**UML**: Unified Modelling Language is a standard used for expressing in the form of object-oriented models. It is used in the representation of Use cases, workflows, system design and models, etc.

**Component diagram:**

The purpose of a component diagram is to show the relationship between different components in a system

There exists a whole development approach that revolves around components: component-based development (CBD). In this approach, component diagrams allow the planner to identify the different components so the whole system does what it's supposed to do.

Component diagrams can help your team:

* Imagine the system’s physical structure.
* Pay attention to the system’s components and how they relate.
* Emphasize the service behavior as it relates to the interface

Component diagrams should communicate:

* The scope of your system
* The overall structure of your software system
* Goals that the system helps human or non-human entities (known as actors) achieve

**Important terms and symbols:**

Component symbol: A component provides and consumes behavior through interfaces, as well as through other components. In UML 1.0, a component is modeled as a rectangular block with two smaller rectangles protruding from the side. In UML 2.0, a component is modeled as a rectangular block with a small image of the old component diagram shape.

Node symbol: Represents hardware or software objects, which are of a higher level than components

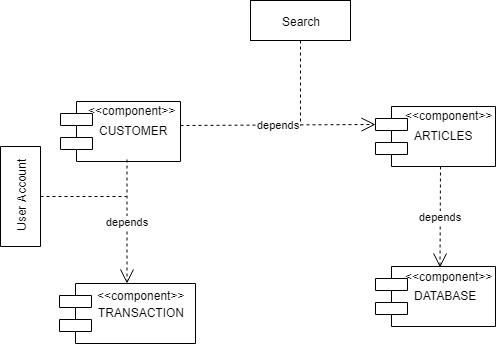
Interface symbol: Shows input or materials that a component either receives or provides. Interfaces can be represented with textual notes or symbols, such as the lollipop, socket, and ball-and-socket shapes

|  |  |
| --- | --- |
|  |  |

Port symbol: Specifies a separate interaction point between the component and the environment. Ports are symbolized with a small square.

Dependency symbol: Shows that one part of your system depends on another. Dependencies are represented by dashed lines linking one component (or element) to another.

Example of component diagram: for ecommerce website NaturoPet



**Deployment Diagram:**

A deployment diagram models the run-time architecture of a system. It shows the configuration of the hardware elements (nodes) and shows how software elements and artifacts are mapped onto those nodes.  
In the context of the Unified Modeling Language (UML), a deployment diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. The information that the software generates is called an artifact.

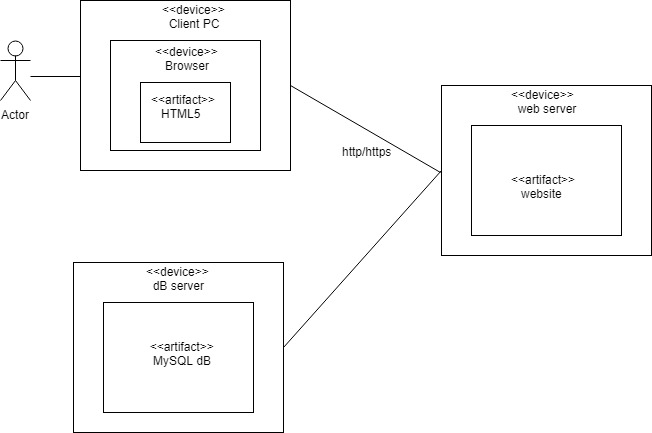
Deployment diagrams:

* Show which software elements are deployed by which hardware elements.
* Illustrate the runtime processing for hardware.
* Provide a view of the hardware system’s topology.

**Important terms:**

* **Artifact:** A product developed by the software, symbolized by a rectangle with the name and the word “artifact” enclosed by double arrows.
* **Association**: A line that indicates a message or other type of communication between nodes.
* **Component:** A rectangle with two tabs that indicates a software element.
* **Dependency:** A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.
* **Interface:** A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.
* **Node:** A hardware or software object, shown by a three-dimensional box.
* **Node as container:** A node that contains another node inside of it—such as in the example below, where the nodes contain components.
* **Stereotype:** A device contained within the node, presented at the top of the node, with the name bracketed by double arrows.

**Example of Deployment Diagram:** for ecommerce website NaturoPet



**Object Diagram:**

A static [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) object diagram is an instance of a [class diagram](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/); it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships at a point in time. It may be considered a special case of a class diagram or a [communication diagram](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-communication-diagram/).

The use of object diagrams is fairly limited, mainly to show examples of data structures.

* During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.
* Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.

**Terms and symbols:**

**Object Names**:

* Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.

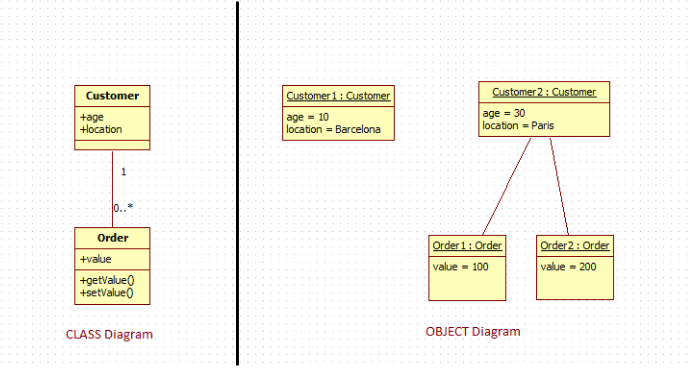
**Object Attributes**:

* Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.

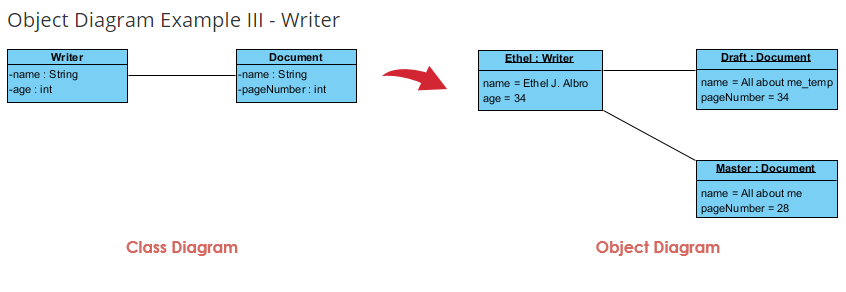
**Links:**

* Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

**Example-1: from the internet:**



**Example-2: from the internet:**



**References:**

<https://sparxsystems.com/resources/tutorials/uml2/deployment-diagram.html>

<https://www.lucidchart.com/pages/uml-deployment-diagram>

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/>