Unit-IV

Structural Modeling: Package Diagram, Composite Structure Diagram, Component Diagram, Deployment Diagram, profile Diagram.

……

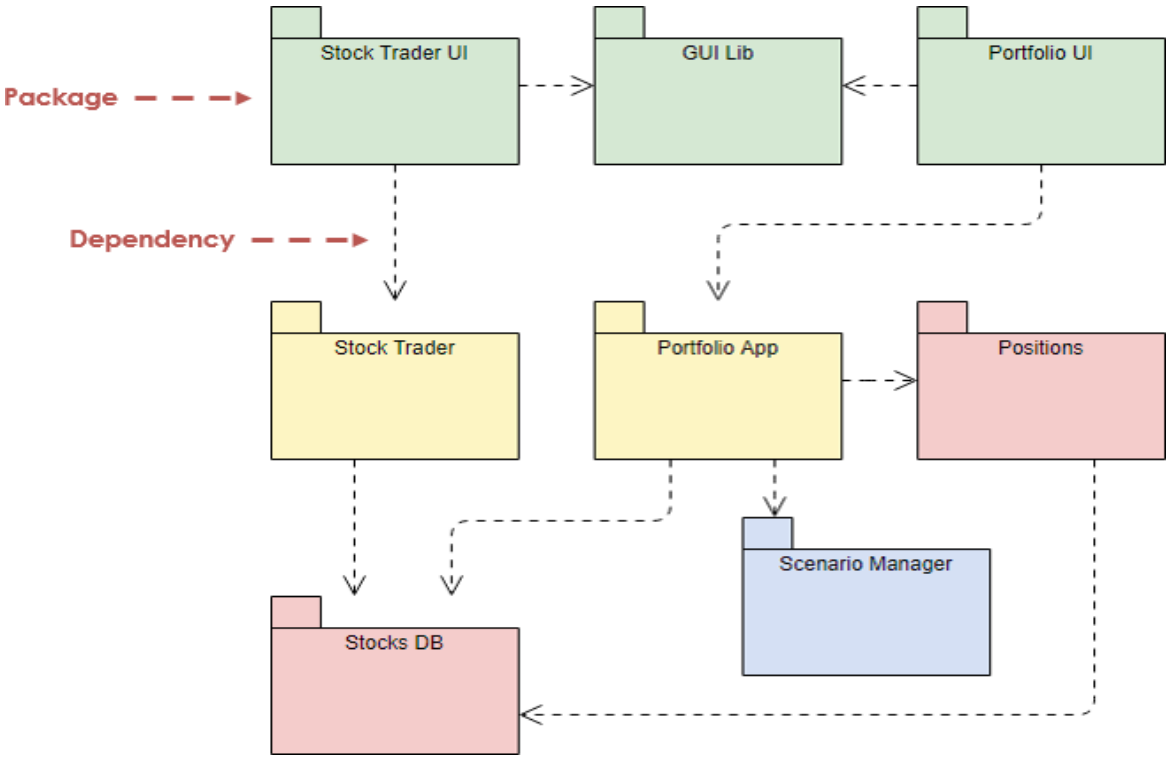
Package Diagram:

A package diagram is a UML diagram composed only of packages and the dependencies between them. A package is a UML construct that enables you to organize model elements, such as use cases or classes, into groups. Packages are depicted as file folders and can be applied on any UML diagram.

Create a package diagram to:

- Depict a high-level overview of your requirements (overviewing a collection of UML Use Case diagrams)
- Depict a high-level overview of your architecture/design (overviewing a collection of UML Class diagrams).
- To logically modularize a complex diagram.
- To organize Java source code.

Package diagram shows the arrangement and organization of model elements in middle to large scale project that can be used to show both structure and dependencies between sub-systems or modules.



Package Diagram

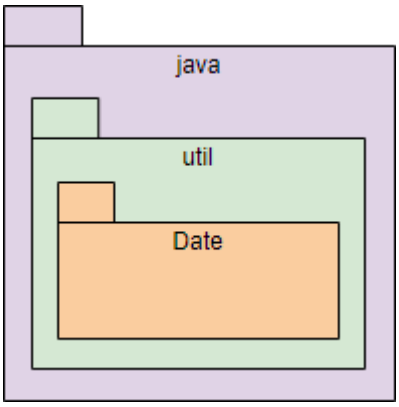
Package Diagram Notations:

Package diagrams are used to structure high level systems. Packages are used for organizing large system which contains diagrams, documents and other key deliverables. In other words, packages can be used as a part of other diagrams also.

1.Nested and Hierarchical Packages:

A package can be represented as a hierarchical structure with nested packages. Atomic module for nested package are usually class diagrams.

The figure below gives an example of package diagram that consists of several nested packages.

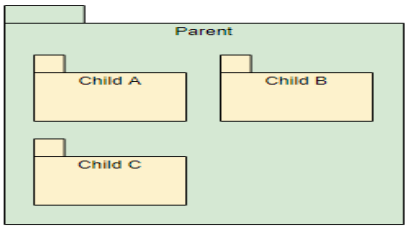


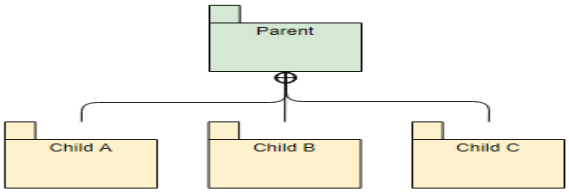
There are few constraints while using package diagrams, they are as follows.

- The name of packages should be unique within a system. However, it is allowed for classes inside different packages to have same name. For Example, Package::Product & Shipping::Product are allowed.
- Users should avoid using package name delivered by the programming language. For Example, Java provides Date as a package. So, programmers should construct package with name Date.
- Packages can include whole diagrams, name of components alone or no components at all.

