

First edition
2012-04-15

**Information technology — Object
Management Group Architecture-Driven
Modernization (ADM) — Knowledge
Discovery Meta-Model (KDM)**

*Technologies de l'information — Modernisation conduite par
l'architecture (ADM) de l'OMG — Métamodèle de découverte de
connaissances (KDM)*

Reference number
ISO/IEC 19506:2012(E)





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from 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.org
Web www.iso.org

Published in Switzerland

Table of Contents

Foreword	xv
Introduction	xvi
1 Scope	1
2 Conformance and Compliance	1
2.1 KDM Domains	2
2.2 Compliance Levels	2
2.3 Meaning and Types of Compliance	3
3 Normative References	6
4 Terms and Definitions	6
5 Symbols	8
6 Additional Information	9
6.1 Changes to Other OMG Specifications	9
6.2 How to Read this International Standard	9
7 Overview	11
8 KDM	15
8.1 Overview	15
8.2 Organization of the KDM Packages	16
<i>Subpart I - Infrastructure Layer</i>	19
9 Core Package	21
9.1 Overview	21
9.2 Organization of the Core Package	21
9.3 CoreEntities Class Diagram	21
9.3.1 Element Class (abstract)	22
9.3.2 ModelElement Class (abstract)	22
9.3.3 KDMEntity Class (abstract)	23
9.4 CoreRelations Class Diagram	24
9.4.1 KDMRelationship Class (abstract)	24
9.4.2 KDMEntity (additional properties)	25

9.5	AggregatedRelations Class Diagram	26
9.5.1	AggregatedRelationship Class	26
9.5.2	KDMEntity (additional properties)	29
9.6	Datatypes Class Diagram	29
9.6.1	Boolean Type (datatype)	29
9.6.2	String Type (datatype)	29
9.6.3	Integer Type (datatype)	30
10	The Package named “kdm”	31
10.1	Overview	31
10.2	Organization of the KDM Framework	31
10.3	Framework Class Diagram	32
10.3.1	KDMFramework Class (abstract)	33
10.3.2	KDMModel Class (abstract)	33
10.3.3	KDMEntity (additional properties)	34
10.3.4	Segment Class	34
10.4	Audit Class Diagram	35
10.4.1	Audit Class	36
10.4.2	KDMFramework (additional properties)	37
10.5	Extensions Class Diagram	37
10.5.1	Stereotype Class	38
10.5.2	TagDefinition Class	40
10.5.3	ExtensionFamily Class	41
10.5.4	ModelElement (additional properties)	42
10.6	ExtendedValues Class Diagram	43
10.6.1	ExtendedValue Class (abstract)	43
10.6.2	TaggedValue Class	44
10.6.3	TaggedRef Class	44
10.7	Annotations Class Diagram	45
10.7.1	Attribute Class	46
10.7.2	Annotation Class	47
10.7.3	Element (additional properties)	47
11	Source Package	49
11.1	Overview	49
11.2	Organization of the Source Package	50
11.3	InventoryModel Class Diagram	51
11.3.1	InventoryModel Class	51
11.3.2	AbstractInventoryElement Class (abstract)	52
11.3.3	AbstractInventoryRelationship Class (abstract)	52
11.3.4	InventoryItem Class (generic)	53

11.3.5 SourceFile Class	53
11.3.6 Image Class	54
11.3.7 Configuration Class	54
11.3.8 ResourceDescription Class	54
11.3.9 BinaryFile Class	54
11.3.10 ExecutableFile Class	55
11.3.11 InventoryContainer Class (generic)	55
11.3.12 Directory Class	55
11.3.13 Project Class	56
11.4 InventoryInheritances Class Diagram	56
11.5 InventoryRelations Class Diagram	57
11.5.1 DependsOn Class	57
11.6 SourceRef Class Diagram	58
11.6.1 SourceRef Class	58
11.6.2 SourceRegion Class	60
11.7 ExtendedInventoryElements Class Diagram	61
11.7.1 InventoryElement Class (generic)	61
11.7.2 InventoryRelationship Class (generic)	61
<i>Subpart II - Program Elements Layer</i>	63
12 Code Package	67
12.1 Overview	67
12.2 Organization of the Code Package	67
12.3 CodeModel Class Diagram	68
12.3.1 CodeModel Class	69
12.3.2 AbstractCodeElement Class (abstract)	69
12.3.3 AbstractCodeRelationship Class (abstract)	70
12.3.4 CodeItem Class (abstract)	70
12.3.5 ComputationalObject Class (generic)	70
12.3.6 Datatype Class (generic)	71
12.4 CodeInheritances Class Diagram	71
12.5 Modules Class Diagram	72
12.5.1 Module Class (generic)	72
12.5.2 CompilationUnit Class	73
12.5.3 SharedUnit Class	73
12.5.4 LanguageUnit Class	74
12.5.5 CodeAssembly Class	74
12.5.6 Package Class	74
12.6 ControlElements Class Diagram	75
12.6.1 ControlElement Class (generic)	75
12.6.2 CallableUnit Class	76
12.6.3 CallableKind Data Type (enumerated)	77

