INTERNATIONAL STANDARD

ISO/IEC 19505-2

First edition
2012-04-15

Information technology — Object Management Group Unified Modeling Language (OMG UML) —

Part 2: **Superstructure**

Technologies de l'information — Langage de modélisation unifié OMG (OMG UML) —

Partie 2: Superstructure

Reference number
ISO/IEC 19505-2:2012(E)

COPYRIGHT PROTECTED DOCUMENT

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

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 part of ISO/IEC 19505 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

This International Standard was prepared by Technical Committee ISO/IEC/TC 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 Unified Modeling Language (UML) specification.

This International Standard is related to:

- ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP).

This International Standard consists of the following parts, under the general title *Information technology - Open distributed processing - UML specification*:

- Part 1: Infrastructure
- Part 2: Superstructure

Apart from this Foreword, the text of this International Standard is identical with that for the OMG specification for UML, v2.4.1, Part 2.

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 that 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 2 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.

The Unified Modeling Language (UML) is a general-purpose modeling language with a semantic specification, a graphical notation, an interchange format, and a repository query interface. It is designed for use in object-oriented software applications, including those based on technologies recommended by the Object Management Group (OMG). As such, it serves a variety of purposes including, but not limited to, the following:

- a means for communicating requirements and design intent,
- a basis for implementation (including automated code generation),
- a reverse engineering and documentation facility.

As an international standard, the various components of UML provide a common foundation for model and metadata interchange:

- between software development tools,
- between software developers, and
- between repositories and other object management facilities.

The existence of such a standard facilitates the communication between standardized UML environments and other environments.

While not limited to this context, the UML standard is closely related to work on the standardization of Open Distributed Processing (ODP).

Information technology - Object Management Group Unified Modeling Language (OMG UML), Superstructure

1 Scope

1.1 General

This International Standard defines the Unified Modeling Language (UML), revision 2. The objective of UML is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes.

The initial versions of UML (UML 1) originated with three leading object-oriented methods (Booch, OMT, and OOSE), and incorporated a number of best practices from modeling language design, object-oriented programming, and architectural description languages. Relative to UML 1, this revision of UML has been enhanced with significantly more precise definitions of its abstract syntax rules and semantics, a more modular language structure, and a greatly improved capability for modeling large-scale systems.

One of the primary goals of UML is to advance the state of the industry by enabling object visual modeling tool interoperability. However, to enable meaningful exchange of model information between tools, agreement on semantics and notation is required. UML meets the following requirements:

- A formal definition of a common MOF-based metamodel that specifies the abstract syntax of the UML. The abstract syntax defines the set of UML modeling concepts, their attributes and their relationships, as well as the rules for combining these concepts to construct partial or complete UML models.
- A detailed explanation of the semantics of each UML modeling concept. The semantics define, in a technology-independent manner, how the UML concepts are to be realized by computers.
- A specification of the human-readable notation elements for representing the individual UML modeling concepts as well as rules for combining them into a variety of different diagram types corresponding to different aspects of modeled systems.
- A detailed definition of ways in which UML tools can be made compliant with this International Standard. This is supported (in a separate specification) with an XML-based specification of corresponding model interchange formats (XMI) that must be realized by compliant tools.

2 Conformance

2.1 General

UML is a language with a very broad scope that covers a large and diverse set of application domains. Not all of its modeling capabilities are necessarily useful in all domains or applications. This suggests that the language should be structured modularly, with the ability to select only those parts of the language that are of direct interest. On the other hand, an excess of this type of flexibility increases the likelihood that two different UML tools will be supporting different subsets of the language, leading to interchange problems between them. Consequently, the definition of compliance for UML requires a balance to be drawn between modularity and ease of interchange.

Experience with previous versions of UML has indicated that the ability to exchange models between tools is of paramount interest to a large community of users. For that reason, this International Standard defines a small number of *compliance levels* thereby increasing the likelihood that two or more compliant tools will support the same or compatible language subsets. However, in recognition of the need for flexibility in learning and using the language, UML also provides the concept of *language units*.

