An Object-Oriented Visual Glossary

An AmbySoft Inc. White Paper

Scott W. Ambler
Object-Oriented Consultant
AmbySoft Inc.

Material for this White Paper has been taken from
Building Object Applications That Work
by Scott W. Ambler
SIGS Books, 1997

This Version: November 29, 1997

Table Of Contents

1. Introduction

The terminology presented in this white paper is taken from by second book, *Building Object Applications That Work* (SIGS Books, 1997). One of the things that you will quickly notice is that the terminology ranges widely. Yes, I describe terms such as inheritance and polymorphism, but I also describe terms such as client/server, function testing, and data model because OO development requires that you have a much wider understanding of development than just "standard" OO terminology. This is why *Building Object Applications That Work* covers such a broad range of topics – there's more to OO development than OO modeling and OO coding.

Building Object Applications That Work covers:

- Architecting your applications so that they're maintainable and extensible.
- Analysis and design techniques using the Unified Modeling Language (UML).
- Going beyond the UML to meet the actual needs of OO development.
- Applying OO patterns to improve the quality of your applications.
- Creating applications for stand-alone, client/server, and distributed environments.
- Using both relational and OO databases for persistence.
- User interface design so your users can actually work with the systems that you build.
- Coding applications in a way that makes them maintainable and extensible.
- Wrapping legacy applications to make them appear OO.
- OO metrics.
- OO testing (it's harder, not easier).

2. Object-Oriented Terminology

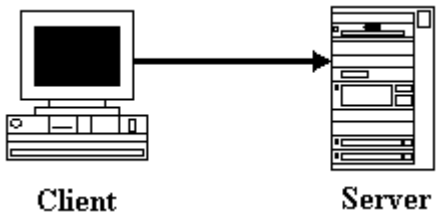
1NF – See First normal form.

1ONF – See First object normal form.

2NF – See Second normal form.

2ONF – See Second object normal form.

2-tier client/server -- Clients and servers communicate with one another in a direct and highly coupled manner.

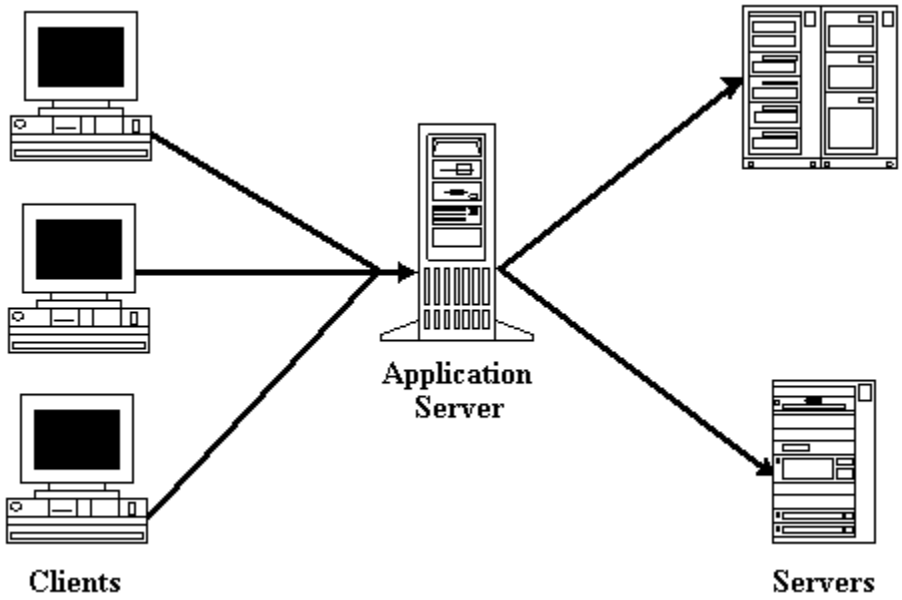


2-tier client/server architecture.

3NF – See Third normal form.

3ONF – See Third object normal form.

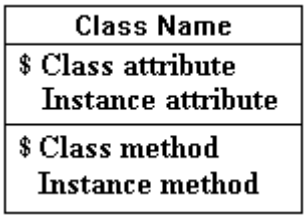
3-tier client/server -- In this client/server architecture client machines send requests to an application server, which then sends requests to other servers on the network to fulfill the original request.



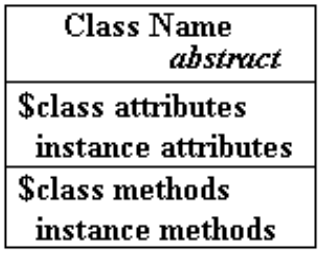
3-tier client/server architecture.

2.1 A-C

Abstract class -- A class that does not have objects instantiated from it, but will provide functionality inherited by its subclasses.



Abstract-class notation (Ambler).



Abstract-class notation (UML).

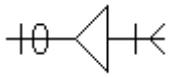
Abstraction -- The definition of the interface of a class (what it knows and does).

Accessor method – A method that is used to either modify or retrieve a single attribute.

Active persistent object -- An object that exists in an object database that can be sent messages both from within and from without the object database.

