

LECTURE 6

Software requirements Engineering

System Modelling for Requirements Engineering

Instructor:

Dr. Natalia Chaudhry,

Assistant Professor, PUCIT, University of the Punjab, Lahore.

Representations for Requirements Engineering


Data flow diagrams (DFDs) are the basis of most traditional modelling method

External Entities: These are the sources and destinations of data outside the system you're modeling. They could be users, other systems, or data stores.

Processes: These are the functions or activities within the system that transform the data flowing through it.

Data Stores: These are repositories where data is stored within the system.

When using the notation, diagrams must be supported by textual descriptions of each process, data store and flow.



Context Diagram: It provides an overview of the entire system, showing the external entities and the interactions between them. It's a high-level diagram that helps you understand the boundaries of the system.

Level 0 DFD: This diagram shows the main processes within the system and how they interact with each other. It provides a more detailed view compared to the context diagram.

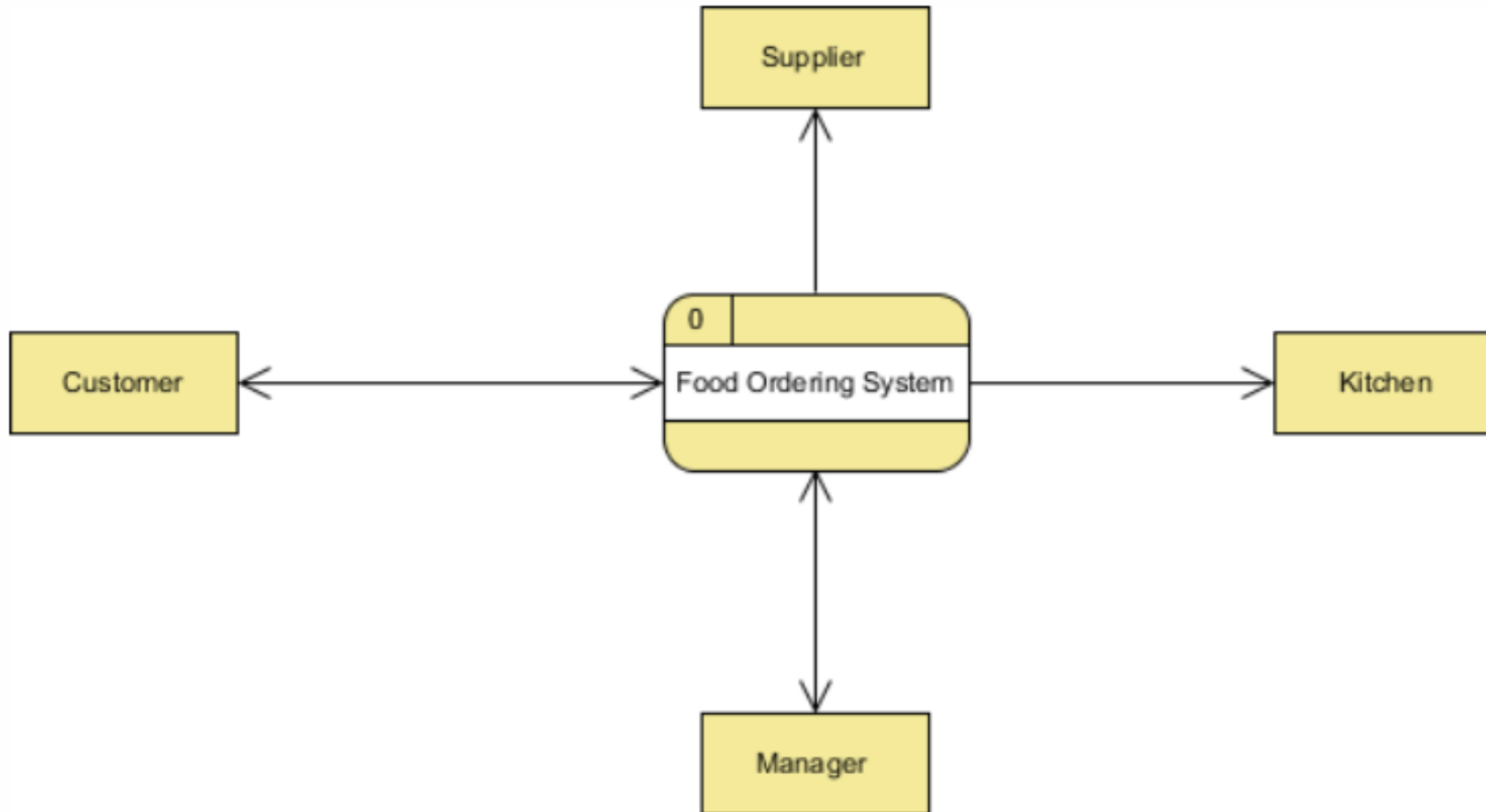
Context to level 0

- Break down the main processes identified in the Level 0 DFD into more detailed subprocesses

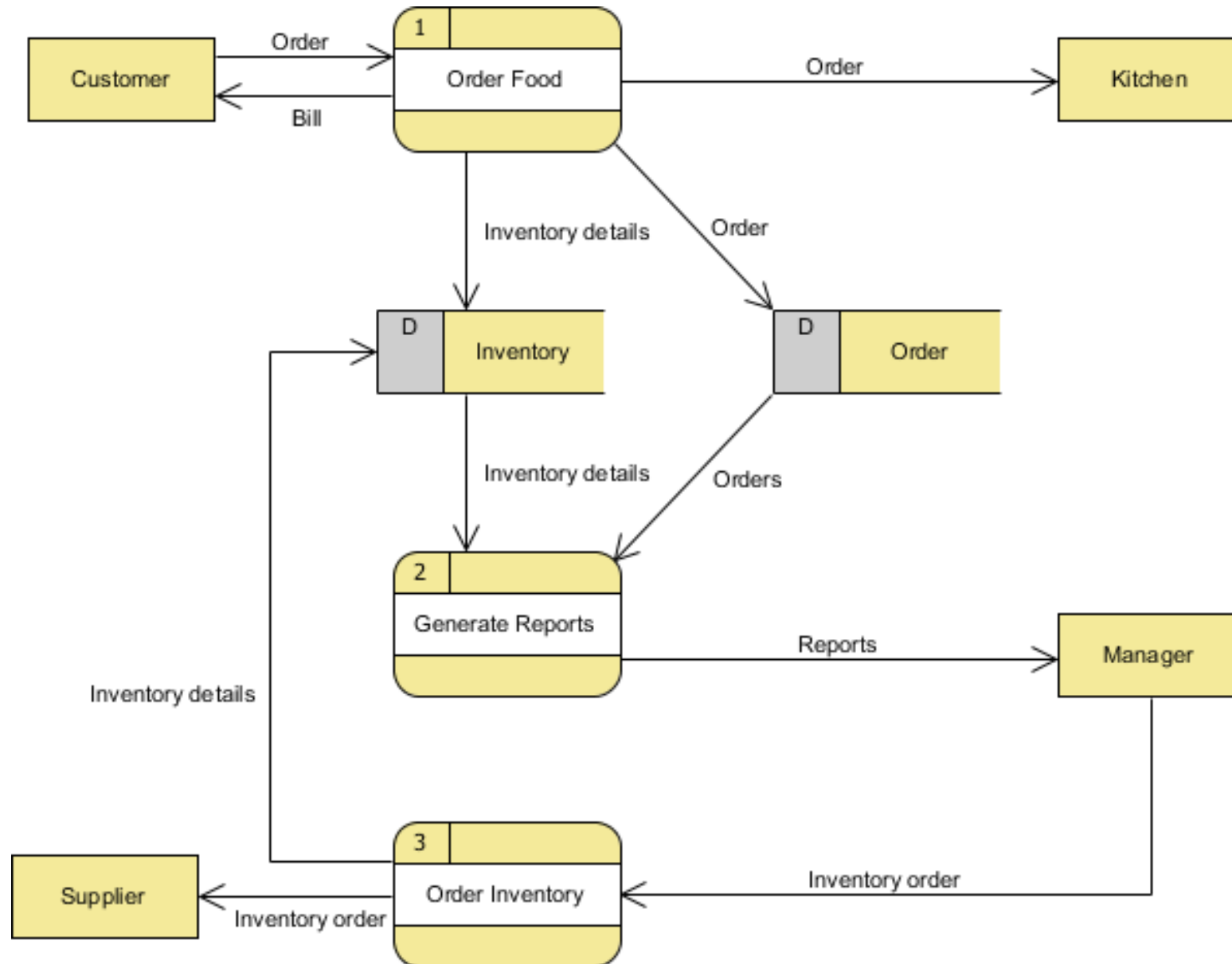
Add Data Stores and Data Flows:

