# Software Architecture (SEM VII)

Presented by: Ms. Drashti Shrimal

# Unit I

Topics:

- 4+1 view model

- Elements of SA

# 4+1 View Model

- The 4+1 View Model was designed by Philippe Kruchten to describe the architecture of a software–intensive system based on the use of multiple and concurrent views. It is a multiple view model that addresses different features and concerns of the system. It standardizes the software design documents and makes the design easy to understand by all stakeholders.
- It is an architecture verification method for studying and documenting software architecture design and covers all the aspects of software architecture for all stakeholders. It provides four essential views.

# 4+1 View Model

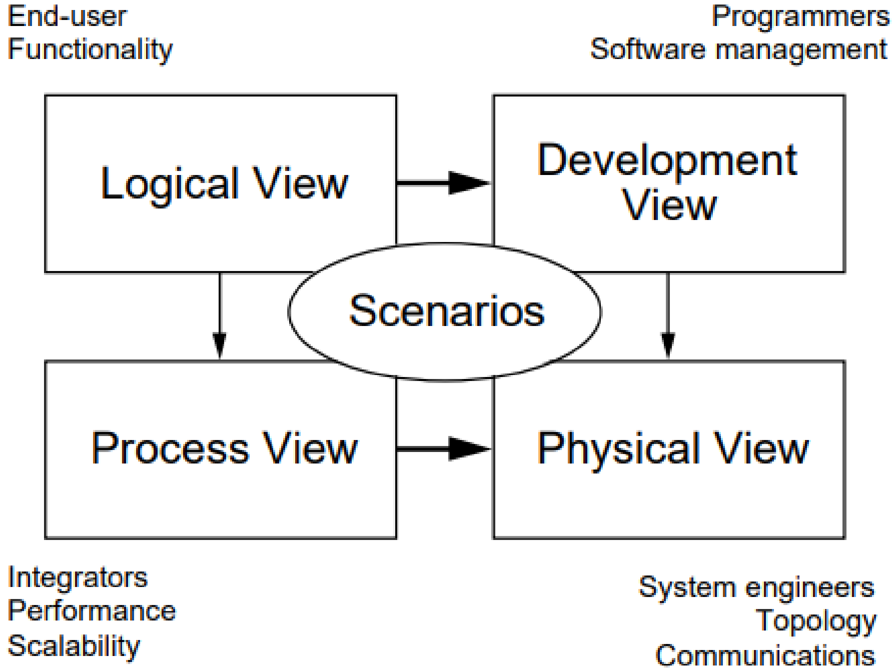


Figure 1 — The “4+1” view model

# 4+1 View Model

1. Logical View:

- The logical architecture primarily supports the functional requirements—what the system should provide in terms of services to its users.
- The system is decomposed into a set of key abstractions, taken (mostly) from the problem domain, in the form of objects or object classes. They exploit the principles of abstraction, encapsulation, and inheritance.
- Examples of Logical view models/diagrams: Sequence and Class diagrams.

# 4+1 View Model

2. Development View:

- The development architecture focuses on the actual software module organization on the software development environment.
- The software is packaged in small chunks—program libraries, or subsystems—that can be developed by one or a small number of developers.
- Provides a view from developers perspective which states where all code and its modules would be placed.
- Examples: Component and Package diagram.

# 4+1 View Model

3. Process View:

- The process architecture takes into account some non-functional requirements, such as performance and availability. It addresses issues of concurrency and distribution, of system’s integrity, of fault-tolerance.
- It provides a view from tasks’ perspective.
- Checks the scalability and performance of system.
- Example: Activity Diagram

