# UML diagrams

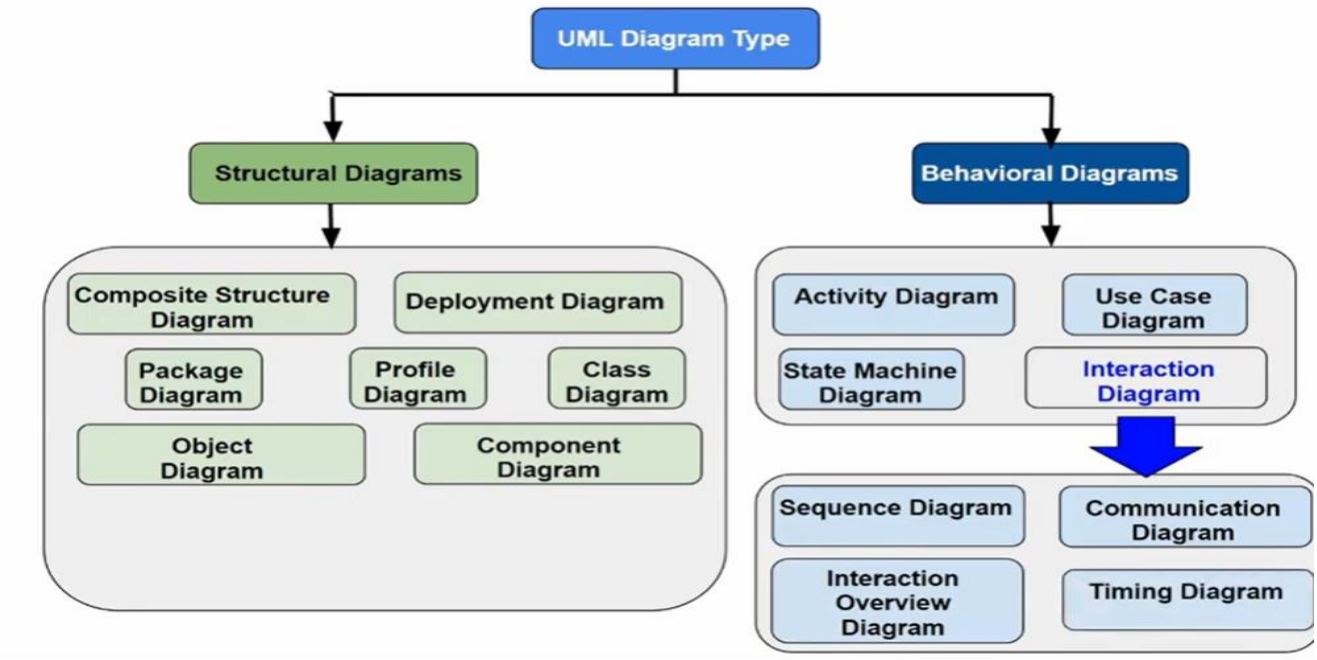
---

## ===➔ This is a brief overview of UML diagrams…

# What are UML diagrams?

UML, which stands for **Unified Modeling Language**, is a way to visually represent the architecture, design, and implementation of complex software systems. When you're writing code, there are thousands of lines in an application, and it's difficult to keep track of the relationships and hierarchies within a software system. UML diagrams divide that software system into components and subcomponents.