12.6.4 MethodUnit Class	77
12.6.5 MethodKind data type (enumeration)	77
12.7 DataElements Class Diagram	79
12.7.1 DataElement Class (generic)	80
12.7.2 StorableUnit Class	80
12.7.3 StorableKind data type (enumeration)	81
12.7.4 ExportKind data type (enumeration)	81
12.7.5 ItemUnit Class	82
12.7.6 IndexUnit Class	82
12.7.7 MemberUnit Class	82
12.7.8 ParameterUnit Class	83
12.8 ValueElements Class Diagram	83
12.8.1 ValueElement Class (generic)	84
12.8.2 Value Class	84
12.8.3 ValueList Class	85
12.9 PrimitiveTypes Class Diagram	86
12.9.1 PrimitiveType Class (generic)	86
12.9.2 BooleanType Class	87
12.9.3 CharType Class	87
12.9.4 OrdinalType Class	87
12.9.5 DateType Class	87
12.9.6 TimeType Class	88
12.9.7 IntegerType Class	88
12.9.8 DecimalType Class	88
12.9.9 ScaledType Class	89
12.9.10 FloatType Class	89
12.9.11 VoidType Class	89
12.9.12 StringType Class	89
12.9.13 BitType Class	90
12.9.14 BitstringType Class	90
12.9.15 OctetType Class	90
12.9.16 OctetstringType Class	90
12.10 EnumeratedTypes Class Diagram	91
12.10.1 EnumeratedType Class	91
12.11 CompositeTypes Class Diagram	92
12.11.1 CompositeType Class (generic)	92
12.11.2 ChoiceType Class	93
12.11.3 RecordType Class	93
12.12 DerivedTypes Class Diagram	95
12.12.1 DerivedType Class (generic)	95
12.12.2 ArrayType Class	96
12.12.3 PointerType Class	96
12.12.4 RangeType Class	97
12.12.5 BagType Class	98
12.12.6 SetType Class	98

12.12.7 SequenceType Class	99
12.13 Signature Class Diagram	99
12.13.1 Signature Class	99
12.13.2 ParameterKind Enumeration Datatype	100
12.14 DefinedTypes Class Diagram	100
12.14.1 DefinedType Class (abstract)	101
12.14.2 TypeUnit Class	101
12.14.3 SynonymUnit Class	102
12.15 ClassTypes Class Diagram	102
12.16 ClassUnit Class	102
12.16.1 InterfaceUnit Class	103
12.17 Templates Class Diagram	103
12.17.1 TemplateUnit Class	104
12.17.2 TemplateParameter Class	104
12.17.3 TemplateType Class	105
12.18 TemplateRelations Class Diagram	105
12.18.1 InstanceOf Class	106
12.18.2 ParameterTo Class	106
12.19 InterfaceRelations Class Diagram	110
12.19.1 Implements Class	111
12.19.2 ImplementationOf Class	112
12.20 TypeRelations Class Diagram	115
12.20.1 HasType Class	116
12.20.2 HasValue Class	116
12.21 ClassRelations Class Diagram	121
12.21.1 Extends Class	121
12.22 Preprocessor Class Diagram	123
12.22.1 PreprocessorDirective Class (generic)	123
12.22.2 MacroUnit Class	125
12.22.3 MacroKind data type (enumeration)	125
12.22.4 MacroDirective Class	125
12.22.5 IncludeDirective Class	126
12.22.6 Conditional Directive Class	126
12.23 PreprocessorRelations Class Diagram	126
12.23.1 Expands Class	127
12.23.2 GeneratedFrom Class	128
12.23.3 Includes Class	130
12.23.4 VariantTo Class	131
12.23.5 Redefines Class	133
12.24 Comments Class Diagram	135