- **Identify Data Stores:** Determine where data is stored within the system and represent them on the diagram.
- **Define Data Flows:** Show how data moves between processes and data stores. Use clear labels to describe the data being transferred.

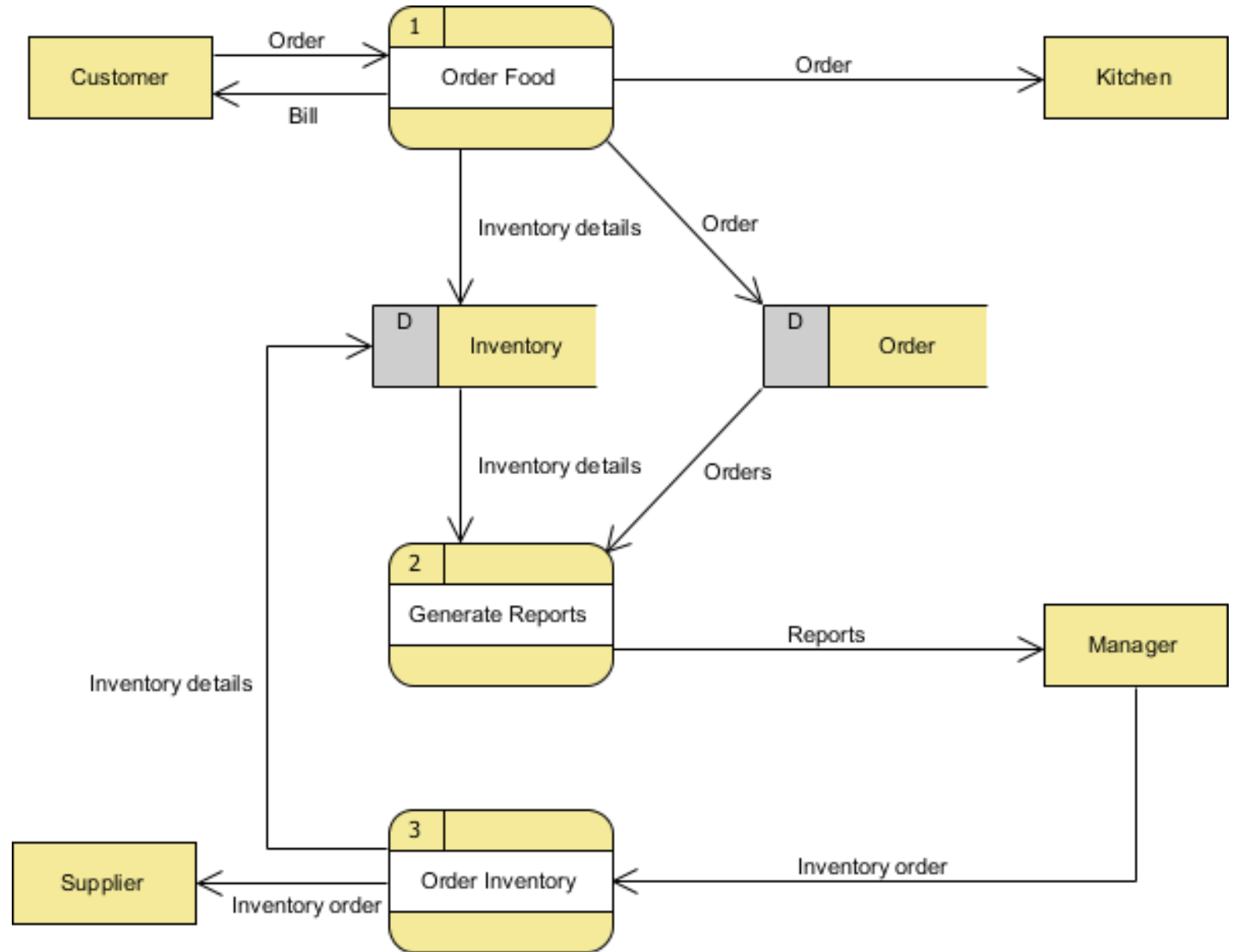
Context DFD is the entrance of a data flow model. It contains one and only one process and does not show any data store.



Level 1



The Food Order System Data Flow Diagram example contains three processes, four external entities, and two data stores.



To Do

Imagine you are tasked with designing a Data Flow Diagram (DFD) for a university registration system. The system allows students to register for courses, drop courses, and view their academic records. The system also allows faculty members to access and update course information and student grades. Design a DFD for this university registration system.



Identify External Entities:

Identify Processes:

Identify Data Stores:

Identify Data Flows:

Connect the elements:

Ensure clarity and correctness:

Identify External Entities:

Students

Faculty Members

Administrative Staff

Identify Processes:

Course Registration Process

Course Management Process

Grade Processing Process

Identify Data Stores:

Student Records (for storing student information)

Course Information (for storing course details)

Grade Records (for storing student grades)

Identify Data Flows:

Students submit registration requests, which flow into the Course Registration Process. Course information is accessed by both the Course Registration Process and the Course Management Process.

Faculty members submit grade updates, which flow into the Grade Processing Process. Student records and course information are accessed by administrative staff for various purposes.

Connect the elements:

Draw arrows to represent the flow of data between external entities, processes, and data stores.

Label each arrow with the data it represents (e.g., "Student Registration Request").