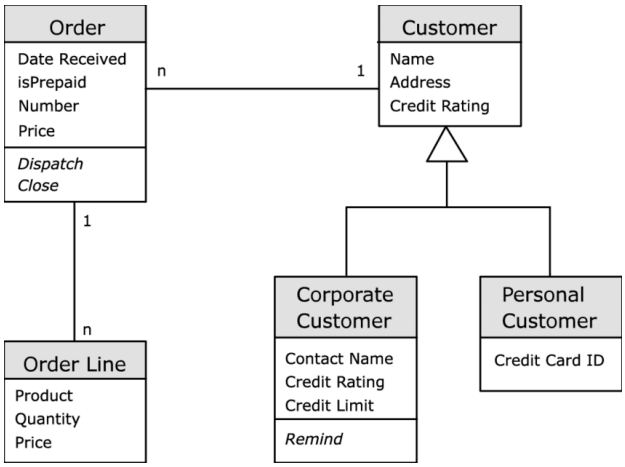
## UML diagrams types…



# Structural diagrams

## Class diagram

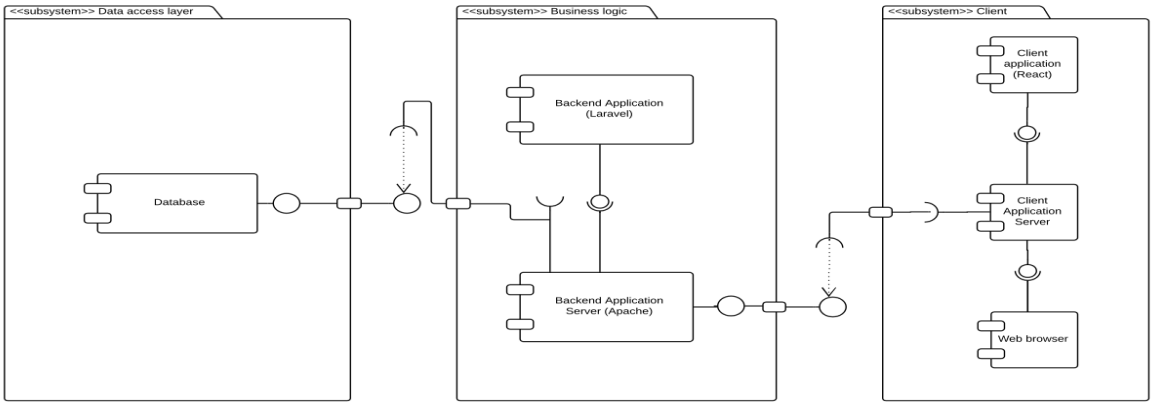
**Is a central modeling technique that is used in all almost object oriented methods this diagram describe the types of objects in the system and different type of static relationships that exist between them.**




## Component diagram

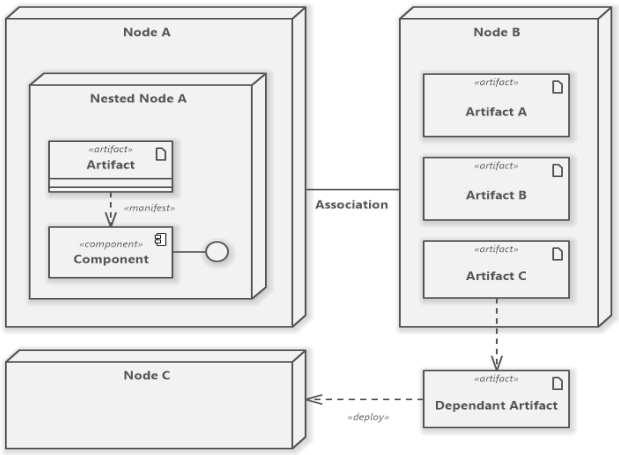
**Illustrate how components are put together to form larger components or software systems and illustrate the architecture of software components**

**And dependencies between them, these software components can include runtime components, executable components and source code components.**