12.24.1 CommentUnit Class	135
12.24.2 AbstractCodeElement Class (additional properties)	136
12.25 Visibility Class Diagram	136
12.25.1 Namespace Class	136
12.26 VisibilityRelations Class Diagram	137
12.26.1 VisibleIn Class	137
12.26.2 Imports Class	138
12.27 ExtendedCodeElements Class Diagram	139
12.27.1 CodeElement Class (generic)	140
12.27.2 CodeRelationship Class (generic)	140
13 Action Package	141
13.1 Overview	141
13.2 Organization of the Action Package	141
13.3 ActionElements Class Diagram	141
13.3.1 ActionElement Class	142
13.3.2 AbstractActionRelationship Class (abstract)	143
13.3.3 BlockUnit Class	143
13.3.4 AbstractCodeElement (additional properties)	144
13.4 ActionInheritances Class Diagram	144
13.5 ActionFlow Class Diagram	145
13.5.1 ControlFlow Class (generic)	145
13.5.2 EntryFlow Class	146
13.5.3 Flow Class	147
13.5.4 TrueFlow Class	147
13.5.5 FalseFlow Class	148
13.5.6 GuardedFlow Class	148
13.6 CallableRelations Class Diagram	149
13.6.1 Calls Class	150
13.6.2 Dispatches Class	151
13.7 DataRelations Class Diagram	152
13.7.1 Reads Class	153
13.7.2 Writes Class	154
13.7.3 Addresses Class	154
13.7.4 Creates Class	154
13.8 ExceptionBlocks Class Diagram	155
13.8.1 ExceptionUnit Class	155
13.8.2 TryUnit Class	156
13.8.3 CatchUnit Class	156
13.8.4 FinallyUnit Class	157
13.9 ExceptionFlow Class Diagram	159

13.9.1 ExitFlow Class	160
13.9.2 ExceptionFlow Class	161
13.10 ExceptionRelations Class Diagram	161
13.10.1 Throws Class	162
13.11 InterfaceRelations Class Diagram	162
13.11.1 CompliesTo Class	163
13.12 UsesRelations Class Diagram	164
13.12.1 UsesType Class	164
13.13 ExtendedActionElements Class Diagram	164
13.13.1 ActionRelationship Class (generic)	165
14 Micro KDM	167
14.1 Overview	167
<i>Subpart III - Runtime Resources Layer</i>	173
15 Platform Package	177
15.1 Overview	177
15.2 Organization of the Platform Package	178
15.3 PlatformModel Class Diagram	179
15.3.1 PlatformModel Class	179
15.3.2 AbstractPlatformElement Class (abstract)	180
15.3.3 AbstractPlatformRelationship Class (abstract)	180
15.4 PlatformInheritances Class Diagram	181
15.5 PlatformResources Class Diagram	181
15.5.1 ResourceType Class	182
15.5.2 NamingResource Class	183
15.5.3 MarshalledResource Class	183
15.5.4 MessagingResource Class	183
15.5.5 FileResource Class	184
15.5.6 ExecutionResource Class	184
15.5.7 LockResource Class	184
15.5.8 StreamResource Class	184
15.5.9 DataManager Class	184
15.5.10 PlatformEvent Class	185
15.5.11 PlatformAction Class	185
15.5.12 ExternalActor Class	185
15.6 PlatformRelations Class Diagram	186
15.6.1 BindsTo Class	186
15.7 ProvisioningRelations Class Diagram	186
15.7.1 Requires Class	187