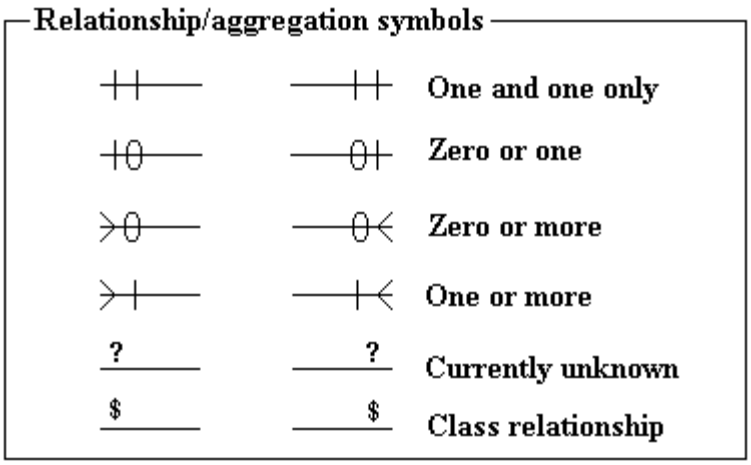
Actor – A person, organization, or external system that interacts with the application that we are currently developing.

Aggregation -- Represents “is-part-of” relationships.



Aggregation notation (Ambler).

Aggregation notation (UML).



Symbol combinations for aggregate and instance relationships (Ambler).

Alpha testing -- A testing period in which pre release versions of software products that are often very buggy are released to users who need access to the product before it is to be officially released. In return these users are willing to report back to the software developers any problems they uncover. Alpha testing is typically followed by a period of beta testing.

Analysis error -- When a user requirement is missed or misunderstood.

Analysis pattern -- An OO pattern that describes a solution to a business/analysis problem.

API – See Application-programming interface.

Application-programming interface (API) -- A set of function/procedure calls that access a component that is external to your system.

Application server -- A component of a 3-tier C/S architecture that encapsulates access to other servers on the network, supplying the business logic for combining the responses from those servers.

Application-specific class -- Any class that is used in a single application.

Association -- Another term for instance relationship.

Associative table -- A table in a relational database that is used to maintain a relationship between two or more other tables. Associative tables are typically used to resolve many-to-many relationships.

Attribute -- Something that an object or class knows. An attribute is basically a single piece of data or information. Attributes can be simple, like a string or integer, or can be a complex object, like an address or customer.

BDE – See Business-domain expert.

Beta testing -- Similar to alpha testing except that the software product is usually less buggy. This approach is typically used by software development companies who want to ensure that they meet as many of their client needs as possible.

Black-box tests -- Test cases that verify that given input A the component/system being tested gives you expected results B.

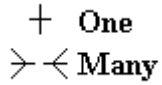
Boundary-value tests -- Test cases that test unusual or extreme situations that your code should be able to handle.

Business classes -- Business classes model the business domain. Business classes are usually found during the analysis process.

Business-domain expert (BDE) -- Someone with intimate knowledge of all or a portion of the problem domain that you are modeling.

Callback methods -- When object A communicates with object B, it may request that at some time in the future object B sends message M whenever a certain event happens (such as a process ending). The method that responds to message M is named a callback method.

Cardinality -- Indicates how many objects are involved in a relationship.



Cardinality symbols (Ambler).

CASE – Computer-aided system engineering.

CASE tool -- A tool that supports the creation of models and potentially both forward and reverse engineering.

Class -- A category of similar objects. A class is effectively a blueprint from which objects are created.

Class attribute -- Information that is applicable to an entire class of objects.

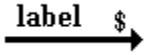
Class diagram -- Class diagrams show the classes of the system, their intrarelationships, and the collaborations between those classes.

Class hierarchy -- A set of classes that are related through inheritance.

Class-integration testing -- The act of ensuring that the classes, and their instances, that form an application perform as defined.

Class library -- A collection of classes, typically purchased off-the-shelf, which you can reuse and extend via inheritance.

Class message – A message that is sent to a class, instead of to the instance of a class.



Class-message notation (Ambler).

Class method -- A method that operates on a class.

Class normalization -- A process in which you organize the behavior within a class diagram in such a way as to increase the cohesion of classes while minimizing the coupling between them.

Class testing -- The act of ensuring that a class and its instances (objects) perform as defined.

Class-type architecture -- The classes of an application are organized into well-encapsulated layers according to their general properties. The interaction between classes is often restricted based on the layer they belong to.

Client -- A client is a single-user PC or workstation that provides presentation services and appropriate computing, connectivity, and interfaces relevant to the business need. A client is also commonly referred to as a "front-end."

Client class -- A class that sends messages but does not receive them.

Client/server class -- A class that both sends a receives messages.

Client/server (C/S) computing -- An environment that satisfies the business need by appropriately allocating the application processing between the client and the server processes.

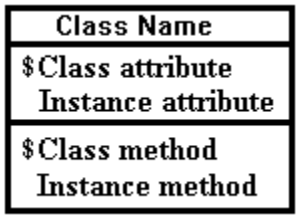
Cohesion - A measure of how much a method or class makes sense.