Ensure clarity and correctness:

Make sure all elements are clearly labeled and connected appropriately.

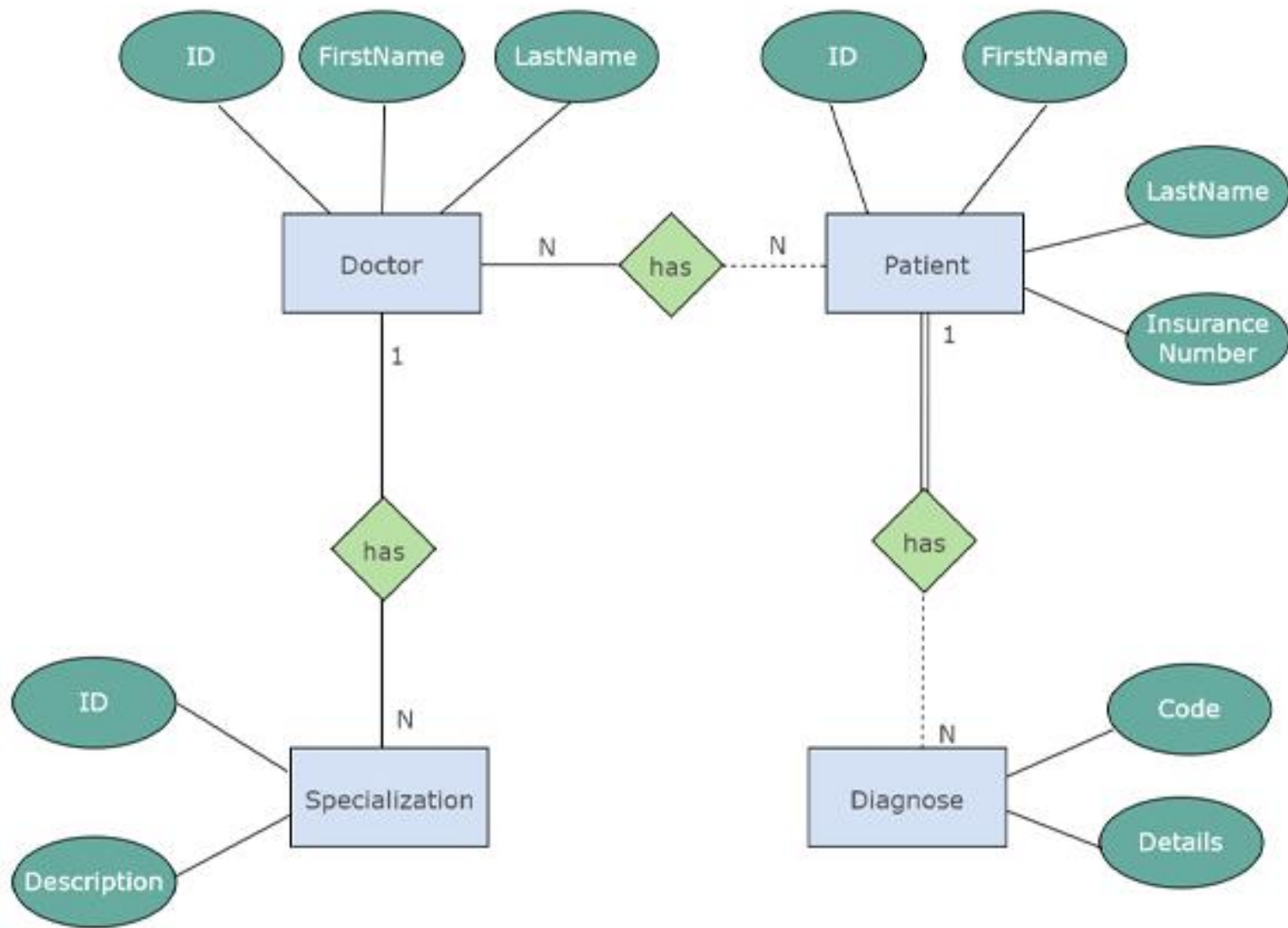
Review the diagram to ensure it accurately reflects the interactions within the system.



Entity–relationship diagrams (ERDs) provide a means of modelling the entities of interest and the relationships that exist between them

An entity is an object that can be distinctly identified, such as customer, supplier, part, or product. A property (or attribute) is information that describes the entity. A relationship has cardinality, which expresses the nature of the association (one-to-one, one-to-many, many-to-many) between entities

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.



To Do

Scenario: Online Bookstore

Imagine you're tasked with designing the database schema for an online bookstore. The bookstore sells books to customers through its website.

Give a brief description of the entities and their relationships:



Entities:

1.Book: Represents a book available for sale in the online bookstore. Each book has a unique ISBN (International Standard Book Number), title, author(s), genre, price, and publication year.

2.Customer: Represents a customer who visits the online bookstore and makes purchases. Each customer has a unique customer ID, name, email address, and shipping address.

3.Order: Represents an order placed by a customer for one or more books. Each order has a unique order ID, order date, total amount, and status (e.g., pending, shipped, delivered).

4.Order Item: Represents a specific book purchased as part of an order. Each order item is associated with an order and a book and includes the quantity purchased and the unit price at the time of purchase.



Relationships:

1.Book-Customer Relationship: Customers can purchase multiple books, and each book can be purchased by multiple customers.

2.Order-Customer Relationship: Each order is placed by a single customer, but a customer can place multiple orders.