15.7.2 PlatformActions Class Diagram	187
15.7.3 ManagesResource Class	188
15.7.4 ReadsResource Class	189
15.7.5 WritesResource Class	189
15.7.6 DefinedBy Class	189
15.7.7 Deployment Class Diagram	190
15.7.8 DeployedComponent Class	191
15.7.9 DeployedSoftwareSystem Class	192
15.7.10 Machine Class	192
15.7.11 DeployedResource Class	193
15.7.12 RuntimeResources Class Diagram.....	193
15.7.13 RuntimeResource (generic)	194
15.7.14 Process Class	194
15.7.15 Thread Class	194
15.7.16 RuntimeActions Class Diagram	194
15.7.17 Loads Class	195
15.7.18 Spawns Class	196
15.7.19 ExtendedPlatformElements Class Diagram	196
15.7.20 PlatformElement Class (generic)	197
15.7.21 PlatformRelationship Class (generic)	197
16 UI Package	199
16.1 Overview	199
16.2 Organization of the UI Package	200
16.3 UIModel Class Diagram	200
16.3.1 UIModel Class	201
16.3.2 AbstractUIElement Class (abstract)	201
16.3.3 AbstractUIRelationship Class (abstract)	202
16.4 UIInheritances Class Diagram	202
16.5 UIResources Class Diagram	203
16.5.1 UIResource Class (generic)	204
16.5.2 UIDisplay Class (generic)	204
16.5.3 Screen Class	204
16.5.4 Report Class	205
16.5.5 UIField Class	205
16.5.6 UIEvent Class	205
16.5.7 UIAction Class	205
16.6 UIRelations Class Diagram	206
16.6.1 UIFlow Class	206
16.6.2 UILayout Class	207
16.7 UIActions Class Diagram	207
16.7.1 Displays Class	208
16.7.2 DisplaysImage Class	208
16.7.3 ManagesUI Class	208
16.7.4 ReadsUI Class	209

16.7.5 WritesUI Class	209
16.8 ExtendedUIElements Class Diagram	210
16.9 UIElement Class (generic)	210
16.9.1 UIRelationship Class (generic)	210
17 Event Package	213
17.1 Overview	213
17.2 Organization of the Event Package	214
17.3 EventModel Class Diagram	214
17.3.1 EventModel Class	215
17.3.2 AbstractEventElement Class (abstract)	215
17.3.3 AbstractEventRelationship Class (abstract)	216
17.4 EventResources Class Diagram	216
17.4.1 EventResource Class (generic)	217
17.4.2 Event Class	217
17.4.3 State Class	218
17.4.4 InitialState Class	218
17.4.5 Transition Class	218
17.4.6 OnEntry Class	218
17.4.7 OnExit Class	219
17.4.8 EventAction Class	219
17.5 EventRelations Class Diagram	219
17.5.1 NextState Class	220
17.6 ConsumesEvent Class	220
17.7 EventActions Class Diagram	220
17.7.1 ReadsState Class	221
17.7.2 ProducesEvent Class	222
17.7.3 HasState Class	222
17.8 ExtendedEventElements Class Diagram	222
17.8.1 EventElement Class (generic)	223
17.8.2 EventRelationship Class (generic)	223
18 Data Package	225
18.1 Overview	225
18.2 Organization of the Data Package	226
18.3 Data Model Class Diagram	226
18.3.1 DataModel Class	227
18.3.2 AbstractDataElement Class (abstract)	228
18.3.3 AbstractDataRelationship Class (abstract)	228
18.4 Data Inheritances Class Diagram	228

18.5	DataResources Class Diagram	229
18.5.1	DataResource Class (generic)	230
18.5.2	DataContainer Class (generic)	230
18.5.3	Catalog Class	231
18.5.4	RelationalSchema Class	231
18.5.5	DataEvent Class	232
18.5.6	DataAction Class	232
18.6	ColumnSet Class Diagram	233
18.6.1	ColumnSet (generic)	233
18.6.2	RelationalTable Class	234
18.6.3	RelationalView Class	237
18.6.4	DataSegment Class	237
18.6.5	RecordFile Class	239
18.7	KeyIndex Class Diagram	244
18.7.1	IndexElement Class (generic)	244
18.7.2	UniqueKey Class	245
18.7.3	ReferenceKey Class	245
18.7.4	Index Class	245
18.8	Key Relations Class Diagram	246
18.8.1	KeyRelationship Class	246
18.9	DataActions Class Diagram	247
18.9.1	ReadsColumnSet Class	247
18.9.2	WritesColumnSet Class	248
18.9.3	ManagesData Class	248
18.9.4	HasContent Class	249
18.10	StructuredData Class Diagram	254
18.10.1	XMLSchema	255
18.10.2	AbstractContentElement (abstract)	255
18.11	ContentElements Class Diagram	255
18.11.1	ContentItem (generic)	256
18.11.2	ComplexContentType	257
18.11.3	SimpleContentType	257
18.11.4	ContentRestriction	257
18.11.5	AllContent Class	260
18.11.6	SeqContent Class	260
18.11.7	ChoiceContent Class	260
18.11.8	GroupContent Class	261
18.11.9	MixedContent Class	261
18.11.10	ContentAttribute Class	261
18.11.11	ContentElement Class	261
18.11.12	ContentReference Class	261
18.12	ContentRelations Class Diagram	266
18.12.1	TypedBy Class	267
18.12.2	DatatypeOf Class	268