2.2 Language Units

The modeling concepts of UML are grouped into *language units*. A language unit consists of a collection of tightly-coupled modeling concepts that provide users with the power to represent aspects of the system under study according to a particular paradigm or formalism. For example, the State Machines language unit enables modelers to specify discrete event-driven behavior using a variant of the well-known statecharts formalism, while the Activities language unit provides for modeling behavior based on a workflow-like paradigm. From the user's perspective, this partitioning of UML means that they need only be concerned with those parts of the language that they consider necessary for their models. If those needs change over time, further language units can be added to the user's repertoire as required. Hence, a UML user does not have to know the full language to use it effectively.

In addition, most language units are partitioned into multiple increments, each adding more modeling capabilities to the previous ones. This fine-grained decomposition of UML serves to make the language 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 language units and their increments do serve to simplify the definition of UML compliance as explained below.

2.3 Compliance Levels

The stratification of language units is used as the foundation for defining compliance in UML. Namely, the set of modeling concepts of UML is partitioned into horizontal layers of increasing capability called *compliance levels*. Compliance levels cut across the various language units, although some language units are only present in the upper levels. As their name suggests, each compliance level is a distinct compliance point.

For ease of model interchange, there are just four compliance levels defined for the whole of UML:

- *Level 0 (L0)*. This compliance level is formally defined in the UML Infrastructure. It contains a single language unit that provides for modeling the kinds of class-based structures encountered in most popular object-oriented programming languages. As such, it provides an entry-level modeling capability. More importantly, it represents a low-cost common denominator that can serve as a basis for interoperability between different categories of modeling tools.
- *Level 1 (L1)*. This level adds new language units and extends the capabilities provided by Level 0. Specifically, it adds language units for use cases, interactions, structures, actions, and activities.
- *Level 2 (L2)*. This level extends the language units already provided in Level 1and adds language units for deployment, state machine modeling, and profiles.
- *Level 3 (L3)*. This level represents the complete UML. It extends the language units provided by Level 2 and adds new language units for modeling information flows, templates, and model packaging.

The contents of language units are defined by corresponding top-tier packages of the UML metamodel, while the contents of their various increments are defined by second-tier packages within language unit packages. Therefore, the contents of a compliance level are defined by the set of metamodel packages that belong to that level.

As noted, compliance levels build on supporting compliance levels. The principal mechanism used in this International Standard for achieving this is *package merge* (see “PackageMerge (from Kernel)” on page 118). Package merge allows modeling concepts defined at one level to be extended with new features. Most importantly, this is achieved *in the context of the same namespace*, which enables interchange of models at different levels of compliance as described in ““Meaning and Types of Compliance” on page 6.

For this reason, all compliance levels are ultimately merged into a single core “UML” model package that defines the common namespace shared by all the compliance levels. Level 0 is defined by the top-level metamodel shown in Figure 2.1. In this model, “L0” is originally an empty package that simply merges in the contents of the *Basic* package from the *UML Infrastructure*. This package is then merged into the UML model. Package L0 contains elementary concepts such as Class, Package, DataType, Operation, etc. merged in from *Basic* (see the *Unified Modeling Language: Infrastructure specification* for the complete list of contents of this package).

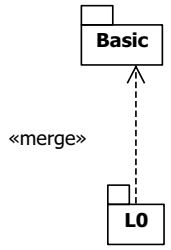

Figure 2.1 - Level 0 package diagram

At the next level (Level 1) the packages merged into Level 0 and their contents are extended with additional packages as shown in Figure 2.2 on page 4. Note that each of the four packages shown in the figure merges in additional packages that are not shown in the diagram. They are defined in the corresponding package diagrams in this Part of ISO/IEC 19505. Consequently, the set of language units that results from this model is more than is indicated by the top-level model in the diagram. The specific packages included at this level are listed in Table 2.3 on page 8.