Package Containment:

- Packages are shown in static diagrams
- Two equivalent ways to show containment:

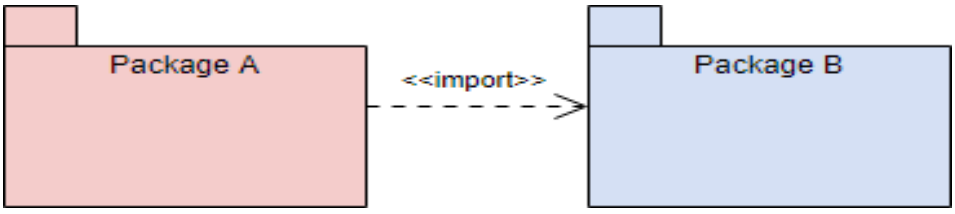




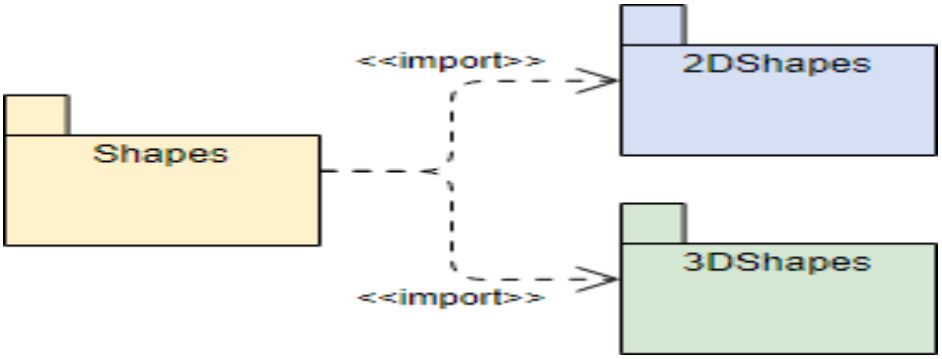
Dependency:

There are two sub-types involved in dependency. They are <<access>> & <<import>>. Though there are two stereotypes users can use their own stereotype to represent the type of dependency between two packages.

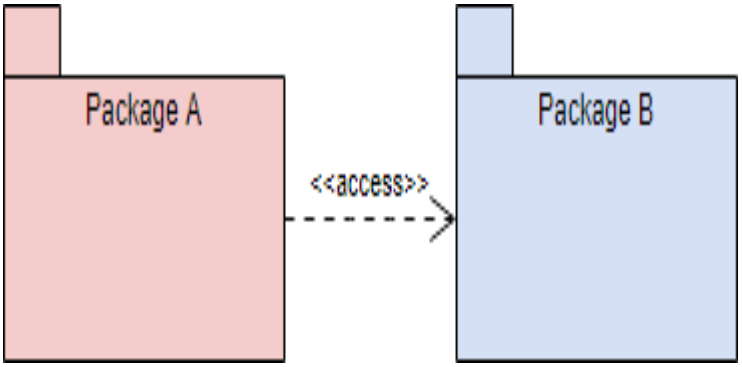
<<import>> - one package imports the functionality of other package



Example – <<import>> Dependency



<<access>> - one package requires help from functions of other package



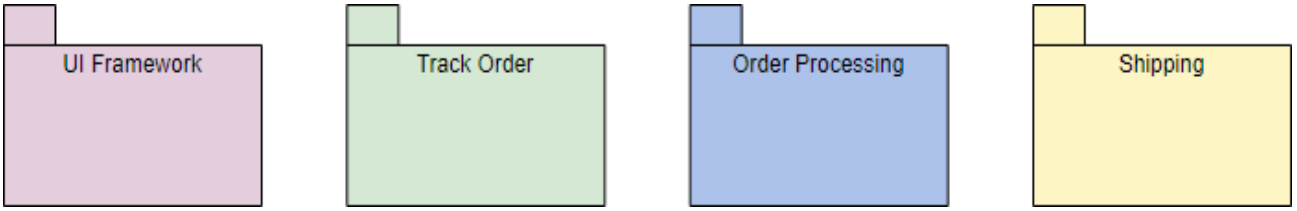
How to Create a Package Diagram?

The following example shows the Track Order Service for an online shopping store.

Track Order Service is responsible for providing tracking information for the products ordered by customers. Customer types in the tracking serial number, Track Order Service refers the system and updates the current shipping status to the customer.

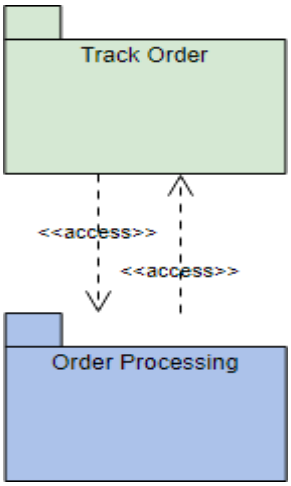
Step 1 - Identify the packages present in the system

- There is a **"track order"** service, it has to talk with other module to know about the order details, let us call it **"Order Processing"**.
- Next after fetching Order Details it has to know about shipping details, let us call that as **"Shipping"**.
- Finally if knows the status of the order it has to update the information to the user, let us call this module as **"UI Framework"**.

