Lab 3: Component Modelling & Architectural Pattern Selection

Objective

Evaluate different architectural styles, select the most appropriate one for a given scenario, and create a comprehensive UML Component Diagram showing modules, interfaces, and dependencies. This lab focuses on understanding how components interact within a system and how architectural decisions impact system design.

Duration

90 minutes

Software Requirements

- UML tool (draw.io, LucidChart, StarUML or any online UML tool)

Learning Outcomes for Lab 3

By the end of this lab, you will be able to:

1. **Evaluate architectural styles:**
    - **Compare and contrast different architectural patterns (Layered, Microservices, Client-Server)**
    - **Analyse pros and cons of each pattern relative to specific scenarios**
2. **Select appropriate architecture:**
    - **Choose the most suitable architectural style based on system requirements**
    - **Justify architectural decisions with concrete reasoning**
3. **Design component diagrams:**
    - **Create UML component diagrams with proper notation**
    - **Show provided and required interfaces between components**
    - **Demonstrate understanding of component dependencies**
4. **Document architectural decisions:**
    - **Write clear, technical justifications for architectural choices**
    - **Explain security and performance implications**

Note: Students may be randomly called for a presentation after completing the lab. Please be prepared to present and discuss your lab deliverables.

Introduction:

Component modelling Fundamentals

What is a Component?

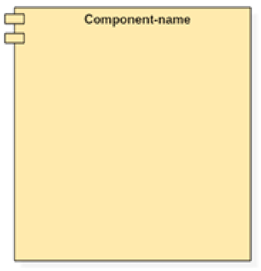
A component is a replaceable and executable piece of a system whose implementation details are hidden. Key characteristics:

- **Modular: Encapsulates specific functionality**
- **Replaceable: Can be swapped with other components that provide the same interface**
- **Black Box: External behavior is defined by interfaces, internal implementation is hidden**
- **Executable: Participates in system execution**

Component Diagram Elements

1. Component Notation

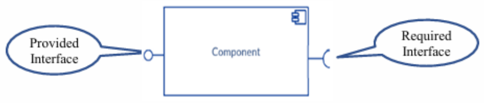
- **Represented as a rectangle with <<component>> stereotype**
- **Component name placed at center**
- **Optional component icon in upper-right corner**
- **Implementation details are hidden from external view**



2. Interfaces

Provided Interface (Ball notation):

- **Services that a component offers to other components**
- **Shown as a circle connected to the component**
- **Represents what the component can do for others**

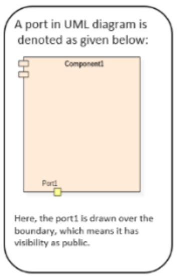


Required Interface (Socket notation):

- **Services that a component needs from other components**
- **Shown as a semicircle connected to the component**
- **Represents what the component needs from others**

3. Ports

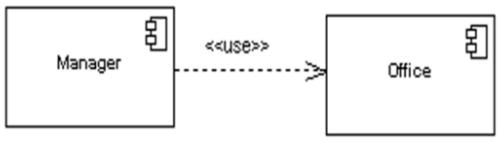
- **Interaction points between components and external environment**
- **Group related provided and required interfaces**
- **Can be public (drawn on boundary) or private (drawn inside)**


A port in UML diagram is denoted as given below:

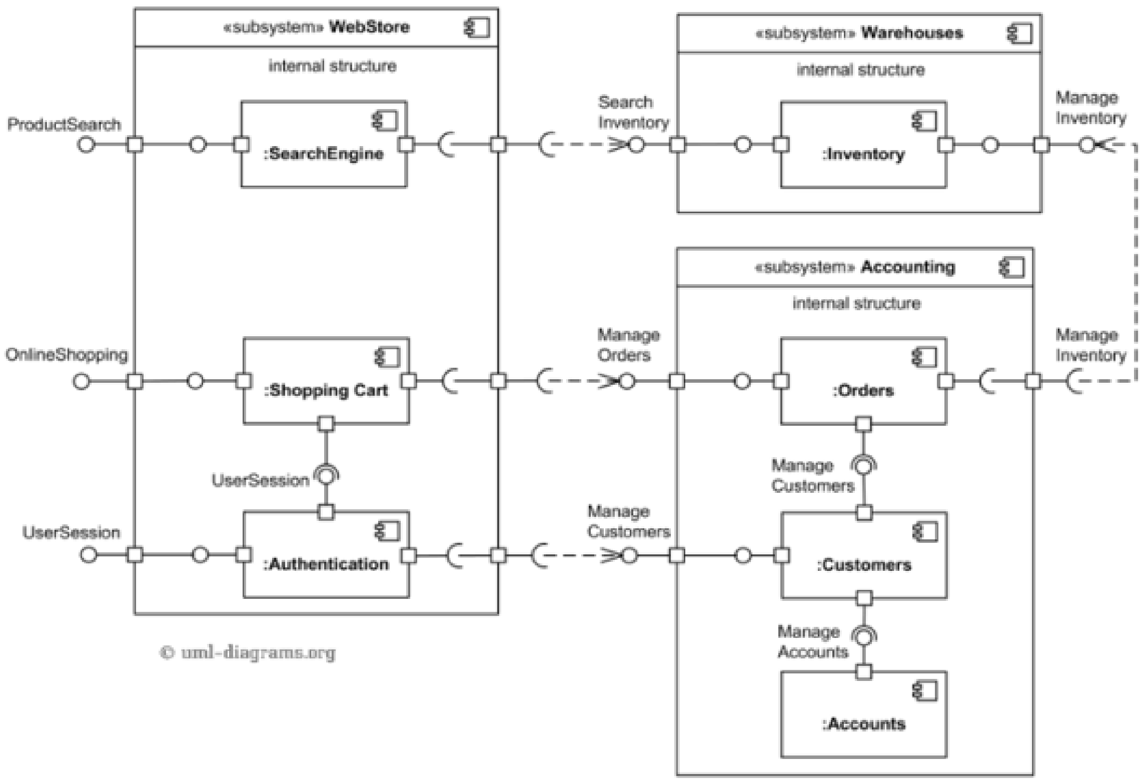
Here, the port1 is drawn over the boundary, which means it has visibility as public.