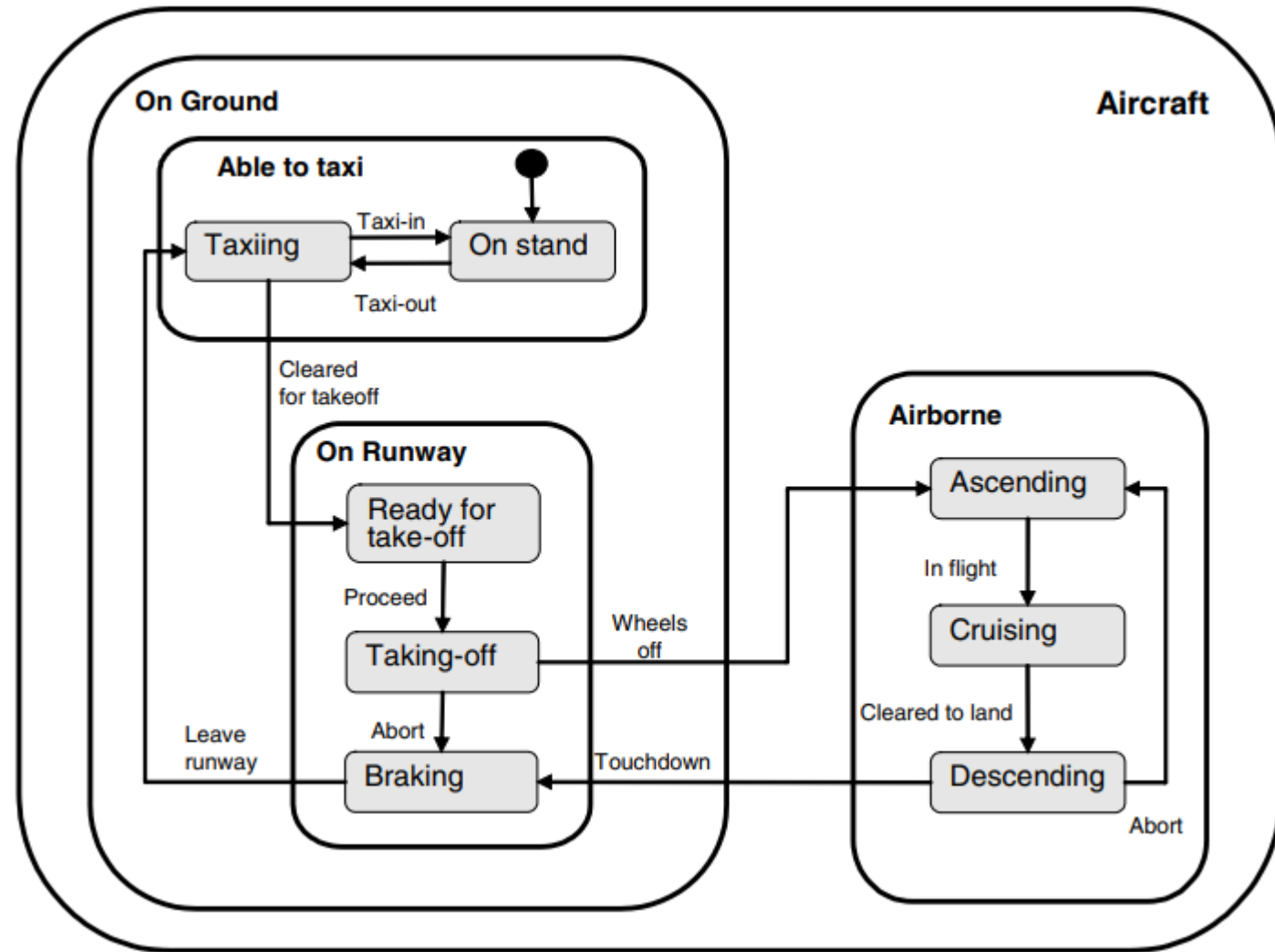
3.Order-Order Item Relationship: Each order can have multiple order items, representing the books included in that order. An order item belongs to exactly one order.

4.Order Item-Book Relationship: Each order item is associated with a single book, representing the book purchased as part of the order. A book can be included in multiple order items across different orders.

Statecharts

- Functionality and data flows are not enough for requirements definition.
- It is also necessary to be able to represent the behaviour of the system and in some circumstances consider the system as having a finite number of possible “states”, with external events acting as triggers that lead to transitions between the states.
- Statecharts are concerned with providing a behavioural description of a system.

- A labelled box with rounded corners denotes a state.
- Hierarchy is represented by encapsulation and directed arcs, labelled with a description of the event, are used to denote a transition between states.



To Do

Scenario: Online Order Tracking System

Imagine you're tasked with designing the state chart diagram for an online order tracking system used by a logistics company. The system allows customers to track the status of their orders from the time they are placed until they are delivered.

Give a brief description of the states, events and transitions:



States:

1.Order Received: This is the initial state when an order is placed by a customer. The system has received the order, but it has not yet been processed.

2.Order Processed: The order has been processed by the logistics company. This state indicates that the order details have been verified, and the necessary preparations for shipment are being made.

3.Order Shipped: The order has been shipped from the warehouse or fulfillment center. The package is in transit and on its way to the customer's shipping address.

4.Out for Delivery: The order has arrived at the local distribution center and is out for delivery by the courier or delivery driver.

5.Order Delivered: The order has been successfully delivered to the customer's shipping address.



Transitions:

1.Place Order: Transition from "Order Received" to "Order Processed" when the customer successfully places an order on the website.

2.Process Order: Transition from "Order Processed" to "Order Shipped" when the logistics company completes the processing of the order and prepares it for shipment.

3.Ship Order: Transition from "Order Shipped" to "Out for Delivery" when the order is dispatched from the warehouse for delivery to the customer.

4.Deliver Order: Transition from "Out for Delivery" to "Order Delivered" when the order is successfully delivered to the customer's shipping address.



Events:

1.Order Placed: Triggers the transition from "Order Received" to "Order Processed" when a customer places an order.

2.Order Processed: Triggers the transition from "Order Processed" to "Order Shipped" when the order is processed by the logistics company.

3.Order Shipped: Triggers the transition from "Order Shipped" to "Out for Delivery" when the order is dispatched for delivery.

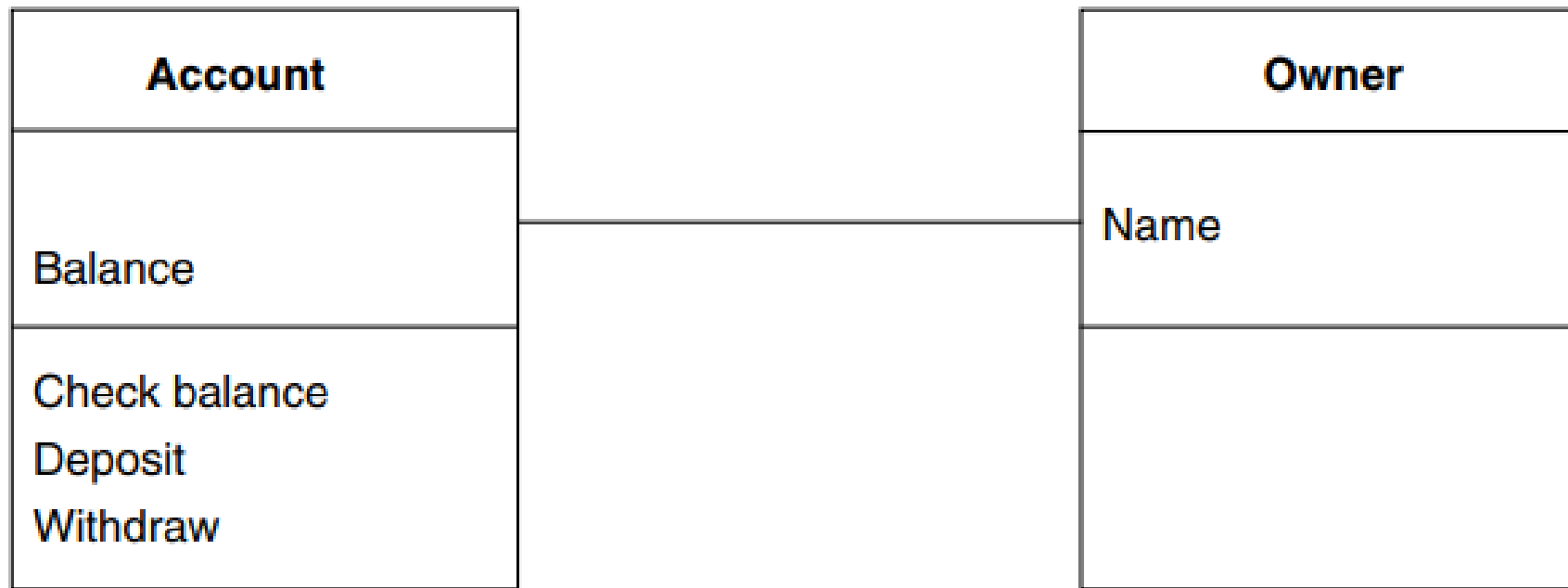
4.Order Delivered: Triggers the transition from "Out for Delivery" to "Order Delivered" when the order is successfully delivered to the customer.

Object-oriented Approaches

- Objects describe stable & re-usable components
- Object orientation focuses on the behaviour of objects and their interrelationships.
- goals of object orientation are to:
 - encapsulate behaviour (states and events), information (data) and actions within the same objects;
 - try to define persistent objects, which can be used within both requirements and design phases;
 - add information by defining the objects in more detail;
 - create new objects by specialization of existing objects, not creation of new objects.

Class Diagrams

- Class diagrams express information about classes of objects and their relationships. In many ways, they are similar to entity–relationship diagrams.



To Do

Scenario: Social Media Analytics Platform

Imagine you're designing a social media analytics platform that collects data from various social media platforms (such as Facebook, Twitter, and Instagram) and provides analytics and insights to businesses and marketers.

Give description of entities, relationships, operations, attributes



Entities:

1.User: Represents a user of the social media analytics platform. Each user has a unique user ID, username, email address, and password.

2.Social Media Account: Represents a social media account linked to the platform for data collection and analysis. Each account has a unique account ID, platform (e.g., Facebook, Twitter), username, and authentication token.

3.Analytics Dashboard: Represents a dashboard provided to users for viewing analytics and insights. Each dashboard has a unique dashboard ID and may contain multiple widgets for displaying different types of analytics data.

4.Analytics Widget: Represents a specific widget within an analytics dashboard, such as a chart, graph, or table. Each widget has a unique widget ID and is associated with a specific type of analytics data (e.g., engagement metrics, audience demographics).



Relationships:

1.User-Social Media Account Relationship: Each user can link multiple social media accounts to their profile for data collection and analysis. Each social media account is linked to exactly one user.

2.User-Analytics Dashboard Relationship: Each user can create multiple analytics dashboards for viewing different sets of analytics data. Each dashboard is associated with exactly one user.

3.Dashboard-Analytics Widget Relationship: Each analytics dashboard can contain multiple analytics widgets for displaying different types of data. Each widget belongs to exactly one dashboard.

Attributes:

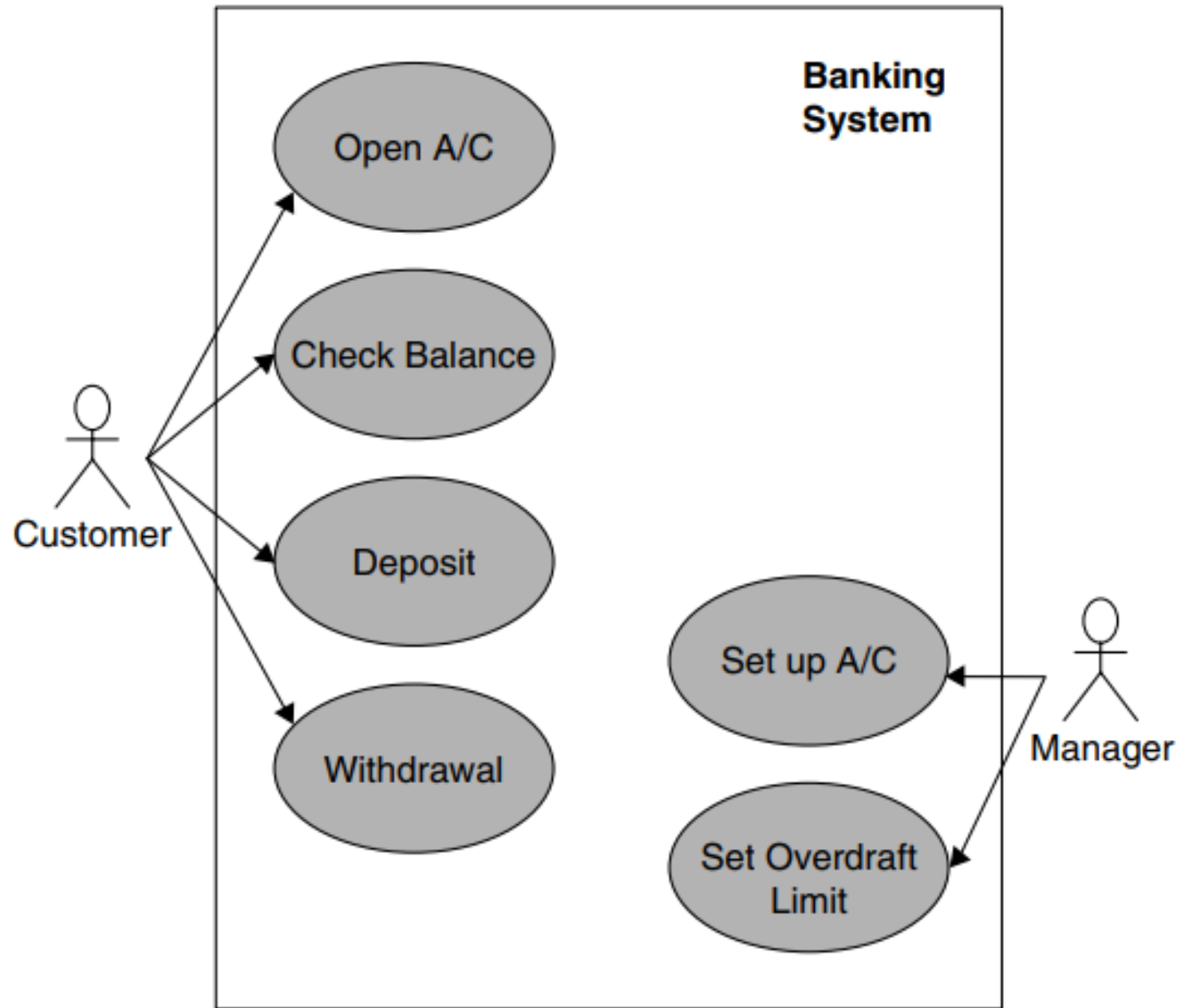
- User: user_id, username, email, password
- Social Media Account: account_id, platform, username, authentication_token
- Analytics Dashboard: dashboard_id
- Analytics Widget: widget_id

Operations:

- User: login(), logout(), createDashboard(), deleteDashboard()
- Social Media Account: connect(), disconnect()
- Analytics Dashboard: addWidget(), removeWidget(), customizeLayout()

use case diagram

- The use case diagram contains the actors and the use cases and shows the relationship between them.
- Each use case defines functional requirements for the system.
- Actors do not need to be human, even though they are represented as stick figures, but in fact represent roles.
- Each of the actors will have an association with at least one use case.
- The system boundary is also defined on the use case diagram by a rectangle, with the name of the system being given within the box.
- Normally significant, and useful, textual information is associated with each use case diagram.



To Do

Scenario: Smart Home Security System

Imagine you're designing a use case diagram for a smart home security system that includes various sensors, alarms, and control mechanisms to ensure the safety and security of the residents.

Give a description of actors, use cases and associations



Actors:

- 1.Homeowner:** Represents the owner or resident of the smart home who interacts with the security system.
- 2.Security System:** Represents the smart home security system itself, including sensors, alarms, and control mechanisms.
- 3.Emergency Services:** Represents external services such as police, fire department, or medical services that may be contacted in case of emergencies.

Use Cases:

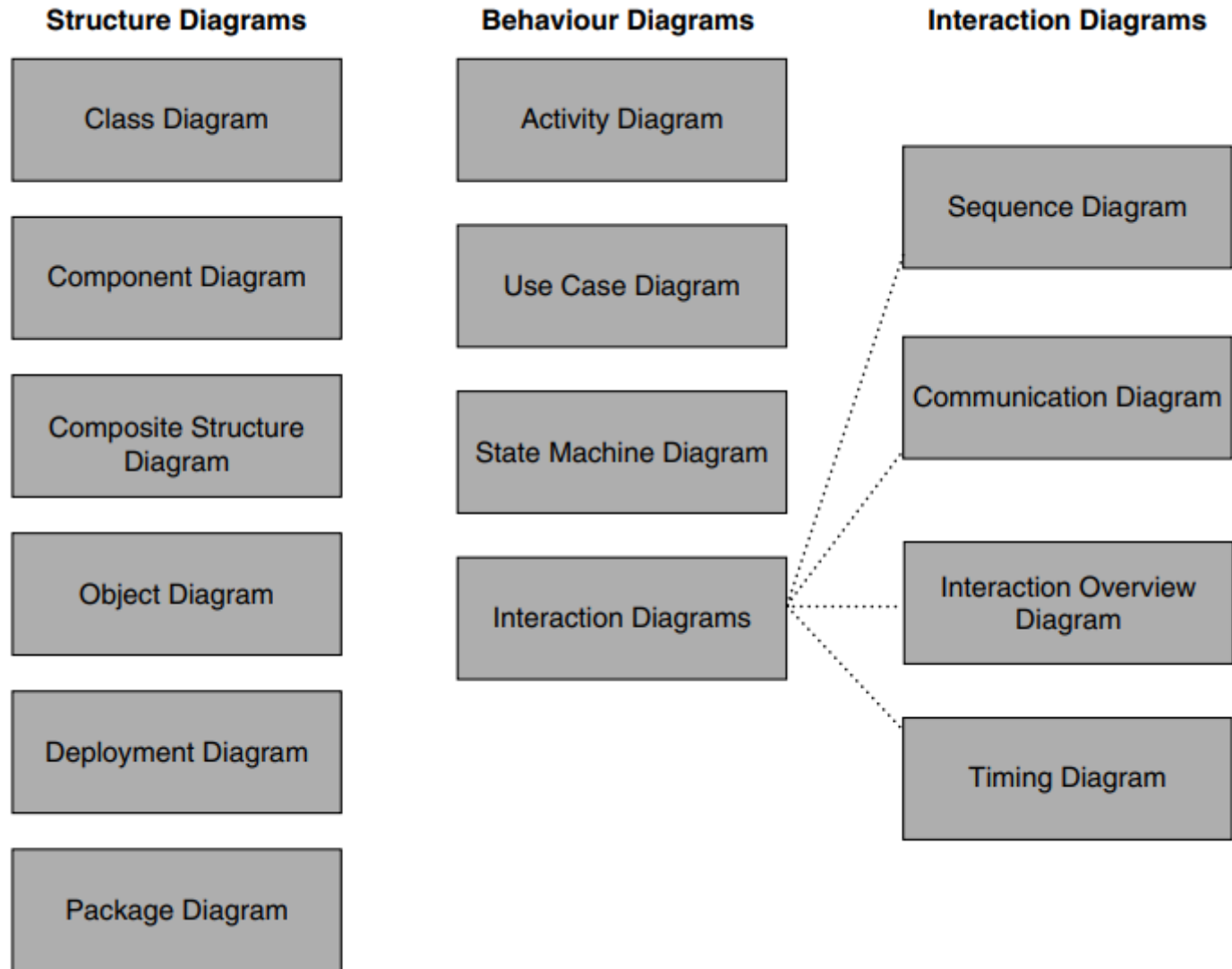
- 1.Arm/Disarm Security System:** The homeowner can arm or disarm the security system to activate or deactivate security measures such as sensors, alarms, and surveillance cameras.
- 2.Detect Intrusion:** The security system detects and alerts the homeowner of any unauthorized entry or intrusion detected by sensors (e.g., door/window sensors, motion sensors).
- 3.Sound Alarm:** The security system sounds an alarm to alert the homeowner and deter intruders in case of a detected intrusion.
- 4.Contact Emergency Services:** In case of emergencies such as a break-in, fire, or medical emergency, the security system can automatically contact emergency services and notify them of the situation.
- 5.Remote Monitoring:** The homeowner can remotely monitor the status of the security system and receive real-time alerts and notifications on their mobile device.
- 6.Emergency Response Coordination:** The security system coordinates with emergency services to provide relevant information and assistance during emergencies, such as providing the location of the incident and status updates.



Associations:

- Homeowner interacts with the Security System for arming/disarming, remote monitoring, and emergency response coordination.
- Security System detects intrusion, sounds alarm, and contacts emergency services based on detected events.
- Emergency Services are contacted by the Security System in case of emergencies and provide assistance as needed.

UML



- Sequence Diagrams are interaction diagrams that detail how operations are carried out.
- They capture the interaction between objects in the context of a collaboration.
- Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

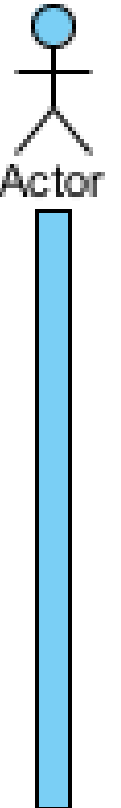
Sequence Diagrams captures:

- the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

Notification

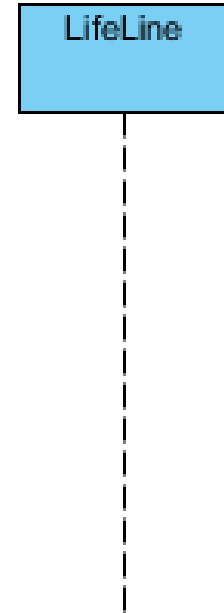
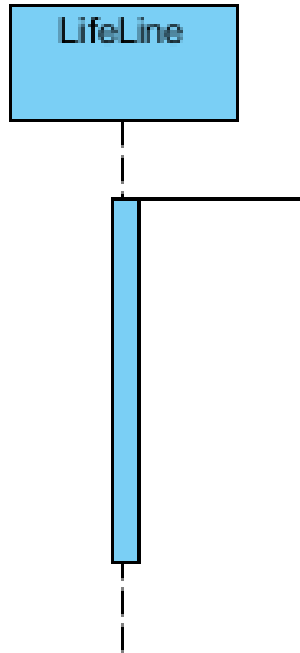
Actor

- a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data)
- represent roles played by human users, external hardware, or other subjects.



Lifeline

A lifeline represents an individual participant in the Interaction.



Activations

- A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
- The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively

Call Message

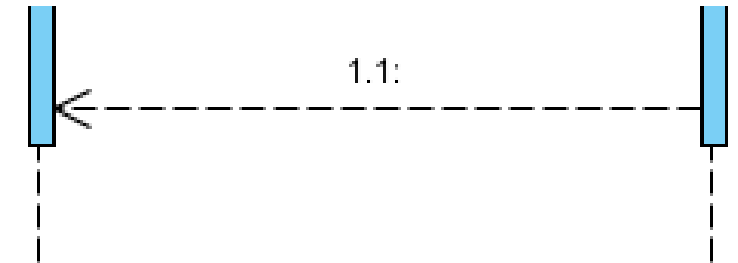
A message defines a particular communication between Lifelines of an Interaction.

Call message is a kind of message that represents an invocation of operation of target lifeline.



Return Message

Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.



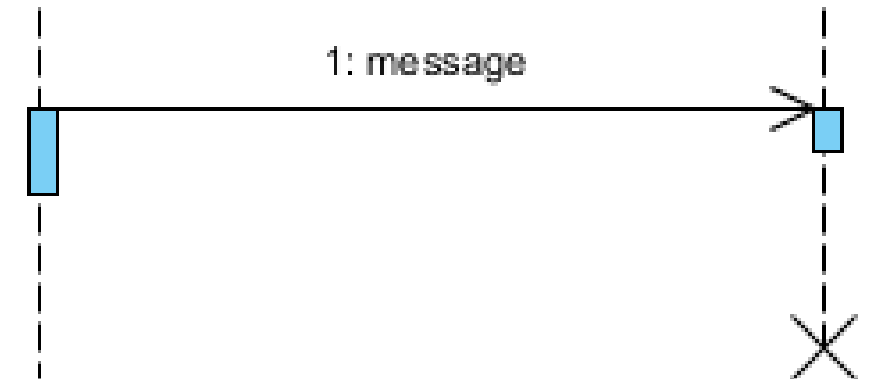
Self Message

Self message is a kind of message that represents the invocation of message of the same lifeline.



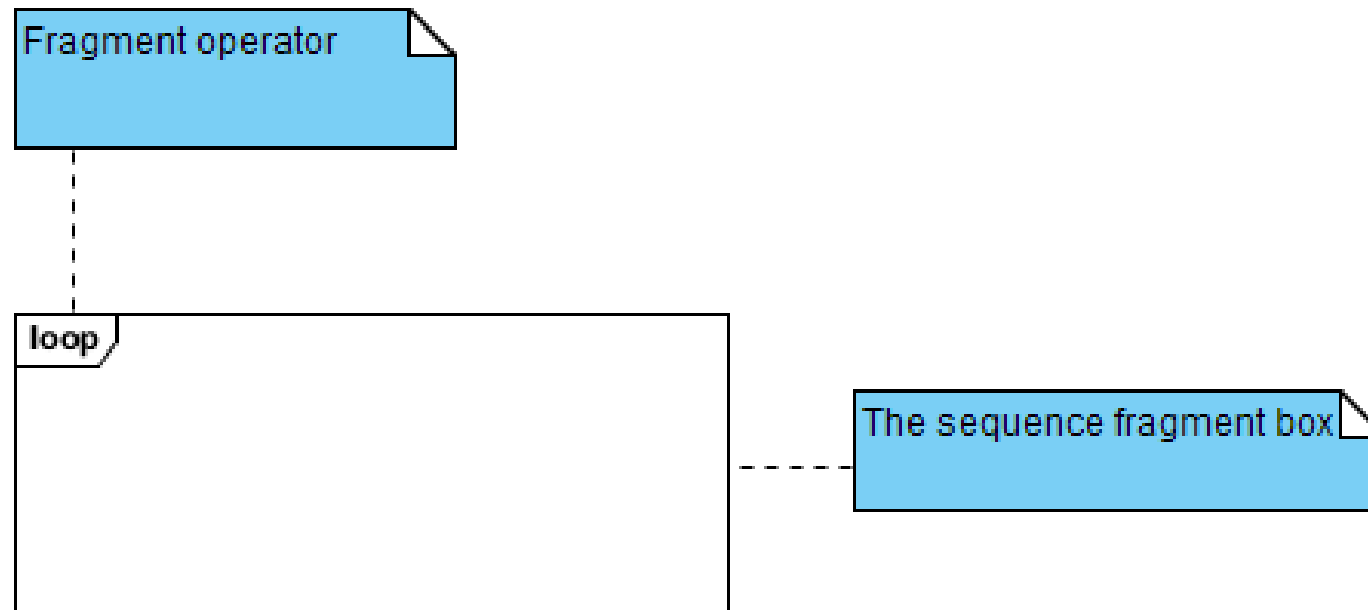
Destroy Message

Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.