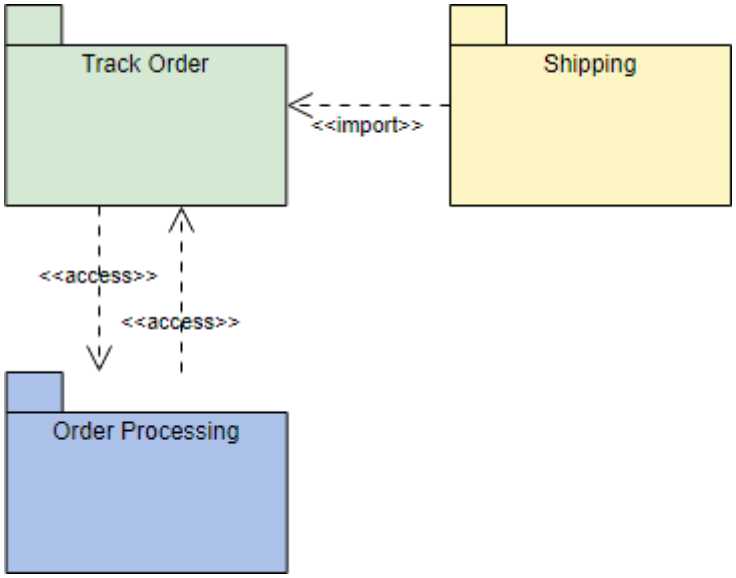


Step 2 - Identify the dependencies

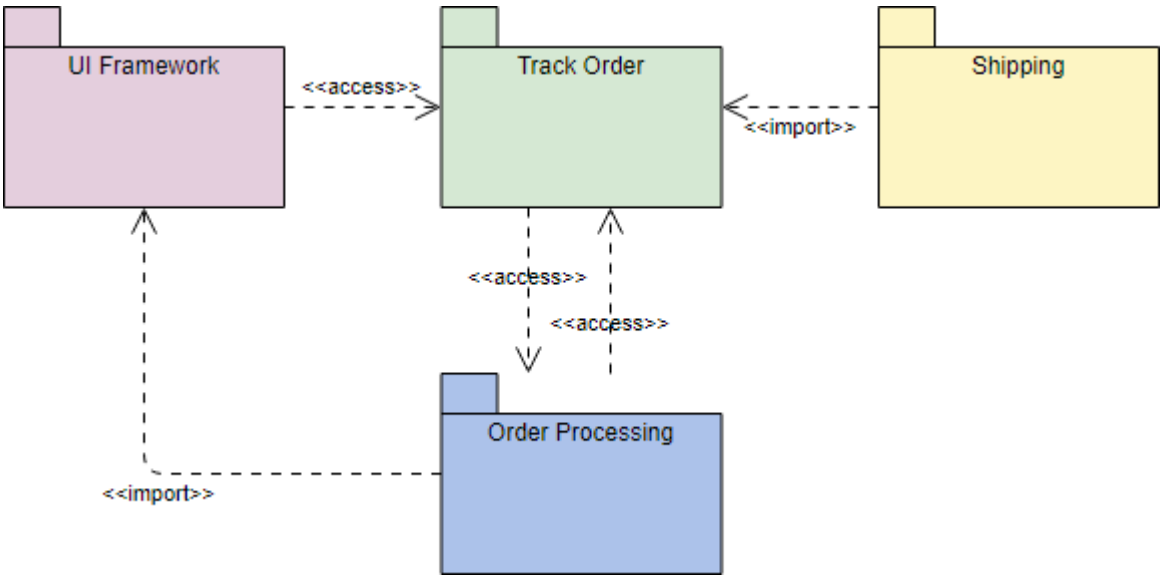
- **"Track order"** package should get order details from **"Order Processing"** and on the other hand, "Order Processing" also requires the tracking information from the "Track Order" package, thus, the two modules are accessing each other which suffices <<access>> dual dependency.



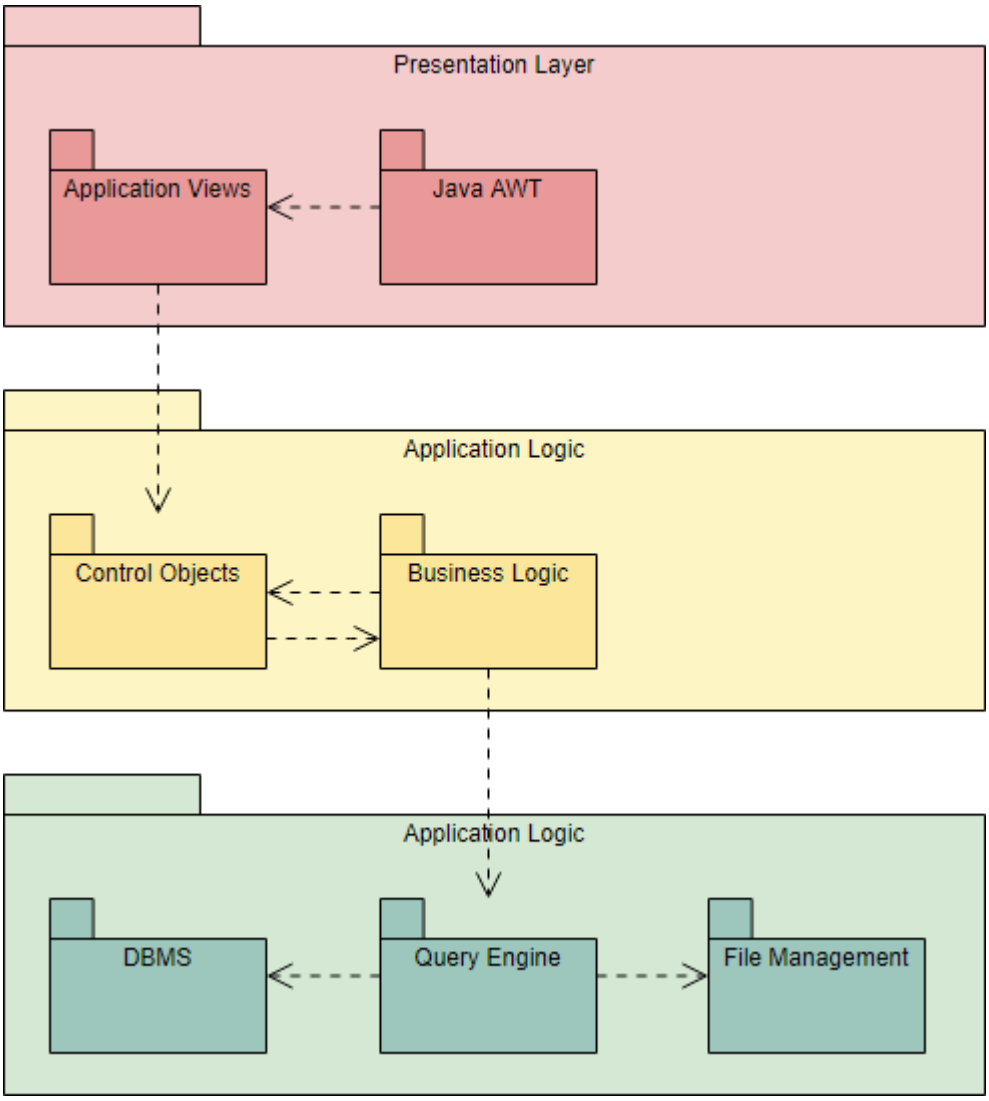
To know shipping information, "Shipping" requires to import "Track Order" to complete the shipping process.



