The Unified Modeling Language (UML)

was created to forge a common, semantically and syntactically rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally. UML has applications beyond software development, such as process flow in manufacturing.

It is analogous to the blueprints used in other fields, and consists of different types of diagrams. In the aggregate, UML diagrams describe the boundary, structure, and the behavior of the system and the objects within it.

UML is not a programming language but there are tools that can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design

UML and its role in object-oriented modeling and design

There are many problem-solving paradigms or models in Computer Science, which is the study of algorithms and data. There are four problem-solving model categories: imperative, functional, declarative and object-oriented languages (OOP). In object-oriented languages, algorithms are expressed by defining ‘objects’ and having the objects interact with each other. Those objects are things to be manipulated and they exist in the real world. They can be buildings, widgets on a desktop, or human beings.

Object-oriented languages dominate the programming world because they model real-world objects. UML is a combination of several object-oriented notations: Object-Oriented Design, Object Modeling Technique, and Object-Oriented Software Engineering.

UML uses the strengths of these three approaches to present a more consistent methodology that's easier to use. UML represents best practices for building and documenting different aspects of software and business system modeling.

UML is linked with **object oriented** design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

1. **Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.
2. **Behavior Diagrams –** Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

The image below shows the hierarchy of diagrams

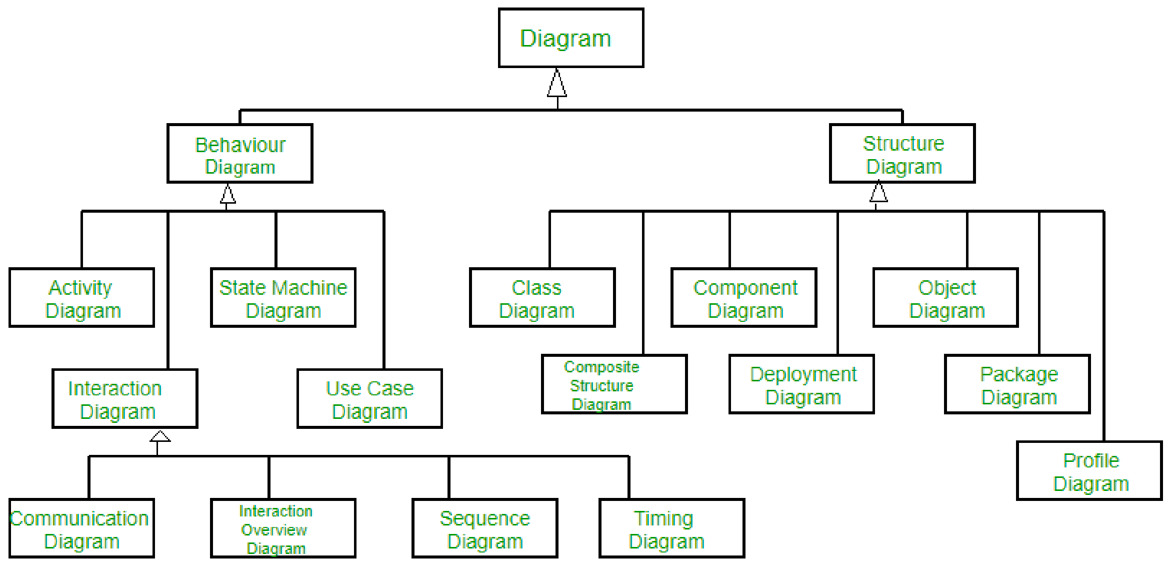
Diagram
Behaviour Diagram
Structure Diagram
Activity Diagram
State Machine Diagram
Interaction Diagram
Use Case Diagram
Communication Diagram
Interaction Overview Diagram
Sequence Diagram
Timing Diagram
Class Diagram
Composite Structure Diagram
Component Diagram
Deployment Diagram
Object Diagram
Package Diagram
Profile Diagram