The Unified Modeling Language

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.

1. Grady Booch

2. James Rumbaugh (OMT)

3. Ivar Jacobson (OOSE)
Building Blocks

1. Things

2. Relationships

3. Diagrams
Things

**Structural things:** classes, interfaces, collaborations, use cases, active classes, Components, nodes.

**Behavioral things:** interactions, state machines.

**Grouping things:** packages.

**Annotational things:** notes.
Relationships

1. Dependency

2. Association

3. Generalization

4. Realization
Diagrams

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Statechart diagram
7. Activity diagram
8. Component diagram
9. Deployment diagram
Structural Thing

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical.

Class: a class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

Attribute: An attribute is a named property of a class that describes a range of values that instances of the property may hold.

Operation: An operation is the implementation of a service that can be requested from any object of the class to affect behavior.
**Use case:** A use case specifies the behavior of a system or a part of a system and is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor.

**actor:** An actor represents a coherent set of roles that users of use cases play when interacting with these use cases.

**include:** An include relationship between use cases means that the base use case explicitly incorporates the behavior of another use case at a location specified in the base.

**extend:** An extend relationship between use cases means that the base use
case implicitly incorporates the behavior of another use case at a location specified indirectly by the extending use case.
**Structural Thing**

**Interface:** An interface is a collection of operations that specify a service of a class or component.

**Collaboration:** A collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that’s bigger than the sum of all the elements.

**Active class:** An active class is a class whose objects own one or more processes or threads and therefore can initiate control activity.

**Component:** A component is a physical and replaceable part of a system that
conforms to and provides the realization of a set of interfaces.

**Node:** A node is a physical element that exists at run time and represents a computational resource.
Behavioral Things

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space.

**Interaction:** An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose.

**State machine:** A state machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its response to those events.
Grouping and Annotational Things

Grouping things are the organizational parts of UML models and Annotational things are the explanatory parts of UML models.

**Package:** A package is a general purpose mechanism for organizing elements into groups.

**Note:** A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.
Relationships

**Dependency:** A dependency is a using relationship that states that a change in specification of one thing may affect another thing that uses it, but not necessarily the reverse.

**Association:** An association is a structural relationship that specifies that objects of one thing are connected to objects of another.

**Aggregation:** An aggregation is a special form of association that specifies a whole-part relationship between the aggregate (the whole) and a component (the part).
**Generalization:** A generalization is a relationship between a general thing and a more specific kind of that thing. Sometimes it is called an ”is-a-kind-of” relationship.

**Realization:** A realization is a semantic relationship between classifiers, wherein, one classifier specifies a contract that another classifier guarantees to carry out.
Diagrams

**Class diagram:** A class diagram shows a set of classes, interfaces, and collaborations and their relationships.

**Object diagram:** An object diagram shows a set of objects and their relationships.

**Use case diagram:** A use case diagram shows a set of use cases and actors and their relationships.

**Sequence diagram:** A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
Collaboration diagram: A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages.

Statechart diagram: A statechart diagram shows a state machine, consisting of states, transitions, events, and activities.

Activity diagram: An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within a system.
**Component diagram:** A component diagram shows the organization and dependencies among a set of components.

**Deployment diagram:** A deployment diagram shows the configuration of runtime processing nodes and the components that live on them.