UML DEPLOYMENT AND COMPONENT DIAGRAMS

Deployment Diagrams : A deployment diagram shows the assignment of concrete software artifacts (such as executable files) to computational nodes (something with processing services). It shows the deployment of software elements to the physical architecture and the communication (usually on a network) between physical elements

Two Nodes of Deployment Diagram :

1) Device Node : This is a Physical Computing Resource representing a computer with memory or mobile.

2) Execution Environment Node :

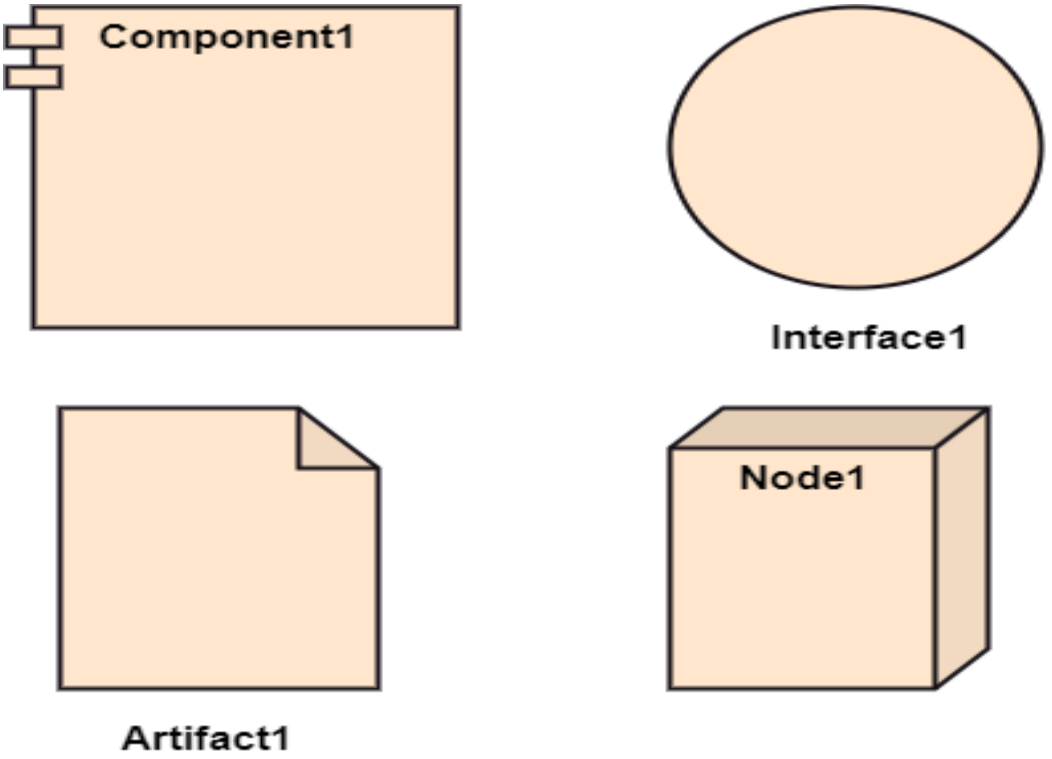
This is a software computing resource that runs within an outer node (such as a computer) and which itself provides a service to host and execute other executable software elements. For example:

- an operating system (OS) is software that hosts and executes programs
- a virtual machine (VM, such as the Java or .NET VM) hosts and executes programs
- a database engine (such as PostgreSQL) receives SQL program requests and executes them, and hosts/executes internal stored procedures (written in Java or a proprietary language)
- a Web browser hosts and executes JavaScript, Java applets, Flash, and other executable technologies
- a workflow engine
- a servlet container or EJB container

Symbol and notation of Deployment diagram

The deployment diagram consist of the following notations:

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

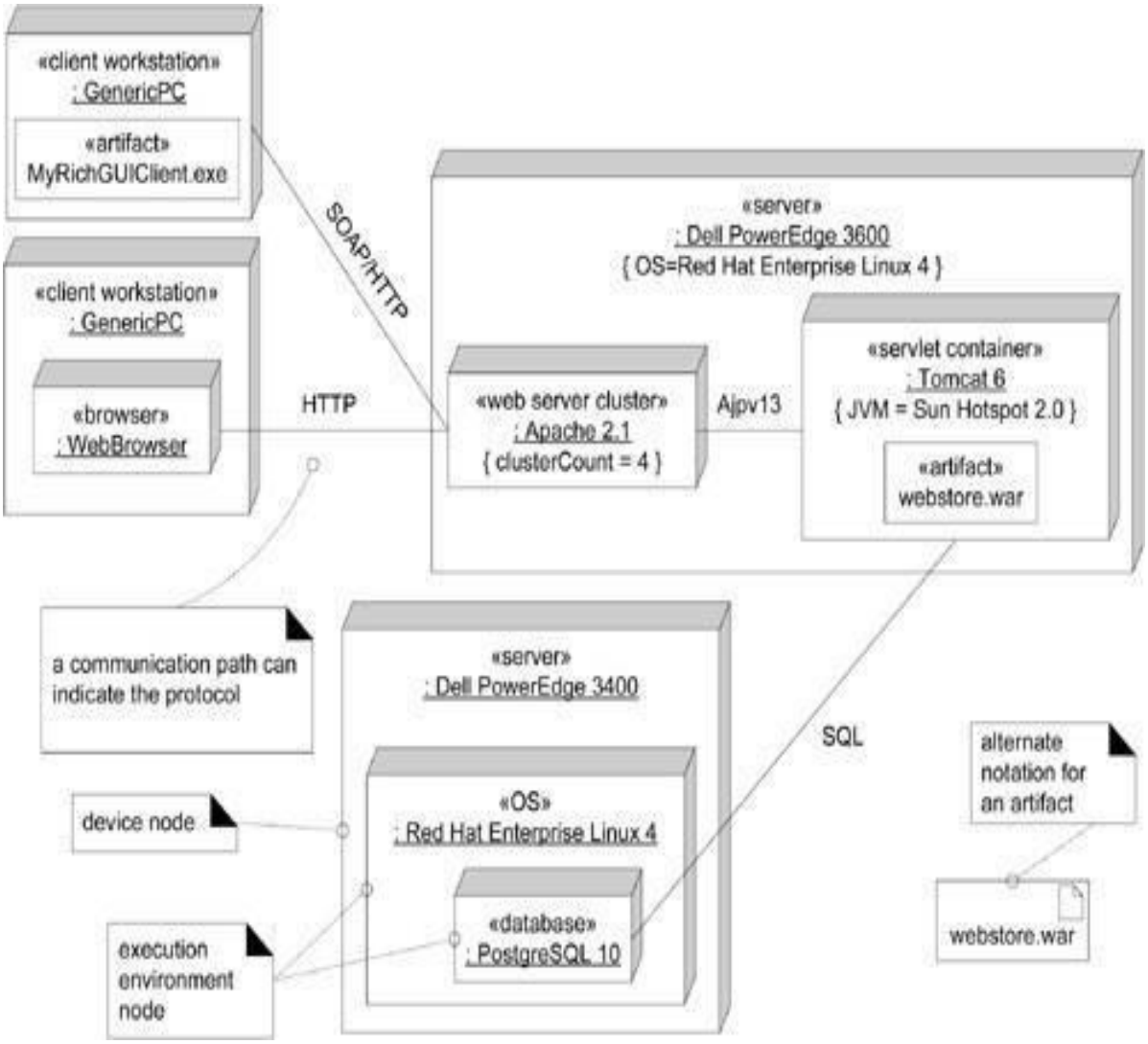


- As the UML specification suggests, many node types may show stereotypes, such as «server», «OS», «database», or «browser», but these are not official predefined UML stereotypes.
- Note that a device node or EEN may contain another EEN. For example, a virtual machine within an OS within a computer.
- A particular EEN can be implied, or not shown, or indicated informally with a
- UML property string; for example, {OS=Linux}.
- The normal connection between nodes is a communication path, which may be labeled with the protocol. These usually indicate the network connections.
- A node may contain and show an artifact a concrete physical element, usually a file. This includes executables such as JARs, assemblies, .exe files, and scripts. It also includes data files such as XML, HTML, and so forth.

