**Week 9**

**System Maintenance**

**System maintenance** is a critical aspect of the software development life cycle (SDLC)

that ensures the continued functionality, performance, and security of a software system after it

has been deployed. Effective maintenance helps to address issues, implement enhancements,

and adapt to changing user needs or technological advancements.

**Types of Maintenance**

**1. Corrective Maintenance**:

* This type of maintenance involves fixing defects or bugs that are

identified after the system has been deployed. It addresses issues that affect

the system's functionality or performance.

**2. Adaptive Maintenance**:

* Adaptive maintenance involves modifying the system to

accommodate changes in the environment, such as updates to operating

systems, hardware, or other software that the system interacts with.

3. **Perfective Maintenance**:

* This type of maintenance focuses on improving the system's

performance or enhancing its features based on user feedback or changing

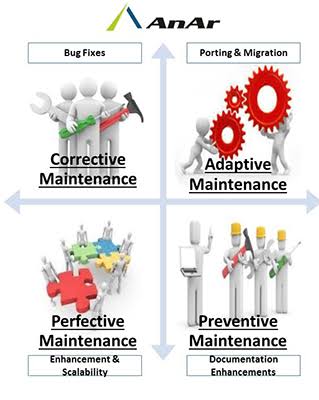
requirements.

4. **Preventive Maintenance**:

* Preventive maintenance involves proactive measures taken to

prevent potential issues before they occur. This type of maintenance aims to

reduce the likelihood of system failures and extend the system's lifespan.



.

**Approaches to Organizing System Maintenance**

Organizing system maintenance effectively is crucial for ensuring that software systems

remain functional, secure, and aligned with user needs. Various approaches can be

adopted to structure maintenance activities, each with its own advantages and

challenges. Below are some common approaches to organizing system maintenance:

**1**. **In-House Maintenance Teams**

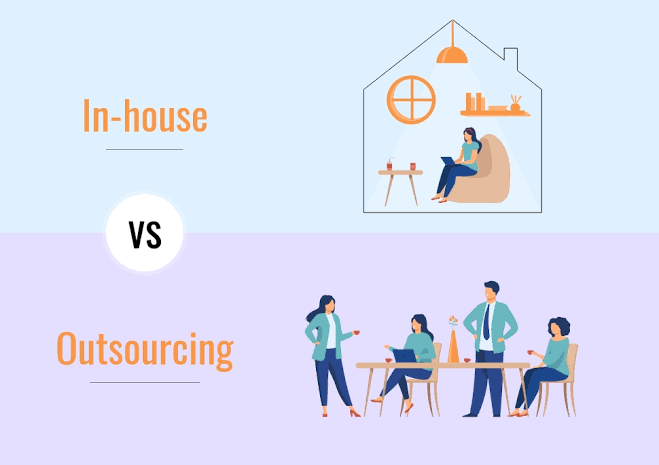
• Description: Organizations maintain a dedicated team of IT professionals

responsible for all maintenance activities.

**2. Outsourced Maintenance**

• Description: Organizations contract third-party vendors to handle

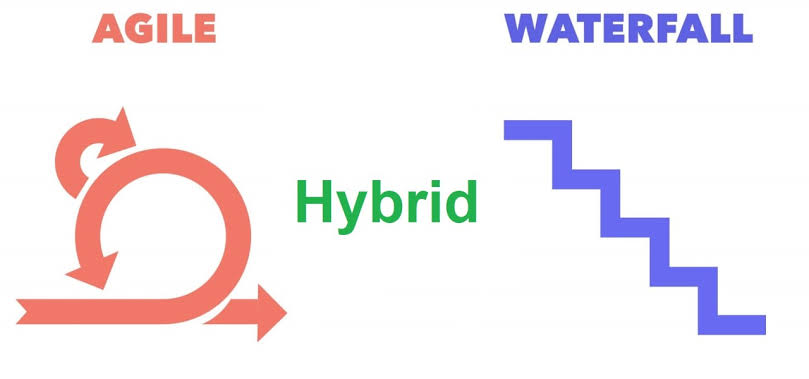
maintenance tasks.