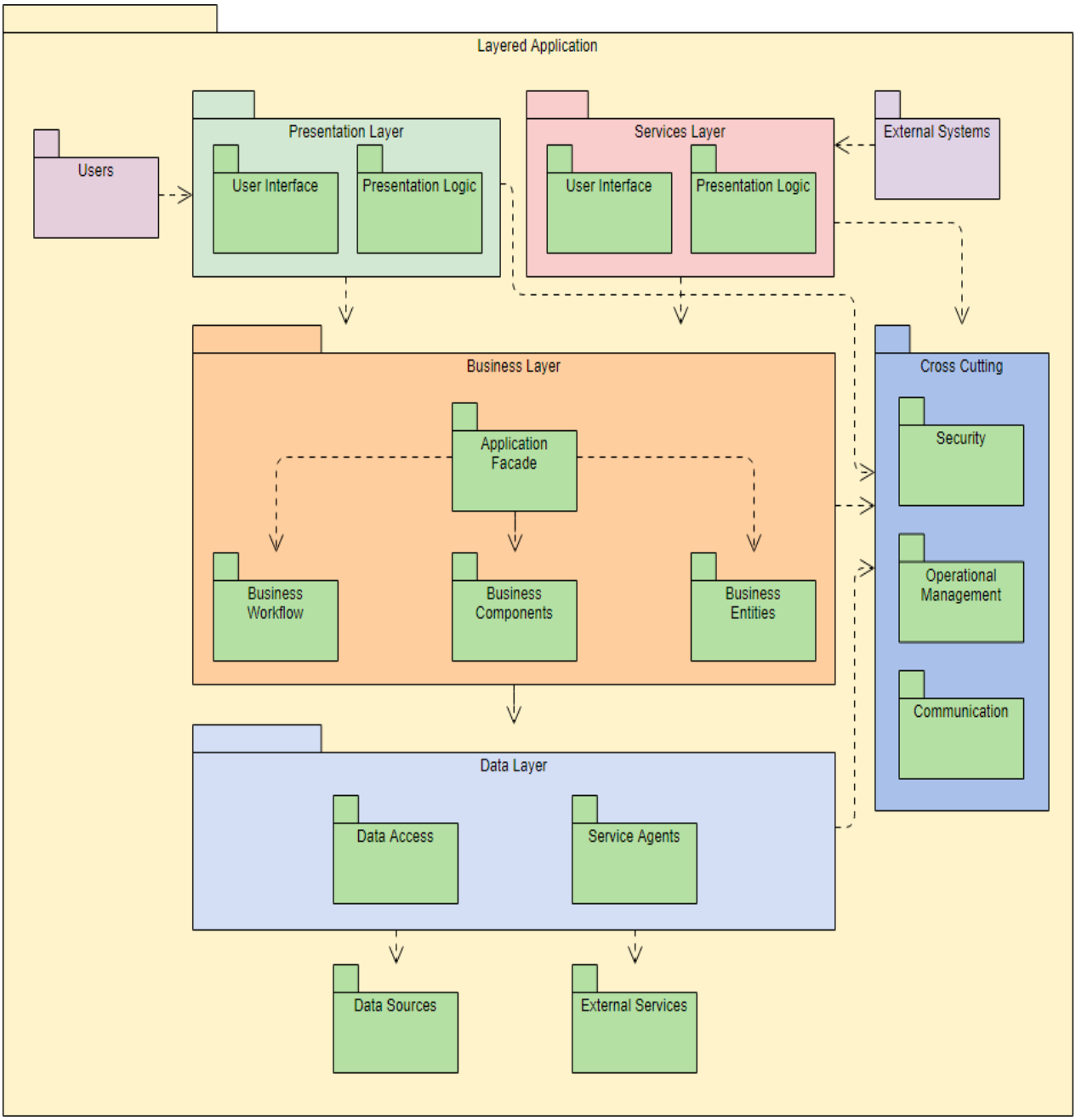
- **Step 3 -** Finally, Track Order dependency to UI Framework is also mapped in to the diagram which completes the Package Diagram for Track Order subsystem.