Deployment Diagram Ex: Next Generation POS System

In the diagram ,there are 2 servers namely DellpowerEdge 3600 with RedHat Linux OS , Tomcat 6, Apache 2.1 are software computing resources , in tomcat server webstore.war file is loaded. In the other server DellpowerEdge 3400 with RedHat Linux OS, database PostgresSQL 10 are software computing resources.

There are 2 client nodes connected to server via HTTP & SOAP /HTTP protocol. In first client exe file is shown as artifact. In the other client , web base application is enabled by browser .



COMPONENT DIAGRAMS

A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. A component serves as a type, whose conformance is defined by these provided and required interfaces.

Features of UML component

1) It has interfaces

2) it is modular, self-contained and replaceable.

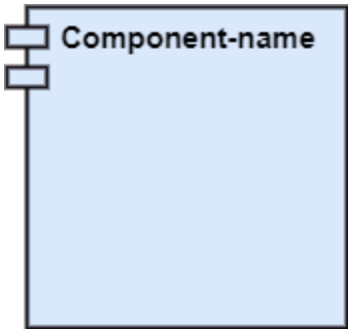
The second point implies that a component tends to have little or no dependency on other external elements . it is a relatively stand-alone module.

Example : A good analogy for software component modeling is a home entertainment system; we expect to be able to easily replace the DVD player or speakers. They are modular, self-contained, replaceable, and work via standard interfaces.

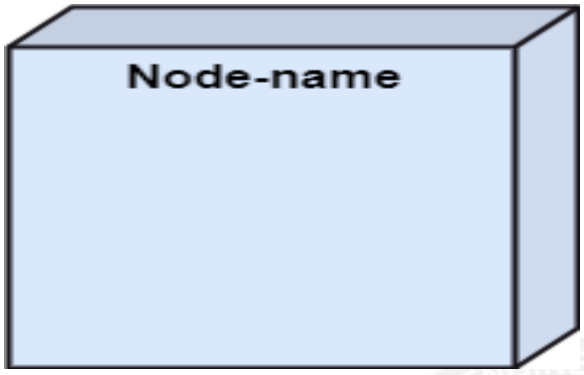
For example, at a large-grained level, a SQL database engine can be modeled as a component; any database that understands the same version of SQL and supports the same transaction semantics can be substituted. At a finer level, any solution that implements the standard Java Message Service API can be used or replaced in a system.

Elements:

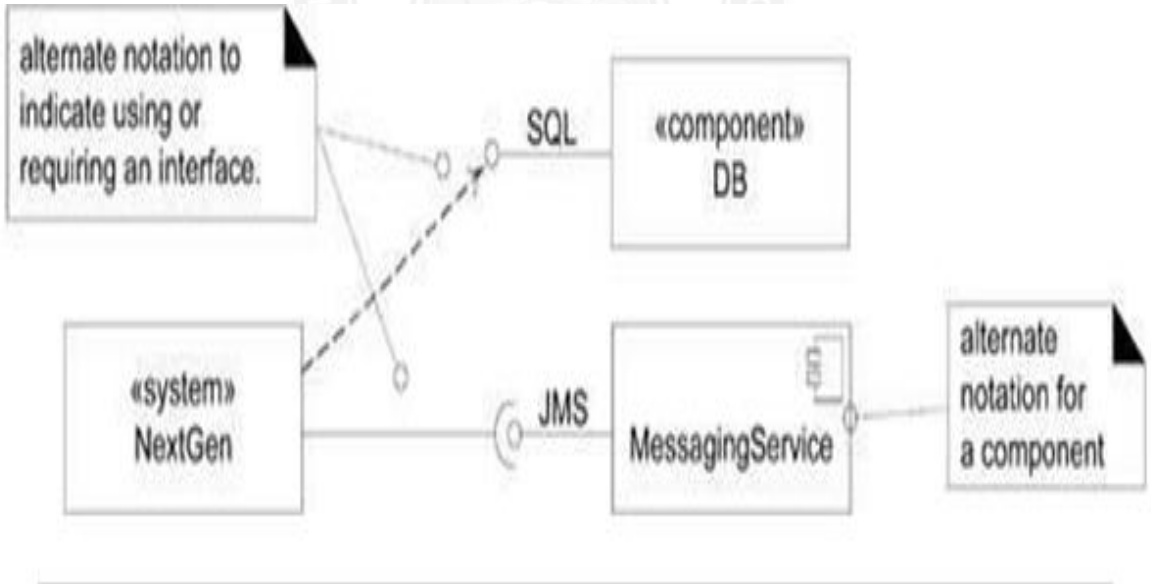
a) A component



b) A Node



Ex: Next Generation POS system



WHEN TO USE COMPONENT AND DEPLOYMENT DIAGRAMS

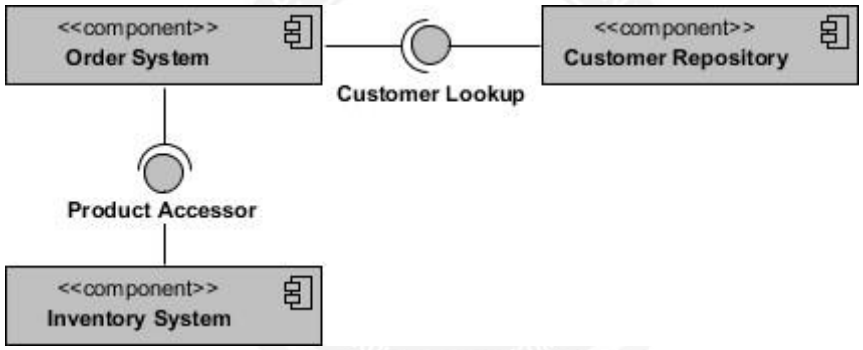
Component diagrams are used to visualize the static implementation view of a system. Component diagrams are special type of UML diagrams used for different purposes.

Component diagrams can be used to −

- Model the components of a system.
- Model the database schema.
- Model the executables of an application.
- Model the system's source code.

Model the components of a system *Ex:*

Order System



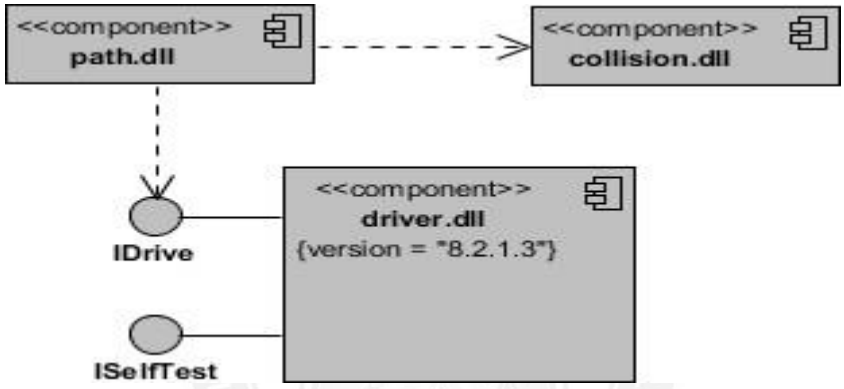
Modeling Source Code

- Either by forward or reverse engineering, identify the set of source code files of interest and model them as components stereotyped as files.
- For larger systems, use packages to show groups of source code files.
- Consider exposing a tagged value indicating such information as the version number of the source code file, its author, and the date it was last changed. Use tools to manage the value of this tag.
- Model the compilation dependencies among these files using dependencies.

Again, use tools to help generate and manage these dependencies

Modeling an Executable Release

- Identify the set of components to model.
- Consider the stereotype of each component in this set. find a small number of different kinds of components (such as executables, libraries, tables, files, and documents).
- For each component in this set, consider its relationship to its neighbors. It shows only dependencies among the comp



Modeling a Physical Database

- Identify the classes in the model that represent the logical database schema.
- Select a strategy for mapping these classes to tables.
- To visualize, specify, construct, and document your mapping, create a component diagram that contains components stereotyped as tables.
- Where possible, use tools to help you transform your logical design into physical design.