**3**. **Hybrid Approach**

* A combination of in-house and outsourced maintenance, where

certain tasks are handled internally while others are outsourced.



**4. Agile Maintenance Teams**

* Maintenance is organized using agile methodologies, focusing

on iterative improvements and responsiveness to user feedback.



**5. Centralized vs. Decentralized Maintenance**

• **Centralized Maintenance:**

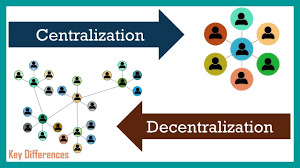
* All maintenance activities are managed by a central

team or department.

• **Decentralized Maintenance:**

* Maintenance responsibilities are distributed across

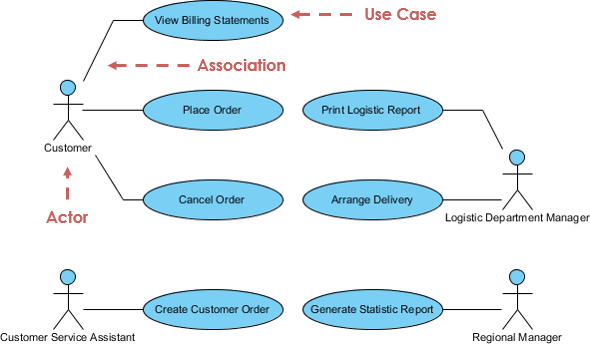
various departments or teams.



**Week 10**

**The Unified Modeling Language (UML)** is a standardized modeling language

used in software engineering to specify, visualize, develop, and document the artifacts

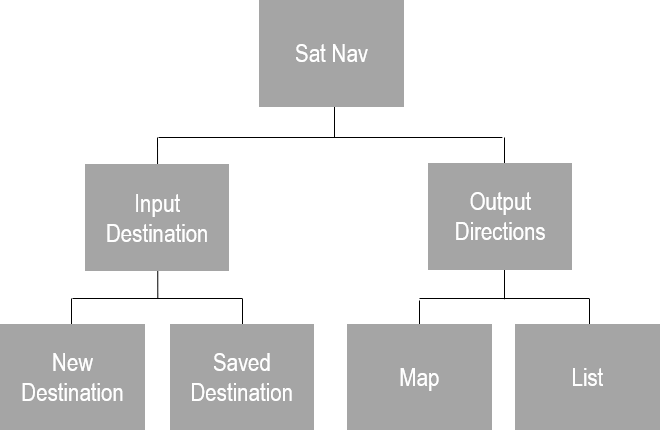


**Types of UML Diagrams**

**1. Structural Diagrams**

These diagrams represent the static aspects of a system, focusing on the

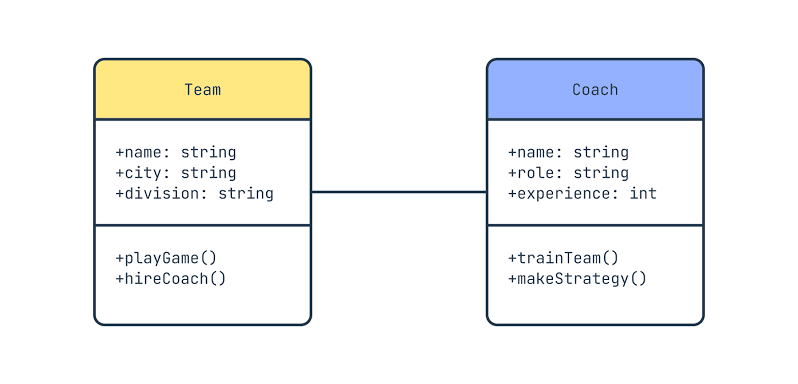
organization and structure of the system's components.