- **Package Diagram Examples:**
- **Package Diagram Example – MVC Structure:**



- **Package Diagram Example - Layering Structure**

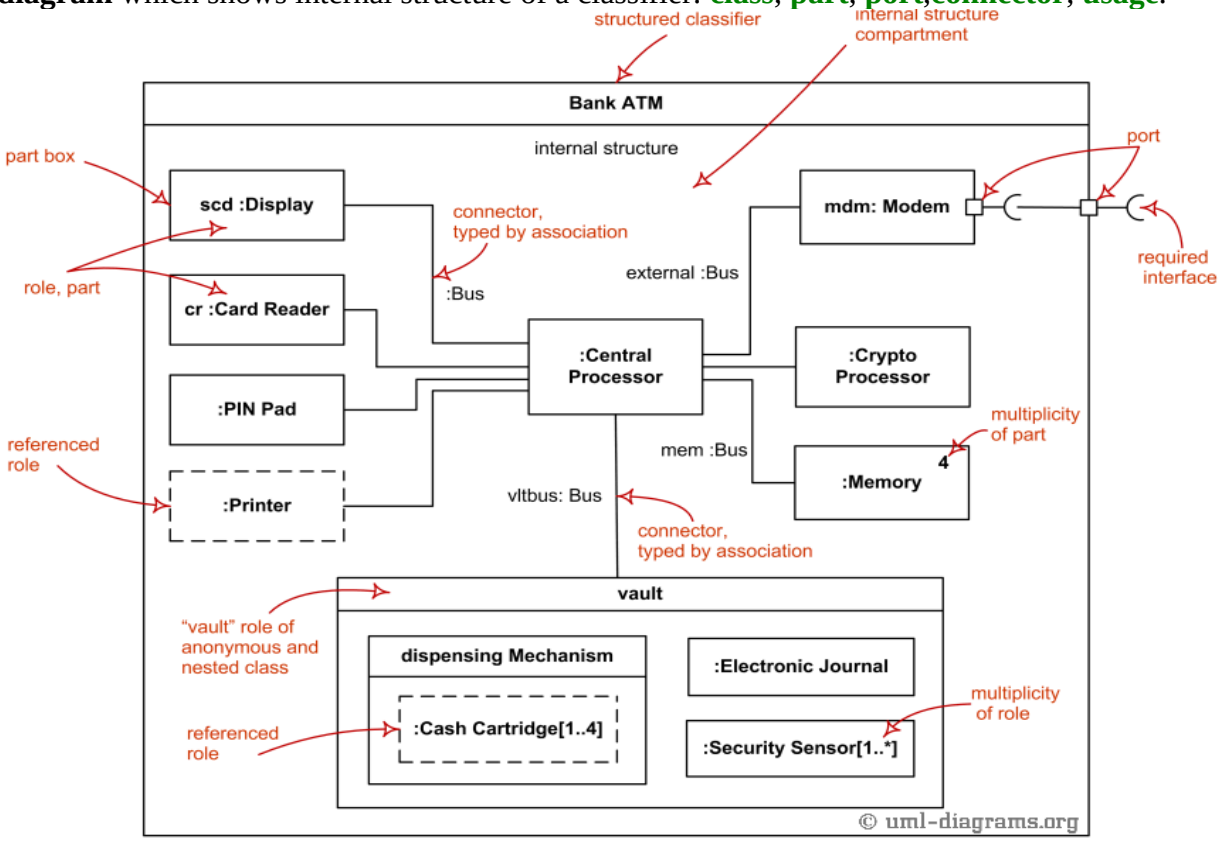


Composite Structure Diagram:

Composite Structure Diagram is one of the new artifacts added to UML 2.0. A composite structure diagram is a UML structural diagram that contains classes, interfaces, packages, and their relationships, and that provides a logical view of all, or part of a software system.

- It shows the internal structure (including parts and connectors) of a structured classifier or collaboration.
- A composite structure diagram performs a similar role to a class diagram, but allows you to go into further detail in describing the internal structure of multiple classes and showing the interactions between them.
- You can graphically represent inner classes and parts and show associations both between and within classes.

The following graphical elements are typically drawn in a **composite structure diagram** which shows internal structure of a classifier: **class**, **part**, **port**,**connector**, **usage**.



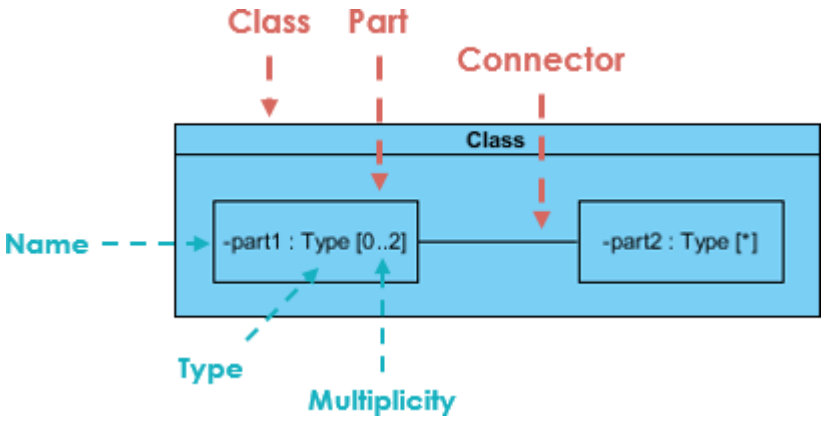
Composite structure diagram overview shows elements of internal structure of structured classifier - roles, parts, connectors.

Purpose of Composite Structure Diagram:

- Composite Structure Diagrams allow the users to "Peek Inside" an object to see exactly what it is composed of.
- The internal actions of a class, including the relationships of nested classes, can be detailed.
- Objects are shown to be defined as a composition of other classified objects.

Composite Structure Diagram at a Glance:

- Composite Structure Diagrams show the internal parts of a class.
- Parts are named: partName:partType[multiplicity]
- Aggregated classes are parts of a class but parts are not necessarily classes, a part is any element that is used to make up the containing class.



Basic Concepts of Composite Structure Diagram:

The key composite structure entities identified in the UML 2.0 specification are structured classifiers, parts, ports, connectors, and collaborations.

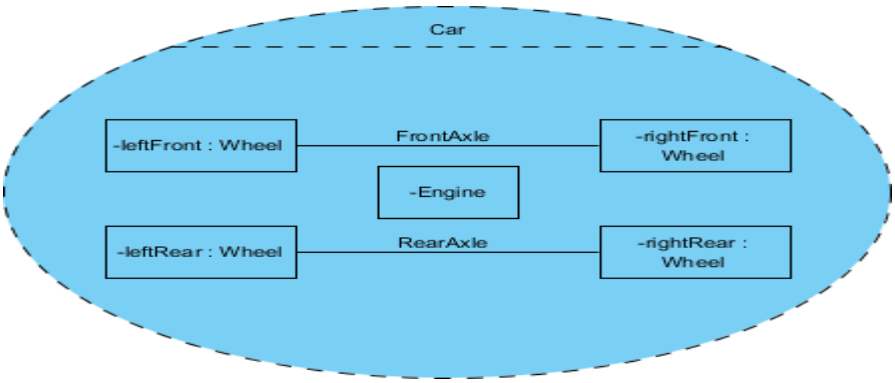
Collaboration:

A collaboration describes a structure of collaborating parts (roles). A collaboration is attached to an operation or a classifier through a Collaboration Use. You use a collaboration when you want to define only the roles and connections that are required to accomplish a specific goal of the collaboration.

For example, the goal of a collaboration can be to define the roles or the components of a classifier. By isolating the primary roles, a collaboration simplifes the structure and clarifies behavior in a model.

Example

In this example the Wheels and the Engine are the Parts of the Collaboration and the FrontAxle and the RearAxle are the Connectors. The Car is the Composite Structure that shows the parts and the connections between the parts.



Parts:

A part is a diagram element that represents a set of one or more instances that a containing structured classifier owns. A part describes the role of an instance in a classifier. You can create parts in the structure compartment of a classifier, and in several UML diagrams such as composite structure, class, object, component, deployment, and package diagrams.

Port:

A port defines the interaction point between a classifier instance and its environment or between the behavior of the classifier and its internal parts.

Interface:

Composite Structure diagram supports the ball-and-socket notation for the provided and required interfaces. Interfaces can be shown or hidden in the diagram as needed.

