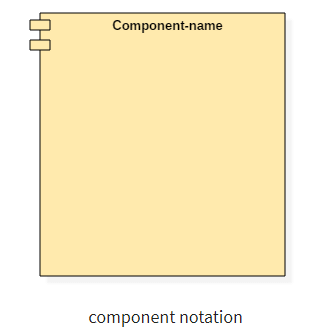
**Component Diagram:**

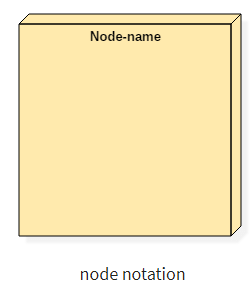
When modeling large object-oriented systems, it is necessary to break down the system into manageable subsystems. UML component diagrams are used for modeling large systems into smaller subsystems which can be easily managed.

A component is a replaceable and executable piece of a system whose implementation details are hidden. A component provides the set of interfaces that a component realizes or implements. Components also require interfaces to carry out a function.

UML Component diagrams are used to represent different components of a system.

**Component diagram Notations:**

****

****

A component is a replaceable and executable piece of a system whose implementation details are hidden. A component provides the set of interfaces that a component realizes or implements. Components also require interfaces to carry out a function.

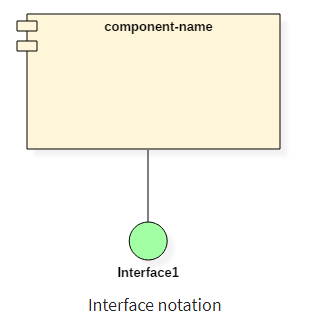
It is a modular part of a system that encapsulates its contents. They are the logical elements of a system that plays an essential role during the execution of a system.

A component is similar to a black box whose external behavior is defined by a provided interface and required interfaces.

**Structure of a component**

A component is represented with classifier rectangle stereotypes as,

A component in UML is represented as follows,



**Interfaces**

The interface is a named set of public features. It separates the specification of functionality from its implementation by a class diagram or a subsystem. An interface symbol cannot be instantiated. It declares a contract that may be realized by zero or more classifiers such as a class or a subsystem.

Anything that realizes an interface accepts the functionalities of the interface and agrees to abide by the contract defined by the interface.

There are two types of interfaces,

1. **Provided interfaces**
2. **Required interfaces**

We can connect provided and required interfaces using assembly connector.

**Advantages:**

* It increases the flexibility and extensibility of a class.
* It decreases the implementation dependencies.

**Disadvantages:**

* Extra flexibility leads to complex classes.
* Too many interfaces make systems hard to understand.

### Subsystems:

It is a component base that acts as a decomposition unit for larger systems. It is a logical construct which is used to break down an extensive system into smaller systems which are known as subsystems. This process makes it easy to manage each subsystem efficiently.

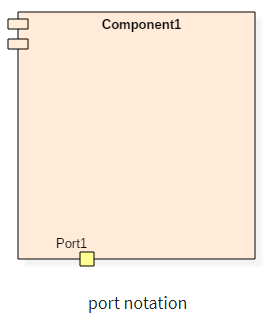
A subsystem cannot be instantiated during runtime, but their contents can be initialized. When subsystems are connected, it creates a single system.

**Port**

A port is an interaction point between a classifier and an external environment. It groups semantically cohesive set of provided and required interfaces. A port can be used in UML without specifying the name of the port. A port may have visibility. When a port is drawn over the boundary of a classifier, then it means that the port is public. It also means that all the interfaces used are made as public.

When a port is drawn inside the classifier, then it is either protected or private.

A port also has multiplicity that indicates the number of instances of the port classifier will have. A port in UML diagram is denoted as given below,



Here the port1 is drawn over the boundary, which means it has visibility as public.

**Why use Component Diagram?**

UML component diagrams have significant importance. Component diagram variously differs from other diagrams. While other diagrams are used to represent the system, working of a system or the architecture of a system. Component diagrams are used to describe the working and behavior of various components of a system.

It represents how each component acts during the execution of a system.

These are the static diagrams of the unified modeling language. A component diagram is used to represent the structure and organization of components during any instance of time.

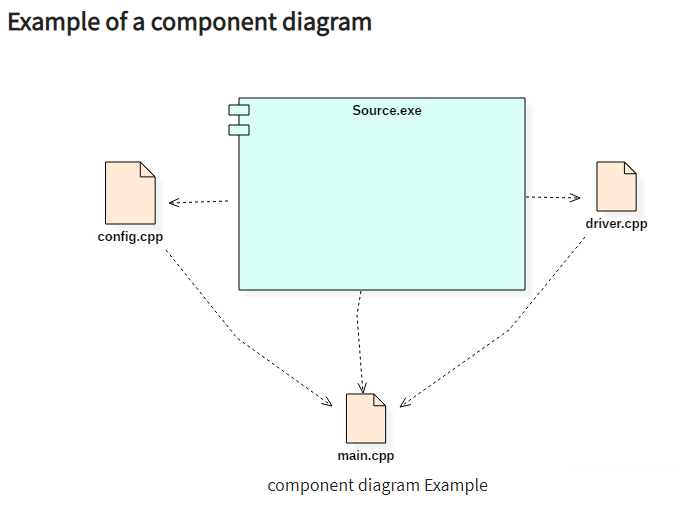
**Component diagrams are used for,**

* To represent the components of any system at runtime.
* It helps during testing of a system.
* It visualizes the connection between various components.

**When to use Component Diagram?**

Component diagrams are different from any other diagrams in UML. Component diagrams are used to display various components of a software system as well as subsystems of a single system. They are used to represent physical things or components of a system. It generally visualizes the structure and an organization of a system.

