UML 2.5 Diagrams Overview

A **UML diagram** is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains **graphical elements** (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The **kind of the diagram** is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is **class diagram**. A diagram which shows **use cases** and **actors** is **use case diagram**. A **sequence diagram** shows sequence of message exchanges between **lifelines**.

UML specification does not preclude **mixing** of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some **UML Tools** do restrict set of available graphical elements which could be used when working on specific type of diagram.

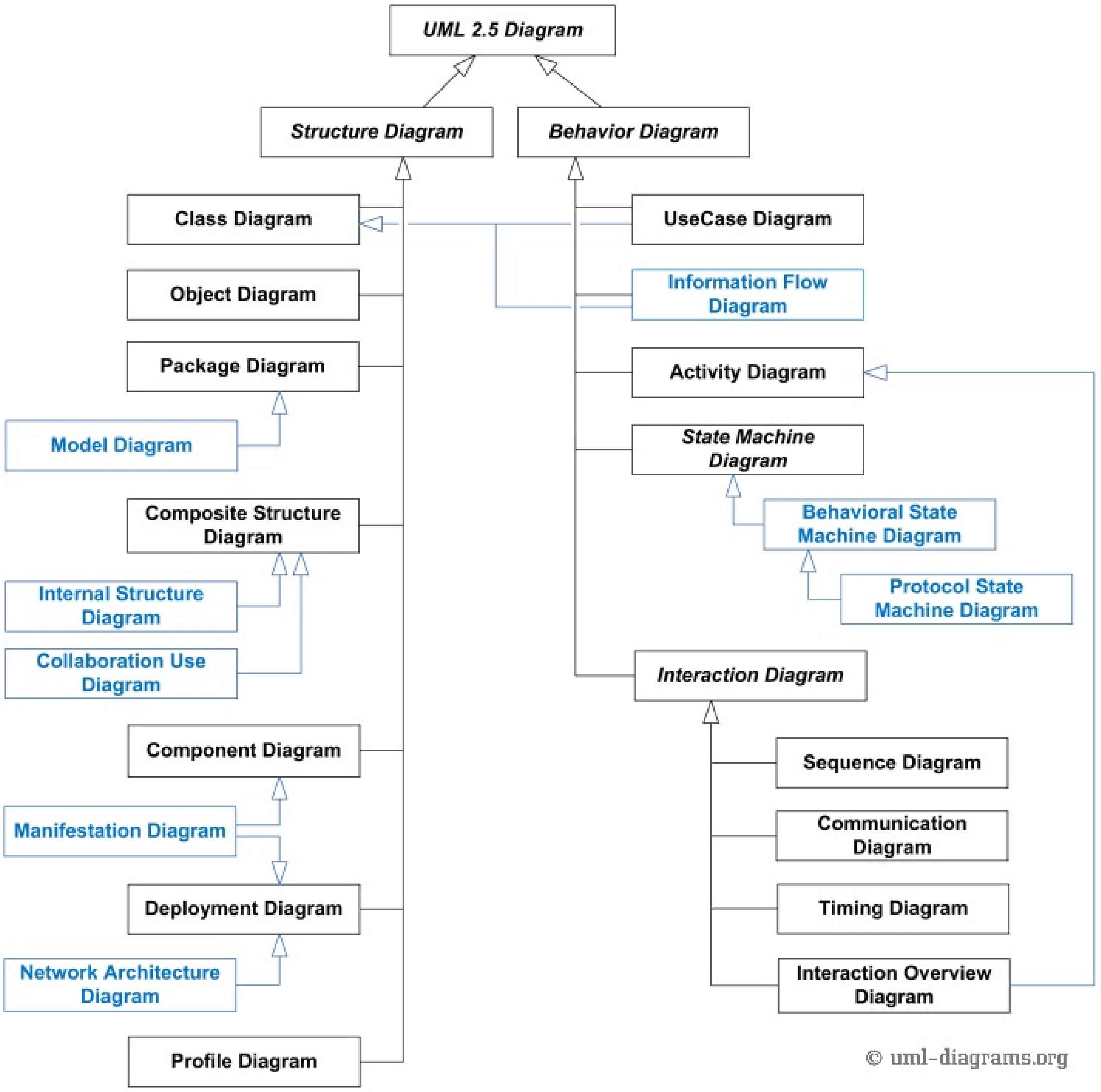
Classification of UML 2.5 Diagrams

UML specification defines two major kinds of UML diagram: **structure diagrams** and **behavior diagrams**.

Structure diagrams show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

UML 2.5 diagrams could be categorized hierarchically as shown below. Note, items shown in blue are **not** part of official UML 2.5 taxonomy of diagrams.



UML 2.5 Diagrams Overview.
Note, items in blue are not part of official taxonomy of UML 2.5 diagrams.

UML 2.5 Structure Diagrams

Structure diagrams show **static structure** of the system and its parts on different abstraction and implementation levels and how those parts are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Structure diagrams are not utilizing **time** related concepts, do not show the details of dynamic behavior. However, they may show relationships to the behaviors of the classifiers exhibited in the structure diagrams.

Diagram	Purpose	Elements
Class diagram	Shows structure of the designed system, subsystem or component as related classes and interfaces, with their features, constraints and relationships - associations, generalizations, dependencies, etc.	**class**, **interface**, **feature**, **constraint**, **association**, **generalization**, **dependency**.
Object diagram	Instance level class diagram which shows instance specifications of classes and interfaces (objects), slots with value specifications, and links (instances of association). **Object diagram** was defined in now obsolete ***UML 1.4.2 Specification*** as *"a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time."* It also stated that object diagram is *"a class diagram with objects and no classes."* **UML 2.5** specification simply provides no definition of **object diagram**.	**instance specification**, **object**, **slot**, **link**.
Package diagram	Shows **packages** and relationships between the packages.	**package**, **packageable**

		element, **dependency**, **element import**, **package import**, **package merge**.
Model diagram	UML auxiliary structure diagram which shows some abstraction or specific view of a system, to describe architectural, logical or behavioral aspects of the system. It could show, for example, architecture of a multi-layered (aka multi-tiered) application - see **multi-layered application model**.	**model**, **package**, **packageable element**, **dependency**.
Composite structure diagram	Diagram could be used to show: • **Internal structure of a classifier** • **A behavior of a collaboration**	
Internal structure diagram	Shows internal structure of a classifier - a decomposition of the classifier into its properties, parts and relationships.	**structured class**, **part**, **port**, **connector**, **usage**.
Collaboration use diagram	Shows objects in a system cooperating with each other to produce some behavior of the system.	**collaboration**, **connector**, **part**, **dependency**.
Component diagram	Shows components and dependencies between them. This type of diagrams is used for **Component-Based Development** (**CBD**), to describe systems with **Service-Oriented Architecture** (**SOA**).	**component**, **interface**, **provided interface**, **required interface**, **class**, **port**, **connector**, **artifact**, **component realization**, **usage**.
Manifestation diagram	While **component diagrams** show components and relationships between components and classifiers, and **deployment diagrams** - **deployments** of artifacts to deployment targets, some missing intermediate diagram is **manifestation diagram** to be used to show **manifestation** (implementation) of **components** by **artifacts** and internal structure of artifacts. Because **manifestation diagrams** are not defined by UML 2.5 specification, manifestation of components by artifacts could be shown using either component diagrams or deployment diagrams.	**manifestation**, **component**, **artifact**.
Deployment diagram	Shows architecture of the system as **deployment** (distribution) of software **artifacts** to **deployment targets**. Note, that **components** were directly deployed to nodes in UML 1.x deployment diagrams. In UML 2.x **artifacts** are deployed to nodes, and artifacts could **manifest** (implement) components. Components are deployed to nodes indirectly through artifacts. **Specification level deployment diagram** (also called type level) shows some overview of **deployment** of **artifacts** to **deployment targets**, without referencing specific instances of artifacts or nodes. **Instance level deployment diagram** shows **deployment** of instances of **artifacts** to specific instances of **deployment targets**. It could be used for example to show differences in deployments to development, staging or production environments with the names/ids of specific build or deployment servers or devices.	**deployment**, **artifact**, **deployment target**, **node**, **device**, **execution environment**, **communication path**, **deployment specification**,
Network architecture diagram	Deployment diagrams could be used to show logical or physical **network architecture** of the system. This kind of deployment diagrams - not formally defined in UML 2.5 - could be called network architecture diagrams.	**node**, **switch**, **router**, **load balancer**, **firewall**, **communication path**, **network segment**, **backbone**.
Profile diagram	Auxiliary UML diagram which allows to define custom stereotypes, tagged values, and constraints as a **lightweight extension mechanism** to the UML standard. Profiles allow to adapt the UML metamodel for different • **platforms** (such as J2EE or .NET), or • **domains** (such as real-time or business process modeling). Profile diagrams were first introduced in UML 2.0.	**profile**, **metaclass**, **stereotype**, **extension**, **reference**, **profile application**.

UML 2.5 Behavior Diagrams

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

Diagram	Purpose	Elements

Use case diagram	Describes a set of actions (**use cases**) that some system or systems (**subject**) should or can perform in collaboration with one or more external users of the system (**actors**) to provide some observable and valuable results to the actors or other stakeholders of the system(s). Note, that **UML 2.4.1 specification** (see "16.4 Diagrams") stated that *Use Case Diagrams are a specialization of Class Diagrams such that the classifiers shown are restricted to being either Actors or Use Cases.* **Class diagrams** are **structure diagrams**.	**use case**, **actor**, **subject**, **extend**, **include**, **association**.
Information flow diagram	Shows exchange of information between system entities at some high levels of abstraction. Information flows may be useful to describe circulation of information through a system by representing aspects of models not yet fully specified or with less details.	**information flow**, **information item**, **actor**, **class**.
Activity diagram	Shows sequence and conditions for coordinating lower-level behaviors, rather than which classifiers own those behaviors. These are commonly called **control flow** and **object flow** models.	**activity**, **partition**, **action**, **object**, **control**, **activity edge**.
State machine diagram	Used for modeling discrete behavior through finite state transitions. In addition to expressing the **behavior** of a part of the system, state machines can also be used to express the **usage protocol** of part of a system. These two kinds of state machines are referred to as **behavioral state machines** and **protocol state machines**.	
Behavioral state machine diagram	Shows discrete **behavior** of a part of designed system through finite state transitions.	**behavioral state**, **behavioral transition**, **pseudostate**.
Protocol state machine diagram	Shows **usage protocol** or a **lifecycle** of some **classifier**, e.g. which operations of the classifier may be called in each state of the classifier, under which specific conditions, and satisfying some optional postconditions after the classifier transitions to a target state.	**protocol state**, **protocol transition**, **pseudostate**.
Interaction diagram	**Interaction diagrams** include several different types of diagrams: • **sequence diagrams**, • **communication diagrams** (known as **collaboration diagrams** in UML 1.x), • **timing diagrams**, • **interaction overview diagrams**.	
Sequence diagram	Most common kind of interaction diagrams which focuses on the message interchange between **lifelines** (objects).	**lifeline**, **execution specification**, **message**, **combined fragment**, **interaction use**, **state invariant**, **destruction occurrence**.
Communication diagram (a.k.a. **Collaboration diagram** in UML 1.x)	Focuses on the interaction between **lifelines** where the architecture of the internal structure and how this corresponds with the **message** passing is central. The sequencing of messages is given through a **sequence numbering** scheme.	**lifeline**, **message**.
Timing diagram	Shows interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among lifelines along a linear time axis.	**lifeline**, **state or condition timeline**, **destruction event**, **duration constraint**, **time constraint**.
Interaction overview diagram	Defines interactions through a variant of **activity diagrams** in a way that promotes overview of the control flow. Interaction overview diagrams focus on the overview of the flow of control where the nodes are interactions or **interaction uses**. The lifelines and the messages do not appear at this overview level.	**initial node**, **flow final node**, **activity final node**, **decision node**, **merge node**, **fork node**, **join node**, **interaction**, **interaction use**, **duration constraint**, **time constraint**.

Noticed a spelling error? Select the text using the mouse and press Ctrl + Enter.

Follow @uml_diagrams   THIS SITE IS ECO-FRIENDLY   by Kirill Fakhroutdinov

This document describes UML versions up to *UML 2.5* and is based on the corresponding **OMG™ Unified Modeling Language™ (OMG**

UML®) specifications. UML diagrams were created in **Microsoft® Visio®** 2007-2016 using ***UML 2.x Visio Stencils***. ***Lucidchart*** is a nice, free UML tool that I recommend for students.
You can send your comments and suggestions to webmaster at **webmaster@uml-diagrams.org**.