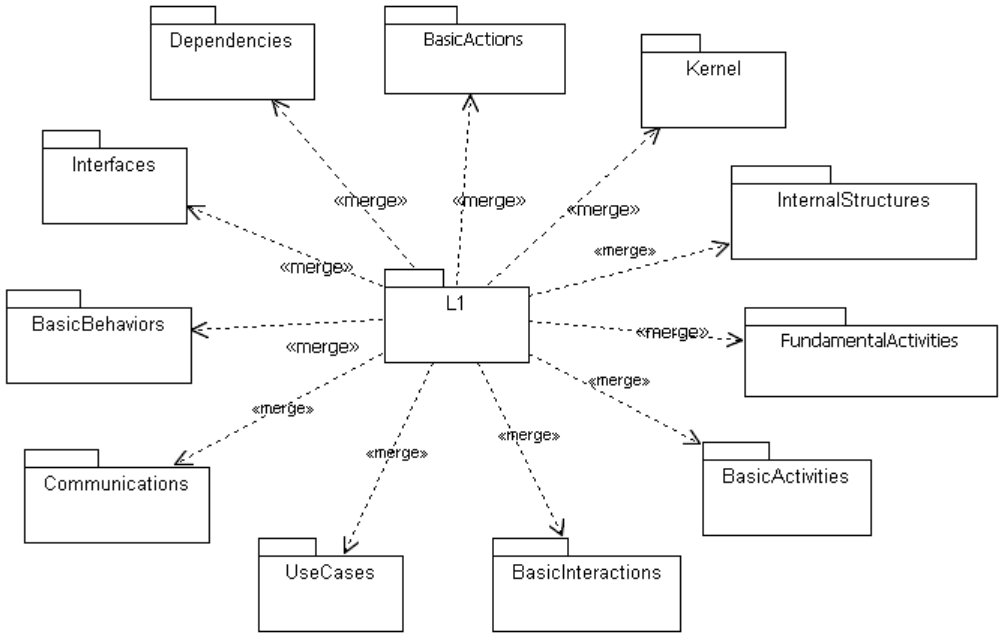

Figure 2.2 - Level 1 top-level package merges

Level 2 adds further language units and extensions to those provided by the Level 1. The actual language units and packages included at this level of compliance are listed in Table 2.4 on page 8.

