OMG Modeling Glossary B

This glossary defines the terms that are used to describe the Unified Modeling Language (UML) and the Meta Object Facility (MOF). In addition to UML and MOF specific terminology, it includes related terms from OMG standards and object-oriented analysis and design methods, as well as the domain of object repositories and meta data managers. Glossary entries are organized alphabetically and MOF specific entries are identified as '[MOF]'.

Notation Conventions

The entries in the glossary usually begin with a lowercase letter. An initial uppercase letter is used when a word is usually capitalized in standard practice. Acronyms are all capitalized, unless they traditionally appear in all lowercase.

When one or more words in a multi-word term is enclosed in brackets, it indicates that those words are optional when referring to the term. For example, *use case [class]* may be referred to as simply *use case*.

The following conventions are used in this glossary:

- Contrast: <term>
  Refers to a term that has an opposed or substantively different meaning.
- See: <term>
  Refers to a related term that has a similar, but not synonymous meaning.
- Synonym: <term>
  Indicates that the term has the same meaning as another term, which is referenced.
- Acronym: <term>
  Indicates that the term is an acronym. The reader is usually referred to the spelled-out term for the definition, unless the spelled-out term is rarely used.

B OMG Modeling Glossary

Glossary Terms

abstract class	A class that cannot be directly instantiated. Contrast: *concrete class*.
abstraction	The essential characteristics of an entity that distinguish it from all other kinds of entities. An abstraction defines a boundary relative to the perspective of the viewer.
action	The specification of an executable statement that forms an abstraction of a computational procedure. An action typically results in a change in the state of the system, and can be realized by sending a message to an object or modifying a link or a value of an attribute.
action sequence	An expression that resolves to a sequence of actions.
action state	A state that represents the execution of an atomic action, typically the invocation of an operation.
activation	The execution of an action.
active class	A class whose instances are active objects. See: *active object.*
active object	An object that owns a thread and can initiate control activity. An instance of active class. See: *active class, thread*.
activity graph	A special case of a state machine that is used to model processes involving one or more classifiers. Contrast: *statechart diagram*.
actor [class]	A coherent set of roles that users of use cases play when interacting with these use cases. An actor has one role for each use case with which it communicates.
actual parameter	Synonym: *argument*.
aggregate [class]	A class that represents the “whole” in an aggregation (whole-part) relationship. See: *aggregation.*
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part. See: *composition.*

analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain. Analysis focuses what to do, design focuses on how to do it. Contrast: *design*.
analysis time	Refers to something that occurs during an analysis phase of the software development process. See: *design time, modeling time.*
architecture	The organizational structure of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.
argument	A binding for a parameter that resolves to a run-time instance. Synonym: *actual parameter.* Contrast: *parameter.*
artifact	A piece of information that is used or produced by a software development process. An artifact can be a model, a description, or software. Synonym: *product.*
association	The semantic relationship between two or more classifiers that specifies connections among their instances.
association class	A model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties.
association end	The endpoint of an association, which connects the association to a classifier.
asynchronous action	A request where the sending object does not pause to wait for results. Contrast: *synchronous action*.
attribute	A feature within a classifier that describes a range of values that instances of the classifier may hold.
behavior	The observable effects of an operation or event, including its results.
behavioral feature	A dynamic feature of a model element, such as an operation or method.

B OMG Modeling Glossary

behavioral model aspect	A model aspect that emphasizes the behavior of the instances in a system, including their methods, collaborations, and state histories.
binary association	An association between two classes. A special case of an n-ary association.
binding	The creation of a model element from a template by supplying arguments for the parameters of the template.
boolean	An enumeration whose values are true and false.
boolean expression	An expression that evaluates to a boolean value.
cardinality	The number of elements in a set. Contrast: *multiplicity.*
child	In a generalization relationship, the specialization of another element, the parent. See: *subclass*, *subtype*. Contrast: *parent*.
call	An action state that invokes an operation on a classifier.
class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See: *interface*.
classifier	A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.
class diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
client	A classifier that requests a service from another classifier. Contrast: *supplier.*
collaboration	The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction. See: *interaction*.