## Deployment diagram

**Helps to model the physical aspects of object oriented software system**

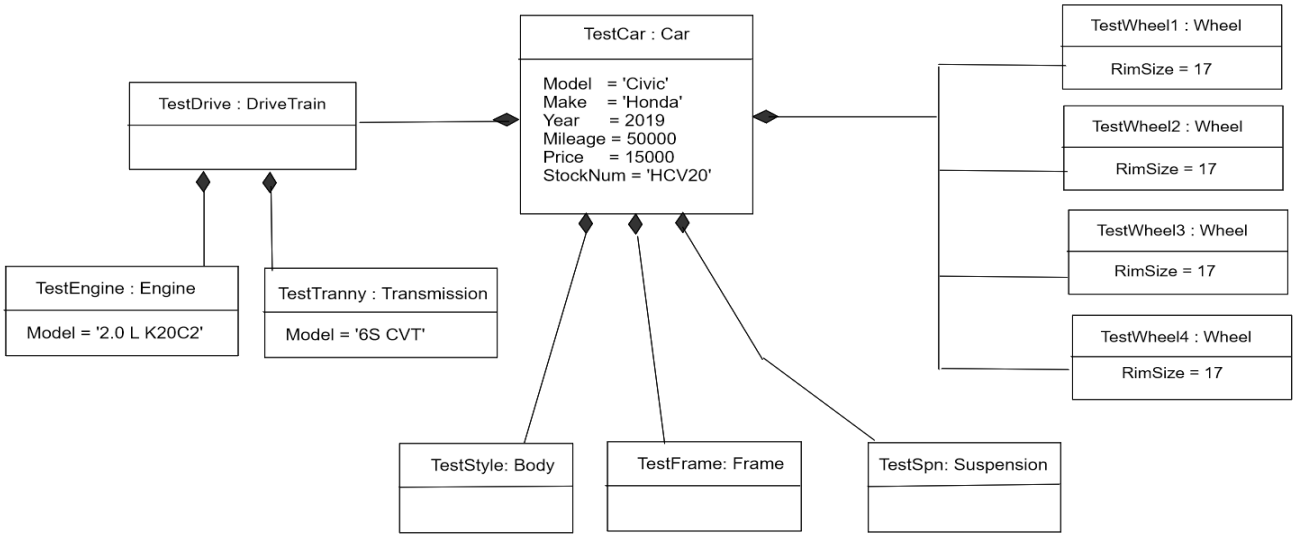



## Object diagram

**Is the instance of class diagram, it shows a detailed snapshot of the system state at a particular point in the time, the difference from a class diagram is an abstract model of classes and their relationships , however the object diagram represent an instance at a specific moment which is concrete in nature**

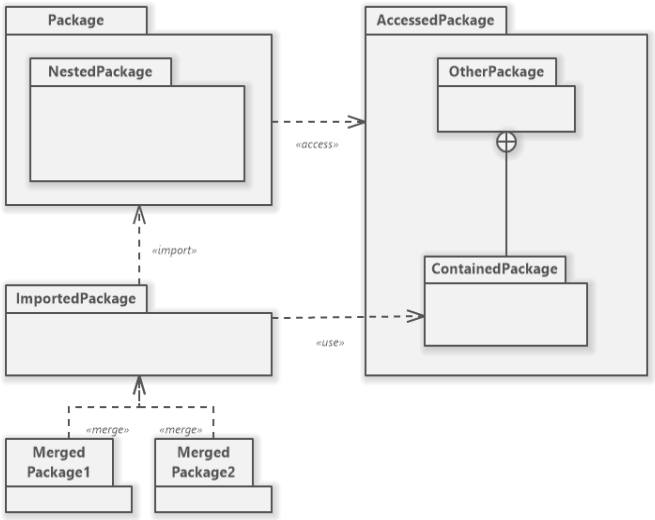
**The use of object diagrams rather limited to shows the example of data structure.**

Object Diagram for specific instance of one car



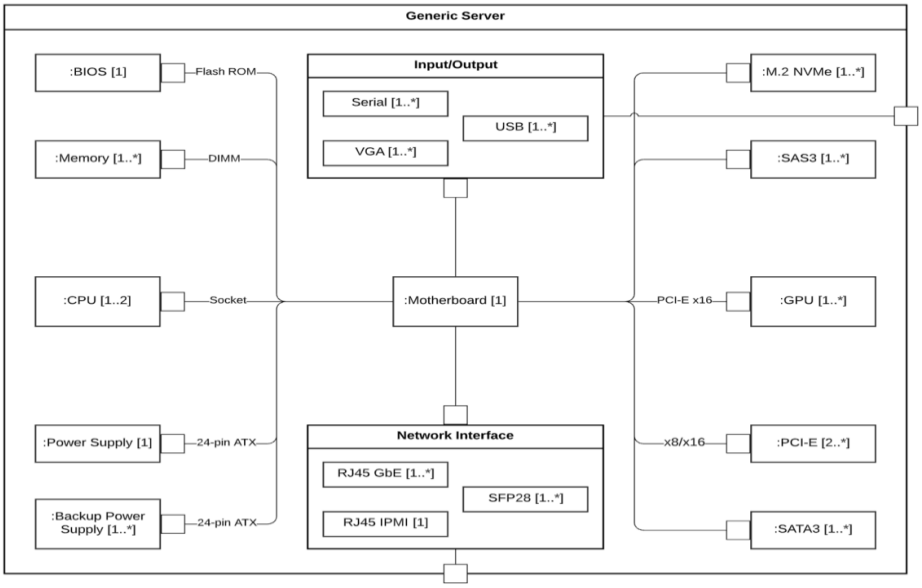
## Packages diagram

**Is a structural diagrams that shows packages and dependencies between them, it allows us to display the different views of the system.**



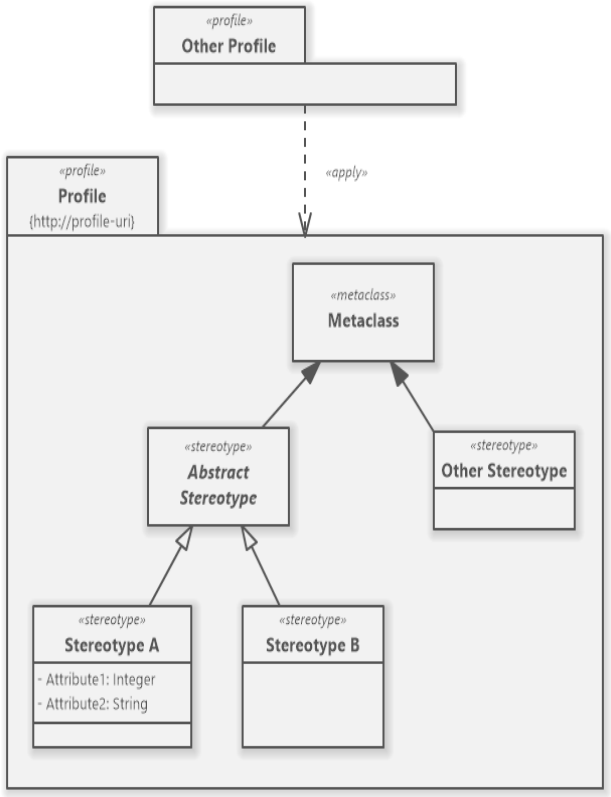

## Composite structural diagram

**Is similar to the class diagram and it is a kind of components diagram used in micro level system modeling but it depicts individual parts instead of whole classes.**



## Profile diagrams

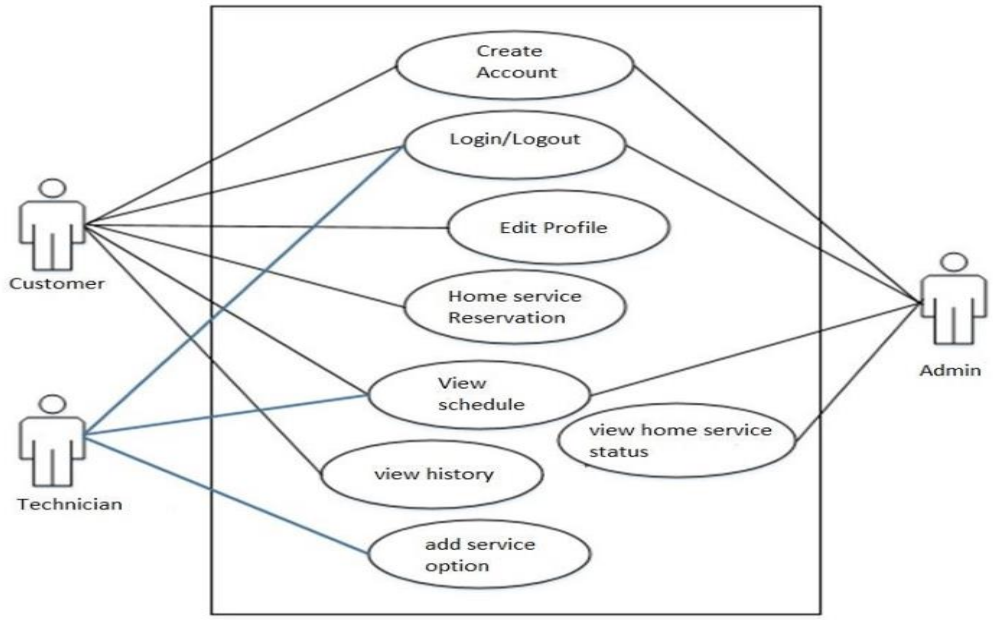
**provides a generic extension mechanism for customizing UML models for particular domains and platforms. Extension mechanisms allow refining standard semantics in strictly additive manner, preventing them from contradicting standard semantics. Profiles are defined using stereotypes, tagged value definitions, and constraints which are applied to specific model elements, like Classes, Attributes, Operations, and Activities. A Profile is a collection of such extensions that collectively customize UML for a particular domain.**