• **Class Diagram:**

* Shows the classes in a system, their attributes, methods, and

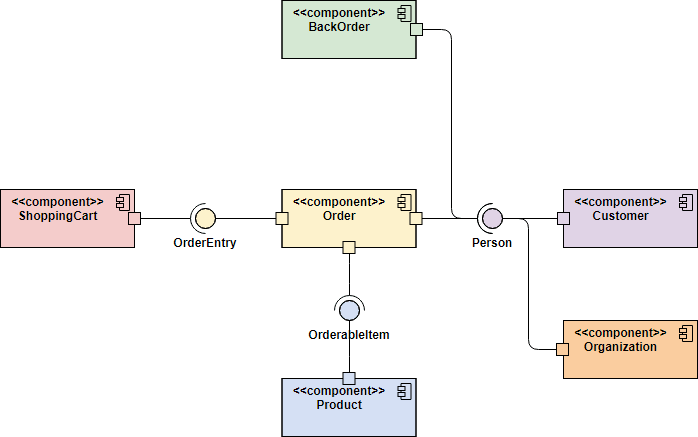
relationships (associations, inheritance, etc.).



• **Component Diagram:**

* Illustrates the components of a system and their

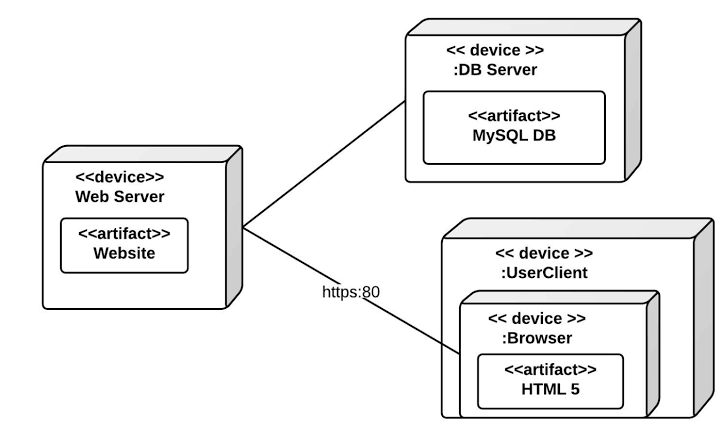
dependencies, focusing on the physical aspects of the system.



• **Deployment Diagram:**

* Represents the physical deployment of artifacts on nodes,

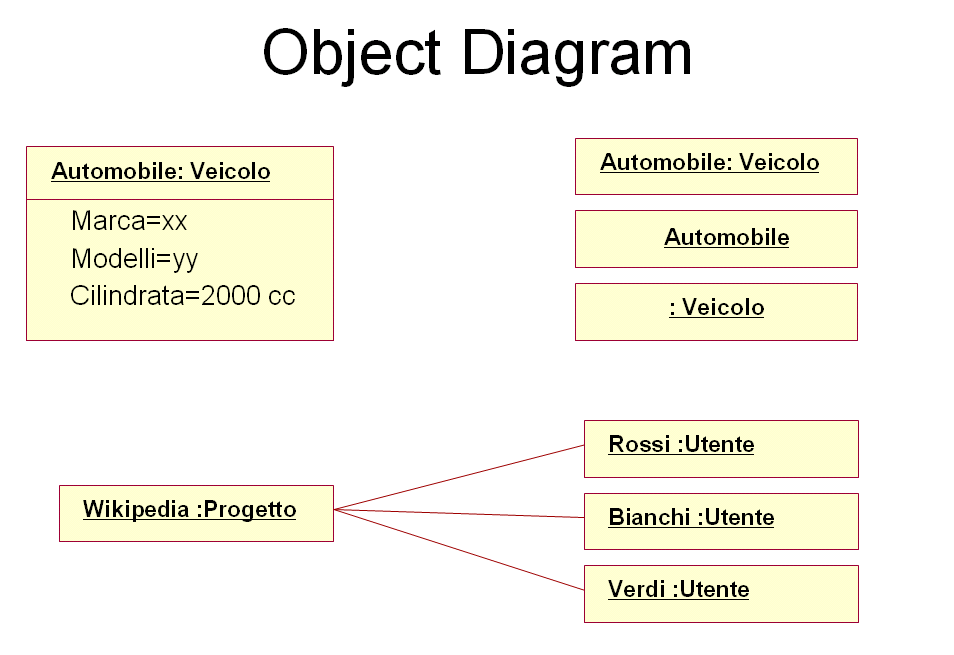
showing how software components are distributed across hardware.