Connector:

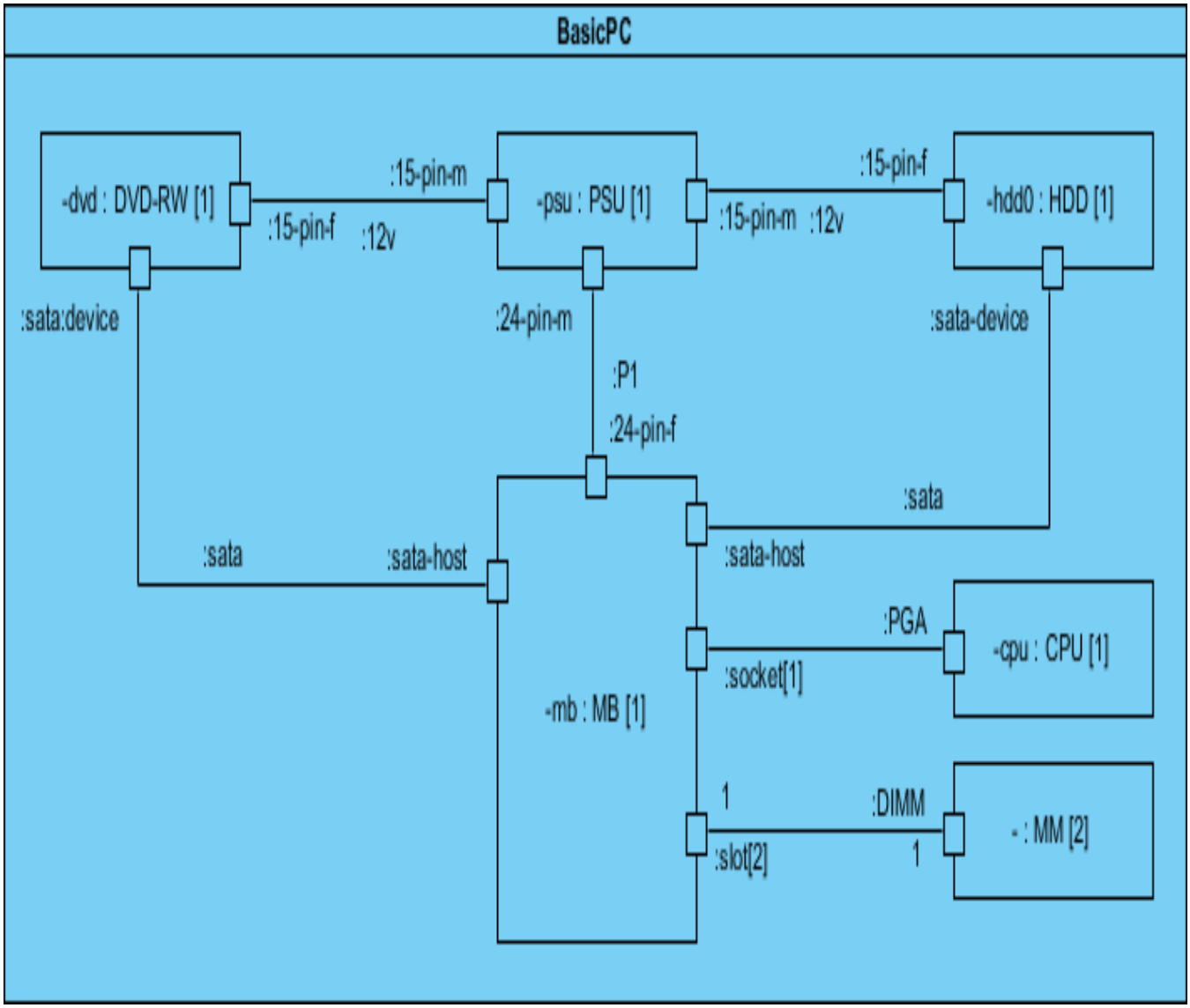
A line that represents a relationship in a model. When you model the internal structure of a classifier, you can use a connector to indicate a link between two or more instances of a part or a port. The connector defines the relationship between the objects or instances that are bound to roles in the same structured classifier and it identifies the communication between those roles. The product automatically specifies the kind of connector to create.

Composite Structure Diagram Example - Computer System

Let's develop the composite structure diagram for a computer system which include the following a list of the components:

- Power Supply Unit (PSU)
- Hard Disk Drive (HDD)
- DVD-RW
- Optical Drive (DVD-RW)
- Memory Module (MM)

We will assume for the moment that the mainboard is of the type that has a sound card and display adapter built in.



Profile diagram:

Profile diagram, a kind of structural diagram in the Unified Modeling Language (UML), provides a generic extension mechanism for customizing UML models for particular domains and platforms.

Extension mechanisms allow refining standard semantics in strictly additive manner, preventing them from contradicting standard semantics.

Profiles are defined using **stereotypes**, **tagged value definitions**, and **constraints** which are applied to specific model elements, like Classes, Attributes, Operations, and Activities.

A Profile is a collection of such extensions that collectively customize UML for a particular domain (e.g., aerospace, healthcare, financial) or platform (J2EE, .NET).

Basic Concepts of Profile Diagram:

Profile diagram is basically an extensibility mechanism that allows you to extend and customize UML by adding new building blocks, creating new properties and specifying new semantics in order to make the language suitable to your specific problem domain.

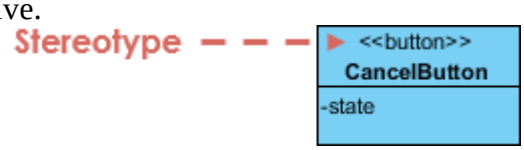
Profile diagram has three types of extensibility mechanisms:

- **Stereotypes**
- **Tagged Values**
- **Constraints**

Stereotypes:

Stereotypes allow you to increase vocabulary of UML. You can add, create new model elements, derived from existing ones but that have specific properties that are suitable to your problem domain. Stereotypes are used to introduce new building blocks that speak the language of your domain and look primitive. It allows you to introduce new graphical symbols.

For example: When modeling a network you might need to have symbols for <<router>>, <<switches>>, <<hub>> etc. A stereotype allows you to make these things appear as primitive.



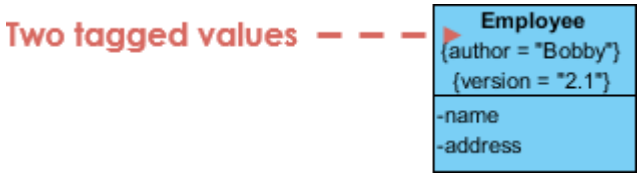
Tagged Values:

Tagged values are used to extend the properties of UML so that you can add additional information in the specification of a model element. It allows you to specify keyword value pairs of a model where keywords are the attributes. Tagged values are graphically rendered as string enclose in brackets.

