Lab # 2

**AVM: Physical View and Development View**

**OBJECTIVE**

* Study Physical View Model and Development or Module View Model.
* Learn Deployment Diagram and Component diagram in Physical model.
* Learn Package Diagram in Development and Module model.

**Theory:**

1. **Physical View Model:**

The physical view describes installation, configuration, and deployment of the software application. It concerns itself with how to deliver the deployable system. The physical view shows the mapping of software onto hardware. It is particularly of interest in distributed or parallel systems. The components are hardware entities (processors), and the links are communication pathways; together these specify how the various elements such as communication protocols and middleware servers found in the logical, process, and development views are mapped onto the various nodes in the runtime environment.

1. **Deployment Diagram:**

A deployment diagram depicts the physical configuration of the software system deployed on hardware server nodes and the network between the nodes (defined as protocols). This diagram is generated in the later phase of the software development life cycle. All components in the system must be deployed on servers to provide services via network protocols. Component diagrams are the basis for deployment diagrams.

1. **Component Diagram:**

A component is neither a class nor an object. A component is a deployable, reusable building block used in software design and development. Each component has an interface to expose its services and hide its implementations. The interface is the contract between a reusable component and its clients.



**Fig1: Component diagram**



**Fig2: Deployment diagram**

1. **Development or Module View Model:**

The development view derives from the logical view and describes the static organization of the system modules. Modules such as namespaces, class library, subsystem, or packages are building blocks that group classes for further development and implementation. The software is packaged and partitioned into small units such as program libraries or subsystems created by many teams of developers. Each package has its own visibility and accessibility as package or default scope visibility.

The development view maps software component elements to actual physical directories and files. UML diagrams such as package diagrams and component diagrams are often used to support this view. The stakeholders of this view can be programmers and software project managers.

1. **Package Diagram:**

A package is represented by a tabbed folder that indicates all included classes and subpackages reside. Packages play a similar role as a directory for grouping files in a file system; they allow the organization of all closely related classes in one “container.”

For example, namespaces in .NET and packages in Java provide well-formed structures for class accessibility and class correlations. We can organize functionally related classes in the same package so that these classes can access each other within a default accessibility or visibility. We can also organize related packages in a same parent package to build a class and package hierarchy just like .NET class library and Java API. Another reason for using the package organization is namespace sharing is that all classes in the same package have a unique name but they may have the same name in different packages (namespaces).

A package diagram shows the dependency relationship between packages in which a change of one package may result in changes in other packages. The package diagram may also specify the contents of a package, i.e., the classes that constitute a package and their relationships. The use of package diagrams to represent system structures can help reduce the dependency complexity and simplify relationships among groups of classes.

****

**Fig3: Package Diagram**

**Exercise**

Implement the Physical View Model and Development or Module View Model diagrams:

Use UML to model the software architecture of an online trusted payment system. The system users are buyer, seller, payer, and security trustee. There are many e-payment selections available. The buyers may be consumers, corporations, or organizations. The sellers may be merchants, service providers, and others. The payers may be banks, financial services, credit card companies, etc. The trustee may be a security service, transaction auditing company, etc. The system supports payment selection, security services, transaction protection, and process flow management. The system administrator is also a special system user.