collaboration diagram	A diagram that shows interactions organized around the structure of a model, using either classifiers and associations or instances and links. Unlike a sequence diagram, a collaboration diagram shows the relationships among the instances. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. See: *sequence diagram.*
comment	An annotation attached to an element or a collection of elements. A note has no semantics. Contrast: *constraint.*
communication association	In a deployment diagram an association between nodes that implies a communication. See: *deployment diagram.*
compile time	Refers to something that occurs during the compilation of a software module. See: *modeling time, run time*.
component	A physical, replaceable part of a system that packages implementation and conforms to and provides the realization of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files.
component diagram	A diagram that shows the organizations and dependencies among components.
composite [class]	A class that is related to one or more classes by a composition relationship. See: *composition*.
composite aggregation	Synonym: *composition*.
composite state	A state that consists of either concurrent (orthogonal) substates or sequential (disjoint) substates. See: *substate*.
composite substate	A substate that can be held simultaneously with other substates contained in the same composite state. Synonym: *region*. See: *composite substate*.
composition	A form of aggregation association with strong ownership and coincident lifetime as part of the whole. Parts with non-fixed multiplicity may be created after the composite itself, but once created they live and die with it (i.e., they share lifetimes). Such parts can also be explicitly removed before the death of the composite. Composition may be recursive. Synonym: *composite aggregation*.

concrete class	A class that can be directly instantiated. Contrast: *abstract class.*
concurrency	The occurrence of two or more activities during the same time interval. Concurrency can be achieved by interleaving or simultaneously executing two or more threads. See: *thread*.
concurrent substate	A substate that can be held simultaneously with other substates contained in the same composite state. See: *composite state.* Contrast: *disjoint substate.*
constraint	A semantic condition or restriction. Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML. See: *tagged value, stereotype.*
container	1. An instance that exists to contain other instances, and that provides operations to access or iterate over its contents. (for example, arrays, lists, sets). 2. A component that exists to contain other components.
containment hierarchy	A namespace hierarchy consisting of model elements, and the containment relationships that exist between them. A containment hierarchy forms an acyclic graph.
context	A view of a set of related modeling elements for a particular purpose, such as specifying an operation.
datatype	A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations.
defining model [MOF]	The model on which a repository is based. Any number of repositories can have the same defining model.
delegation	The ability of an object to issue a message to another object in response to a message. Delegation can be used as an alternative to inheritance. Contrast: *inheritance*.
dependency	A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

deployment diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes, and objects that live on them. Components represent run-time manifestations of code units. See: *component diagrams*.
derived element	A model element that can be computed from another element, but that is shown for clarity or that is included for design purposes even though it adds no semantic information.
design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
design time	Refers to something that occurs during a design phase of the software development process. See: *modeling time*. Contrast: *analysis time*.
development process	A set of partially ordered steps performed for a given purpose during software development, such as constructing models or implementing models.
diagram	A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). UML supports the following diagrams: class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, and deployment diagram.
disjoint substate	A substate that cannot be held simultaneously with other substates contained in the same composite state. See: composite state. Contrast: *concurrent substate*.
distribution unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
domain	An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.
dynamic classification	A semantic variation of generalization in which an object may change type or role. Contrast: *static classification*.
element	An atomic constituent of a model.

entry action	An action executed upon entering a state in a state machine regardless of the transition taken to reach that state.
enumeration	A list of named values used as the range of a particular attribute type. For example, RGBColor = {red, green, blue}. Boolean is a predefined enumeration with values from the set {false, true}.
event	The specification of a significant occurrence that has a location in time and space. In the context of state diagrams, an event is an occurrence that can trigger a transition.
exit action	An action executed upon exiting a state in a state machine regardless of the transition taken to exit that state.
export	In the context of packages, to make an element visible outside its enclosing namespace. See: *visibility.* Contrast: *export* [OMA], *import*.
expression	A string that evaluates to a value of a particular type. For example, the expression "(7 + 5 * 3)" evaluates to a value of type number.
extend	A relationship from an extension use case to a base use case, specifying how the behavior defined for the extension use case can be inserted into the behavior defined for the base use case.
feature	A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.
final state	A special kind of state signifying that the enclosing composite state or the entire state machine is completed.
fire	To execute a state transition. See: *transition*.
focus of control	A symbol on a sequence diagram that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.
formal parameter	Synonym: *parameter.*
framework	A micro-architecture that provides an extensible template for applications within a specific domain.
generalizable element	A model element that may participate in a generalization relationship. See: *generalization*.

generalization	A taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed. See: *inheritance.*
guard condition	A condition that must be satisfied in order to enable an associated transition to fire.
implementation	A definition of how something is constructed or computed. For example, a class is an implementation of a type, a method is an implementation of an operation.
implementation inheritance	The inheritance of the implementation of a more specific element. Includes inheritance of the interface. Contrast: *interface inheritance*.
import	In the context of packages, a dependency that shows the packages whose classes may be referenced within a given package (including packages recursively embedded within it). Contrast: *export*.
include	A relationship from a base use case to an inclusion use case, specifying how the behavior defined for the inclusion use case can be inserted into the behavior defined for the base use case.
inheritance	The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior. See *generalization*.
instance	An entity to which a set of operations can be applied and which has a state that stores the effects of the operations. See: *object*.
interaction	A specification of how stimuli are sent between instances to perform a specific task. The interaction is defined in the context of a collaboration. See *collaboration.*
interaction diagram	A generic term that applies to several types of diagrams that emphasize object interactions. These include: collaboration diagrams, sequence diagrams, and activity diagrams.
interface	A named set of operations that characterize the behavior of an element.

interface inheritance	The inheritance of the interface of a more specific element. Does not include inheritance of the implementation. Contrast: *implementation inheritance.*
internal transition	A transition signifying a response to an event without changing the state of an object.
layer	The organization of classifiers or packages at the same level of abstraction. A layer represents a horizontal slice through an architecture, whereas a partition represents a vertical slice. Contrast: *partition*.
link	A semantic connection among a tuple of objects. An instance of an association. See: *association.*
link end	An instance of an association end. See: *association end*.
message	A specification of the conveyance of information from one instance to another, with the expectation that activity will ensue. A message may specify the raising of a signal or the call of an operation.
metaclass	A class whose instances are classes. Metaclasses are typically used to construct metamodels.
meta-metamodel	A model that defines the language for expressing a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.
metamodel	A model that defines the language for expressing a model.
metaobject	A generic term for all metaentities in a metamodeling language. For example, metatypes, metaclasses, metaattributes, and metaassociations.
method	The implementation of an operation. It specifies the algorithm or procedure associated with an operation.
model	A semantically closed abstraction of a subject system. See: *system*.
[MOF]	Usage note: In the context of the MOF specification, which describes a meta-metamodel, for brevity the meta-metamodel is frequently to as simply the model.

model aspect	A dimension of modeling that emphasizes particular qualities of the metamodel. For example, the structural model aspect emphasizes the structural qualities of the metamodel.
model elaboration	The process of generating a repository type from a published model. Includes the generation of interfaces and implementations which allows repositories to be instantiated and populated based on, and in compliance with, the model elaborated.
model element	An element that is an abstraction drawn from the system being modeled. Contrast: *view element.*
[MOF]	In the MOF specification model elements are considered to be metaobjects.
modeling time	Refers to something that occurs during a modeling phase of the software development process. It includes analysis time and design time. Usage note: When discussing object systems, it is often important to distinguish between modeling-time and run-time concerns. See: *analysis time, design time*. Contrast: *run time*.
module	A software unit of storage and manipulation. Modules include source code modules, binary code modules, and executable code modules. See: *component.*
multiple classification	A semantic variation of generalization in which an object may belong directly to more than one class. See: *dynamic classification*.
multiple inheritance	A semantic variation of generalization in which a type may have more than one supertype. Contrast: *single inheritance*.
multiplicity	A specification of the range of allowable cardinalities that a set may assume. Multiplicity specifications may be given for roles within associations, parts within composites, repetitions, and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the non-negative integers. Contrast: *cardinality.*
multi-valued [MOF]	A model element with multiplicity defined whose Multiplicity Type:: upper attribute is set to a number greater than one. The term multi-valued does not pertain to the number of values held by an attribute, parameter, etc. at any point in time. Contrast: *single-valued*.

B OMG Modeling Glossary

n-ary association	An association among three or more classes. Each instance of the association is an n-tuple of values from the respective classes. Contrast: *binary association*.
name	A string used to identify a model element.
namespace	A part of the model in which the names may be defined and used. Within a namespace, each name has a unique meaning. See: *name*.
node	A node is classifier that represents a run-time computational resource, which generally has at least a memory and often processing capability. Run-time objects and components may reside on nodes.
object	An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class. See: *class, instance.*
object diagram	A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram. See: *class diagram, collaboration diagram.*
object flow state	A state in an activity graph that represents the passing of an object from the output of actions in one state to the input of actions in another state.
object lifeline	A line in a sequence diagram that represents the existence of an object over a period of time. See: *sequence diagram.*
operation	A service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.
package	A general purpose mechanism for organizing elements into groups. Packages may be nested within other packages.
parameter	The specification of a variable that can be changed, passed, or returned. A parameter may include a name, type, and direction. Parameters are used for operations, messages, and events. Synonyms: *formal parameter.* Contrast: *argument*.
parameterized element	The descriptor for a class with one or more unbound parameters. Synonym: *template*.

parent	In a generalization relationship, the generalization of another element, the child. See: *subclass*, *subtype*. Contrast: *child*.
participates	The connection of a model element to a relationship or to a reified relationship. For example, a class participates in an association, an actor participates in a use case.
partition	1. activity graphs: A portion of an activity graphs that organizes theresponsibilities for actions. See: *swimlane*. 2. architecture: A subset of classifiers or packages at the same level of abstraction. A partition represents a vertical slice through an architecture, whereas a layer represents a horizontal slice. Contrast: *layer.*
persistent object	An object that exists after the process or thread that created it has ceased to exist.
postcondition	A constraint that must be true at the completion of an operation.
precondition	A constraint that must be true when an operation is invoked.
primitive type	A pre-defined basic datatype without any substructure, such as an integer or a string.
process	1. A heavyweight unit of concurrency and execution in an operating system. Contrast: *thread*, which includes heavyweight and lightweight processes. If necessary, an implementation distinction can be made using stereotypes. 2. A software development process—the steps and guidelines by which to develop a system. 3. To execute an algorithm or otherwise handle something dynamically.
projection	A mapping from a set to a subset of it.
property	A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others may be user defined. See: tagged value.
pseudo-state	A vertex in a state machine that has the form of a state, but doesn't behave as a state. Pseudo-states include initial and history vertices.

published model [MOF]	A model which has been frozen, and becomes available for instantiating repositories and for the support in defining other models. A frozen model's model elements cannot be changed.
qualifier	An association attribute or tuple of attributes whose values partition the set of objects related to an object across an association.
receive [a message]	The handling of a stimulus passed from a sender instance. See: *sender, receiver.*
receiver [object]	The object handling a stimulus passed from a sender object. Contrast: *sender.*
reception	A declaration that a classifier is prepared to react to the receipt of a signal.
reference	1. A denotation of a model element. 2. A named slot within a classifier that facilitates navigation to other classifiers. Synonym: *pointer.*
refinement	A relationship that represents a fuller specification of something that has already been specified at a certain level of detail. For example, a design class is a refinement of an analysis class.
relationship	A semantic connection among model elements. Examples of relationships include associations and generalizations.
repository	A storage place for object models, interfaces, and implementations.
requirement	A desired feature, property, or behavior of a system.
responsibility	A contract or obligation of a classifier.
reuse	The use of a pre-existing artifact.
role	The named specific behavior of an entity participating in a particular context. A role may be static (e.g., an association end) or dynamic (e.g., a collaboration role).
run time	The period of time during which a computer program executes. Contrast: *modeling time*.

scenario	A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of a use case instance. See: *interaction.*
schema [MOF]	In the context of the MOF, a schema is analogous to a package which is a container of model elements. Schema corresponds to an MOF package. Contrast: *metamodel, package*.
semantic variation point	A point of variation in the semantics of a metamodel. It provides an intentional degree of freedom for the interpretation of the metamodel semantics.
send [a message]	The passing of a stimulus from a sender instance to a receiver instance. See: *sender, receiver.*
sender [object]	The object passing a stimulus to a receiver object. Contrast: *receiver.*
sequence diagram	A diagram that shows object interactions arranged in time sequence. In particular, it shows the objects participating in the interaction and the sequence of messages exchanged. Unlike a collaboration diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. See: *collaboration diagram*.
signal	The specification of an asynchronous stimulus communicated between instances. Signals may have parameters.
signature	The name and parameters of a behavioral feature. A signature may include an optional returned parameter.
single inheritance	A semantic variation of generalization in which a type may have only one supertype. Synonym: *multiple inheritance* [OMA]. Contrast: *multiple inheritance*.

single valued [MOF]	A model element with multiplicity defined is single valued when its Multiplicity Type:: upper attribute is set to one. The term single-valued does not pertain to the number of values held by an attribute, parameter, etc., at any point in time, since a single-valued attribute (for instance, with a multiplicity lower bound of zero) may have no value. Contrast: *multi-valued*.
specification	A declarative description of what something is or does. Contrast: *implementation*.
state	A condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event. Contrast: *state* [OMA].
statechart diagram	A diagram that shows a state machine. See: *state machine*.
state machine	A behavior that specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses and actions.
static classification	A semantic variation of generalization in which an object may not change type or may not change role. Contrast: *dynamic classification.*
stereotype	A new type of modeling element that extends the semantics of the metamodel. Stereotypes must be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML. See: *constraint, tagged value.*
stimulus	The passing of information from one instance to another, such as raising a signal or invoking an operation. The receipt of a signal is normally considered an event. See: *message*.
string	A sequence of text characters. The details of string representation depend on implementation, and may include character sets that support international characters and graphics.
structural feature	A static feature of a model element, such as an attribute.

structural model aspect	A model aspect that emphasizes the structure of the objects in a system, including their types, classes, relationships, attributes, and operations.
subactivity state	A state in an activity graph that represents the execution of a non-atomic sequence of steps that has some duration.
subclass	In a generalization relationship, the specialization of another class; the superclass. See: *generalization*. Contrast: *superclass*.
submachine state	A state in a state machine which is equivalent to a composite state but its contents is described by another state machine.
substate	A state that is part of a composite state. See: *concurrent state, disjoint state*.
subsystem	A subsystem is a grouping of model elements, of which some constitute a specification of the behavior offered by the other contained model elements. See *package*. See: *system*.
subtype	In a generalization relationship, the specialization of another type; the supertype. See: *generalization*. Contrast: *supertype*.
superclass	In a generalization relationship, the generalization of another class; the subclass. See: *generalization*. Contrast: *subclass*.
supertype	In a generalization relationship, the generalization of another type; the subtype. See: *generalization*. Contrast: *subtype*.
supplier	A classifier that provides services that can be invoked by others. Contrast: *client*.
swimlane	A partition on a activity diagram for organizing the responsibilities for actions. Swimlanes typically correspond to organizational units in a business model. See: *partition*.
synch state	A vertex in a state machine used for synchronizing the concurrent regions of a state machine.
synchronous action	A request where the sending object pauses to wait for results. Contrast: *asynchronous action*.

system	1. A collection of connected units that are organized to accomplish a specific purpose. A system can be described by one or more models, possibly from different viewpoints. Synonym: physical system. 2. A top-level subsystem.
tagged value	The explicit definition of a property as a name-value pair. In a tagged value, the name is referred as the tag. Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML. See: *constraint, stereotype.*
template	Synonym: *parameterized element*.
thread [of control]	A single path of execution through a program, a dynamic model, or some other representation of control flow. Also, a stereotype for the implementation of an active object as lightweight process. See *process*.
time	A value representing an absolute or relative moment in time.
time event	An event that denotes the time elapsed since the current state was entered. See: *event.*
time expression	An expression that resolves to an absolute or relative value of time.
timing mark	A denotation for the time at which an event or message occurs. Timing marks may be used in constraints.
trace	A dependency that indicates a historical or process relationship between two elements that represent the same concept without specific rules for deriving one from the other.
transient object	An object that exists only during the execution of the process or thread that created it.
transition	A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire.
type	A stereotype of class that is used to specify a domain of instances (objects) together with the operations applicable to the objects. A type may not contain any methods. See: *class, instance*. Contrast: *interface*.

type expression	An expression that evaluates to a reference to one or more types.
uninterpreted	A placeholder for a type or types whose implementation is not specified by the UML. Every uninterpreted value has a corresponding string representation. See: *any* [CORBA].
usage	A dependency in which one element (the client) requires the presence of another element (the supplier) for its correct functioning or implementation.
use case [class]	The specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system. See: *use case instances.*
use case diagram	A diagram that shows the relationships among actors and use cases within a system.
use case instance	The performance of a sequence of actions being specified in a use case. An instance of a use case. See: *use case class*.
use case model	A model that describes a system’s functional requirements in terms of use cases.
utility	A stereotype that groups global variables and procedures in the form of a class declaration. The utility attributes and operations become global variables and global procedures, respectively. A utility is not a fundamental modeling construct, but a programming convenience.
value	An element of a type domain.
vertex	A source or a target for a transition in a state machine. A vertex can be either a state or a pseudo-state. See: *state, pseudo-state.*
view	A projection of a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective.

view element	A view element is a textual and/or graphical projection of a collection of model elements.
view projection	A projection of model elements onto view elements. A view projection provides a location and a style for each view element.
visibility	An enumeration whose value (public, protected, or private) denotes how the model element to which it refers may be seen outside its enclosing namespace.