# 4+1 View Model

4. Physical View:

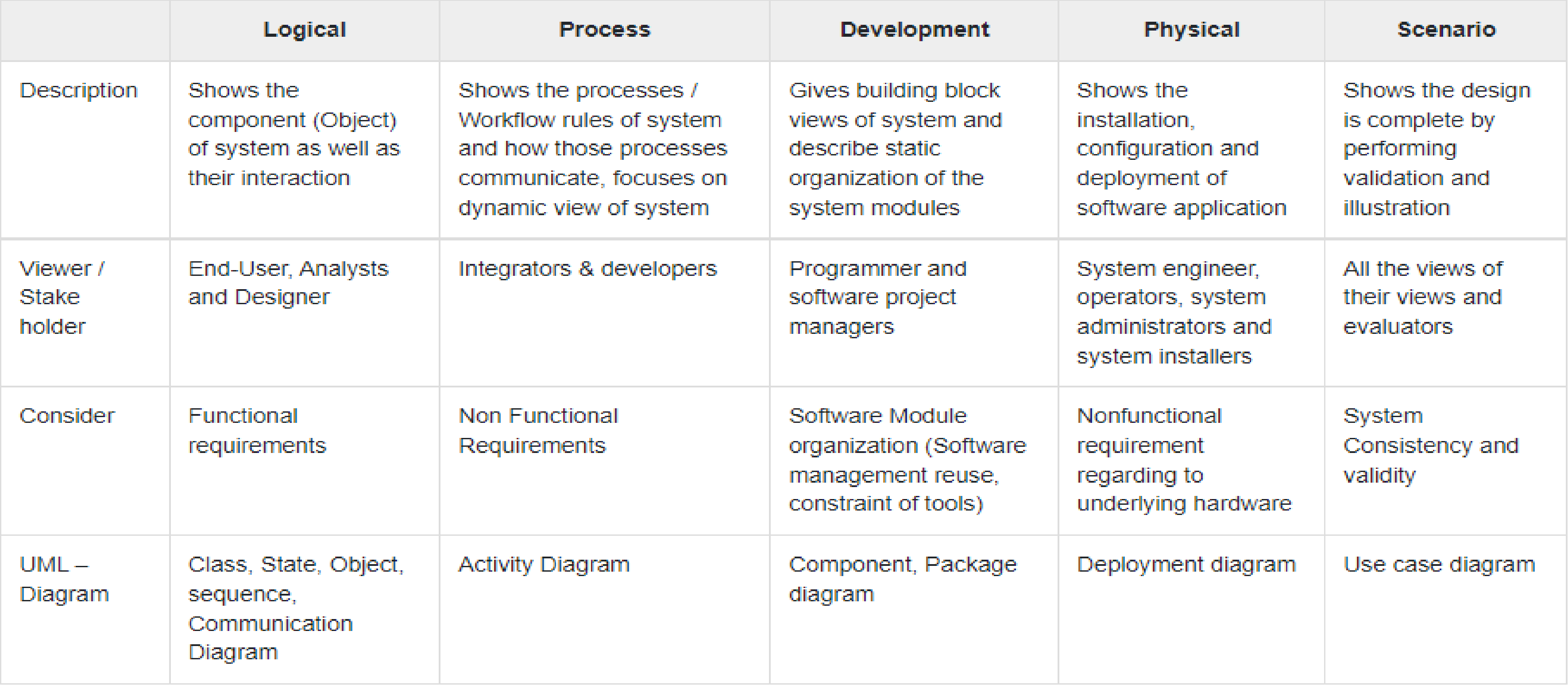
- It describes the mapping of software onto hardware and reflects its distributed aspect.
- It looks after the deployment of the system, tools and environment in which the product is installed.
- It takes a system engineer’s point of view in consideration.
- Example: Deployment diagram.

# 4+1 View Model

+1 Scenarios:

- This view model can be extended by adding one more view called scenario view or use case view for end-users or customers of software systems.
- It is coherent with other four views and are utilized to illustrate the architecture serving as “plus one” view, (4+1) view model.