• **Object Diagram:**

* Displays instances of classes (objects) and their relationships at

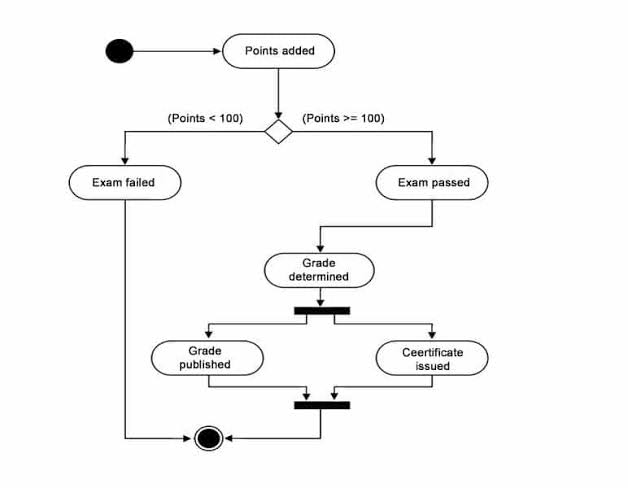
a specific point in time.



**2**. **Behavioral Diagrams**

These diagrams represent the dynamic aspects of a system, focusing on how the

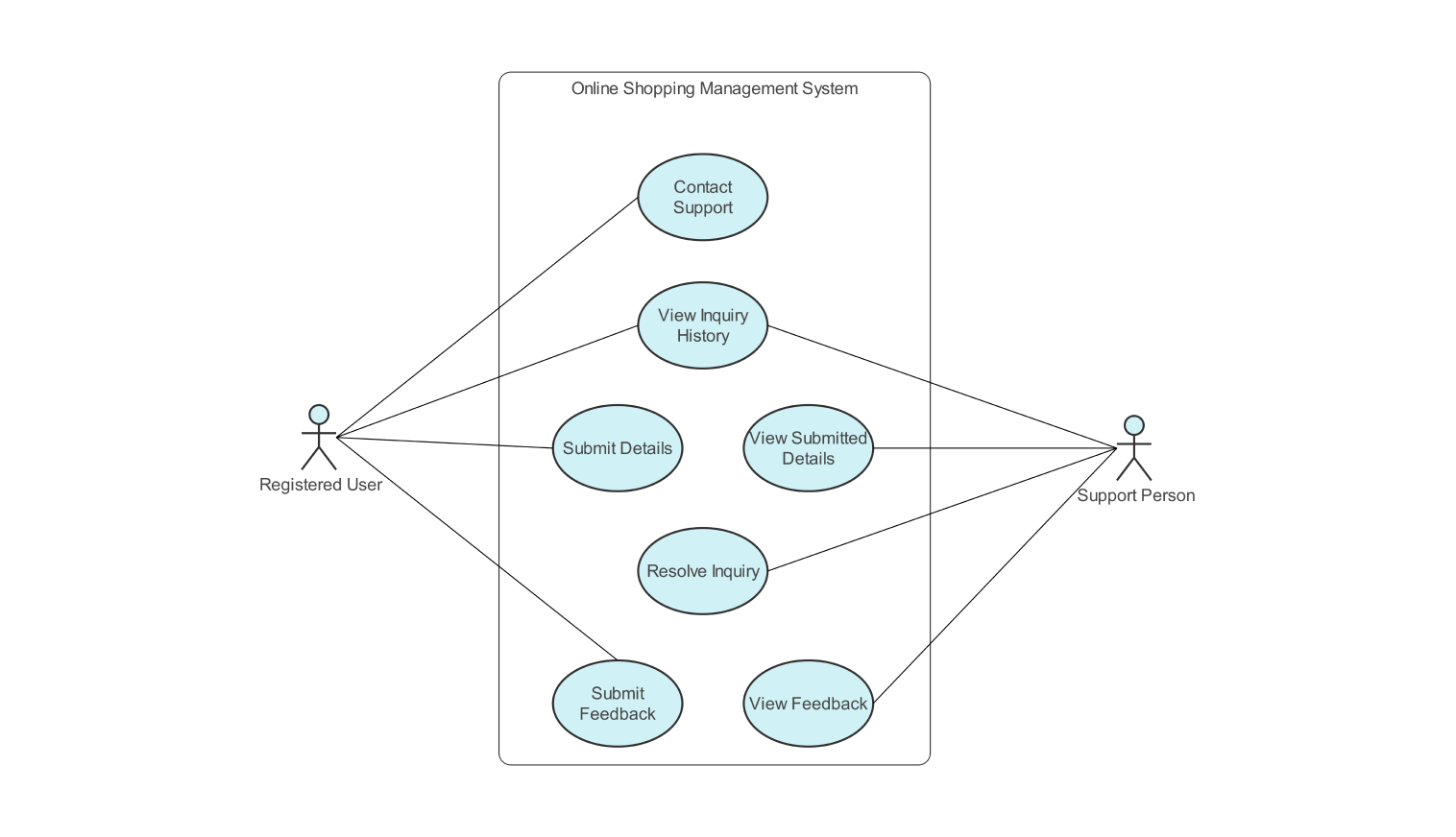
system behaves and interacts over time.