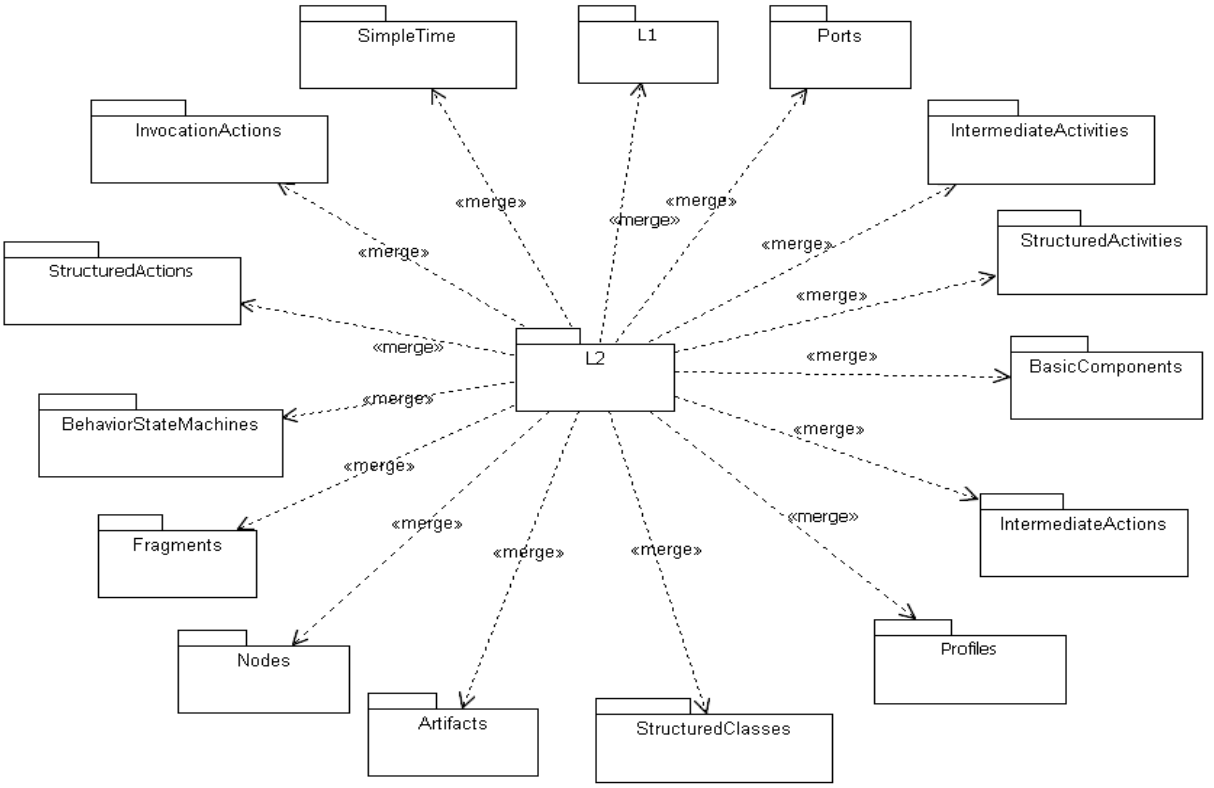

Figure 2.3 - Level 2 top-level package merges

Finally, Level3, incorporating the full UML definition, is shown in Figure 2.4 on page 6. Its contents are described in Table 2.5 on page 9.