# Comparison of Views



| | Logical | Process | Development | Physical | Scenario |
|---|---|---|---|---|---|
| Description | Shows the component (Object) of system as well as their interaction | Shows the processes / Workflow rules of system and how those processes communicate, focuses on dynamic view of system | Gives building block views of system and describe static organization of the system modules | Shows the installation, configuration and deployment of software application | Shows the design is complete by performing validation and illustration |
| Viewer / Stake holder | End-User, Analysts and Designer | Integrators & developers | Programmer and software project managers | System engineer, operators, system administrators and system installers | All the views of their views and evaluators |
| Consider | Functional requirements | Non Functional Requirements | Software Module organization (Software management reuse, constraint of tools) | Nonfunctional requirement regarding to underlying hardware | System Consistency and validity |
| UML – Diagram | Class, State, Object, sequence, Communication Diagram | Activity Diagram | Component, Package diagram | Deployment diagram | Use case diagram |

# Elements of SA

- **A software architecture is defined by a configuration of architectural elements--components, connectors, and configuration.**
- **The elements are:**

1. **Components**
2. **Connectors**
3. **Configuration**

# Elements of SA

1. Components:

- **“A Software component is an architectural entity that -**

  **1)encapsulates a subset of the system’s functionality**

  **2)restricts access to that subset via an explicitly defined interface**

  **3)has explicitly defined dependencies on itself”**

- A Component can be as simple as a single operation or as complex as an entire system.
- It can be “seen” by its end users completely (if the developer has made it public or it can be seen as a black box.
- It can be usable and reuseable, which is an important aspect.

# Elements of SA

1. Components: (Contd..)

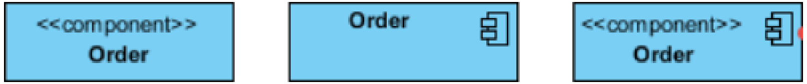
- One component can depend for its functionality on another component, and both can be linked by an ***interface***.
- The extent of the context captured by a component can include:

  - The interfaces it uses to link with other components

  - The availability of resources like the data files, directories

  - The required system software such as programming language run time environments, etc.

  - The hardware configurations.



# Elements of SA

2. Connector:

- **“ A software connector is an architectural element tasked with effecting and regulating interaction among components”.**
- In box and line diagrams, the boxes represent the Components and lines represent the connectors.
- Most widely used connector is the Procedure call. They are directly implemented in programming languages where they directly enable the synchronous exchange of data between components.

# Elements of SA

3. Configuration:

- **“An architectural configuration is a set of specific associations between the components and connectors of a software’s system architecture.”**
- A configuration may also be called as a graph where in nodes are the components and edges are the the connectors, the entire graph will be called as the configuration.