4. Dependencies

- **Usage Dependencies: Shown as dashed arrows with <<use>> stereotype**
- **Assembly Connectors: Connect provided interfaces to required interfaces**



Example Component diagram:



Online shopping UML component diagram example with three related subsystems - WebStore, Warehouses, and Accounting.

Architectural Styles Overview

1. Layered Architecture

- **Structure: Organized in horizontal layers (Presentation, Business, Data)**
- **Pros: Clear separation of concerns, easy to understand, good for traditional applications**
- **Cons: Performance overhead, tight coupling between layers, difficult to scale individual components**

2. Microservices Architecture

- **Structure: Small, independently deployable services**
- **Pros: Independent scaling, fault isolation, technology diversity, independent deployment**
- **Cons: Operational complexity, network latency, data consistency challenges**

3. Client-Server Architecture

- **Structure: Centralized server with multiple clients**
- **Pros: Centralized control, simple deployment, easy data consistency**
- **Cons: Single point of failure, scalability bottleneck, limited fault tolerance**

Scenario:

Self-Service Coffee Kiosk System

You are designing the software architecture for a Self-Service Coffee Kiosk system in a busy café. The kiosk must provide a complete coffee ordering experience with the following requirements:

Core Functionality:

- Customers can select from 3 coffee types (Espresso, Americano, Latte)
- Customers can choose from 2 drink sizes (Small, Large)
- Customers can pay via credit card only
- System prints receipts with order details

Technical Constraints:

- Must handle touch screen interface interactions
- Must connect to receipt printer hardware
- Must store menu data and pricing information

Deliverables

1. Component Diagram (PNG or PDF)

Your diagram must include atleast 5 components (Adding attributes is optional).

Interface Requirements:

- Show atleast 4 interfaces between components:
- Use proper UML notation for provided/required interfaces
- Show data flow and component interactions

2. Written Justification (Word document, max 1 page)

Lab 3 Steps (90 minutes)

Step 1: Scenario Review (10 minutes)

- Read the scenario thoroughly
- Identify key functional and non-functional requirements
- List the main challenges (performance, usability, security)

Step 2: Architectural Style Analysis (20 minutes)

Analysis (10 minutes): Analyse each architectural style.

Step 3: Architecture Selection & Component Identification (15 minutes)

Select Architecture (5 minutes)

Identify Components (10 minutes): Based on your chosen architecture, identify all necessary components *(Two components are given below, please identify the remaining 3):*

1. Order Manager Component
2. Payment Service Component

Step 4: Component Diagram Creation (35 minutes)

Setup (5 minutes):

- Open your UML tool (draw.io recommended)

- Set up the canvas with adequate space

Draw Components (15 minutes):

- Create component rectangles for each of the 5 identified components
- Use <<component>> stereotype or clear component notation
- Arrange components logically on the canvas
- Group related components (e.g., customer-facing, data storage)

Add Interfaces (15 minutes): Define and draw the required interfaces between components:

Key Interfaces to Show (Min 4):

Given:

- Order Manager ↔ Payment Service: Payment processing requests

Interface Notation Guidelines:

- Use solid lines for internal service communications
- Label interfaces with protocol/technology (API calls, hardware interfaces, database queries)
- Use provided interface (ball) and required interface (socket) notation where appropriate

Create a one-page Word document with the following structure:

Architecture Selection: "We chose [Architecture] for the Self-Service Coffee Kiosk System."

Your justification must include:

- Architectural Choice: State which architectural style you selected
- Two Reasons: Provide two specific, scenario-related reasons for your choice
- Security Advantage: Explain one way your architecture addresses security concerns
- Performance Benefit: Describe one performance advantage of your chosen architecture

Step 6: Diagram Finalization & Export (5 minutes)

- Review diagram for completeness
- Ensure all required components are present
- Verify interface directions and labels are correct
- Check UML notation compliance
- Export as PNG or PDF
- Save justification document as PDF