UML Glossary and Terms

- **Abstract Class** - A class that will never be instantiated. An instance of this class will never exist.
- **Actor** - An object or person that initiates events the system is involved with.
- **Activity**: A step or action within an Activity Diagram. Represents an action taken by the system or by an Actor.
- **Activity Diagram**: A glorified flowchart that shows the steps and decisions and parallel operations within a process, such as an algorithm or a business process.
- **Aggregation** - Is a part of another class. Shown with a hollow diamond next to the containing class in diagrams.
- **Artifacts** - Documents describing the output of a step in the design process. The description is graphic, textual, or some combination.
- **Association** - A connection between two elements of a Model. This might represent a member variable in code, or the association between a personnel record and the person it represents, or a relation between two categories of workers, or any similar relationship. By default, both elements in an Association are equal, and are aware of each other through the Association. An Association can also be a Navigable Association, meaning that the source end of the association is aware of the target end, but not vice versa.
- **Association Class**: A Class that represents and adds information to the Association between two other Classes.
- **Attributes** - Characteristics of an object which may be used to reference other objects or save object state information.
- **Base Class**: A Class which defines Attributes and Operations that are inherited by a Subclass via a Generalization relationship.
- **Branch**: A decision point in an Activity Diagram. Multiple Transitions emerge from the Branch, each with a Guard Condition. When control reaches the Branch, exactly one Guard Condition must be true; and control follows the corresponding Transition.
- **Class**: A category of similar Objects, all described by the same Attributes and Operations and all assignment-compatible.
- **Class Diagram** - Shows the system classes and relationships between them.
- **Classifier**: A UML element that has Attributes and Operations. Specifically, Actors, Classes, and Interfaces.
- **Collaboration**: A relation between two Objects in a Communication Diagram, indicating that Messages can pass back and forth between the Objects.
- **Communication Diagram** - A diagram that shows how operations are done while emphasizing the roles of objects.

- **Component**: A deployable unit of code within the system.
- **Component Diagram**: A diagram that shows relations between various Components and Interfaces.
- **Concept** - A noun or abstract idea to be included in a domain model.
- **Construction Phase** - The third phase of the Rational Unified Process during which several iterations of functionality are built into the system under construction. This is where the main work is done.
- **Dependence**: A relationship that indicates one Classifier knows the Attributes and Operations of another Classifier, but isn't directly connected to any instance of the second Classifier.
- **Deployment Diagram**: A diagram that shows relations between various Processors.
- **Domain** -The part of the universe that the system is involved with.
- **Elaboration Phase** - The second phase of the Rational Unified Process that allows for additional project planning including the iterations of the construction phase.
- **Element**: Any item that appears in a Model.
- **Encapsulation** - Data in objects is private.
- **Generalization** - Indicates that one class is a subclass on another class (superclass). A hollow arrow points to the superclass.
- **Event**: In a State Diagram, this represents a signal or event or input that causes the system to take an action or switch States.
- **Final State**: In a State Diagram or an Activity Diagram, this indicates a point at which the diagram completes.
- **Fork**: A point in an Activity Diagram where multiple parallel control threads begin.
- **Generalization**: An inheritance relationship, in which a Subclass inherits and adds to the Attributes and Operations of a Base Class.
- **GoF** - Gang of Four set of design patterns.
- **High Cohesion** - A GRASP evaluative pattern which makes sure the class is not too complex, doing unrelated functions.
- **Low Coupling** - A GRASP evaluative pattern which measures how much one class relies on another class or is connected to another class.
- **Inception Phase** - The first phase of the Rational Unified Process that deals with the original conceptualization and beginning of the project.
- **Inheritance** - Subclasses inherit the attributes or characterics of their parent (superclass) class. These attributes can be overridden in the subclass.