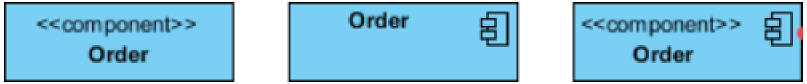
# Notations

## 1. Component:

1. A rectangle with the component's name

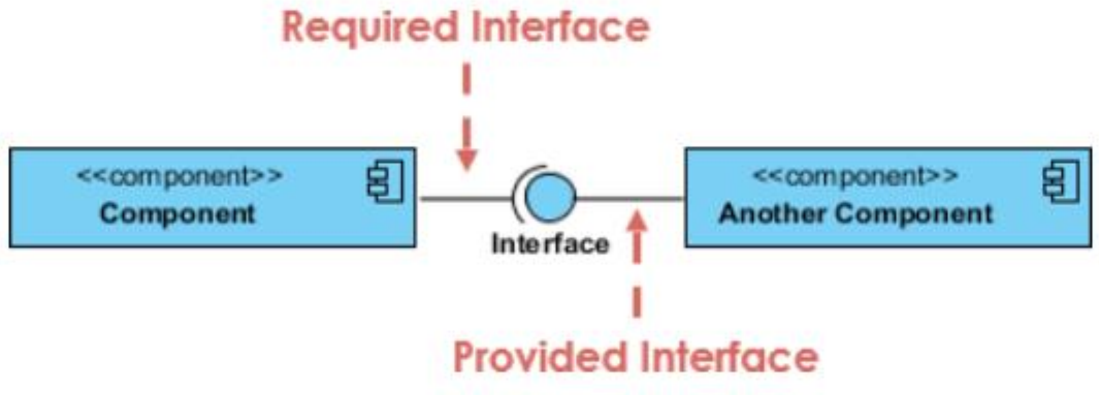
2. A rectangle with the component icon

3. A rectangle with the stereotype text and/or icon



# Notations

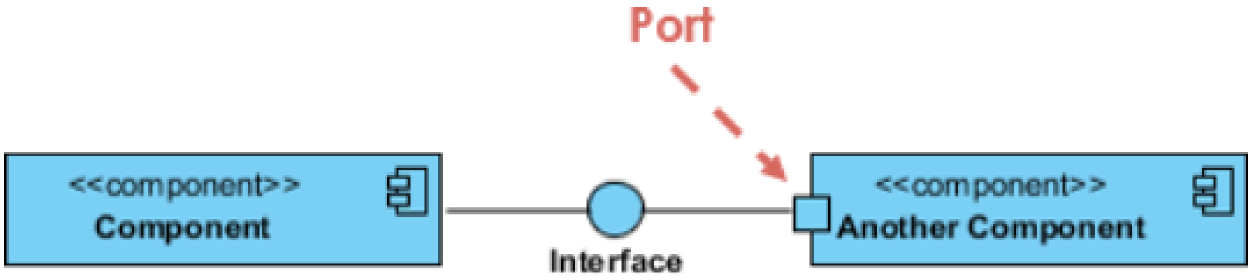
1. Interfaces:



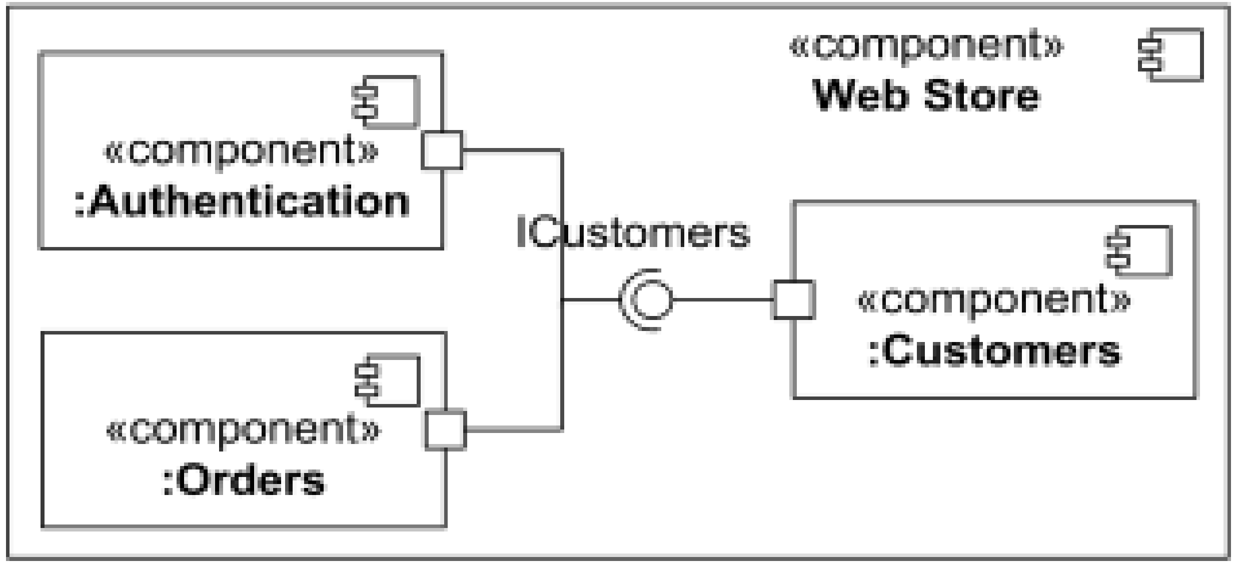
# Notations

## 3. Ports

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



# Example: Notations



# Lecture Takeaway

1. Define Software architecture and state needs.

2. Give advantages and disadvantages of Software Architecture.

3. Draw a Component Diagram for an Online Food Delivery App or Airline Management system. Assume necessary components for the entire flow of the application.

4. Define Software Architecture. Explain the need and significance of Software Architecture in detail.

5.  Build the use case diagram “ Railway Management System”.

6. Compare and contrast the SDLC Models.

7. List the various challenges in Software Architectural Design.

# THANK YOU