18.12.3 ReferenceTo Class	268
18.12.4 ExtensionTo Class	268
18.12.5 RestrictionOf Class	269
18.13 ExtendedDataElements Class Diagram	269
18.13.1 ExtendedDataElement Class	270
18.13.2 DataRelationship Class	270
<i>Subpart IV - Abstractions Layer</i>	273
19 Structure Package	275
19.1 Overview	275
19.2 Organization of the Structure Package	276
19.3 StructureModel Class Diagram	276
19.3.1 StructureModel Class	277
19.3.2 AbstractStructureElement Class (abstract)	277
19.3.3 AbstractStructureRelationship Class (abstract)	278
19.3.4 Subsystem Class	278
19.3.5 Layer Class	278
19.3.6 Component Class	278
19.3.7 SoftwareSystem Class	279
19.3.8 ArchitectureView Class	279
19.4 StructureInheritances Class Diagram	279
19.5 ExtendedStructureElements Class Diagram	280
19.5.1 StructureElement Class (generic)	280
19.5.2 StructureRelationship Class (generic)	280
20 Conceptual Package	283
20.1 Overview	283
20.2 Organization of the Conceptual Package	285
20.3 ConceptualModel Class Diagram	285
20.3.1 ConceptualModel	286
20.3.2 AbstractConceptualElement (abstract)	287
20.3.3 AbstractConceptualRelationship Class (abstract)	288
20.4 ConceptualInheritances Class Diagram	288
20.5 ConceptualElements Class Diagram	288
20.5.1 ConceptualContainer Class	289
20.5.2 TermUnit	290
20.5.3 FactUnit	290
20.5.4 RuleUnit	290
20.5.5 ConceptualRole	291
20.5.6 BehaviorUnit Class	291
20.5.7 ScenarioUnit Class	291

20.6	ConceptualRelations Class Diagram	292
20.6.1	ConceptualFlow Class	292
20.7	ExtendedConceptualElements Class Diagram	299
20.7.1	ConceptualElement Class (generic)	300
20.7.2	ConceptualRelationship Class (generic)	300
21	Build Package	303
21.1	Overview	303
21.2	Organization of the Build Package	303
21.3	BuildModel Class Diagram	304
21.4	BuildModel Class	304
21.4.1	AbstractBuildElement Class (abstract)	305
21.4.2	AbstractBuildRelationship Class (abstract)	305
21.4.3	Supplier Class	305
21.4.4	Tool Class	305
21.4.5	SymbolicLink Class	306
21.5	BuildInheritances Class Diagram	306
21.6	BuildResources Class Diagram	306
21.6.1	BuildResource Class	307
21.6.2	BuildComponent Class	308
21.6.3	BuildDescription Class	308
21.6.4	BuildStep Class	308
21.7	BuildRelations Class Diagram	308
21.7.1	LinksTo Class	309
21.7.2	Consumes Class	310
21.7.3	Produces Class	310
21.7.4	SupportedBy Class	311
21.7.5	SuppliedBy Class	311
21.7.6	DescribedBy Class	312
21.8	ExtendedBuildElements Class Diagram	313
21.8.1	BuildElement Class (generic)	313
21.8.2	BuildRelationship Class (generic)	314
	Annex A - Semantics of the Micro KDM Action Elements	315
	Index	327

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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

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

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

ISO/IEC 19500-1 was prepared by Technical Committee ISO/IEC JTC1, Information technology, in collaboration with the Object Management Group (OMG), following the submission and processing as a Publicly Available Specification (PAS) of the OMG Common Object Request Broker Architecture (CORBA) specification Part 1 Version 3.1 CORBA Interfaces.