# Behavioral diagram

## Use case diagram

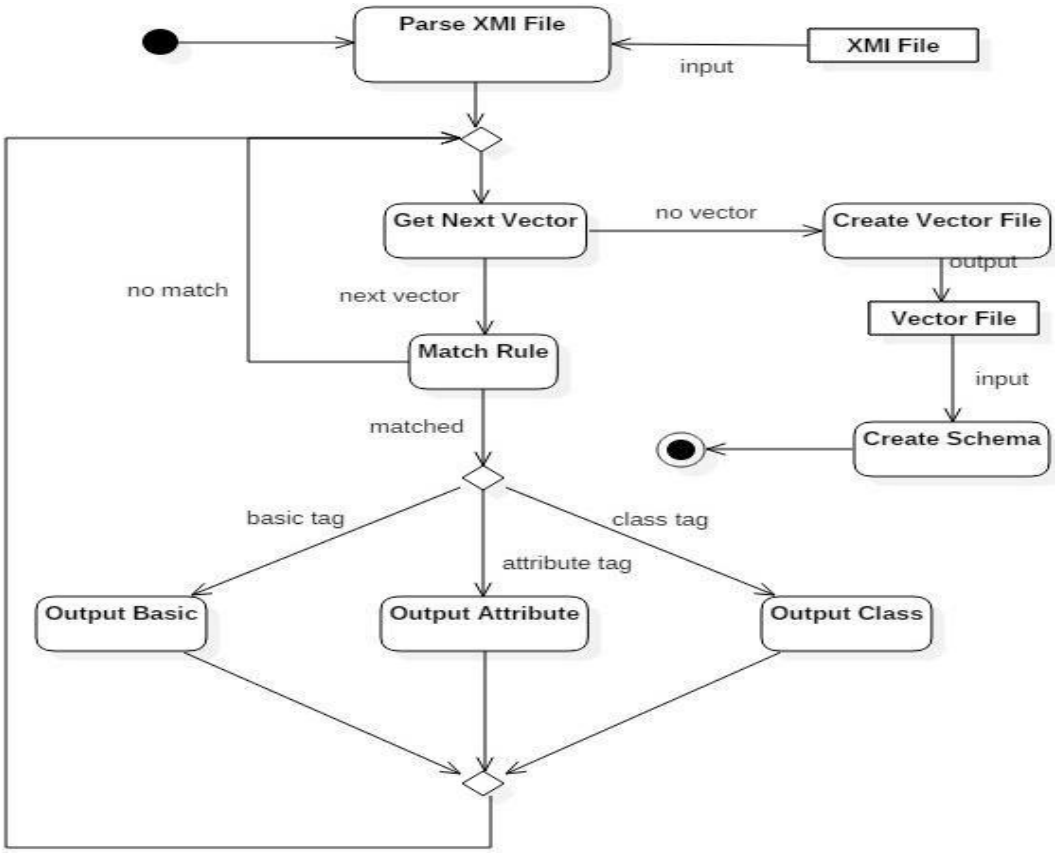
**A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.**



## Activity diagrams

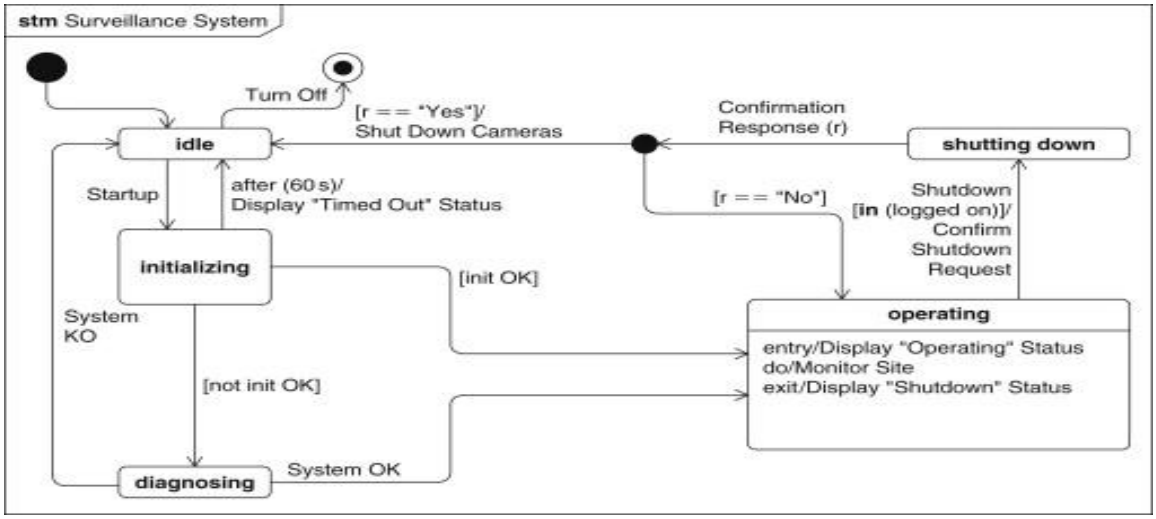
**are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes, as well as the data flows intersecting with the related activities.**

**Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.**



## State machine diagram

**also known as UML statechart, is an extension of the mathematical concept of a finite automaton in computer science applications.**




## The interaction diagram

**is similar to the activity diagram, in that both visualize a sequence of activities. The difference is that, for an interaction overview, each individual activity is pictured as a frame which can contain a nested interaction diagram.**

**This makes the interaction overview diagram useful to deconstruct a complex scenario that would otherwise require multiple if-then-else paths to be illustrated as a single sequence diagram.**