UML Diagrams: Concepts, Theory, and Graphical Models

1 Introduction

Unified Modeling Language (UML) is a standardized modeling language used to visualize, specify, construct, and document software systems. UML diagrams are divided into structural and behavioral diagrams. This document explains six important UML diagrams along with example models for each.

2 Use Case Diagram

A Use Case Diagram represents system functionality from the user's perspective. It focuses on what the system does rather than how it is implemented.

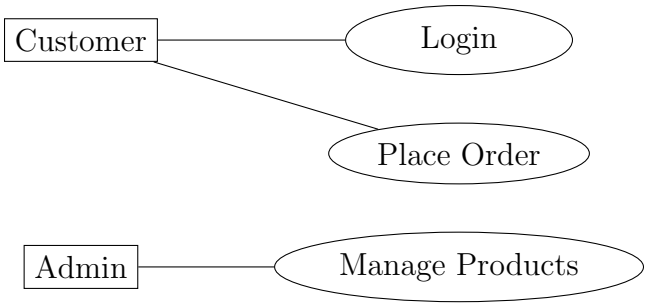
Key Elements

- Actor
- Use Case
- System Boundary
- Relationships (Include, Extend)

Example Model: Online Shopping System

- Actors: Customer, Admin
- Use Cases: Login, Browse Products, Place Order, Make Payment, Manage Products

Use Case Diagram (Graphical)



3 Class Diagram

A Class Diagram describes the static structure of a system by showing classes, attributes, methods, and relationships.

Key Elements

- Class name
- Attributes
- Methods
- Relationships

Class Diagram (Graphical)

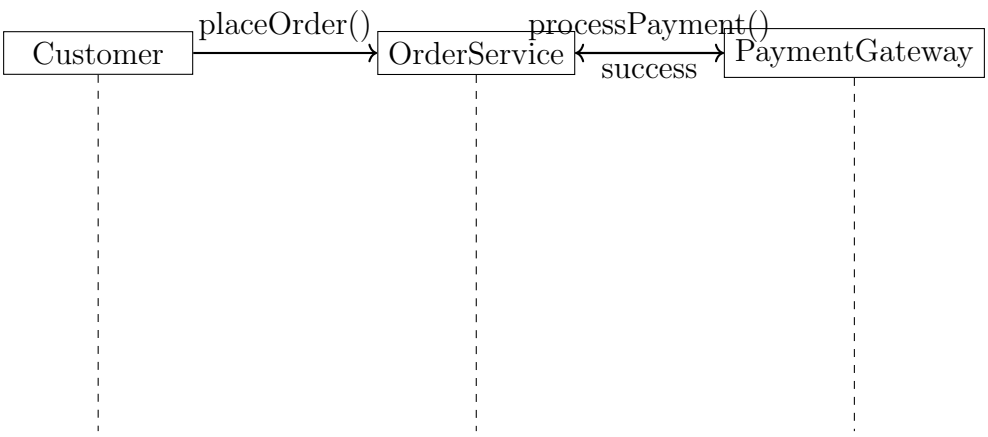
name : string
email : string
---
login()
logout() ;
[class, right=3cm of user] (order) **Order**
---
orderId : int
date : Date
---
placeOrder()
cancelOrder() ;

[-¿] (user) – node[above]places (order);

4 Sequence Diagram

A Sequence Diagram illustrates how objects interact with each other in a time sequence.

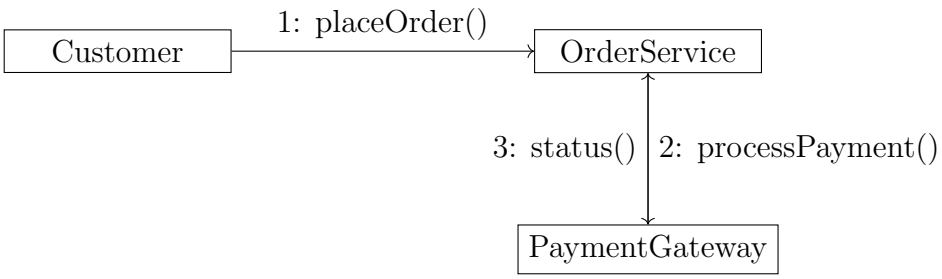
Sequence Diagram (Graphical)



5 Collaboration Diagram

A Collaboration Diagram focuses on how objects interact and are connected rather than the time sequence.

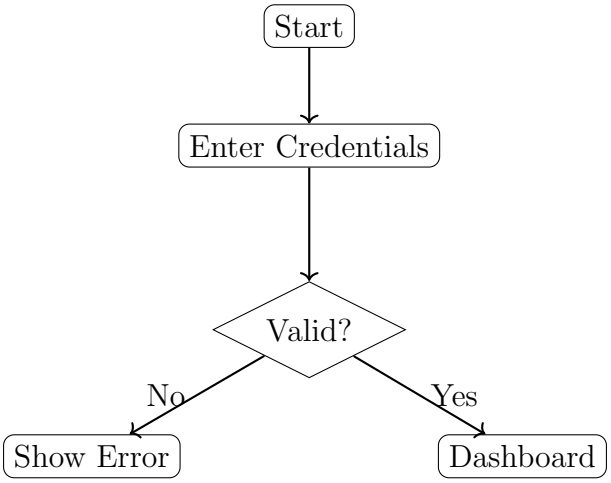
Collaboration Diagram (Graphical)



6 Activity Diagram

An Activity Diagram represents the workflow or business process of a system.

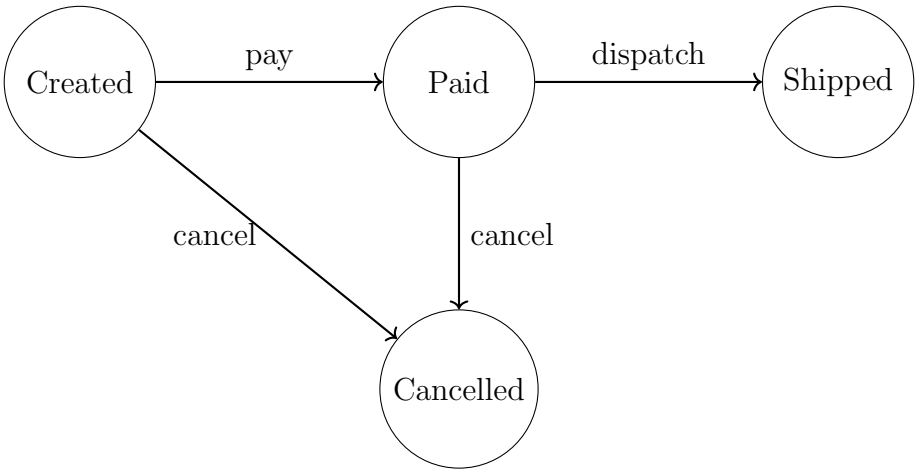
Activity Diagram (Graphical)



7 State Machine Diagram

A State Machine Diagram models the lifecycle of an object.

State Machine Diagram (Graphical)



8 Conclusion

Each UML diagram serves a specific purpose in modeling software systems. Including both theoretical explanations and graphical representations improves clarity, understanding, and communication in software engineering.