Collaboration -- Classes work together (collaborate) to get things done.

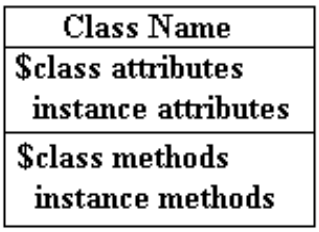
Common Object Request Broker Architecture (CORBA) -- An OMG specification defining a distributed-object architecture. CORBA specifies how to develop OO applications that are able to connect and communicate with other CORBA-compliant (and potentially non-OO) applications.

Component -- Reusable code, typically purchased off-the-shelf, which you can reuse but not extend via inheritance.

Concrete class -- A class that has objects instantiated (created) from it.



Concrete-class notation (Ambler).



Concrete-class notation (UML).

Concurrency -- The issues involved with allowing multiple people simultaneous access to your persistence mechanism.

Constructor -- A C++ member function (method) that allocates the memory needed for an instance of a class. Constructors are also used to set the initial value for attributes too.

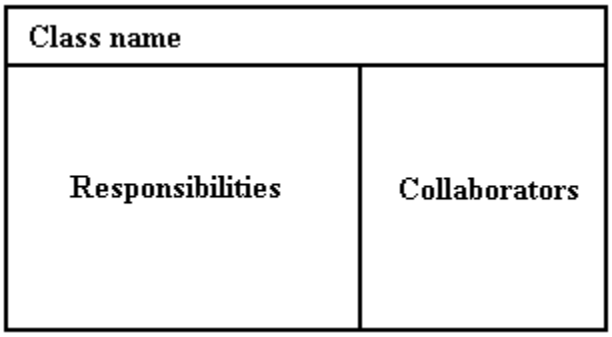
Contract -- Any service/behavior of an object or server that other objects or servers request of it.

CORBA – See Common Object Request Broker Architecture.

Coupling - A measure of how connected two classes are.

Coverage testing -- The act of ensuring that all lines of code were exercised at least once.

CRC (Class Responsibility Collaborator) card -- A standard index card divided into three sections showing the name of the class, the responsibilities of the class, and the collaborators of the class.



CRC card notation.

CRC model -- A collection of CRC cards that describe the classes that make up a system or a component of a system.

CRC modeling -- The act of creating a CRC model.

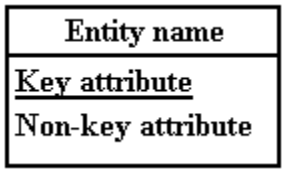
CRUD -- Acronym for create, retrieve, update, delete. The basic functionality that a persistence mechanism must support.

2.2 D-F

Database proxies -- An object that represents a business object stored in a database. To every other object in the system the database proxy appears to be the object that it represents. When other objects send the proxy a message it immediately fetches the object from the database and replaces itself with the fetched object, passing the message onto it. See the Proxy pattern in chapter 4 for more details.

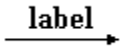
Data dictionary -- A repository of information about the layout of a database, the layout of a flat file, the layout of a class, and any mappings among the three.

Data entity – A person, place, thing, event, or concept. Data entities are drawn on data models and are similar to classes with the exception that they have data attributes, but do not have functionality (methods).



Data-entity notation.

Data flow – In a process model a data flow represents the movement of information, either physical or electronic, from one source to another.

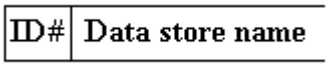


Data-flow notation.

Data model -- A diagram used to communicate the design of a (typically relational) database. Data models are often referred to as entity-relationship (ER) diagrams.

Data normalization -- A process in which data in a relational database is organized in such a way as to reduce and even eliminate data redundancy.

Data store – In a process model it is a place where information is stored, such as a database or filing cabinet.



Gane & Sarson data-store notation.

Data warehouse -- A large database, almost always relational, that is used to store data for reporting purposes.

Design pattern -- An OO pattern that describes a solution to a design problem.

Development/maintenance trade-off -- Development techniques that speed up the development process often have a negative impact on your maintenance efforts, whereas techniques that lead to greater maintainability negatively impact your development efforts, at least in the short term.

DFD – See Data-flow diagram.

Distributed classes -- An architecture in which logic of your applications is organized by putting classes on computers on your network based on their specific behaviors.

Distributed objects -- An architecture in which objects dynamically reside on the machine that is most appropriate at the time. Objects move freely about the network and are not limited to where they may go.

do/ -- A keyword used on a state diagram to document actions taken by an object while in a state.

Drag and drop -- A technique in which a person uses a pointing device (typically a mouse) to select an object on the screen and then uses the mouse to move the object on top of another screen object.

DSL – See Dynamic shared library.

Dynamic model – See State diagram.

Dynamic shared library (DSL) – A library of function/procedure calls that exists outside of an application that is linked in at run time. Also known as a dynamic-link library.

Dynamic SQL -- An SQL statement that is generated by an application at run time and then processed against the database. This is more flexible but much slower than static SQL.

Dynamic typing – The class/type of an object is associated to it at run time.

Electronic commerce -- Any form of commerce in which the buyer of a product or service uses a computer to interact with the computer system of the seller of that product or service.

Encapsulation -- The hiding of the implementation of what a class/object knows or does, without telling anyone how it's done.

Extensibility -- A measure of how easy it is to add new features to a system. The easier it is to add new features, the more extensible we say the system is.

External entity – In a process model it is the source or destination of data that is external to the system being modeled. In a class diagram we would call this an actor class.

External-entity notation.

Fat-client approach -- A 2-tier C/S architecture in which client machines implement both the user interface and the business logic of an application. Servers typically only supply data to client machines without little or no processing done to it.

Final state – A state (in a state diagram) from which no transitions lead out of. Objects will have zero or more final states.

First normal form (1NF) -- A data entity is in 1NF when it contains no repeating groups of data. A database is in 1NF when all of its data entities are in 1NF.

First object normal form (1ONF) -- A class is in 1ONF when specific behavior required by an attribute that is actually a collection of similar attributes is encapsulated within its own class. A class diagram is in 1ONF when all of its classes are in 1ONF.

Flat file -- A single data file in which information is stored.

FLOOT – See Full life cycle object-oriented testing.

Foreign key -- An attribute(s) of a data entity that make up the primary key of another data entity. Foreign keys are used to maintain relationships between data entities.

Forward engineering -- The generation of source code from a model by a CASE tool.

Framework -- A collection of classes that together provide sophisticated, generic functionality, which can be extended via the addition of subclasses to meet your specific needs.

Full life cycle object-oriented testing (FLOOT) -- A testing methodology for object-oriented development that comprises testing techniques that taken together provide methods to verify that your application works correctly at each stage of development.

Function testing -- A part of systems testing in which development staff confirm that their application meets the user requirements specified during analysis.

2.3 G-K

Garbage collection – Memory management in an OO application.

Garbage collector – The object(s) responsible for garbage collection.

Getter method – An accessor method that retrieves the value of an attribute.

Human factors -- The study of how people interact with machines.

Hybrid OO language – An OO language that supports both structured/procedural and OO programming constructs.

IDL – See Interface-definition language.

Ignored method -- An inherited method that is overridden with a method that has no functionality.

Information hiding --The restriction of access to attributes.

Inheritance – A concept that allows us to take advantage of similarities between classes by representing "is-a" and "is-like" relationships.

Inheritance notation (Ambler).

Inheritance notation (UML).

Inheritance-regression testing -- The act of running the test cases of all the superclasses, both direct and indirect, on a given subclass.

Inheritance-tree depth -- The maximum number of classes from the root of a class hierarchy to it's lowest node, including the root class.

Initial state – The state in which an object is in when it is first created. All objects have an initial state.

Initialize method -- A Smalltalk method that sets the initial values for attributes. In Smalltalk memory is allocated automatically when objects are instantiated.

Installation testing -- The act of ensuring that your application can be installed successfully.

Instance -- Another word for object. We say that an object is an *instance* of a class.

Instance attribute -- Information that is specific to a single object.

Instance method -- A method that operates on an individual object.

Instance relationship -- Relationships, or associations, exist between objects. For example, customers BUY products.

Instantiate -- To create an instance. When we create an object we say that we *instantiate* it from a class.

Interface -- The set of messages an object or class will respond to.

Interface classes -- Interface classes provide the ability for users to interact with the system. Interface classes typically define a graphical user interface (GUI) for an application, although other interface styles such as voice command or handwritten input are also implemented via interface classes.

Interface-definition language (IDL) -- A standard language for defining the interface of an object.

Interface-flow diagram – A diagram that models the interface objects of your system and the relationships between them. Also referred to as storyboards.

Interface object – An object displayed as part of the user interface for an application. This includes widgets such as buttons and list boxes, icons, screens, and reports.

Internet -- A collection of interconnected computers that people can log onto to share information, to communicate, to be entertained, and to perform electronic-commerce transactions.

Intranet -- A network internal to your organization that is built either partially or completely from Internet-based technology.

ISO 9001 -- A standard defined by the International Standards Organization (ISO) that defines how organizations should manage their quality-assurance programs.

ISO 9003 -- The standards defining how organizations should manage their software quality-assurance programs.

Java applet – Small programs written in Java that are transmitted to, and run on, a client workstation from a server.

Java virtual machine (Java VM) – The interpretive environment that runs Java applets.

Key -- One or more columns in a relational data table that when combined form a unique identifier for each record in the table.

2.4 L-N

Layer -- A (class-type) layer encapsulates the broad functionality of a collection of classes that exhibit similar behaviors. Layers help us to identify, define, and potentially restrict how classes interact with one another.

Lazy initialization -- An approach in which the initial value of attributes are set in their corresponding getter methods.

Leaf class -- A class within an inheritance hierarchy that doesn't have other class inheriting from it.

Legacy application -- Any application or system that is difficult, if not impossible, to maintain and enhance.

Lock -- An indication that a table, record, class, object, and so on is reserved so that work can be accomplished on the item being locked. A lock is established, the work is done, and the lock is removed.

Maintainability -- A measure of how easy it is to add, remove, or modify existing features of a system. The easier a system is to change the more maintainable that system is.

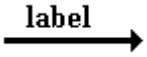
Master test plan -- A document, typically created at the beginning of the project but that is updated throughout, that describes your testing strategies. Other test plans are included as components of the master test plan.

Member function -- The C++ term for method.

Memory leak – When you haven't properly managed the removal of an object from memory and have effectively lost some of the memory space that it takes up.

Mental model -- An internal representation of a person's conceptualization and understanding of a system.

Message -- A message is either a request for information or a request to do something.



Message notation (Ambler & UML).

Message dispatcher -- An object that exists solely to pass messages onto other objects. Objects will often register themselves with a message dispatcher to inform it of the events that they are interested in being informed about.

Message-invocation box – The long, thin vertical boxes that appear on sequence diagrams that represent a method invocation in an object.

Message-invocation box notation.

Messaging -- In order to collaborate, classes send messages to each other.

Metaphor -- A set of concepts, terms, and objects that the user is familiar with that are used in the design of a user interface to make it easier to understand and use.

Method -- Something that an object or class does. A method is similar to a function or procedure in structured programming.

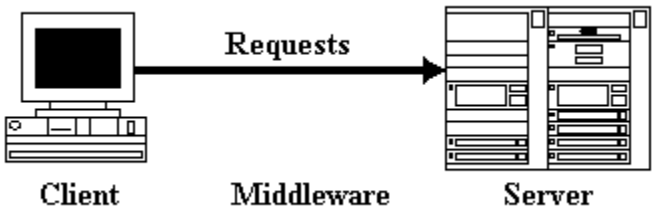
Method response -- A count of the total number of messages that are sent as a result of a method being invoked.

Method testing -- The act of ensuring that a method (member function) performs as defined.

Methodology – In the context of systems development, it is the collection of techniques and approaches that you take when creating systems.

Metric -- A measurement. In our case, a measurement of some factor involved in OO development.

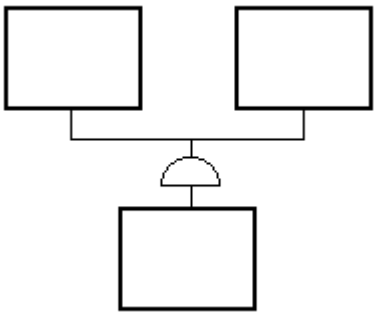
Middleware -- The technology that allows clients and servers to communicate with one another. This includes the network itself, its operating system, and anything needed to connect computers to the network.



Middleware.

MIS -- Management Information System.

Multiple inheritance -- When a class directly inherits from more than one class, we say that we have multiple inheritance. Note that not all OO languages support multiple inheritance.



Multiple inheritance (Ambler).

Non key attribute -- A non key attribute is any attribute of a relational table that is not part of the primary key.

Notation – The set of symbols that are used to document the analysis/design of a system.

2.5 O-R

Object -- Any person, place, thing, event, screen, report, or concept that is applicable to the design of the system. Objects have both data and functionality that define its behavior.

Object adapter -- A mechanism that both converts objects to records that can be written to a persistence mechanism and converts records back into objects again. Object adapters can also be used to convert between objects and flat file records.

Object Database Management Group (ODMG) -- A consortium of most of the ODBMS vendors who together set standards for object databases.

Object identifier -- An attribute that uniquely identifies an object. The object-oriented equivalent of a key.

Object Management Group (OMG) -- A consortium of organizations that work together to create standards for the distributed object computing.

Object-oriented database management system (OODBMS) -- A persistence mechanism that stores objects, including both their attributes and their methods.

Object-oriented database system manifesto -- The “13 golden rules” that define what it is to be an OODBMS.

Object-oriented test case -- A collection of objects that are in states that are appropriate to what is being tested, message sends to those objects, and the expected results of those message sends that together verify that a specific feature within your application works properly.’

Object-oriented user interface (OOUI) -- A style of user interface in which users directly interact with objects using the computer just as they work with them in the real world.

Object query language (OQL) -- A standard proposed by the ODMG for the selection of objects. Basically SQL with object-oriented extensions that provide the ability to work with classes and objects instead of tables and records.

Object/relational database – A persistence mechanism that encompasses the features of both relational databases and OODBMSs by allowing you to store both data and objects.

Object/relational impedance mismatch -- The difference resulting from the fact that relational theory is based on relationships between tuples that are queried, whereas the object paradigm is based on relationships between objects that are traversed.

Object request broker (ORB) -- A middleware technology that allows objects to send messages across the network to other objects.

Object streaming -- A process in which an object is converted to data so that it can be stored or transmitted, eventually being converted back into an object afterward.

ODBC – See Open database connectivity.

ODMG – See Object Database Management Group.

OID – See Object identifier.

OOCASE tool -- A CASE tool that supports OO development.

OOCRUD -- The object-oriented create, retrieve, update, and delete functionality performed by persistence classes.

OOD – Object-oriented design.

OODBMS – See Object-oriented database management system.

OO pattern -- A model of several classes that work together to solve a common problem in your application's business or technical domain.

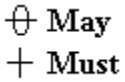
OOUI – See Object-oriented user interface.

Open database connectivity (ODBC) -- A Microsoft standard for accessing relational databases. Effectively a standard for defining a database access wrapper that allows database vendors to provide a common interface to their product.

Operations testing -- The act of ensuring that the needs of operations personnel who have to support the application are met.

Optimistic locking -- An approach to concurrency in which an item is locked only for the time that it is accessed in the persistence mechanism. This strategy allows many people to work with an object simultaneously, but also presents the opportunity for people to overwrite the work of others (we'll discuss this later).

Optionality -- Indicates whether or not it is mandatory that other objects are involved in a relationship.



Optionality symbols (Ambler).

OQL – See Object query language.

Overload -- See override.

Override -- A term used to indicate that we redefine attributes and/or methods in subclasses to provide slightly or completely different behavior.

Paragraphing -- The indenting of your code to make it more readable.

Path testing -- The act of ensuring that all logic paths within your code were exercised at least once. This is a superset of coverage testing.

Peer-to-peer architecture -- An architecture based on the concept that any computer can potentially send messages to any other computer on the network.

Peer-to-peer wrapping – A legacy-application wrapping technique in which wrapper code in an OO application communicates with wrapper code on the machine running the legacy application that directly accesses its functionality.

Persistence -- The issue of how to store objects to permanent storage. Objects need to be persistent if they are to be available to you and/or to others the next time your application is run.

Persistence classes -- Persistence classes provide the ability to permanently store objects. By encapsulating the storage and retrieval of objects via persistence classes you are able to use various storage technologies interchangeably without affecting your applications.

Persistence layer -- The collection of classes that provide business objects the ability to be persistent. The persistence layer effectively wraps your persistence mechanism.

Persistence mechanism -- The permanent-storage facility used to store objects, such as a relational database, a flat file, or an object database.

Persistent memory -- Main memory plus all available storage space on the network.

Persistent object -- An object that is saved to permanent storage making it retrievable for future use.

Pessimistic locking -- An approach to concurrency in which an item is locked for the entire time that it is in memory. This strategy guarantees that an item won't be updated in the persistence mechanism while the item is in memory, but at the same time disallows other to work with it while someone else does.

Pilot testing -- A testing process that is equivalent to beta testing that is used by organizations to test applications that they have developed for their own internal use.

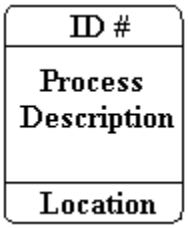
Polymorphic – Two classes are polymorphic when they exhibit the same public interface.

Polymorphism -- Polymorphism says that an object can take any of several forms, and that other objects can interact with the object without having to know what specific form it takes.

Portability -- A measure of how easy it is to move an application to another environment. Application environments may vary by the configuration of both their software and hardware. The easier it is to move an application to another environment the more portable we say that application is.

pre/ -- A keyword used on a state diagram to document state preconditions.

Process – In a process model a process takes some data as input, does something to it, and then outputs it.



Gane & Sarson process notation.

Process model – A diagram that shows the movement of data within a system. Similar in concept to a DFD but not as rigid and documentation heavy.

Prototype – A mock-up of the user interface of your application.

Prototype walkthrough – A process in which your users work through a collection of use-cases using the prototype of the application as if it was the real thing. The main goal is to test whether or not the design of the prototype meets their needs.

Pure inheritance -- A subclass inherits everything from its superclass.

Pure OO language – An OO language that supports only OO programming constructs.

Reading into memory – When you obtain an object from the persistence mechanism but don't intend to update it.

Read lock -- A type of lock indicating that a table, record, class, object, and so on is currently being read by someone else. Other people may also obtain read locks on the item, but no one may obtain a write lock until all read locks are cleared.

Recursive transition – A transition is considered recursive when it leads into the same state that it originated.

Refactoring – The act of reorganizing OO development efforts. Refactoring will often comprise the renaming of methods, attributes, or classes; the redefinition of methods, attributes, or classes; or the complete rework or methods, attributes, or classes. Your analysis, design, or coding efforts can often be refactored.

Regression testing -- The act of ensuring that previously tested functionality still works as expected after changes have been made to an application.

Repository -- A centralized database in which you can check-in and check-out versions of your development work, including documentation, models, and source code.

Requirement-verification matrix -- A document that is used to relate use cases to the portions of your application that implement those requirements. For OO applications, the names of classes are listed across the top of the matrix, the use cases are listed along the left-hand axis of the matrix, and in the squares are listed the main method(s) involved in fulfilling each use case.

	Class #1	Class #2

Requirement #1		
Requirement #2		

Requirement-verification matrix format.

Responsibility -- A responsibility is anything that a class/object knows or does.

Retrieving into memory – When you obtain an object from the persistence mechanism and will potentially update it.

Reverse engineering -- The generation of a model from source code by a CASE tool.

Root - The topmost class in a class hierarchy. Also called a root class.

Router class -- A class that accepts incoming messages sent to a server and passes them onto the appropriate classes, providing an interface through which clients interact with the server.

2.6 S-T

Scaffolding -- A technique in which one or more legacy applications are accessed through an OO wrapper to provide users with a modern user interface usually on a new hardware platform.

Screen scraping – A wrapping technique in which the wrapper simulates the keystrokes that a user would normally make to drive the functionality of a legacy application.

Second normal form (2NF) -- A data entity is in 2NF when it is in first normal form (1NF) and when all of its non key attributes are fully dependent on its primary key. A database is in 2NF when all of its entities are in 2NF.

Second object normal form (2ONF) -- A class is in 2ONF when it is in first object normal form (1ONF) and when "shared" behavior that is needed by more than one instance of the class is encapsulated within its own class(es). A class diagram is in 2ONF when all of its classes are in 2ONF.

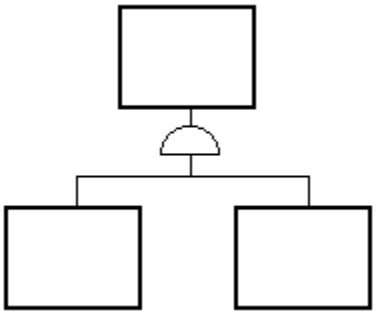
Sequence diagram – A diagram that shows the types of objects involved in a use-case scenario, including the messages they send to one another and the values that they return.

Server -- A server is one or more multiuser processors with shared memory that provides computing connectivity, database services, and interfaces relevant to the business need. A server is also commonly referred to as a "back-end."

Server class -- A class that receives messages but does not send them.

Setter method – An accessor method that modifies the value of an attribute.

Single inheritance -- When a class directly inherits from only one class, we say that we have single inheritance.



Single-inheritance notation (Ambler).

Software quality assurance (SQA) -- The process and techniques by which the development of software is verified, tested, and ensured to be of sufficient levels of excellence for your organization.

SQA – See Software quality assurance.

SQL – See Structured query language.

SQL3 -- The latest version of SQL that includes extensions that support object-oriented concepts.

SQL statement -- A piece of SQL code.

State -- A state represents a stage in the behavior pattern of an object. A state can also be said to represent a condition of an object to which a defined set of policies, regulations, and physical laws apply. On state diagrams a state is shown as a horizontal rectangle.

State notation.

State diagram – A model that describes the states that an object may be in, as well as the transitions between states.

State precondition – A condition that must be met before a state can be entered.

State-transition model – See state diagram.

Static binding – The class/type of an object is associated to it at compile time.

Static SQL -- An SQL statement that is defined and bound to the database at compile time. This is much less flexible but faster than dynamic SQL.

Stress testing -- The act of ensuring that the system performs as expected under high volumes of transactions, high numbers of users, and so on.

Stress-test plan -- The test plan that describes how you intend to go about stress testing your application.

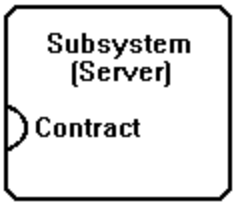
String of message sends -- A series of messages is sent to the same object.

Structured query language -- A standard language used to access and modify data stored in relational databases.

Subclass -- If class "B" inherits from class "A," then we say that "B" is a subclass of "A."

Substate – A specific state that is part of a more generalized superstate.

Subsystem -- A set of classes that collaborate among themselves to support a set of cohesive set of contracts.



Subsystem notation (Ambler).

Subsystem contracts -- The collection of the class contracts of all of the classes that respond to messages sent from classes external to the subsystem.

Superclass -- If class "B" inherits from class "A," then we say that "A" is a superclass of "B."

Superstate – A general state that is decomposed into several substates.

System classes -- System classes provide operating-system-specific functionality for your applications, or they wrap functionality provided by other tool/application vendors. System classes isolate you from the operating system (OS), making your application portable between environments by wrapping OS specific features.

System testing -- A testing process in which you find and fix any known problems to prepare your application for user-acceptance testing.

Technical-design review -- A testing technique in which the design of your application is examined critically by a group of your peers. This process is often referred to as a walkthrough.

Technical-review plan -- A document that describes the goal of a technical review, who is to attend and why, the information that the reviewers require before the review, and the records and documentation that will be produced by the review.

Test log -- A chronological tracking of your testing activities.

Test plan -- A document that prescribes the approach to be taken during the testing process.

Test-procedure scripts -- The description of the steps that must be carried out to perform all or part of a test plan.

Testing in the small – Any testing technique that concentrates on testing components of, or portions of, a system.

Testing in the large – Any testing technique that concentrates on testing the entire system as a whole, or at least very large components of it.

Testing tool -- Any tool that aids in either the definition of test cases, the storage of test cases, or the running of test cases.

Thin-client approach -- A 2-tier C/S architecture in which client machines implement only the user interface of an application.

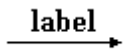
Third normal form (3NF) -- A data entity is in 3NF when it is in second normal form (2NF) and when all of its attributes are directly dependent on the primary key. A database is in 3NF when all of its data entities are in 3NF.

Third object normal form (3ONF) -- A class is in 3ONF when it is in second object normal form (2ONF) and when it encapsulates only one set of cohesive behaviors. A class diagram is in 3ONF when all of its classes are in 3ONF.

Transaction -- A transaction is a single unit of work performed in a persistence mechanism. A transaction may be one or more updates to a persistence mechanism, one or more reads, one or more deletes, or any combination thereof.

Transient object -- See transitory object.

Transition -- A transition is a progression from one state to another. A transition will be triggered by an event (either internal or external to the object). A transition is shown on a state diagram as an arrow leading from one state to another.



Transition notation.

Transitory object -- An object that is not persistent.

2.7 U-Z

UAT – See User acceptance test.

Unified Modeling Language (UML) –The industry standard OO modeling notation proposed by Grady Booch, James Rumbaugh, and Ivar Jacobson. At the time of this writing the UML is being considered by the OMG to make it the OMG standard.

Unit testing -- The act of testing small components of a system to ensure that they work. In the object world this is both method and class testing.

Use case -- A description of a real-world scenario that a system may or may not be able to handle.

Use-case diagram – A diagram that shows the use cases and actors for the application that we are developing.

Use-case scenario testing -- The process of having a group of BDEs act out use-case scenarios to ensure that their CRC model handles the use-cases correctly.

User-acceptance testing (UAT) -- The act of having users verify that an application meets their needs as they see them.

User-requirement review -- A testing process in which a facilitated group of users verify and prioritize the user requirements gathered by a development team.

Volume testing -- A subset of stress testing that deals specifically with determining how many transactions or database accesses that an application can handle during a defined period of time.

White-box tests -- Test cases that verify that specific lines of code work as defined. This is also referred to as clear box testing.

Whitespace -- Blank lines in your code that are used to distinguish between sections.

World Wide Web (WWW) -- A component of the Internet that provides users with the ability to move from computer system to computer system by following predefined links between those systems.

Wrapping -- Wrapping is the act of encapsulating non-OO functionality within a class making it look and feel like any other object within the system.

Wrapper -- A collection of one or more classes that encapsulates access to non-OO technology to make it appear as if it is OO.

Wrapper class – Any class that is part of a wrapper.

Write lock -- A type of lock indicating that a table, record, class, object, and so on is currently being written to by someone else. No one may obtain either a read or a write lock until this lock is cleared.

3. About the Author

Scott W. Ambler is a object development consultant living in the village of Sharon, Ontario, which is 60 km north of Toronto, Canada. He has worked with OO technology since 1990 in various roles: Business Architect, System Analyst, System Designer, Smalltalk programmer, Java programmer, and C++ programmer. He has also been active in education and training as both a formal trainer and as an object mentor.

Scott has a Master of Information Science and a Bachelor of Computer Science from the University of Toronto and is the author of the best-selling books *The Object Primer* and *Building Object Applications That Work*, both published by SIGS Books (http://www.sigs.com). Scott has published several white papers about OO development, including the *AmbySoft Inc. Java Coding Standards* which can be downloaded free of charge from his personal web site (http://www.ambysoft.com). Scott is a contributing editor with *Software Development* (http://www.sdmagazine.com), writes a column for *Computing Canada* (http://www.plesman.com), and has had feature articles appear in *Object Magazine* and *Client/Server Computing*.

He can be reached via e-mail at:

scott@ambySoft.com

and you can visit his personal web site:

http://www.ambysoft.com

About The Object Primer

The Object Primer is a straightforward, easy to understand introduction to object-oriented analysis and design techniques. Object-orientation is the most important change to system development since the advent of structured methods. While OO is often used to develop complex systems, OO itself does not need to be complicated. This book is different than any other book ever written about object-orientation (OO) – It's written from the point of view of a real-world developer, somebody who has lived through the difficulty of learning this exciting new approach. Readers of *The Object Primer* have found it to be one of the easiest introductory books in OO development on the market today, many of whom have shared their comments and kudos with me. Topics include **CRC modeling**, **use cases**, **use-case scenario testing**, and **class diagramming**.

About Building Object Applications That Work

Building Object Applications That Work is about: **architecting** your applications so that they're maintainable and extensible; analysis and design techniques using the **Unified Modeling Language (UML)**; creating applications for stand-alone, **client/server**, and **distributed** environments; using both **relational** and object-oriented (OO) databases for persistence; OO **metrics**; applying OO **patterns** to improve the quality of your applications; OO **testing** (it's harder, not easier); **user interface design** so your users can actually work with the systems that you build; and **coding** applications in a way that makes them **maintainable** and **extensible**.

Uses the

Unified Modeling Language

About the AmbySoft Inc. Java Coding Standards

The *AmbySoft Inc. Java Coding Standards* summarizes in one place the common coding standards for Java, as well as presents several guidelines for improving the quality of your code. It is in Adobe PDF format and can be **downloaded** from http://www.ambysoft.com.

4. Index

I

J

K

L

M