UML Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

Notation of a Component Diagram

a) A component

b) A node

Purpose of a Component Diagram

Since it is a special kind of a UML diagram, it holds distinct purposes. It describes all the individual components that are used to make the functionalities, but not the functionalities of the system. It visualizes the physical components inside the system. The components can be a library, packages, files, etc.

The component diagram also describes the static view of a system, which includes the organization of components at a particular instant. The collection of component diagrams represents a whole system.

The main purpose of the component diagram are enlisted below:

1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.

Why use Component Diagram?

The component diagrams have remarkable importance. It is used to depict the functionality and behavior of all the components present in the system, unlike other diagrams that are used to represent the architecture of the system, working of a system, or simply the system itself.

In UML, the component diagram portrays the behavior and organization of components at any instant of time. The system cannot be visualized by any individual component, but it can be by the collection of components.

Following are some reasons for the requirement of the component diagram:

1. It portrays the components of a system at the runtime.
2. It is helpful in testing a system.
3. It envisions the links between several connections.

When to use a Component Diagram?

It represents various physical components of a system at runtime. It is helpful in visualizing the structure and the organization of a system. It describes how individual components can together form a single system. Following are some reasons, which tells when to use component diagram:

1. To divide a single system into multiple components according to the functionality.
2. To represent the component organization of the system.

How to Draw a Component Diagram?

The component diagram is helpful in representing the physical aspects of a system, which are files, executables, libraries, etc. The main purpose of a component diagram is different from that of other diagrams. It is utilized in the implementation phase of any application.

Once the system is designed employing different UML diagrams, and the artifacts are prepared, the component diagram is used to get an idea of implementation. It plays an essential role in implementing applications efficiently.

Following are some artifacts that are needed to be identified before drawing a component diagram:

1. What files are used inside the system?
2. What is the application of relevant libraries and artifacts?
3. What is the relationship between the artifacts?

Following are some points that are needed to be kept in mind after the artifacts are identified:

1. Using a meaningful name to ascertain the component for which the diagram is about to be drawn.
2. Before producing the required tools, a mental layout is to be made.
3. To clarify the important points, notes can be incorporated.

Example of a Component Diagram

A component diagram for an online shopping system is given below:

Where to use Component Diagrams?

The component diagram is a special purpose diagram, which is used to visualize the static implementation view of a system. It represents the physical components of a system, or we can say it portrays the organization of the components inside a system. The components, such as libraries, files, executables, etc. are first needed to be organized before the implementation.

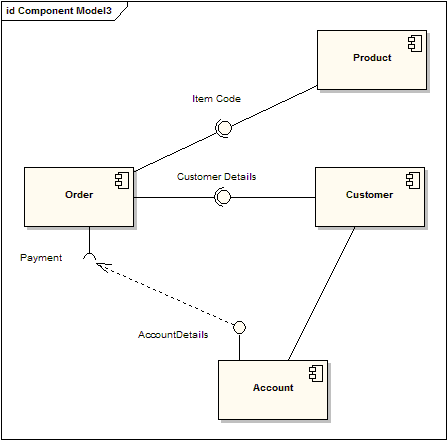
The component diagram can be used for the followings:

1. To model the components of the system.
2. To model the schemas of a database.
3. To model the applications of an application.
4. To model the system's source code.

# UML 2 Tutorial - Component Diagram

## Component Diagrams

Component diagrams illustrate the pieces of software, embedded controllers, etc., that will make up a system. A component diagram has a higher level of abstraction than a Class Diagram - usually a component is implemented by one or more classes (or objects) at runtime. They are building blocks so a component can eventually encompass a large portion of a system.

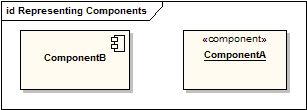


The diagram above demonstrates some components and their inter-relationships. Assembly connectors "link" the provided interfaces supplied by "Product" and "Customer" to the required interfaces specified by "Order". A dependency relationship maps a customer's associated account details to the required interface; "Payment", indicated by "Order".

Components are similar in practice to package diagrams, as they define boundaries and are used to group elements into logical structures. The difference between package diagrams and component diagrams is that Component Diagrams offer a more semantically rich grouping mechanism. With component diagrams all of the model elements are private, whereas package diagrams only display public items.

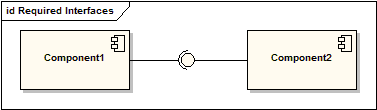
### Representing Components

Components are represented as a rectangular classifier with the keyword «component»; optionally the component may be displayed as a rectangle with a component icon in the right-hand upper corner.



### Assembly Connector

The assembly connector bridges a component’s required interface (Component1) with the provided interface of another component (Component2); this allows one component to provide the services that another component requires.



### Components with Ports

Using Ports with component diagrams allows for a service or behavior to be specified to its environment as well as a service or behavior that a component requires. Ports may specify inputs and outputs as they can operate bi-directionally. The following diagram details a component with a port for online services along with two provided interfaces order entry and tracking as well as a required interface payment.