• **Use Case Diagram:**

* Captures the functional requirements of a system by showing

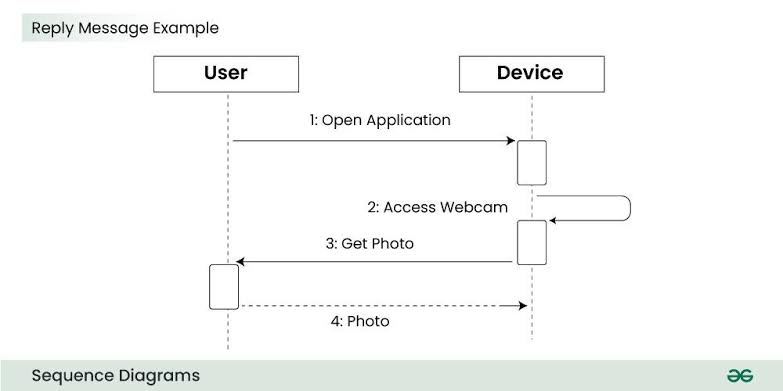
the interactions between users (actors) and the system's use cases.



• **Sequence Diagram:**

* Illustrates how objects interact in a particular scenario of a

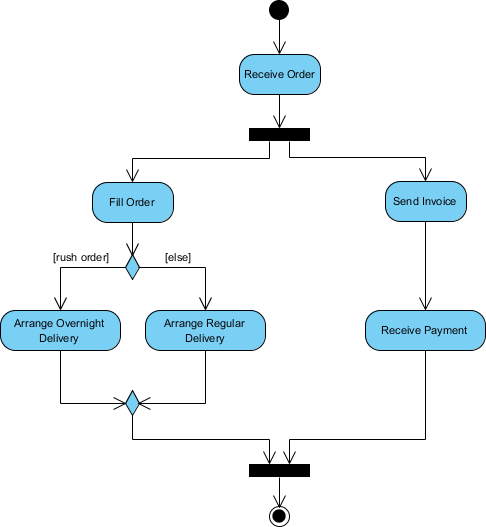
use case, showing the sequence of messages exchanged.



• **Activity Diagram:**

* Represents the flow of control or data in a system, showing the

sequence of activities and decisions.



• **State Diagram:**

* Describes the states of an object and the transitions between

those states in response to events.

