I	Type-Instance	“Type-Instance” is a conceptualization type between a type and its instance. Also, the class diagram in a model package and its object diagram may be included (see Figure7).
S-S	Super-Sub	“Super-Sub” is a conceptualization type between a super class and its inherited sub classes. Also a model package and its sub packages may be included (see Figure7).
B-V	Base-Variant	“Base-Variant” is a conceptualization type between a base model and its variant models that are created by modifying the base model according to the permitted operation (see Figure7). There are operations such as renaming, specifying, refining, substituting, extending and merging. See Table F1 in Annex F. In the conceptualization 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. Super-Sub is a special case of Base-Variant limit to a class.
A-E	Abstract Syntax-	“Abstract Syntax-Expression” is a conceptualization type between an upper metamodel and a lower model (see Figure7). In this case, a ModelByMOF specifying the ModelComponent in a ModelDomainProfile provides a

	Expression	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.
--	------------	--

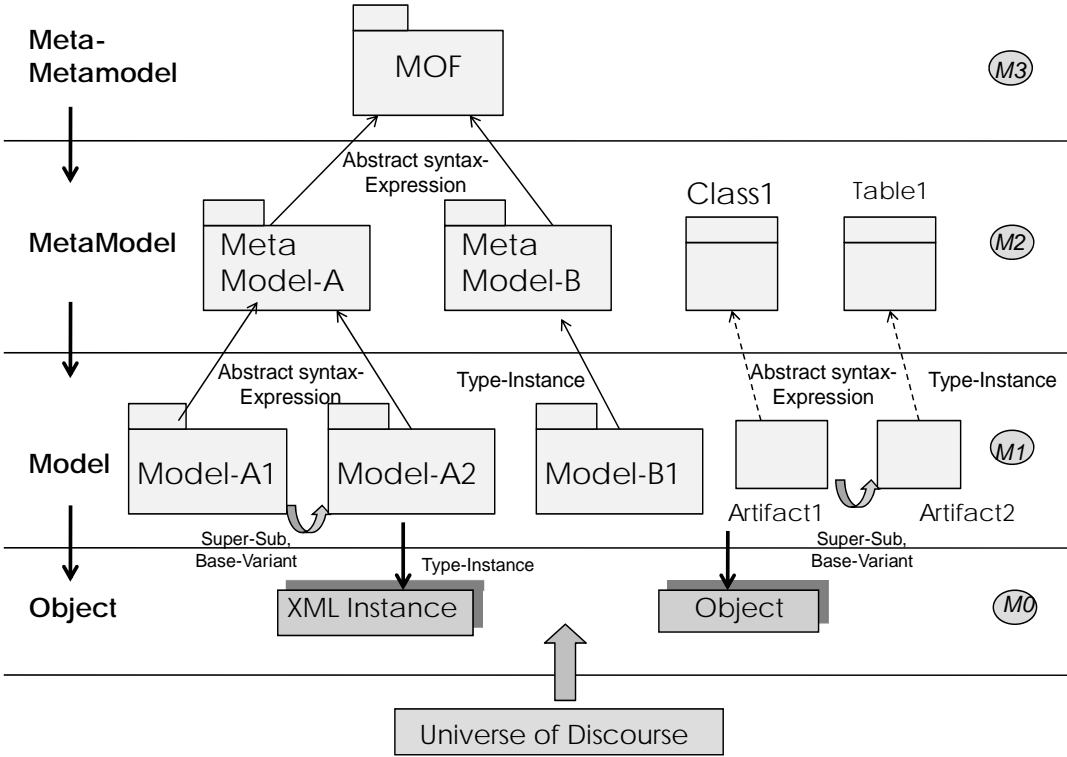


Figure 4- Conceptualization in registered target artefacts

5.2.2 ModelConcept

In the core MFI model, ModelConcept is a metaclass. A ModelConcept shall be specified by a ModelDomainProfile and shall be classified by a ModelClassifier. A ModelConcept is an Administered Item.

ModelConcept associates the concept that is classified by a ModelClassifier with the context that is specified by a ModelDomainProfile.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

none

(3) References

- 1. specified by: ModelDomainProfile[1..1]

-Use: Mandatory

-Description: The ModelDomainProfile that specifies the context for the ModelConcept.

2. classified by: ModelClassifier[1..1]

-Use: Mandatory

-Description: The ModelClassifier that classifies the ModelConcept.

(4) Constraints

-designated by: ModelSign[0..*] is exposed non-navigable reference.

-conceptualizes: ModelInstances[0..*] is exposed non-navigable reference.

-binding constraint 2: The ModelClassifier should be included in the ModelDomainProfile as a ModelClassifier that should belong to some ModelComponent of the ModelDomainProfile.

NOTE 1 A ModelClassifier and its code list should be specified in an MFI registry

NOTE 2 A ModelConcept should be managed and maintained by the Registration Authority within the user community.

5.2.3 ModelSign

In the core MFI model, ModelSign is a metaclass that shall hold a sign and a namespace where the sign shall be unique within the namespace. ModelSign instances each identify a thing of interest, such as a package, class, or association. A ModelSign shall designate the associated ModelConcept and may label multiple ModelSelections. A ModelSign is an Administered Item.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

1. namespace: string[1..1]

-Use: Mandatory

-Description: A string identifying the namespace where the sign is uniquely specified.

2. sign: string[1..1]

-Use: Mandatory

-Description: A string identifying the named element invoking the concept specified by the ModelSign.

(3) References

1. designates: ModelConcept[1..1]

-Use: Mandatory

-Description: The ModelConcept that is designated by the ModelSign.

(4) Constraints

-labels: *ModelSelection[0..*]* is exposed non-navigable reference.

-The namespace is to follow the rules of the particular environment (MOF, XML, etc) from which the namespace originates. When registering an XML element name, an XML namespace shall be used. When registering a MOF model element, the name of the package shall be used, but it shall be qualified with globally unique identifier such as URL.

NOTE A namespace should be managed and maintained by the Registration Authority within the user community.

5.2.4 ModelSelection

In the core MFI model, ModelSelection is a metaclass that may hold a condition. A ModelSelection shall select a modelInstances and shall be labelled by a ModelSign. A ModelSelection relates a ModelSign to a ModelInstances.

A subset of ModelInstances may be retrieved using the condition in ModelSelection from the ModelInstances. A ModelComponent may include other ModelSelections as a sub component by external reference. A ModelSelection is an Administered Item.

(1) Superclasses

Administered Item (from MDR)

(2) Attributes

1. condition: *string[0..1]*

-Use: Optional

-Description: A string specifying the filtering condition for the ModelInstances.

(3) References

1. labelled by: *ModelSign[1..1]*

-Use: Mandatory

-Description: The ModelSign that labels the set of ModelComponent selected by the ModelSelection.

2. selects: *ModelInstances[1..1]*

-Use: Mandatory

-Description: The ModelInstances that is selected by the ModelSelection.

(4) Constraints

-binding constraint 3: The ModelConcept designated by the ModelSign labelling the ModelSelection, shall be same as one conceptualized by its 'selected by' ModelInstances.

NOTE A ModelSelection should be managed and maintained by the Registration Authority within the user community.

5.3 Model mapping package

5.3.1 LevelPair Rule

LevelPair Rule is a metaclass designating a set of transformation rules and transformed model components such as ModelDomainProfile / ModelComponentSet. LevelPair Rules are applied to four model components including source upper / target upper for ModelDomainProfile and source lower / target lower for ModelComponentSet.

In general, transformations are uni-directional or bi-directional. Each direction defines a source and/or target of a transformation. A transformation can be executed in checking all constraints, or enforcing all derivations of that transformation in one chosen direction.

(1) Superclass

Administered Item (from MDR-ByMOF)

(2) Attributes

1. **name:** string[1..1]

-Use: Mandatory

-Description: An identifier of the transformation rule

2. **definitionType:** typeCode[1..1]

-Use: Mandatory

-Description: The type of language describing the transformation rule, it may be specified using types representing 'QVT-Relations', 'QVT-Core', 'QVT-Operational' and so on (see Table 3 and Annex A).

Table 3- Type of Definition

Type	Description
QVT-Relations	Defined by the declarative language of MOF QVT.
QVT-Core	Defined by the pattern matching language of MOF QVT.
QVT-Operational	Defined by the mechanism of invoking the operational mapping of MOF QVT.
QVT-Black-box	Defined by non-standard Black-box MOF operational implementation.
CWM	Defined by transformation framework of CWM.
XSLT	Defined by the XSL transformation language XSLT.

3. **upperRuleType:** typeCode[1..1]

-Use: Mandatory

-Description: The upperRuleType specifies the type of relationship between source upper and target upper model components. It may be specified using types for 'Relationship', 'Mapping', or 'Computation' (see Table 4 and 5).

4. **lowerRuleType:** typeCode[1..1]

-Use: Mandatory

-Description: The lowerRuleType specifies the type of the relationship between source lower and target lower model components. It may be specified using types for 'Relationship', 'Mapping', or 'Computation' (see Tables 4 and 5).

Table 4- Transformation Rule Types

Type	Source model		Target model		Function
	Element	Instance	Element	Instance	
Relationship	shall be specified	Pattern of Class / Association	shall be specified	Pattern of Class / Association	Constraint
Mapping	shall be specified	Pattern of Class / Association and Object	generated from source	Pattern of Class / Association and Object	Constraint/ Derivation
Computation	specified at execution time	Object (Value)	generated from source	Object (Value)	Derivation

Table 5- Transformation Operation Types

Type	Operation	Description
Relationship	Renaming	Change a named element of source model component into another name of target model component
	Equivalence	Check if named elements between the source and target model component are equal, return the true or false.
	Enhancement	Provide additional elements into the target model component
Mapping	Instantiation	Create instances of elements of the source model component into the target model component
	Generation	Compile the source model component and Issue codes for target model component
	Migration	Embed part of the source model component into the target model component
	Derivation	Extract elements of the source model component and build the target model component based on those elements
Computation	Aggregation	Apply arithmetic and logical operations for values of elements for the source model component and aggregate the results into values of the target model component
	Conversion	Convert values of data types of the source model component into appropriate ones of the target model component

5. ruleDefinitionURI: URI[1..1]

-Use: Mandatory

-Description: The URI of the file containing transformation rule definitions

(3) References

1. sourceUpper: ModelDomainProfile[1..1]

-Use: Mandatory

-Description: The packages of transformed source upper model component

2. targetUpper: ModelDomainProfile[1..1]

-Use: Mandatory

-Description: The packages of transformed target upper model component

3. sourceLower: ModelComponentSet[1..1]

-Use: Mandatory

-Description: The packages of transformed source lower model component

4. targetLower: ModelComponentSet[1..1]

-Use: Mandatory

-Description: The packages of transformed target lower model component

(4) Constraints**-Metamodel-Model Rule:**

The Metamodel-Model Rule is constraints of the metaclass LevelPair Rule between Metamodel level and Model level.

Metamodel-Model Rule provides constraints and derivations defined between upper/lower source and upper/lower target model components. The transformed lower model components are sets of objects, in MOF extents, of the types of the registered packages only. Metamodel-Model Rules are defined in the context of those model components.

-upperRuleType shall be the type of 'Relationship'

-lowerRuleType shall be the type of 'Mapping' or 'Relationship'

-sourceUpper shall be in Metamodel level, and sourceLower shall be in Model level

-targetUpper shall be in Metamodel level, and targetLower shall be in Model level

-Model-Value Rule:

The Model-Value Rule is constraints of the metaclass LevelPair Rule between Model level and Value level.

Model-Value Rule provides constraints and derivations defined between upper / lower source and upper / lower target model components. The transformed lower model components are sets of objects, in MOF extents, of the types of the not registered packages only. Model-Value Rules are defined in the context of those model components.

-upperRuleType shall be the code of 'Mapping' or 'Relationship'

-lowerRuleType shall be the code of 'Computation'

-sourceUpper shall be in Model level, and sourceLower may be in Model level

-targetUpper shall be in Model level, and targetLower may be in Model level

5.3.2 Transformation

Transformation is a subclass of AdministeredItem and a metaclass designating the transformation definition between source and target of ModelDomainProfile / ModelComponentSet.

(1) Superclass

AdministeredItem (from MDR-ByMOF)

(2) Attributes

1. **name:** string[1..1]

-**Use:** Mandatory

-**Description:** The identifier of transformation between ModelDomainProfile / ModelComponentSet

2. **transformationType:** typeCode[1..1]

-**Use:** Mandatory

-**Description:** The type of transformation, it may be specified using a code for 'Compilation type', 'Projection type I', 'Projection type II' or 'Transfer type' (see Table 6).

Table 6- Transformation Types

Type	Description
Compilation type	Compile the source into the target by generating
Projection type I	Project subset of the source on the target by generating
Projection type II	Project subset of the source on the target by matching
Transfer type	Transfer the modified source into the target by matching

(3) References

1. **levelPairRule:** LevelPair Rule[1..*]

-**Use:** Mandatory

-**Description:** The definition of mapping rule between source and target of model component for Metamodel-Model transformation or Model-Value transformation.

(4) Constraints

-**Metamodel-Model Transformation:**

Metamodel-Model Transformation is constraints of the metaclass Transformation between source and target model components.

-upperRuleType (Metamodel Level) and lowerRuleType (Model Level) of levelPairRule shall be specified using the types provided in Table 7 (see Figures 3 and 4).

-A mapping is part of one transformation and performs between ModelComponentSet. Each ModelComponentSet of a mapping has (is in) one direction of the transformation. When a user executes a transformation to derive a model, all mappings that are part of that transformation shall be executed, deriving the ModelComponentSet that are in one chosen direction.

Table 7- Transformation Combinations on Metamodel-Model Transformation

Type	Metamodel Level	Model Level	Value Level
Compilation type	Relationship	Mapping	N/A
Projection type I	Relationship	Mapping	N/A
Projection type II	Relationship	Relationship	N/A
Transfer type	Relationship	Relationship	N/A

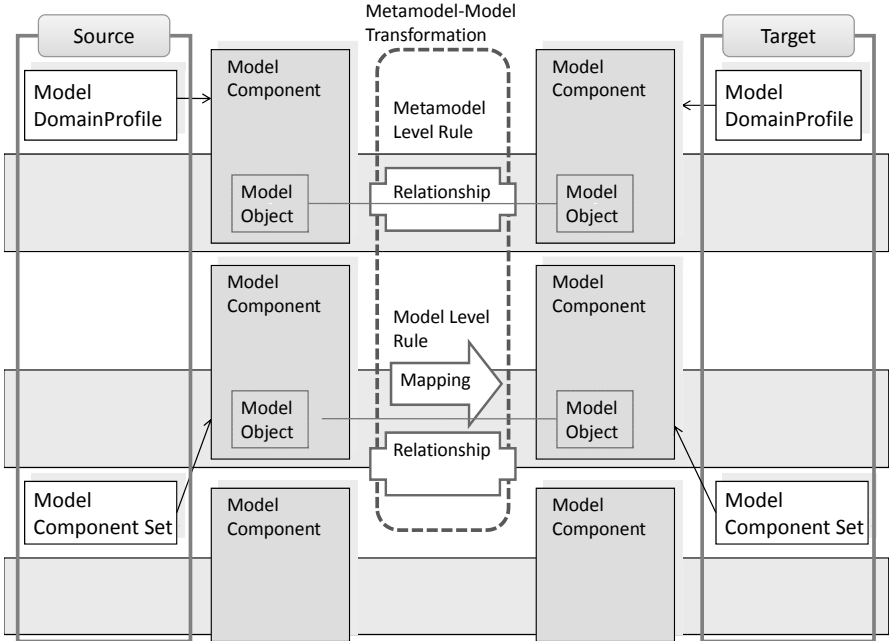


Figure 5- Metamodel-Model Transformation Combination

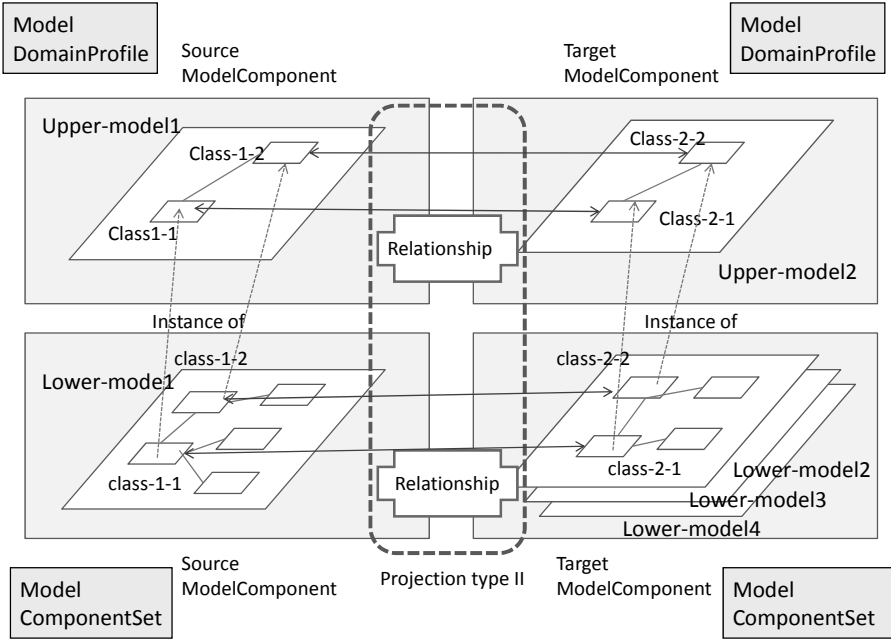


Figure 6- Metamodel-Model Transformation Detail (Projection type II)

-Model-Value Transformation:

Model-Value Transformation is constraints of the metaclass Transformation between source and target values.

-upperRuleType (Model Level) and lowerRuleType (Value Level) of levelPairRule shall be specified using the types provided in table 8 (see Figures 5 and 6).

-A mapping is part of one transformation and performs between ModelComponent of ModelComponentSet. Each ModelComponent of a mapping has (is in) one direction of the transformation. When a user executes a transformation to derive a model, all mappings that are part of that transformation shall be executed, deriving the ModelComponentSet that are in one chosen direction.

-A computation is part of one transformation and performs between ModelComponent (Values) of ModelComponentSet. Each Values of a computation has (is in) one direction of the transformation. When a user executes a transformation to derive a model, all computations that are part of that transformation shall be executed, deriving the Values that are in one chosen direction.

Table 8- Transformation Combinations on Model-Value Transformation

Type	Metamodel Level	Model Level	Value Level
Compilation type	N/A	Mapping	Computation
Projection type I	N/A	Mapping	Computation
Projection type II	N/A	Relationship	Computation

Transfer type	N/A	Relationship	Computation
---------------	-----	--------------	-------------

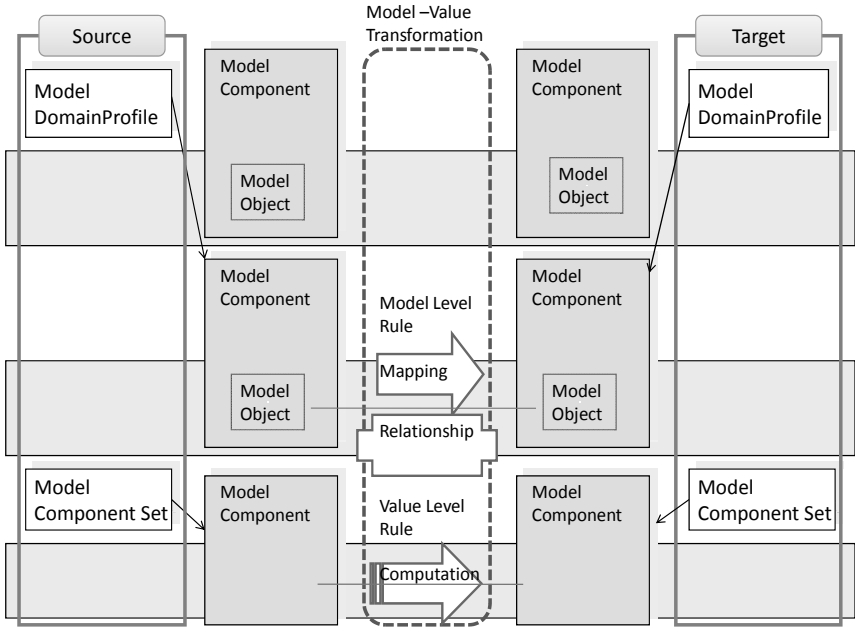


Figure 7- Model-Value Transformation Combination

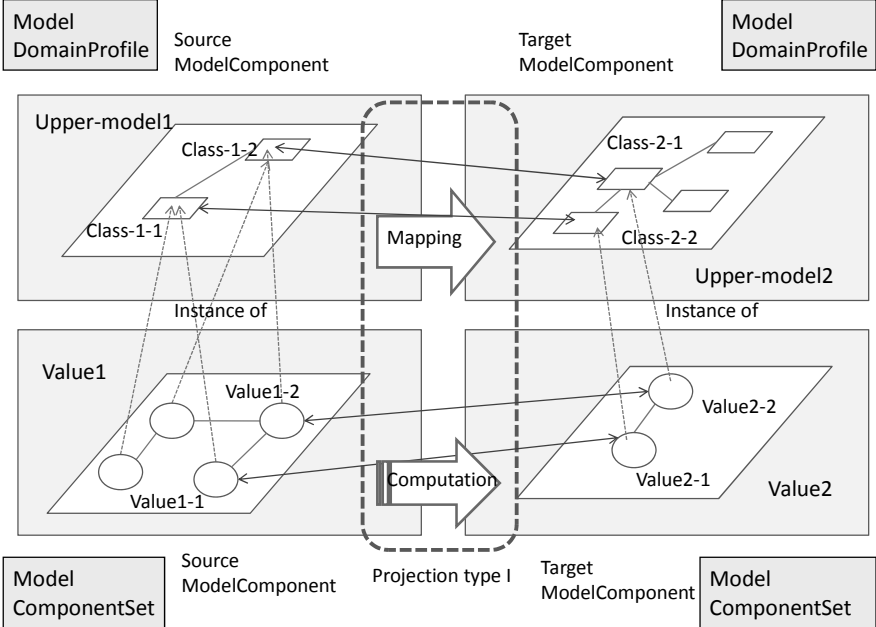


Figure 8- Model-Value Transformation Detail (Projection type I)

5.4 Standard formats for interchanging models

In this part of ISO/IEC 19763, no concrete metadata format is specified. However, the XMI schema for MFI core model is a primary recommended metadata interchange format for interoperability. Other metadata formats, including bindings of ISO/IEC 20944, may be used.

Annex A (informative)

Model Classifier

A.1 General

A kind of a model element has various granularities. For MFI, a model element may be defined as a model classifier. For example, subclass of a model classifier, methodology, a product, a website, a tag value, data type, a pattern (communication, message, component and framework), vocabulary (term and coded value) are candidates of registered model types. A model classifier is expressed by a package of UML, a class diagram or an XML instance of XMI to be concrete. This annex of this document specifies the subclasses of ModelClassifier as examples.

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 C.1 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.

Table A.1- Code list of model types (informative)

Code	Name	Description
STY	Stereotype	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. See Annex C.1.

CDV	CodedValue	A CodedValue can be specified against the datatype having coded values for the ModelDomainProfile and ModelComponentSet. See Annex C.2.
PAT	Pattern	The pattern is a kind of model construct element that is a reusable piece of model and generally applicable to a similar model. It can be treated as a type (or template). See Annex C.3.
COM	Communication	Communication provides a facility to build composed and nested collaborations based on Pattern. Individual portions of nested and/or layered collaboration may be standardized if necessary. See Annex C.4.
CMP	Component	Component provides a facility to build composed and nested components based on Pattern. Individual portions of the resulting components may be standardized if necessary. The operations attached to the pattern may be connected in assembling the components. See Annex C.5
FRM	Framework	Framework provides a facility to build composed and nested frameworks based on Pattern. Individual portions of the resulting frameworks may be standardized if necessary. The operations attached to the pattern may be connected in assembling the components and frameworks. See Annex C.6

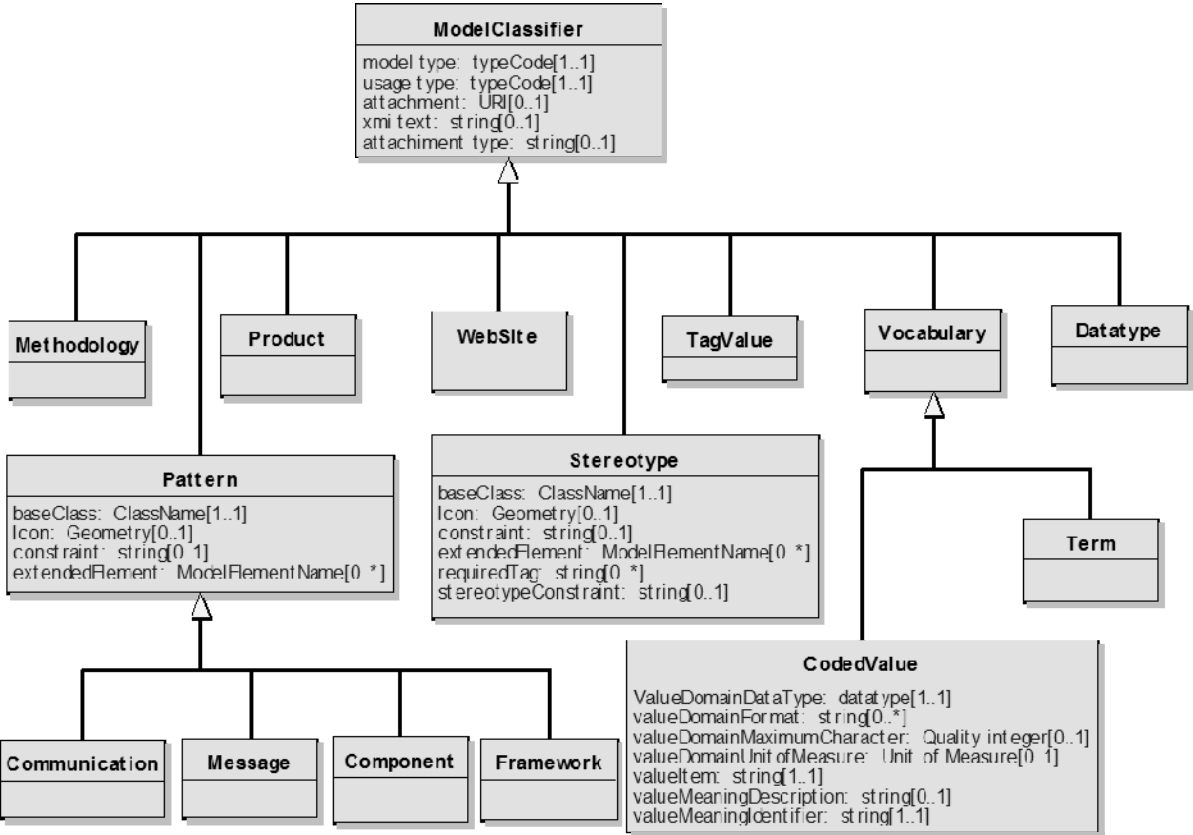


Figure A.1- A Examples of ModelClassifier Package

Annex B (informative)

Transformation Languages

B1. General

For interoperability of shared models, model transformation using a model pattern is a useful technique. Model transformation may be parameterized using a model pattern. The transformation definition also shall be described with a formal transformation language. Simply stated, a model **compiler** is a program that reads a transformation definition written in one formal language (defined by a metamodel) – the source model – and translates it into an equivalent model in another metamodel – the target model.

A tool based on MOF is useful for specifying and implementing the transformation rules. Such supporting tools use specialized transformation languages. A transformation definition tool provides support for creating and modifying transformation definitions. A formal transformation language is needed to define complex transformation rule and to share transformations among different tools.

For example, OMG MOF QVT (Query, View and Transformation) is a formal transformation language used to specify transformation rules precisely and available. A transformation engine such as a model compiler generates simple code from a model. An important function supporting MDA (Model Driven Architecture) promoted by OMG is a function for completing transformations between one model and another model.

B2. Kinds of Transformation Languages

-QVT-Relations : [1]

This is a declarative specification for relationships between MOF models. The Relations language supports complex object pattern matching, and implicitly creates trace classes and their instances to record what occurred during a transformation execution. Relations can assert that other relations also hold between particular model elements matched by their patterns.

-QVT-Core : [1]

This is a small model/language, which only supports pattern matching over a flat set of variables by evaluating conditions over those variables against a set of models. It treats all of the model elements of source, target and trace models symmetrically. In addition, the trace models must be explicitly defined, and these trace models are not deduced from the transformation description, as is the case with Relations. The core model may be implemented directly, or it may be simply used as a reference for the semantics of Relations, which are mapped to the Core, using the transformation language itself.

-QVT-Operational, QVT-Black-box: [1]

In addition to the declarative Relations and Core Languages which embody the same semantics at two different levels of abstraction, there are two mechanisms for invoking imperative implementations of transformations from Relations or Core: one standard language, Operational Mappings, as well as non-standard Black-box MOF Operational implementations. Each relation defines a class which will be instantiated to trace between model elements being transformed, and it has a one-to-one mapping to an Operation signature that the Operational Mapping or Black-box implementations.

-CWM:[3]

The transformation framework defined in the OMG's Common Warehouse Metamodel(CWM) Specification and transformation implemented using XSLT. The CWM transformation framework provides a mechanism for linking source and target elements, but the derivation of the target elements has to be implemented in some concrete language, which is not prescribed by CWM. Effectively, CWM gives a general model, but no actual mechanism to implement model transformations.

-XSLT:[4]

XSLT is another model transformation language. Since MOF models can be serialized as XML using the XML Metadata Interchange (XMI), implementing model transformations using XSLT is possible.

Annex C (informative)

MFI Registry and Model Mapping

C.1 General

This standard provides solutions to resolve problems typically found in a heterogeneous environment that consists of different software platforms and middleware. Even in a single environment, which consists of similar platforms, a modelling artefact might be implemented or installed in different formats or syntaxes. In such environments, model mapping will be needed to exchange modelling artefacts and concrete data.

Specialized diagram languages like UML are used in virtually every area of information technology. Source and Target models are equally as diversified; a target model may be another model or programming language, or the machine language of any computer between a microprocessor and a supercomputer.

C.2 Use Case

There are many use cases of MFI registry and model mapping. Figure B1 illustrates an example of an e-business application. The MFI Registry holds registered model components and model transformation including minimal set of metamodel for representing model mapping. Model mappings are required between different company's systems as shown in the figure.

A document / message format from Company-A could be transformed into a format of Company-B using registered transformation rule.

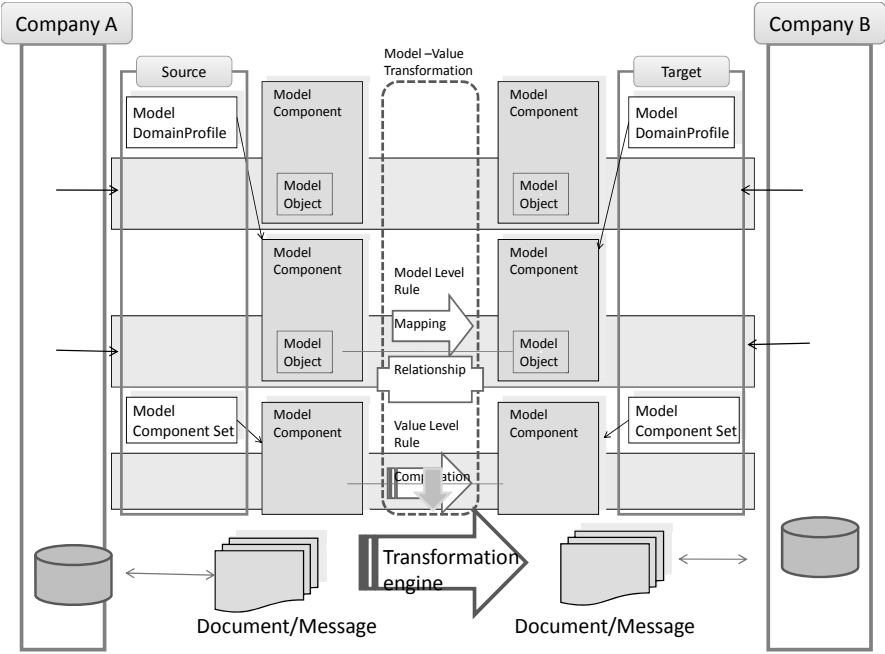


Figure C1- An Environment of MFI Registry and Model Mapping

Annex D (informative)

Use Cases of Model Transformation

D1. General

Business information is being shared or exchanged amongst and between enterprises, governmental agencies, and/or other organizations in e-business and e-commerce over worldwide environment.

There are many kinds of business information models developed by business analysts, business users and information technology specialists. Those business information models are used to supply the content of messages and implement applications.

For example, the UN/CEFACT Core Component Library (CCL) will be stored in a repository and identified in an ebXML compliant registry. Model Transformation is needed in the case of exchange business information using different CCLs.

There are several cases of model transformation as follows:

- Case 1: Transformation on HL7 (see Figures C1 and C2)
- Case 2: Transformation on ebXML (see Figures C3 and C4)
- Case 3: Transformation on CWM (see Figures C5 and C6)
- Case 4: Transformation on XSLT (see Figures C7)

D2. Examples

(1) Case 1:[12]

HL7HDF (HL7 Development Framework) provides model transformation from DMIM (Domain Message Information Model) to RMIM (Refined Message Information Model) based on RIM (Reference Information Model).

-Upper Model:

RIM is a model specifies the grammar of HL7 messages and, specifically, the basic building blocks of the language and their permitted relationships. The RIM backbone has just five core classes (Act, Participations, Role, Entity and Act-Relationship) and a number of permitted relationships between them.

-Lower Model:

DMIM is a model organizing classes of detailed view of the domain. Those are large models derived from RIM that contain all information specific to a particular functional domain (such as patient administration).

RMIM is a model displaying the structure of a message as a colour-coded diagram and used to design messages and explain what each HL7 message consists of RMIMs are refinements of the DMIM.

HMD (Hierarchical message descriptor) is a model derived/refinement of an RMIM, potentially multiple HMDs per RMIM. It is a linear format for generating the XML schema.

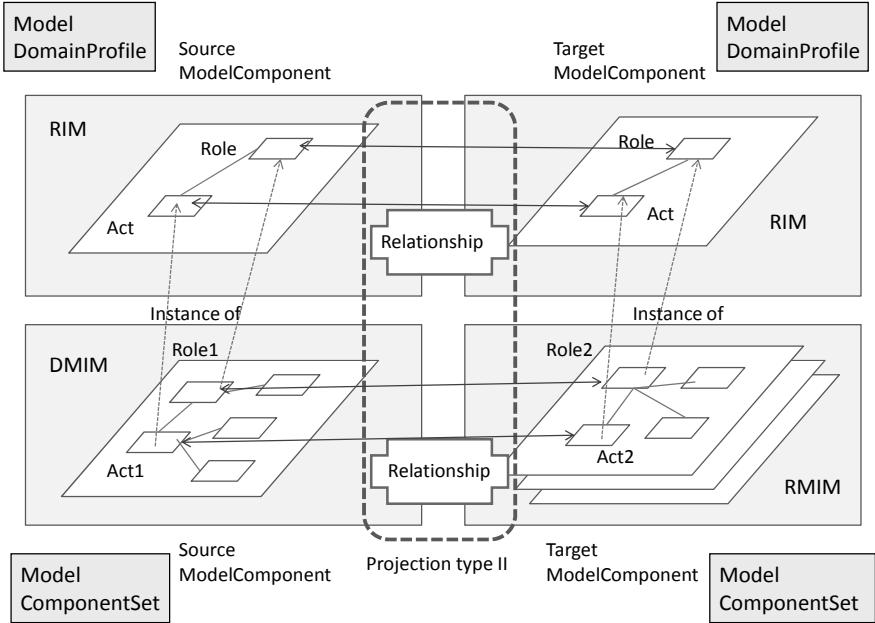


Figure D1- Transformation on HL7 (1)

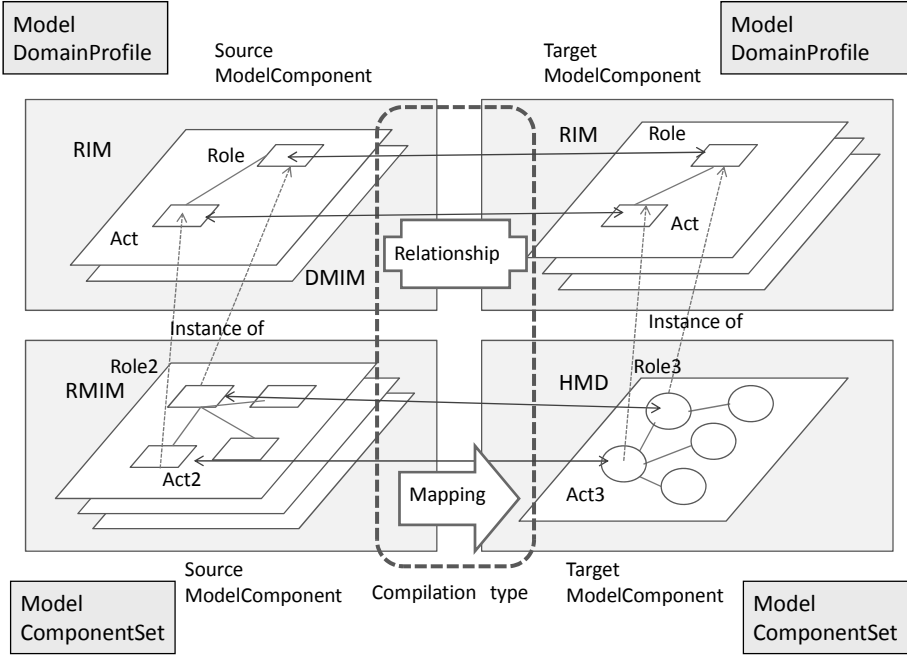


Figure D2- Transformation on HL7 (2)

(2) Case 2:[11]

Another approach of building a message model is ebXML, which was developed by OASIS and UN/CEFACT. For example, the mapping from CC (Core Component) to BIE (Business Information Entity) based on the metamodel of CCL (Core Component Library) is a typical model mapping.

-Upper Model:

CCL is a model specifies the grammar of describing the relationship between CC and BIE, specifically, the basic building blocks of the language and their permitted relationships. The CCL backbone has several core classes (CC, ACC, BIE, ABIE and so on) and a number of permitted relationships between them. A building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept.

-Upper / Lower Model:

CC is a model capturing information about a real world business concept, and relationships between that concept and other business concepts. A Core Component can be either an individual piece of business information, or a family of business information pieces.

It has a unique Business Semantic definition. A Basic Core Component (BCC) represents a Basic Core Component Property and is therefore of a Data Type, which defines its set of values. Basic Core Component (BCC)s function as the Properties of Aggregate Core Component(ACC)s.

-Lower Model:

BIE is a model specializing CCs into domain specific one. A collection of related pieces of business information that together convey a distinct business meaning, independent of any specific Business Context. Expressed in modelling terms, it is the representation of an Object Class such as Basic BIE (BBIE) and Aggregate BIE (ABIE) independent of any specific Business Context.

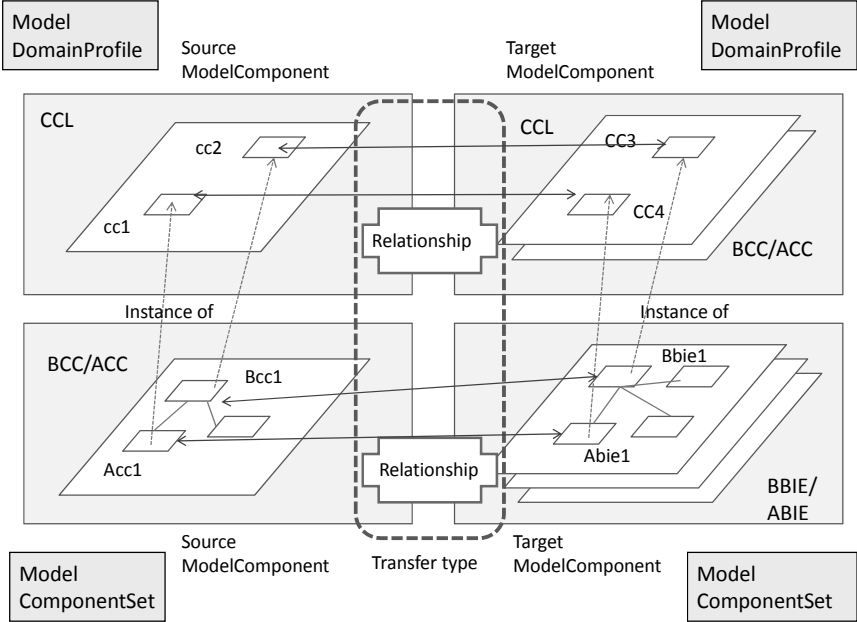


Figure D3- Transformation on ebXML (1)

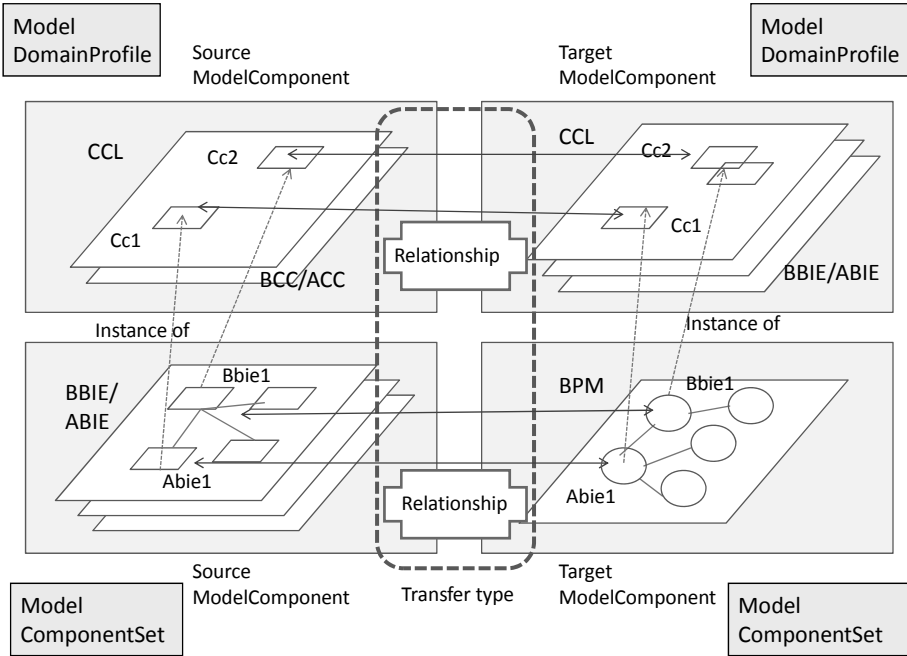


Figure D4- Transformation on ebXML (2)

(3) Case 3:[3]

A model transformation between different metamodels in CWM (Common Warehouse Metamodel) is shown. For example, the ETL (Extract, Transform and Load) tool provides model transformation between SQL tables.

-Upper Model:

SQL shcema is a model specifying the structure of tables.

CWM is a model specifying the structure of ER model and SQL model.

-Lower Model / Value:

Row is a value designating a row of the Table

ER is a model specifying the specific structure with ER model

SQL is a model specifying the specific structure with SQL model

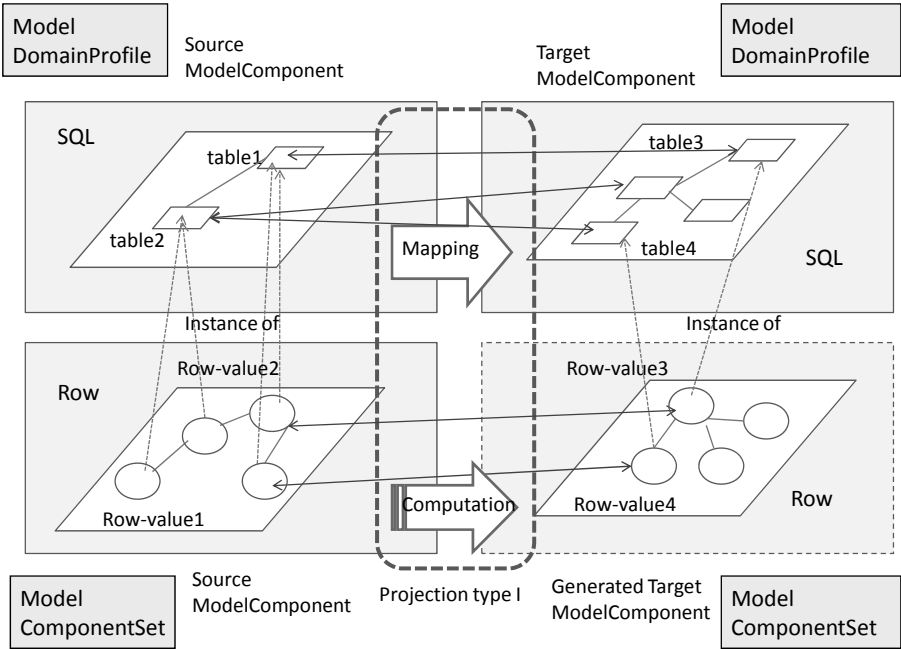


Figure D5- Transformation on CWM (1)

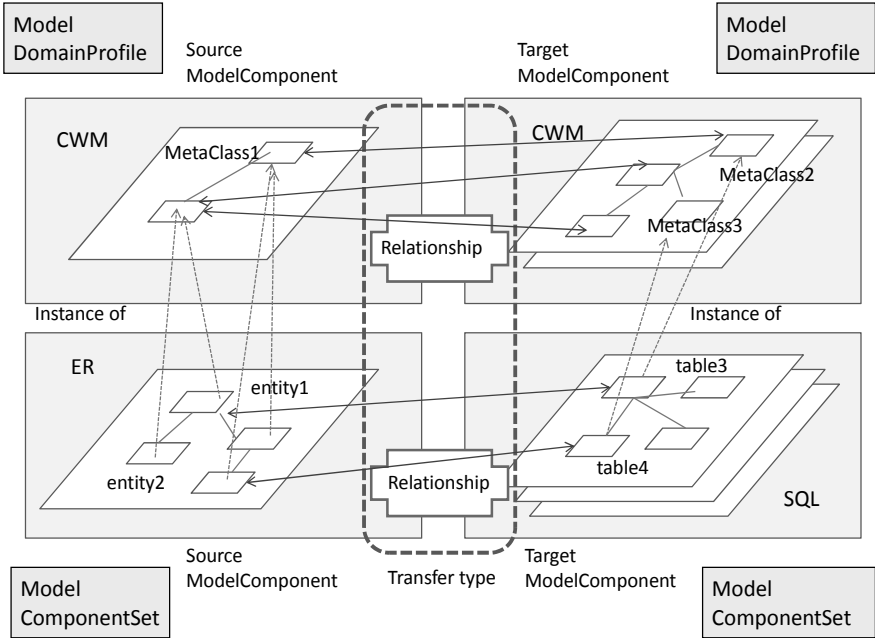


Figure C6- Transformation on CWM (2)

(4) Case 4:[4]

Also, the transformation related to XSL, XPath, and XSLT is a typical example of this type transformation.

-Upper Model:

XML Shema is a model specifying the structure of XML instance.

-Lower Model:

XML instance is a value of XML format according to the upper XML shcema.

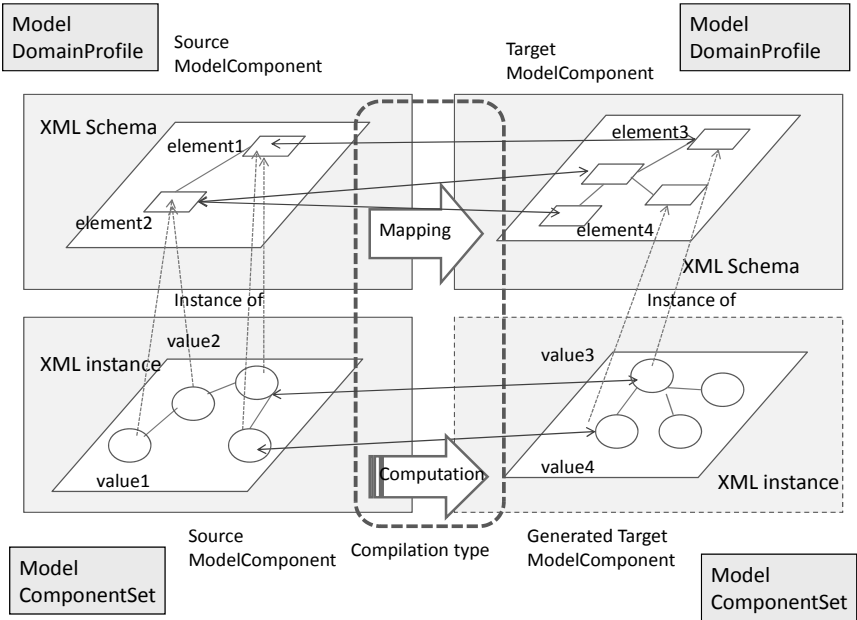


Figure D7- Transformation on XSLT

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 20943-1:2003, *Information technology – Achieving metadata registry content consistency – Part 1:Data elements*
- [4] ISO/IEC TR 20943-3:2004 , *Information technology – Achieving metadata registry content consistency – Part 3:Value domains*
- [5] ISO/IEC 20944, Information Technology -Metadata Registry Interoperability and Bindings (MDRIB)
- [6] ISO 1087-1:2000, Terminology work -- Vocabulary -- Part 1: Theory and application
- [7] ISO 8601:2000, Data elements and interchange formats – Information exchange – Representation of dates and times
- [8] ISO/IEC 11179-1, Information technology – Metadata registries (MDR) - Part 1: Framework
- [9] ISO/IEC 11179-2, Information technology – Metadata registries (MDR) - Part 2: Classification
- [10]ISO/IEC 11179-4, Information technology – Metadata registries (MDR) - Part 4: Formulation of data definitions
- [11]ISO/IEC 11179-5, Information technology – Metadata registries (MDR) - Part 5: Naming and identification principles
- [12]ISO 12620:1999, Computer applications in terminology – Data categories