ISO/IEC 19506 is related to:

- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1995, Information Technology - Open Distributed Processing - Reference Model: Foundations
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1995, Information Technology - Open Distributed Processing - Reference Model: Architecture
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1997, Information Technology - Open Distributed Processing - Interface Definition Language

ISO/IEC 19506 consists of the following, under the general title *Information technology - Open distributed processing - Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM)*.

Apart from this Foreword, the text of this International Standard is identical with that for the OMG specification for Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.3.

Introduction

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated.

RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. The scopes and objectives of the RM-ODP Part 2 and the UML, while related, are not the same and, in a number of cases, the RM-ODP Part 2 and the UML specification use the same term for concepts which are related but not identical (e.g., interface). Nevertheless, a specification using the Part 2 modeling concepts can be expressed using UML with appropriate extensions (using stereotypes, tags, and constraints).

RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2. Given the relation between UML as a modeling language and Part 3 of the RM-ODP standard, it is easy to show that UML is suitable as a notation for the individual viewpoint specifications defined by the RM-ODP.

This International Standard for Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM) is a standard for the technology specification of an ODP system. It defines a technology to provide the infrastructure required to support functional distribution of an ODP system, specifying functions required to manage physical distribution, communications, processing and storage, and the roles of different technology objects in supporting those functions.

Information technology - Object Management Group Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM)

1 Scope

This International Standard defines a meta-model for representing *existing software assets*, their associations, and operational environments, referred to as the Knowledge Discovery Meta-model (KDM). This is the first in the series of specifications related to Software Assurance (SwA) and Architecture-Driven Modernization (ADM) activities. KDM facilitates projects that involve *existing software systems* by insuring interoperability and exchange of data between tools provided by different vendors.

One common characteristic of various tools that address SwA and ADM challenge is that they analyze *existing software assets* (for example, source code modules, database descriptions, build scripts, etc.) to obtain explicit knowledge. Each tool produces a portion of the knowledge about existing software assets. Such tool-specific knowledge may be implicit (“hard-coded” in the tool), restricted to a particular source language, and/or particular transformation, and/or *operational environment*. All the above may hinder interoperability between different tools. The meta-model for Knowledge Discovery provides a common repository structure that facilitates the exchange of data contained within individual tool models that represent *existing software assets*. The meta-model represents the physical and logical assets at various levels of abstraction. The primary purpose of this meta-model is to provide a common interchange format that will allow interoperability between existing modernization and software assurance tools, services, and their respective intermediate representations.

2 Conformance and Compliance

KDM is a meta-model with a very broad scope that covers a large and diverse set of applications, platforms, and programming languages. Not all of its capabilities are equally applicable to all platforms, applications, or programming languages. The primary goal of KDM is to provide the capability to exchange models between tools and thus facilitate cooperation between tool suppliers by allowing integration information about a complex *enterprise application* from multiple sources, as the complexity of modern *enterprise applications* involves multiple platform technologies and programming languages. In order to achieve interoperability and especially the integration of information about different facets of an *enterprise application* from multiple analysis tools, this International Standard defines several compliance levels thereby increasing the likelihood that two or more compliant tools will support the same or compatible meta-model subsets. This suggests that the meta-model should be structured modularly, following the principle of separation of concerns, with the ability to select only those parts of the meta-model that are of direct interest to a particular tool vendor. Consequently, the definition of compliance for KDM requires a balance to be drawn between modularity and ease of interchange. Separation of concerns in the design of KDM is embodied in the concept of KDM domains.

2.1 KDM Domains

Separate facts of knowledge discovery in enterprise application in KDM are grouped into several KDM domains (refer to Figure 2.1). Each KDM domain consists of a single KDM package that defines meta-model elements to represent particular aspects of the system under study. KDM domains correspond to the well-known concept of *architecture views*. For example, the Structure domain enables users to discover architectural elements of source code from the system under study, while the Business Rules domain provides users with behavioral elements of the same system such as features or process rules.

The following domains of knowledge have been identified as the foundation for defining compliance in KDM: Build, Structure, Data, Business Rules, UI, Event, Platform, and micro KDM.

