UML Semantics
Appendix M1-
UML Glossary

version 1.0
13 January 1997

ad/97-01-04

RATIONAL
SOFTWARE CORPORATION
2800 San Tomas Expressway
Santa Clara, CA 95051-0951
http://www.rational.com

1. INTRODUCTION

This glossary defines the terms that are used to describe the Unified Modeling Language (UML). In addition to UML-specific terminology it includes related terms from OMG standards and object-oriented analysis and design methods. The glossary is intended for use by anyone who wants to understand UML concepts and background. Glossary entries are organized alphabetically.

Please send any comments via e-mail to: *uml_feedback@rational.com.*

1.1 SCOPE

This glossary includes terms from the following primary sources:

- UML Semantics
- UML Notation Guide

In addition the following secondary sources have been used:

- Object Management Architecture object model [OMA]
- CORBA 2.0 [CORBA]
- Object Analysis & Design RFP-1 [OA&D RFP]
- Rational Process (work-in-progress by Philippe Kruchten, et al.) [RATLPROC]

[OMA], [CORBA] and [OA&D RFP] have been used to promote OMG-compliancy and provide distributed object terms that complement UML. (When there are inconsistencies among the three OMG sources, we have ranked their authority in the order listed.) [RATLPROC] has been used to provide architectural and process terms that complement UML.

1.2 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 by 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>*. This 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.

The glossary is extensively cross-referenced to assist in the location of terms that may be found in multiple places. The cross-references that are underlined are hyperlinks in the hypertext version. The hypertext version of the UML Glossary will appear on Rational Software's web site, *http://www.rational.com/uml.*

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 kind of entities. An abstraction defines a boundary relative to the perspective of the viewer.

action
A computational or algorithmic procedure.

action expression
An expression that resolves to a collection of actions.

action state
A state with an internal action and one or more outgoing transitions involving the completion of an internal action.

activation
The execution of an action. Contrast: activation [OMA].

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.

activity diagram
A special case of a state diagram in which all or most of the states are action states and in which all or most of the of the transitions are triggered by completion of actions in the source states. Contrast: state diagram.

actor [class]
A predefined stereotype of type denoting an entity outside the system that interacts with use cases.

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. Contrast: 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.

argument
A specific value corresponding to a parameter. 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.

association
A relationship that describes a set of links.

association class
A modeling 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 role
The role that a type or class plays in an association.

asynchronous message
A message where the sending object does not pause to wait for results. Synonym: asynchronous request [OMA]. Contrast: synchronous message.

attribute
A named property of a type. Synonym: attribute [OMA].

behavior
The observable effects of an operation or event, including its results. Synonym: behavior [OMA].

behavioral model aspect
A model aspect that emphasizes the behavior of the objects in a system, including their methods, interactions, collaborations, and state histories

binary association
An association between two classes. A special case of an n-ary association.

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.

class
A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class is an implementation of type. Synonym: class [OMG]. See: type, implementation.

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

client
A type, class, or component that requests a service from another type, class or component. Synonym: client object [OMA]. Contrast: supplier.

collaboration
A context that supports a set of interactions. See: interaction.

collaboration diagram
A diagram that shows object interactions organized around the objects and their links to each other. Unlike a sequence diagram a collaboration diagram shows the relationships among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. See: sequence diagram.

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
An executable software module with identity and a well-defined interface. Contrast: component [OMA].

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 substates. Contrast: substate.

composition
A form of aggregation 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 concurrent 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 extendibility mechanisms in UML. See: tagged value, stereotype. Contrast: note.

container
1. An object that exists to contain other objects, and that provides operations to access or iterate over its contents. For example, arrays, sets, dictionaries.
2. A component that exists to contain other components.

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

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. Synonym: delegation [OMA]. 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 concurrent 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. In the UML 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.

enumeration
A list of named values used as the range of a particular attribute type. For example, Color = {Red, Green, Blue}.

event
A significant occurrence. An event has a location in time and space and may have parameters. In the context of state diagrams, an event is an occurrence that can trigger a state transition.

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.

extends
A relationship from one use case to another, specifying how the behavior defined for the first use case can be inserted into the behavior defined for the second use case.

fire
To cause 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. Synonym: generalization [OMA]. See: inheritance.

guard condition
A condition that must be satisfied in order to cause 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. Synonym: implementation [OMA].

implementation inheritance

The inheritance of the implementation of a more specific element. Includes inheritance of the interface. Synonym: implementation inheritance [OMA]. 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: import [OMA], export.

inheritance

The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior. See generalization. Synonym: inheritance [OMA].

instance

An individual member described by a type or a class. Usage note: According to a strict interpretation of the metamodel an individual member of a type is an instance and a member of a class is an object. In less formal usage it is acceptable to refer to a member of a class as an object or an instance. See: type. Contrast: object.

interaction

A behavioral specification that comprises a set of message exchanges among a set of objects within a particular context to accomplish a specific purpose. An interaction may be illustrated by one or more scenarios.

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. See: activity diagram, collaboration diagram, sequence diagram.

interface

The use of a type to describe the externally visible behavior of a class, object, or other entity. In the case of a class or object, the interface includes the signatures of the operations. Synonym: interface [OMA]. See: type.

interface inheritance

The inheritance of the interface of a more specific element. Does not includes inheritance of the implementation. Synonym: interface inheritance [OMA]. Contrast: implementation inheritance.

layer

A specific way of grouping packages in a model at the same level of abstraction.

link

A semantic connection among a tuple of objects. An instance of an association. Synonym: link [OMA]. See: association.

link role

An instance of an association role. See: association role.

list
A container whose contents are ordered.

member
A part of a type or class denoting either an attribute or an operation.

message
A communication between objects that conveys information with the expectation that activity will ensue. The receipt of a message is normally considered an event.

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. An instance of a meta-metamodel.

metaobject
A generic term for all metaentities in a metamodeling language. For example, metatypes, metaclasses, metaattributes, and metaassociations. Synonym: metaobject [OMA].

metatype
A type whose instances are types. Metatypes are typically used to construct metamodels.

method
The implementation of an operation. The algorithm or procedure that effects the results of an operation. Synonym: method [OMA].

model
A semantically closed abstraction of a system. See: system.

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

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.

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 a run-time physical object that represents a computational resource, generally having at least a memory and often processing capability as well. Run-time objects and components may reside on nodes.

note

A comment attached to an element or a collection of elements. A note has no semantics. Contrast: constraint.

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 and methods. An object is an instance of a class. Synonym: object [OMA]. 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 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. Synonym: operation [OMA].

package
A general purpose mechanism for organizing elements into groups. Packages may be nested within other packages. A system may be thought of as a single high-level package, with everything else in the system contained in it.

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: parameter [OMA], *formal parameter*. Contrast: argument.

parameterized class
The descriptor for a class with one or more unbound parameters. Synonym: *template*.

participates
A relationship that indicates the role that an instance plays in a modeling element. For example, a class participates in an association, an actor participates in a use case. Contrast: participate [OMA].

persistent object
An object that exists after the process or thread that created it. Synonym: persistent object [OMA].

postcondition
An constraint that must be true at the completion of an operation.

powertype
A metatype whose instances are subtypes of another type. For example, TreeSpecies is a powertype on the Tree type. The subtypes of Tree (e.g., Ash, Birch, Cherry) are therefore all instances of TreeSpecies.

precondition
An constraint that must be true when an operation is invoked.

primitive type
A predefined basic type, such as an integer or a string.

process
A thread that can execute concurrently with other threads.

product
The artifacts of development, such as models, code, documentation, work plans.

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. Synonym: property [OMA].

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, final, and history connections.

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 message passed from a sender object. See: sender, receiver.

receiver [object]
The object handling a message passed from a sender object. Contrast: sender.

reference
A denotation of a model element.

refinement
A relationship that represents the 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.

requirement
A desired feature, property, or behavior of a system.

responsibility
A contract or obligation of a type or class.

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 role) 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. See: interaction.

semantic variation
A particular interpretation choice for a semantic variation point. See: semantic variation point.

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. See: semantic variation.

send [a message]
The passing of a message from a sender object to a receiver object. See: sender, receiver.

sender [object]
The object passing a message 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
A named event that can be explicitly invoked ("raised"). Signals may have parameters. A signal may be broadcast or directed toward a single object or a set of objects.

signature
The name and parameters of an operation, message, or event. Parameters may include an optional returned parameter. Synonym: signature [OMA].

single inheritance
A semantic variation of generalization in which a type may have only one supertype. Synonym: single inheritance [OMA]. Contrast: multiple inheritance.

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].

state 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 extendibility mechanisms in UML. See: constraint, tagged value.

string

A sequence of text characters. The details of string representation depends on implementation, and may include character sets that support international characters and graphics.

structural model aspect

A model aspect that emphasizes the structure of the objects in a system, including their types, classes, relationships, attributes and operations.

subclass

In a generalization relationship the specialization of another class, the superclass. See: generalization. Contrast: superclass.

substate

A state that is part of a composite state. A substate can either be a concurrent or disjoint substate. See: concurrent state, disjoint state.

subsystem

A subordinate system within a larger system. In the UML a subsystem is modeled as a package of components. Contrast: 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. Synonym: supertype [OMA]. See: generalization. Contrast: subtype.

supplier

A type, class or component that provides services that can be invoked by others. Synonym: server object [OMA]. Contrast: client.

swimlane

A package on interaction diagrams for organizing responsibilities for actions. They often correspond to organizational units in a business model.

synchronous message

A message where the sending object pauses to wait for results. Synonym: synchronous request [OMA]. Contrast: asynchronous message.

system

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.

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 extendibility mechanisms in UML. See: constraint, stereotype.

template

Synonym: parameterized class.

thread [of control]

A single path of execution through a program, a dynamic model, or some other representation of control flow. Synonym: *thread of control*. See process.

time

A value representing an absolute or relative moment in time.

time event

An event that occurs at a particular time. It may be expressed as a time expression. 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.

transient object

An object that exists only during the execution of the process or thread that created it. Synonym: transient object [OMA].

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 description of a set of instances that share the same operations, abstract attributes and relationships, and semantics. A type may define an operation specification (such as a signature) but not an operation implementation (such as a method). Usage note: Type is sometimes used synonymously with interface, but it is not an equivalent term. Synonym: type [OMA]. 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].

use case [class]
A class that defines a set of use case instances.

use case diagram
A diagram that shows the relationships among actors and use cases within a system.

use case instance
A sequence of actions a system performs that yields an observable result of value to a particular actor. Usually scenarios illustrate prototypical use case instances. An instance of a use case class. See: use case class.

use case model
A model that describes a system's functional requirements in terms of use cases.

uses
A relationship from a concrete use case to an abstract use case in which the behavior defined for the concrete use case employs the behavior defined for the abstract use case.

utility
A stereotype of type 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. Contrast: value [OMA].

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, private, or implementation) denotes how the model element to which it refers may be seen outside its enclosing name space.