For Example: Consider a release team responsible for assembling, testing and deployment of a system. In such case it is necessary to keep a track on version and test results of the main subsystem. Tagged values are used to add such info.

Tagged Value can be useful for adding properties to the model for some useful purposes:

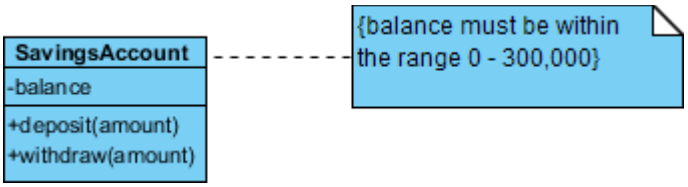
- Code generation
- Version control
- Configuration management
- Authorship
- Etc



Constraints:

They are the properties for specifying semantics or conditions that must be held true at all the time. It allows you to extend the semantics of UML building block by adding new protocols. Graphically a constraint is rendered as string enclose in brackets placed near associated element.

For example: In development of a real time system it is necessary to adorn the model with some necessary information such as response time. A constraint defines a relationship between model elements that must be use *{subset}* or *{xor}*. Constraints can be on attributes, derived attributes and associations. It can be attached to one or more model elements shown as a note as well.



Profile Diagram - How it Works

The extension mechanism in UML 1.1 is relatively imprecise in the sense that extensions could be made only on the basis of the primitive data type string. UML 2.0 lets you use arbitrary data structures for extended elements, which means that more extensive and more precise model extensions are now possible.

The profiles mechanism is not a first-class extension mechanism. It does **NOT allow** to:

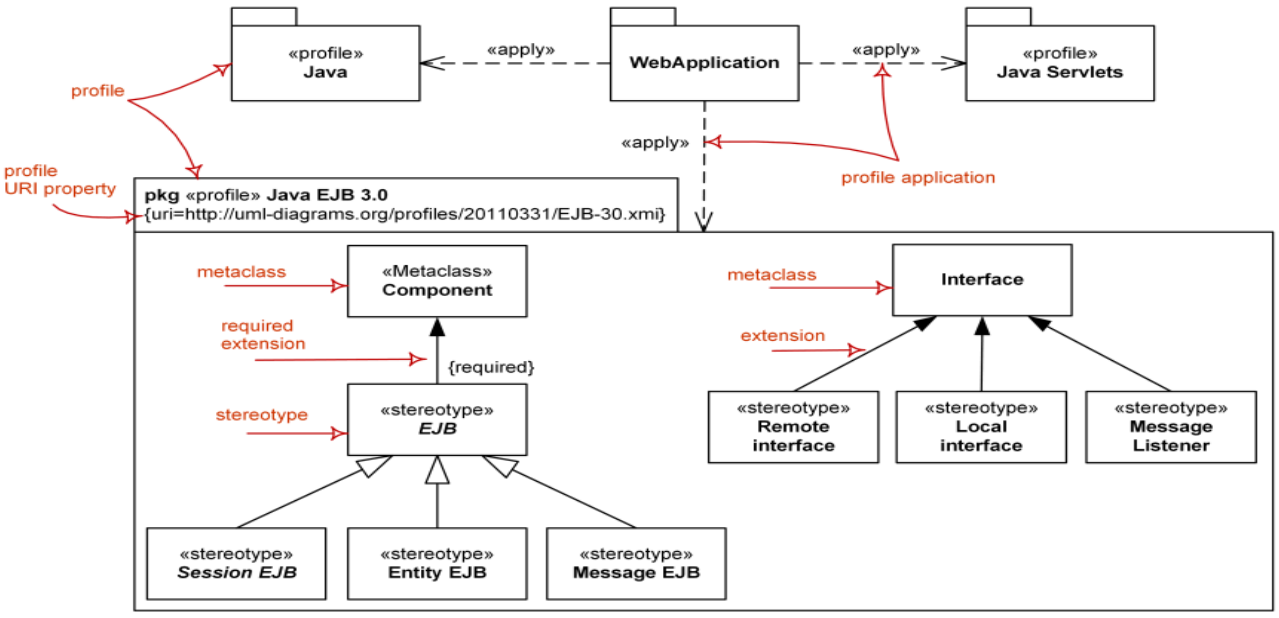
- Modify existing metamodels
- Create a new metamodel like MOF does

Profile only allows adaptation or customization of an existing metamodel. In UML 2.0 or above, profiles can also be dynamically combined so that several profiles will be applied at the same time on the same model.

In the figure below, we define a profile of EJB as a package. The bean itself is extended from component metamodel as an abstract bean.

The abstract bean can be concreted as either an Entity Bean or Session Bean. An EJB has two types of remote and home interfaces. An EJB also contains a special kind of artifact called JAR file for storing a collection of Java code.

Graphical nodes and edges used on profile diagrams
are: **profile**, **metaclass**, **stereotype**, **extension**, **reference**, **profile application**.

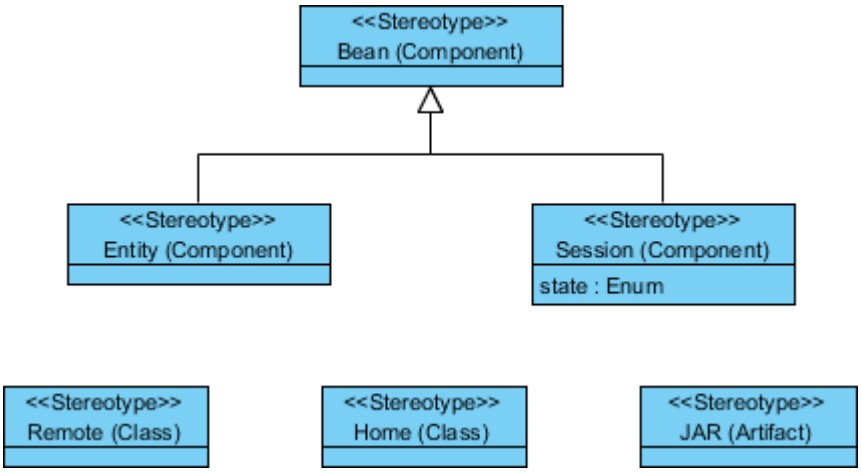


*Major elements of UML profile diagram - **profile**, **stereotype**, **metaclass**, **extension**, **profile application**.*