From the user’s perspective, this partitioning of KDM means that they need only to be concerned with those parts of the KDM that they consider necessary for their activities. If those needs change over time, further KDM domains can be added to the user’s repertoire as required. Hence, a KDM user does not have to know the full meta-model to use it effectively. In addition, most KDM domains are partitioned into multiple increments, each adding more knowledge capabilities to the previous ones. This fine-grained decomposition of KDM serves to make the KDM easier to learn and use, but the individual segments within this structure do not represent separate compliance points. The latter strategy would lead to an excess of compliance points and result to the interoperability problems described above. Nevertheless, the groupings provided by KDM domains and their increments do serve to simplify the definition of KDM compliance as explained below.

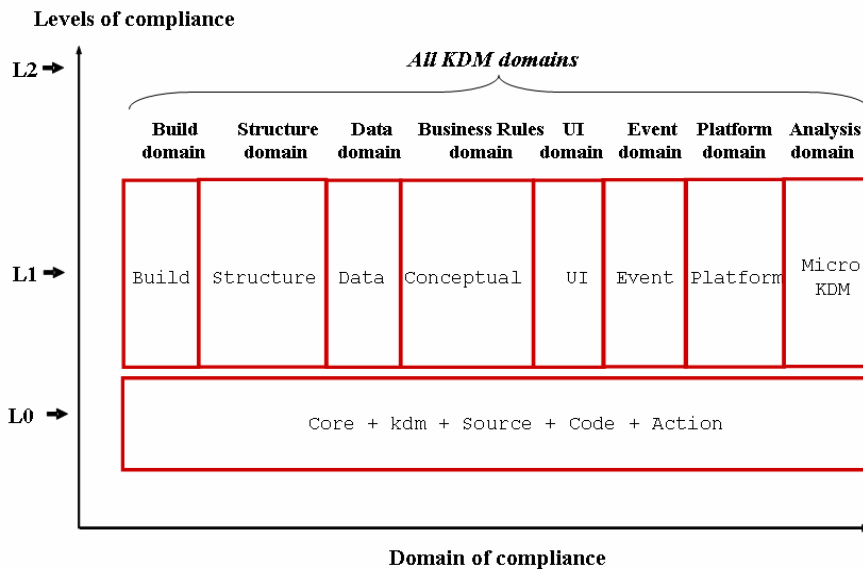


Figure 2.1- Domains and levels of KDM compliance

2.2 Compliance Levels

In addition, the total set of KDM packages is further partitioned into layers of increasing capability called compliance levels. There are two KDM compliance levels:

- Level 0 (L0) - This compliance level contains the following KDM packages: Core, kdm, Source, Code, and Action packages. It provides an entry-level of knowledge discovery capability. More importantly, it represents a common denominator that can serve as a basis for interoperability between different categories of KDM tools.

To be L0 compliant, a tool must completely support all model elements within all packages for L0 level.

- Level 1 (L1) - This level addresses KDM domains and extends the capabilities provided by Level 0. Specifically, it adds the following packages: Build, Structure, Data, Conceptual, UI, Event, Platform, as well as the set of constraints for the micro KDM domain defined in sub clause 14 “Micro KDM,” and Annex A “Semantics of the Micro KDM Action Elements.” These packages are grouped to form above-mentioned domains. More importantly, this level represents a layer where tools could be complimentary since their focus would be in different areas of concern. This would be an additional reason why L0 interoperability (which at this level would be viewed as information sharing between tools) is mandated. In this case interoperability at this level would be viewed as correlation between tools to complete knowledge puzzle that end user might need to perform a particular task.

To be L1 compliant for a given KDM domain, a tool must completely support all model elements defined by the package for that domain and satisfy all semantic constraints specified for that domain.

- Level 2 (L2) - This level is the union of L1 levels for all KDM domains.

2.3 Meaning and Types of Compliance

Compliance to Level 1 (L1) for a certain KDM domain entails full realization of all KDM packages for the corresponding KDM Domain. This also implies full realization of all KDM packages in all the levels below that level (in this case Level 0 (L0)). It is not meaningful to claim compliance to Level 1 without also being compliant with the Level 0. A tool that is compliant at a Level 1 must be able (at least) to import models from tools that are compliant to Level 0 without loss of information. So, “full realization” for a KDM domain means supporting the complete set of concepts defined for that KDM domain at L1 and complete set of concepts defined at L0.