1. Component diagrams are used to model the component organization of a system.
2. They are used to divide a single system into various subsystems as per the functionality.



**Deployment Diagram:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

The deployment diagram maps the software architecture created in design to the physical system architecture that executes it. In distributed systems, it models the distribution of the software across the physical nodes.

The software systems are manifested using various **artifacts**, and then they are mapped to the execution environment that is going to execute the software such as **nodes**. Many nodes are involved in the deployment diagram; hence, the relation between them is represented using communication paths.

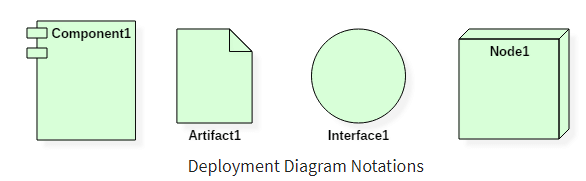
There are two forms of a deployment diagram.

* **Descriptor form**
  + It contains nodes, the relationship between nodes and artifacts.
* **Instance form**
  + It contains node instance, the relationship between node instances and artifact instance.
  + An underlined name represents node instances.

## Purpose of a deployment diagram:

Deployment diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality. It is used to describe software to hardware interaction and vice versa.

**Deployment Diagram Symbol and notations:**

****

**A deployment diagram consists of the following notations:**

1. A node
2. A component
3. An artifact
4. An interface

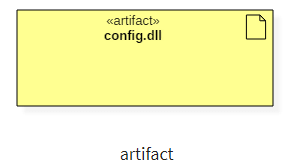
**What is an artifact?**

An artifact represents the specification of a concrete real-world entity related to software development. You can use the artifact to describe a framework which is used during the software development process or an executable file. Artifacts are deployed on the nodes. The most common artifacts are as follows,

1. Source files
2. Executable files
3. Database tables
4. Scripts
5. DLL files
6. User manuals or documentation
7. Output files

Artifacts are deployed on the nodes. It can provide physical manifestation for any UML element.

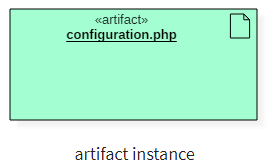
Generally, an artifact is represented as follows in the unified modeling language.



**Artifact Instances**

An artifact instance represents an instance of a particular artifact. An artifact instance is denoted with same symbol as that of the artifact except that the name is underlined.

Generally, an artifact instance is represented as follows in the unified modeling language.

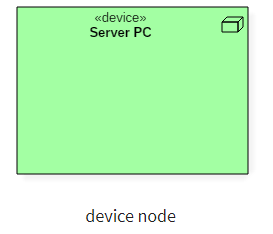


**What is a node?**

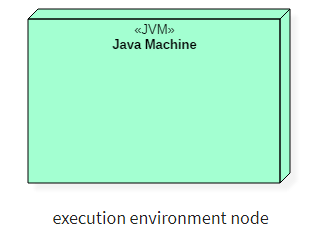
Node is a computational resource upon which artifacts are deployed for execution. A node is a physical thing that can execute one or more artifacts. A node may vary in its size depending upon the size of the project.

Node is an essential UML element that describes the execution of code and the communication between various entities of a system. It is denoted by a 3D box with the node-name written inside of it. Nodes help to convey the hardware which is used to deploy the software.

**Following is a representation of a device in UML:**



Following is a representation of an execution environment in UML:



**How to draw a deployment diagram?**

Deployment diagram visualizes the topological view of an entire system. It represents the deployment of a system.

A deployment diagram consists of nodes which describe the physical devices used inside the system. On these nodes, artifacts are deployed. We can also have node instances on which artifact instances are going to be implemented.

Node and artifacts of a system participate in the final execution of a system.

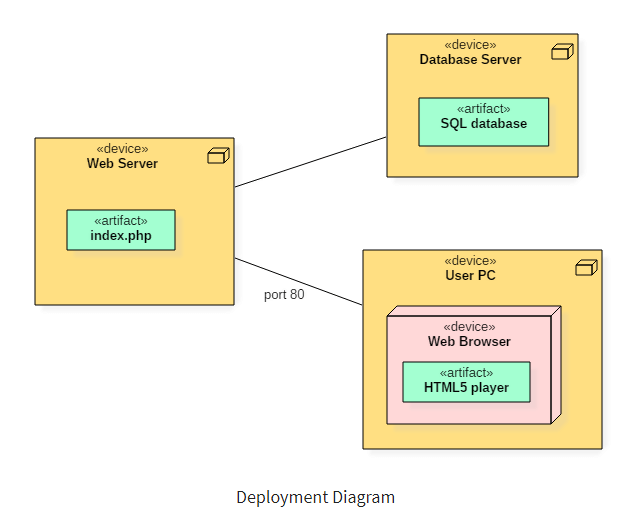
A deployment diagram plays a critical role during the administrative process, and it must satisfy the following parameters,

* High performance
* Maintainability
* Scalability
* Portability
* Easily understandable

Nodes and artifacts are the essential elements of deployment. Before actually drawing the deployment diagram, all nodes and the relationship between every node of the system must be identified.

**Example of a Deployment diagram**

Following deployment diagram represents the working of HTML5 video player in the browser:



**When to use a deployment diagram?**

Deployment diagrams are mostly used by system administrators, network engineers, etc. These diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality.

To make the software work efficiently and at a faster rate, the hardware also must be of good quality. It must be designed efficiently to make software work properly and produce accurate results in quick time.

Deployment diagrams can be used for,

1. Modeling the network topology of a system.
2. Modeling distributed systems and networks.
3. Forward and reverse engineering processes.