Profile Diagram at a Glance:

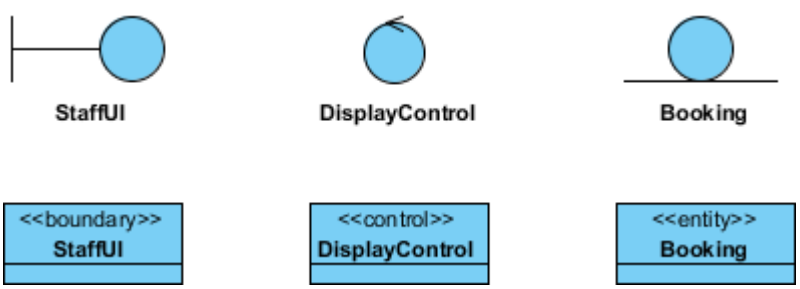
In this example, we can see that a stereotype can extend from one or more metaclasses. This extension relationship is depicted as an arrow with a continuous line and filled arrowhead. The arrow points away from the stereotype to the metaclass.

In the figure below, we define a profile of EJB as a package. The bean itself is extended from component metamodel as an abstract bean. The abstract bean can be concreted as either an Entity Bean or Session Bean. An EJB has two types of remote and home interfaces. An EJB also contains a special kind of artifact called JAR file for storing a collection of Java code.



Textual vs Graphic Icon Stereotype:

Stereotypes can be in textual or graphical representation. The icon can also replace the normal class box. For Example: People often use these 3 stereotyped class representations to model the software MVC framework:



Other Popular Usages for UML profiles:

Every technical target, i.e. programming language, middleware, library or database is a natural candidate for defining UML profile. For examples:

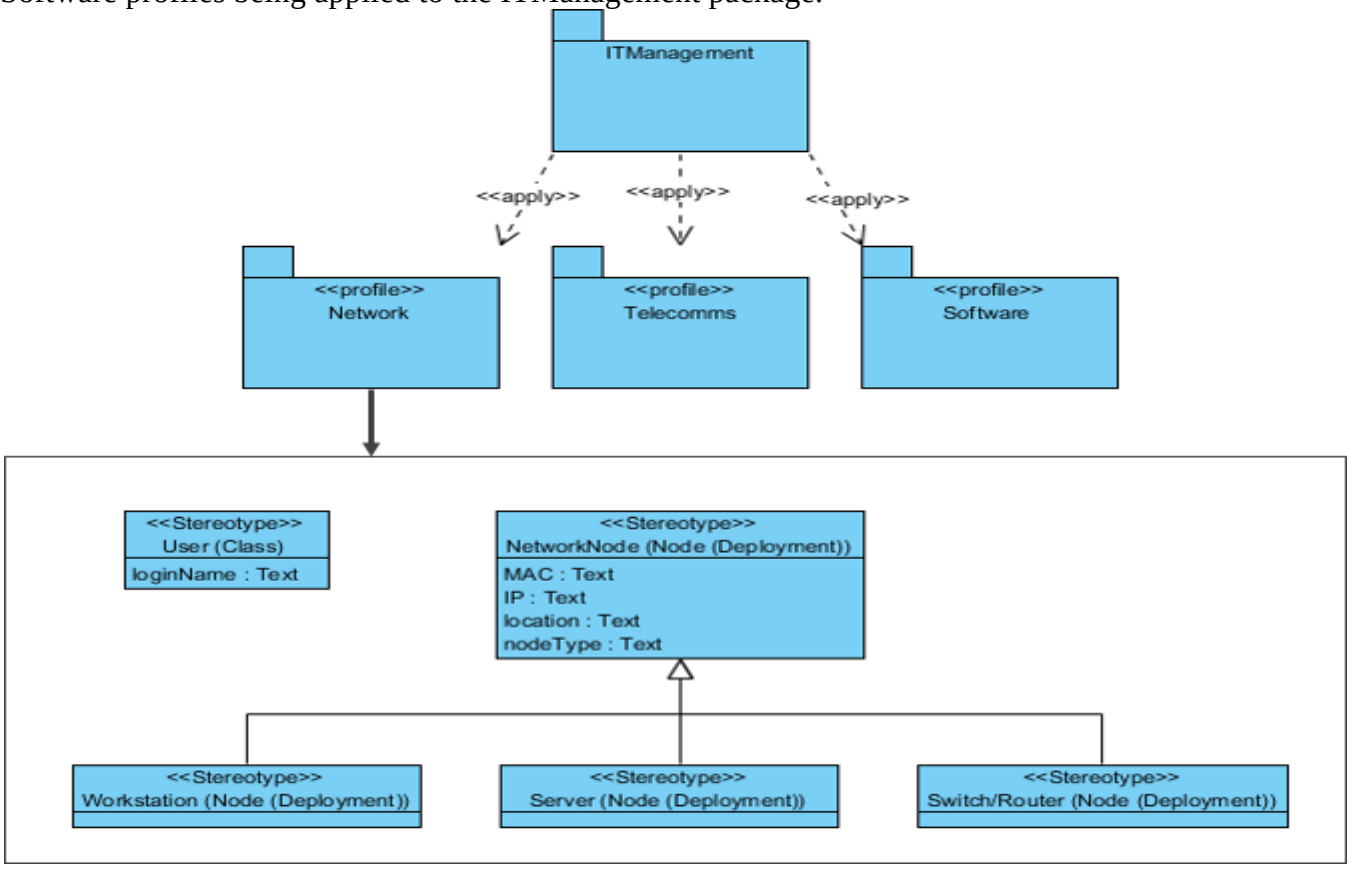
- CORBA
- EJB
- C++ or JAVA
- ORACLE or MYSQL

Applying Stereotype of a Profile:

- To use stereotypes in a specific application, you must first integrate the profile that contains the stereotypes. You do this with a dashed arrow than open arrowhead pointing away from the package of the application towards the profile. This arrow is labelled with the keyword <<apply>>.

Profile Diagram Example I - IT Management

A profile is applied to another package in order to make the stereotypes in the profile available to that package. The illustration below shows the Network, Telecomms and Software profiles being applied to the ITManagement package.



Profile Diagram Example II - EJB Application

In this example, we define a customer - session bean based on the EJB Profile we had developed previously: