Glossary *A*

This glossary contains terms used throughout this guide. Text in *italics* is for clarification only.

For more information about object-oriented terminology, refer to the Semaphore *Glossary of Object-Oriented Terminology*. (Semaphore email: 74743.16@compuserve.com).

activation Copying the persistent form of methods and stored data into an executable address space to allow execution of the methods on the stored data.

application A dynamic object-based application is the (end-user) functionality provided by one or more programs consisting of a collection of interoperating objects. *In common terminology this is usually referred to as a running application or process.* A static object-based application is a set of related types and classes specific to a particular (end user) objective. *In common terminology this is usually referred to as a program*.

application facilities Comprise facilities that are useful within a specific application domain. See *common facilities*.

application objects Applications and their components that are managed within an object-oriented system. Example operations on such objects are open, install, move and remove.

asynchronous request A request where the client object does not pause to wait for results.

atomicity The property that ensures that an operation either changes the state associated with all participating objects consistent with the request, or changes none at all. If a set of operations is atomic, then multiple requests for those operations are serializable.

attribute A conceptual notion. An attribute of an object is an identifiable association between the object and some other entity or entities. Typically, the association is revealed by an operation with a single parameter identifying the object. See related definition for *property*.

behavior The behavior of a request is the observable effects of performing the requested service (including its results).

behavior consistency Ensures that the behavior of an object maintains its state consistency.

binding (or, more specifically, **method binding**) The selection of the method to perform a requested service and of the data to be accessed by that method. See also *dynamic binding* and *static binding*.

class An implementation that can be instantiated to create multiple objects with the same behavior. An object is an instance of a class. Types classify objects according to a common interface; classes classify objects according to a common implementation.

class inheritance The construction of a class by incremental modification of other classes.

class object An object that serves as a class. A class object serves as a **factory**. See *factory*.

client object An object issuing a request for a service. See also *server object. A given object may be a client for some requests and a server for other requests.*

common facilities Provides facilities useful in many application domains and which are made available through OMA-compliant class interfaces. See also *application facilities*.

component A conceptual notion. A component is an object that is considered to be part of some containing object.

compound object A conceptual notion. A compound object is an object that is viewed as standing for a set of related objects.

conformance A relation defined over types such that type x conforms to type *y* if any value that satisfies type *x* also satisfies type *y*.

context-independent operation An operation where all requests that identify the operation have the same behavior. (In contrast, the effect of a context-dependent operation might depend upon the identity or location of the client object issuing the request.)

core object model Basic object model that forms the basis for the OMG's Object Management Architecture; formally defined in Chapter 4 of this guide. The Core Object Model defines concepts such as object and non-object types, operations, signatures, parameters, return values, interfaces, substitutability, inheritance, and subtyping.

data model A collection of entities, operators, and consistency rules.

delegation The ability for a method to issue a request in such a way that self-reference in the method performing the request returns the same object(s) as self-reference in the method issuing the request. See *self-reference*.

dynamic binding Binding that is performed after the request is issued (see *binding*).

embedding Creating an object out of a non-object entity by wrapping it in an appropriate shell.

exchange format The form of a description used to import and export objects.

export To transmit a description of an object to an external entity.

extension of a type The set of values that satisfy the type.

factory A conceptual notion. A factory provides a service for creating new objects.

generalization The inverse of the specialization relation.

generic operation A conceptual notion. An operation is generic if it can be bound to more than one method.

handle A value that unambiguously identifies an object. See also *object name*.

implementation A definition that provides the information needed to create an object, allowing the object to participate in providing an appropriate set of services. An implementation typically includes a description of the data structure used to represent the core state associated with an object, as well as definitions of the methods that access that data structure. It also typically includes information about the intended type of the object.

implementation inheritance The construction of an implementation by incremental modification of other implementations.

import Creating an object based on a description of an object transmitted from an external entity.

inheritance The construction of a definition by incremental modification of other definitions. See also *implementation inheritance*.

instance An object created by instantiating a class. An object is an instance of a class.

instantiation Object creation.

interface A description of a set of possible uses of an object. Specifically, an interface describes a set of potential requests in which an object can participate meaningfully. See also *object interface*, *principal interface*, and *type interface*.

interface inheritance The construction of a new interface using one or more existing interfaces as its basis. The new interface is called a subtype and the existing interfaces are its supertypes.

interface type A type that is satisfied by any object (literally, by any value that identifies an object) that satisfies a particular interface. See also *object type*.

interoperability The ability to exchange requests using the ORB in conformance with the OMG Architecture Guide. *Objects interoperate if the methods of one object request services of another.*

link A conceptual notion. A relation between two objects.

literal A value that identifies an entity that is not an object. See also *object name*.

meaningful request A request where the actual parameters satisfy the signature of the named operation.

metaobject An object that represents a type, operation, class, method, or other object model entity that describes objects.

method Code that may be executed to perform a requested service. *Methods associated with an object may be structured into one or more programs.*

method binding See *binding*.

multiple inheritance The construction of a definition by incremental modification of more than one other definition.

non-object A member of the set of denotable values. Non-objects are not labeled by an object reference.

object A combination of a state and a set of methods that explicitly embodies an abstraction characterized by the behavior of relevant requests. An object is an instance of a class. *An object models a real world entity and is implemented as a computational entity that encapsulates state and operations (internally implemented as data and methods) and responds to requests for services.*

A basic characteristic of an object is its distinct object identity, which is immutable, persists for as long as the object exists, and is independent of the object's properties or behavior.

Methods can be owned by one or more objects.

Requests can be sent to zero, one, or more objects.

State data can be owned by one or more objects.

State data and methods can be located at one or more locations.

object creation An event that causes an object to exist that is distinct from any other object.

object destruction An event that causes an object to cease to exist and its associated resources to become available for reuse.

object interface A description of a set of possible uses of an object. Specifically, an interface describes a set of potential requests in which an object can meaningfully participate as a parameter. It is the union of the object's type interfaces.

object name A value that identifies an object. See also *handle*.

object services A collection of interfaces and objects that support basic functions for using and implementing objects. Object Services are necessary to construct any distributed application and are independent of application domains. Interfaces for object services are specified by OMG in *CORBAservices* and currently include Life Cycle, Events, Naming, Persistent Object, Transaction, Concurrency Control, Relationships, and Externalization.

object type A type the extension of which is a set of objects (literally, a set of values that identify objects). In other words, an object type is satisfied only by (values that identify) objects. See also *interface type*.

OMA-compliant application An application consisting of a set of interworking classes and instances that interact via the ORB. Compliance therefore means conformance to the OMA protocol definitions and interface specifications outlined in this document.

OMG IDL Object Management Group Interface Definition Language. A programming language–independent way to specify object interfaces. OMG IDL must be used to specify all object interfaces in a CORBA-compliant system; it is used only for specifications, not for programming. The specification for OMG IDL is contained in *CORBA*.

operation A service that can be requested. An operation has an associated signature, which may restrict which actual parameters are possible in a meaningful request.

operation name A name used in a request to identify an operation.

ORB (Object Request Broker) provides the means by which objects make and receive requests and responses.

parameter Part of an operation's signature. It gives the type and name of an argument to the operation.

participate An object participates in a request if one or more of the actual parameters of the request identifies the object.

passivation The reverse of activation.

persistent object An object that can survive the process or thread that created it. A persistent object exists until it is explicitly deleted.

principal interface The interface that describes all requests in which an object is meaningful.

property A conceptual notion. An attribute the value of which can be changed.

protection The ability to restrict the client objects for which a requested service will be performed.

query An activity that involves selecting objects from implicitly or explicitly identified collections based on a specified predicate.

referential integrity The property that ensures that a handle which exists in the state associated with another object reliably identifies a single object.

request An event consisting of an operation and zero or more actual parameters. A client issues a request to cause a service to be performed. Also associated with a request are the results that may be returned to the client. *A message can be used to implement (carry) a request and/or a result.*

result The information returned to the client, which may include values as well as status information indicating that exceptional conditions were raised in attempting to perform the requested service.

security domain An identifiable subset of computational resources used to define security policy.

self-reference The ability of a method to determine the object(s) identified in the request for the service being performed by the method. (Self-reference in Smalltalk is indicated by the keyword *self*. See also *delegation*.)

server object An object providing response to a request for a service. See also *client object*. *A given object may be a client for some requests and a server for other requests.*

service A computation that may be performed in response to a request.

signature Defines the types of the parameters for a given operation.

single inheritance The construction of a definition by incremental modification of one definition. See also *multiple inheritance*.

specialization A class *x* is a specialization of a class *y* if *x* is defined to directly or indirectly inherit from *y*.

state The information about the history of previous requests needed to determine the behavior of future requests.

state consistency Ensures that the state associated with an object conforms to the data model.

state integrity Requires that the state associated with an object is not corrupted by external events.

state-modifying request A request that by performing the service alters the results of future requests.

state variable Part of the state of an object.

static binding Binding that is performed prior to the actual issuing of the request. See also *binding*.

supertype When an object of type A is substitutable for an object of type B, A is a subtype of B, and B is a supertype of A. Although the Core Object Model effectively defines its inheritance to be the same as subtyping, the two concepts are separate. Subtyping determines substitutability, whereas inheritance is a mechanism for specifying one entity in terms of another.

synchronous request A request where the client object pauses to wait for completion of the request.

transient object An object the existence of which is limited by the lifetime of the process or thread that created it.

type A predicate (Boolean function) defined over values that can be used in a signature to restrict a possible parameter or characterize a possible result. Types classify objects according to a common interface; classes classify objects according to a common implementation.

type interface Defines the requests in which instances of this type can meaningfully participate as a parameter. *Example: given that document type and product type the interface to document type comprises edit and print, and the interface to product type comprises set price and check inventory, then the object interface of a particular document which is also a product comprises all four requests.*

type object An object that serves as a type.

value Any entity that can be a possible actual parameter in a request. Values that serve to identify objects are called "object names." Values that identify other entities are called "literals."

value-dependent operation An operation where the behavior of the corresponding requests depends upon which names are used to identify object parameters (if an object can have multiple names).