**UML:**

"The Unified Modeling Language (UML) is a language for

Specifying,

Visualizing,

Constructing and

Documenting

The artifacts of software systems.

UML is a graphical notation for modeling various aspects of software systems.

Architectural blueprints including elements such as

Activities,

Actors,

Business processes,

Database schemas,

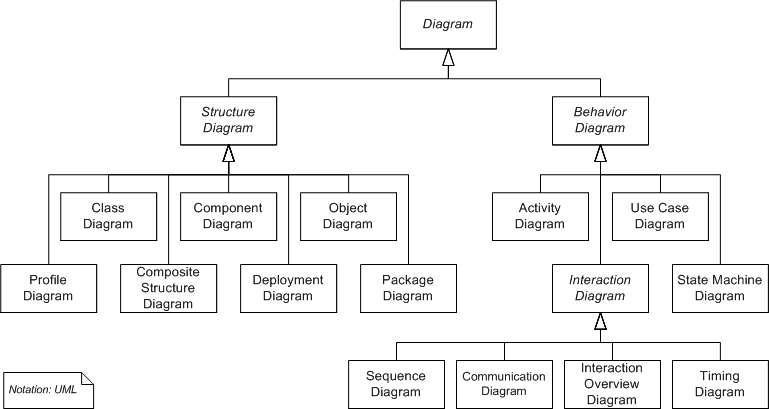
Components,

Programming language statements and

Reusable software components.

UML diagrams represent two different views of a system model:

* Static (or structural) view: emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams.
* Dynamic (or behavioral) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.



#### Structure Diagrams

Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

* **Class diagram**: describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.
* **Component diagram:** describes how a software system is split up into components and shows the dependencies among these components.
* **Composite structure diagram**: describes the internal structure of a class and the collaborations that this structure makes possible.
* **Deployment diagram**: describes the hardware used in system implementations and the execution environments and artifacts deployed on the hardware.
* **Object diagram:** shows a complete or partial view of the structure of a modeled system at a specific time.
* **Package diagram:** describes how a system is split up into logical groupings by showing the dependencies among these groupings.
* **Profile diagram:** operates at the metamodel level to show stereotypes as classes with the <<stereotype>> stereotype, and profiles as packages with the <<profile>> stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what metamodel element a given stereotype is extending.

#### Behaviour Diagrams

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

* **Use case diagram:** describes the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.
* **Activity diagram:** describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
* **State machine diagram:** describes the states and state transitions of the system.

#### Interaction Diagrams

Interaction diagrams, a subset of behaviour diagrams, emphasize the flow of control and data among the things in the system being modeled:

* **Sequence diagram**: shows how objects communicate with each other in terms of a sequence of messages. Also indicates the life spans of objects relative to those messages.
* **Communication diagram:** shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.
* **Interaction overview diagram:** provides an overview in which the nodes represent communication diagrams.
* **Timing diagrams:** a specific type of interaction diagram where the focus is on timing constraints.