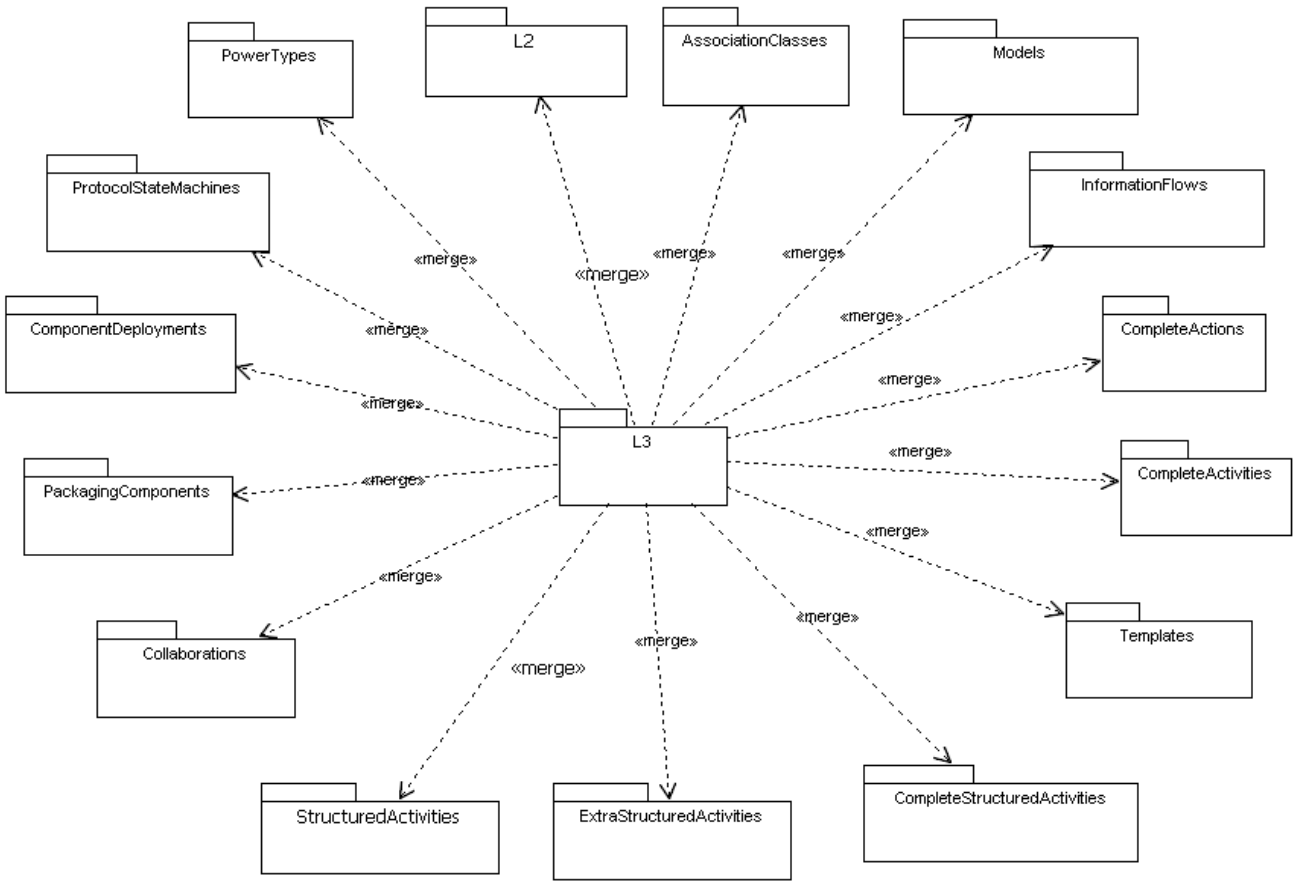

Figure 2.4 - Level 3 top-level package merges

2.4 Meaning and Types of Compliance

Compliance to a given level entails full realization of *all language units* that are defined for that compliance level. This also implies full realization of all language units in all the levels below that level. “Full realization” for a language unit at a given level means supporting the *complete set of modeling concepts* defined for that language unit *at that level*.

Thus, it is not meaningful to claim compliance to, say, Level 2 without also being compliant with the Level 0 and Level 1. A tool that is compliant at a given level must be able to import models from tools that are compliant to lower levels without loss of information.

There are two distinct types of compliance. They are:

1. *Abstract syntax compliance*. For a given compliance level, this entails:
   - compliance with the metaclasses, their structural relationships, and any constraints defined as part of the merged UML metamodel for that compliance level and,
   - the ability to output models and to read in models based on the XMI schema corresponding to that compliance level.

2. *Concrete syntax compliance*. For a given compliance level, this entails:
   - Compliance to the notation defined in the "Notation" sub clauses in this part of ISO/IEC 19505 for those metamodel elements that are defined as part of the merged metamodel for that compliance level and, by implication, the diagram types in which those elements may appear. And, optionally:
   - the ability to output diagrams and to read in diagrams based on the XMI schema defined by the Diagram Interchange specification for notation at that level. This option requires abstract syntax and concrete syntax compliance.

Concrete syntax compliance does not require compliance to any presentation options that are defined as part of the notation.

Compliance for a given level can be expressed as:

- abstract syntax compliance
- concrete syntax compliance
- abstract syntax with concrete syntax compliance
- abstract syntax with concrete syntax and diagram interchange compliance

Table 2.1 - Example compliance statement

Compliance Summary			
Compliance level	**Abstract Syntax**	**Concrete Syntax**	**Diagram Interchange Option**
Level 0	YES	YES	YES
Level 1	YES	YES	NO
Level 2	YES	NO	NO

In case of tools that generate program code from models or those that are capable of executing models, it is also useful to understand the level of support for the run-time semantics described in the various "Semantics" sub clauses of the specification. However, the presence of numerous variation points in these semantics (and the fact that they are defined informally using natural language), make it impractical to define this as a formal compliance type, since the number of possible combinations is very large.

A similar situation exists with presentation options, since different implementors may make different choices on which ones to support. Finally, it is recognized that some implementors and profile designers may want to support only a subset of features from levels that are above their formal compliance level. (Note, however, that they can only claim compliance to the level that they fully support, even if they implement significant parts of the capabilities of higher levels.) Given this potential variability, it is useful to be able to specify clearly and efficiently, which capabilities are supported by a given implementation. To this end, in addition to a formal statement of compliance, implementors and profile designers may also provide informal *feature support statements*. These statements identify support for additional features in terms of language units and/or individual metamodel packages, as well as for less precisely defined dimensions such as presentation options and semantic variation points.