For a given compliance level, a *KDM implementation* can provide:

- The capability to analyze physical artifacts of existing applications and export their representations based on the XMI schema corresponding to the given compliance level.
- The capability to import representations of existing software systems based on the XMI schema corresponding to the given compliance level and perform operations suggested by the corresponding packages.

Table 2.1- Compliance Statements

Compliance Statement				
Compliance Level		Import-Analysis	Import API	Export
L0		Import KDM models based on complete KDM XMI schema into existing tool; support specified mapping between KDM and existing model in the tool; extend operations of existing tool to support meta-model elements of KDM framework; extend operations of existing tool to support meta-model elements of Code and Action packages; extend operations of existing tool to traceability to the physical artifacts of the application from Source package.	Import KDM models based on complete KDM XMI schema; support KDM API defined by the KDM Core package; support KDM framework as defined in the Kdm package; support KDM API defined by the Code and Action packages; support traceability to the physical artifacts of the application as defined in the Source package.	Provide capability to analyze artifacts of an application for specified programming language or multiple languages; Generate XMI documents corresponding to the KDM XMI schema; Support KDM framework as defined by the Kdm package; Support Code and Action packages; Provide traceability back to the physical artifacts as defined by the Source package.
L1	STRUCTURE	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Structure package.	L0 compliance for import; Support KDM API as defined by the Structure package.	L0 compliance for export; Provide capability to analyze architecture components of existing application and generate KDM Structure model according to Structure package.
	DATA	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Data package.	L0 compliance for import; Support KDM API as defined by the Data package.	L0 compliance for export; Provide capability to analyze persistent data components of existing application for specified database system and generate KDM Data model according to Data package.
	PLATFORM	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Platform package.	L0 compliance for import; Support KDM API as defined by the Platform and Runtime packages.	L0 compliance for export; Provide capability to analyze platform artifacts for specified platform and generate KDM Platform model according to Platform package.
	BUILD	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Build package.	L0 compliance for import; Support KDM API as defined by the Build package.	L0 compliance for export; Provide capability to analyze build artifacts for specified build environment and generate KDM Build model according to Build package.

Table 2.1- Compliance Statements

	UI	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the UI package.	L0 compliance for import; Support KDM API as defined by the UI package.	L0 compliance for export; Provide capability to analyze user interface artifacts for specified user interface system and generate KDM UI model according to UI package.
	EVENT	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Event package.	L0 compliance for import; Support KDM API as defined by the Event package.	L0 compliance for export; Provide capability to analyze artifacts related to event-driven runtime frameworks and state-transition behavior and generate KDM Event model according to Event package.
	BUSINESS	L0 compliance for analysis; extend operations of existing tool to support meta-model elements of the Conceptual package.	L0 compliance for import; Support KDM API as defined by the Conceptual package.	L0 compliance for export; Provide capability to analyze conceptual and behavior artifacts (e.g., domain concepts, business rules, scenarios) of existing application and generate KDM Conceptual model according to Conceptual package.
	MICRO KDM	L0 compliance for analysis; extend operations of existing tool to support micro KDM actions as specified in sub clause 14 micro KDM and Annex A.	L0 compliance for import; Support micro KDM actions as specified in sub clause 14 micro KDM and Annex A.	L0 compliance for export; Provide capability to analyze artifacts of existing application to the level of detail specified in sub clause 14 and Annex A provide the mapping of semantics of the existing application as it is determined by the programming languages and the runtime platform into KDM micro actions and generate KDM models that represent the same meaning
L2		L0 import compliance for analysis; L1 import-analysis compliance for all KDM domains.	L0 compliance for import; Support KDM API as defined by all KDM packages.	L0 export compliance; L1 export compliance for all KDM domains.

3 Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to or revisions of any of these publications do not apply.

- OMG UML Infrastructure Specification, v2.3, formal/2010-05-03
- OMG Meta-Object Facility (MOF) Specification, v2.0, formal/2006-01-01
- OMG MOF XML Metadata Interchange (XMI) Specification, v2.1, formal/2005-09-01
- OMG Semantics of Business Vocabularies and Business Rules (SBVR) Specification, v1.0, formal/2008-01-02
- ISO/IEC 11404:2007 Information technology -- General Purpose Datatypes (GPD)