- **Initial State**: In a State Diagram or an Activity Diagram, this indicates the point at which the diagram begins.
- **Instance** - A class is used like a template to create an object. This object is called an instance of the class. Any number of instances of the class may be created.
- **Interface**: A Classifier that defines Attributes and Operations that form a contract for behavior. A provider Class or Component may elect to Realize an Interface (i.e., implement its Attributes and Operations). A client Class or Component may then Depend upon the Interface and thus use the provider without any details of the true Class of the provider.
- **Iteration** - A mini project section during which some small piece of functionality is added to the project. Includes the development loop of analysis, design and coding.
- **Join**: A point in an Activity Diagram where multiple parallel control threads synchronize and rejoin.
- **Member**: An Attribute or an Operation within a Classifier.
- **Merge**: A point in an Activity Diagram where different control paths come together.
- **Message** - A request from one object to another asking the object receiving the message to do something. This is basically a call to a method in the receiving object.
- **Method** - A function or procedure in an object.
- **Model** - The central UML artifact. Consists of various elements arranged in a hierarchy by Packages, with relations between elements as well.
- **Multiplicity** - Shown in a domain model and indicated outside concept boxes, it indicates object quantity relationship to quantiles of other objects.
- **Navigability**: Indicates which end of a relationship is aware of the other end. Relationships can have bidirectional Navigability (each end is aware of the other) or single directional Navigability (one end is aware of the other, but not vice versa).
- **Notation** - Graphical document with rules for creating analysis and design methods.
- **Note**: A text note added to a diagram to explain the diagram in more detail.
- **Object** - Object: In an Activity Diagram, an object that receives information from Activities or provides information to Activities. In a Collaboration Diagram or a Sequence Diagram, an object that participates in the scenario depicted in the diagram. In general: one instance or example of a given Classifier (Actor, Class, or Interface).
- **Package** - A group of UML elements that logically should be grouped together.
- **Package Diagram**: A Class Diagram in which all of the elements are Packages and Dependencies.
- **Pattern** - Solutions used to determine responsibility assignment for objects to interact. It is a name for a successful solution to a well-known common problem.

- **Parameter**: An argument to an Operation.
- **Polymorphism** - Same message, different method. Also used as a pattern.
- **Private**: A Visibility level applied to an Attribute or an Operation, indicating that only code for the Classifier that contains the member can access the member.
- **Processor**: In a Deployment Diagram, this represents a computer or other programmable device where code may be deployed.
- **Protected**: A Visibility level applied to an Attribute or an Operation, indicating that only code for the Classifier that contains the member or for its Subclasses can access the member.
- **Public**: A Visibility level applied to an Attribute or an Operation, indicating that any code can access the member.
- **Reading Direction Arrow** - Indicates the direction of a relationship in a domain model.
- **Realization**: Indicates that a Component or a Class provides a given Interface.
- **Role** - Used in a domain model, it is an optional description about the role of an actor.
- **Sequence Diagram**: A diagram that shows the existence of Objects over time, and the Messages that pass between those Objects over time to carry out some behavior. State chart diagram - A diagram that shows all possible object states.
- **State**: In a State Diagram, this represents one state of a system or subsystem: what it is doing at a point in time, as well as the values of its data.
- **State Diagram**: A diagram that shows States of a system or subsystem, Transitions between States, and the Events that cause the Transitions.
- **Static**: A modifier to an Attribute to indicate that there's only one copy of the Attribute shared among all instances of the Classifier. A modifier to an Operation to indicate that the Operation stands on its own and doesn't operate on one specific instance of the Classifier.
- **Stereotype**: A modifier applied to a Model element indicating something about it which can't normally be expressed in UML. In essence, Stereotypes allow you to define your own "dialect" of UML.
- **Subclass**: A Class which inherits Attributes and Operations that are defined by a Subclass via a Generalization relationship.
- **Swimlane**: An element of an Activity Diagram that indicates what parts of a system or a domain perform particular Activities. All Activities within a Swimlane are the responsibility of the Object, Component, or Actor represented by the Swimlane.
- **Time Boxing** - Each iteration will have a time limit with specific goals.
- **Transition**: In an Activity Diagram, represents a flow of control from one Activity or Branch or Merge or Fork or Join to another. In a State Diagram, represents a change from one State to another.

- **Transition Phase** - The last phase of the Rational Unified Process during which users are trained on using the new system and the system is made available to users.
- **UML** - Unified Modeling Language utilizes text and graphic documents to enhance the analysis and design of software projects by allowing more cohesive relationships between objects.
- **Use Case**: In a Use Case Diagram, represents an action that the system takes in response to some request from an Actor.
- **Use Case Diagram**: A diagram that shows relations between Actors and Use Cases.
- **Visibility**: A modifier to an Attribute or Operation that indicates what code has access to the member. Visibility levels include Public, Protected, and Private.
- **Workflow** - A set of activities that produces some specific result.