An example feature support statement is shown in Table 2.2 for an implementation whose compliance statement is given in Table 2.1. In this case, the implementation adds two new language units from higher levels.

Table 2.2 - Example feature support statement

Feature Support Statement					
Language Unit	**Packages**	**Abstract Syntax**	**Concrete Syntax**	**Semantics**	**Presentation Options**
Deployments	Deployments::Artifacts (L2) Deployments::Nodes (L2)	**YES**	**YES**	**Note (4)**	**Note (5)**
State Machines	StateMachines::BehaviorStateMachines (L2) StateMachines::ProtocolStateMachines (L3)	**Note (1)**	**YES**	**Note (2)**	**Note (3)**

Note (1): States and state machines are limited to a single region
Shallow history pseudostates not supported

Note (2): FIFO queueing in event pool

Note (3): Inherited elements indicated using grey-toned lines, etc.

2.5 Compliance Level Contents

Table 2.3 - Metamodel packages added in Level 1

Language Unit	Metamodel Packages
Actions	Actions::BasicActions
Activities	Activities::FundamentalActivities
	Activities::BasicActivities
Classes	Classes::Kernel
	Classes::Dependencies
	Classes::Interfaces
General Behavior	CommonBehaviors::BasicBehaviors
	CommonBehaviors::Communications
Structures	CompositeStructure::InternalStructures
Interactions	Interactions::BasicInteractions
UseCases	UseCases

Table 2.4 - Metamodel packages added in Level 2

Language Unit	Metamodel Packages
Actions	Actions::StructuredActions
	Actions::IntermediateActions
Activities	Activities::IntermediateActivities
	Activities::StructuredActivities
Components	Components::BasicComponents

Table 2.4 - Metamodel packages added in Level 2

Language Unit	Metamodel Packages
Deployments	Deployments::Artifacts
	Deployments::Nodes
General Behavior	CommonBehaviors::SimpleTime
Interactions	Interactions::Fragments
Profiles	AuxilliaryConstructs::Profiles
Structures	CompositeStructures::InvocationActions
	CompositeStructures::Ports
	CompositeStructures::StructuredClasses
State Machines	StateMachines::BehaviorStateMachines

Table 2.5 - Metamodel packages added in Level 3

Language Unit	Metamodel Packages
Action	Actions::CompleteActions
Activities	Activities::CompleteActivities
	Activities::CompleteStructuredActivities
	Activities::ExtraStructuredActivities
Classes	Classes::AssociationClasses
	Classes::PowerTypes
Components	Components::PackagingComponents
Deployments	Deployments::ComponentDeployments
Information Flows	AuxilliaryConstructs::InformationFlows
Models	AuxilliaryConstructs::Models
State Machines	StateMachines::ProtocolStateMachines
Structures	CompositeStructures::Collaborations
	CompositeStructures::StructuredActivities
Templates	AuxilliaryConstructs::Templates

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 19505. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- RFC2119, http://ietf.org/rfc/rfc2119, Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, March 1997.

- ISO/IEC 19505-1, Information technology — OMG Unified Modeling Language (OMG UML) Version 2.4.1 — Part 1: Infrastructure (pas/2011-08-11)
- OMG Specification formal/2011-08-05, UML Infrastructure, v2.4.1
- OMG Specification formal/2010-02-01, Object Constraint Language, v2.2
- OMG Specification formal/2011-08-07, Meta Object Facility (MOF) Core, v2.4.1
- OMG Specification formal/2011-08-09, XML Metadata Interchange (XMI), v2.4.1
- OMG Specification formal/2006-04-04, UML 2.0 Diagram Interchange

Note – UML 2 is based on a different generation of MOF and XMI than that specified in ISO/IEC 19502:2005 Information technology - Meta Object Facility (MOF) and ISO/IEC 19503:2005 Information technology - XML Metadata Interchange (XMI) which are compatible with ISO/IEC 19501 UML version 1.4.1.