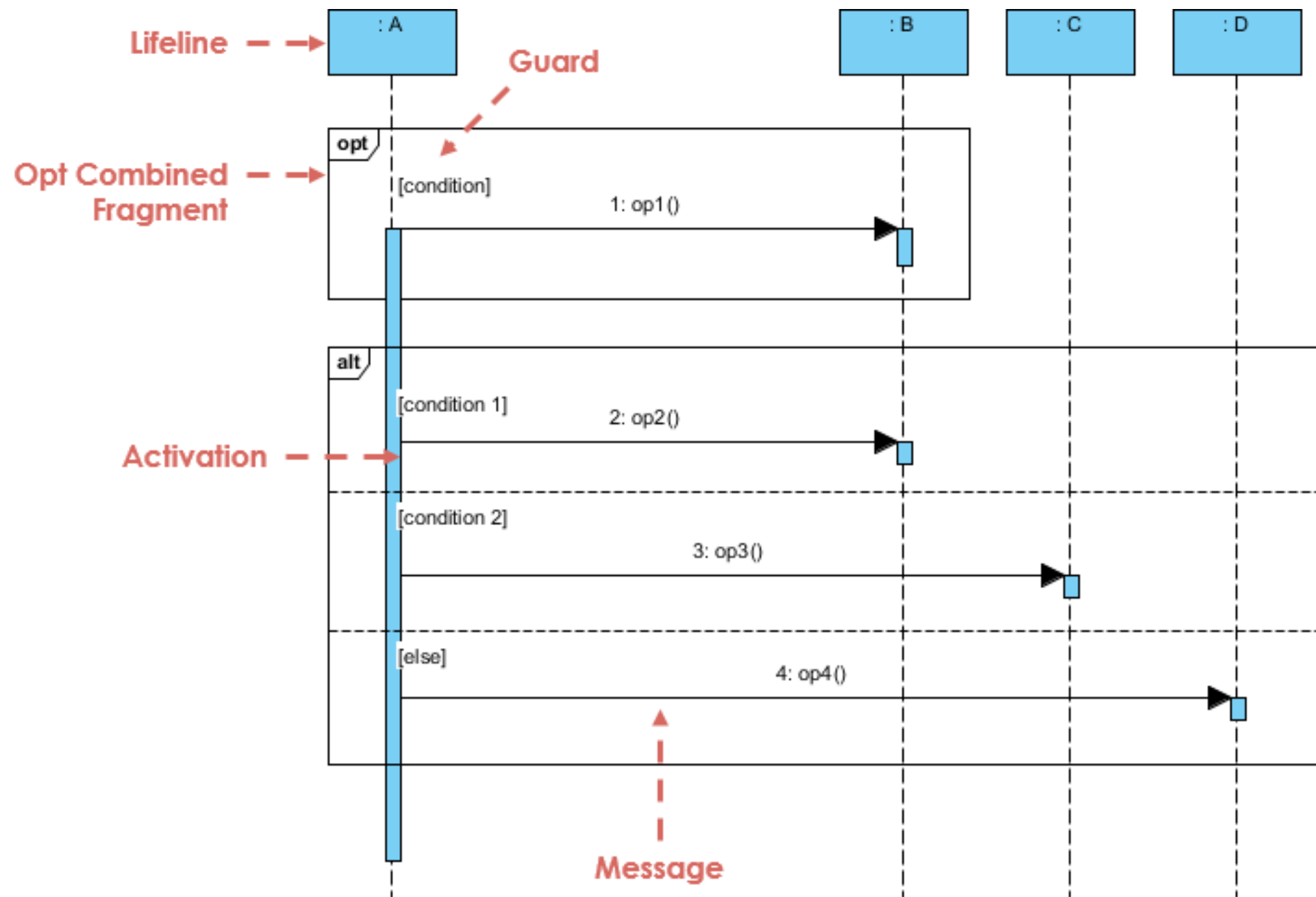


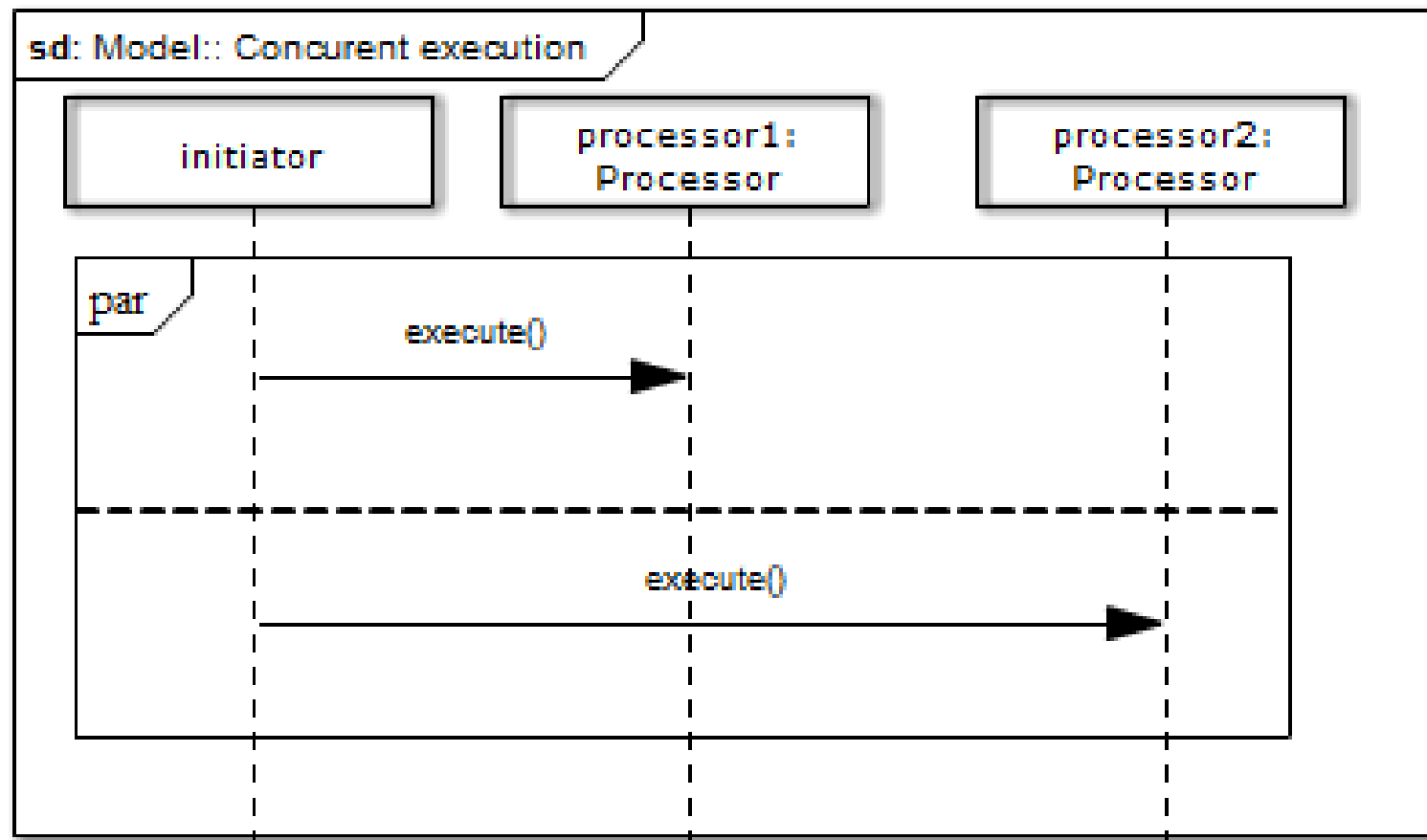
Sequence Fragments

- A sequence fragment is represented as a box, called a combined fragment, which encloses a portion of the interactions within a sequence diagram
- The fragment operator (in the top left corner) indicates the type of fragment
- Fragment types: ref, assert, loop, break, alt, opt, neg

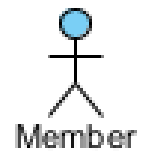


Operator	Fragment Type
alt	Alternative multiple fragments: only the one whose condition is true will execute.
opt	Optional: the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	Parallel: each fragment is run in parallel.
loop	Loop: the fragment may execute multiple times, and the guard indicates the basis of iteration.
region	Critical region: the fragment can have only one thread executing it at once.
neg	Negative: the fragment shows an invalid interaction.
sd	Sequence diagram: used to surround an entire sequence diagram.





sd Place Order Scenario



Member

: Order

: Courier

: Mail

: Notification

loop

1: For each line [for each order item]

alt

[Member type = VIP]

1.1: dispatch

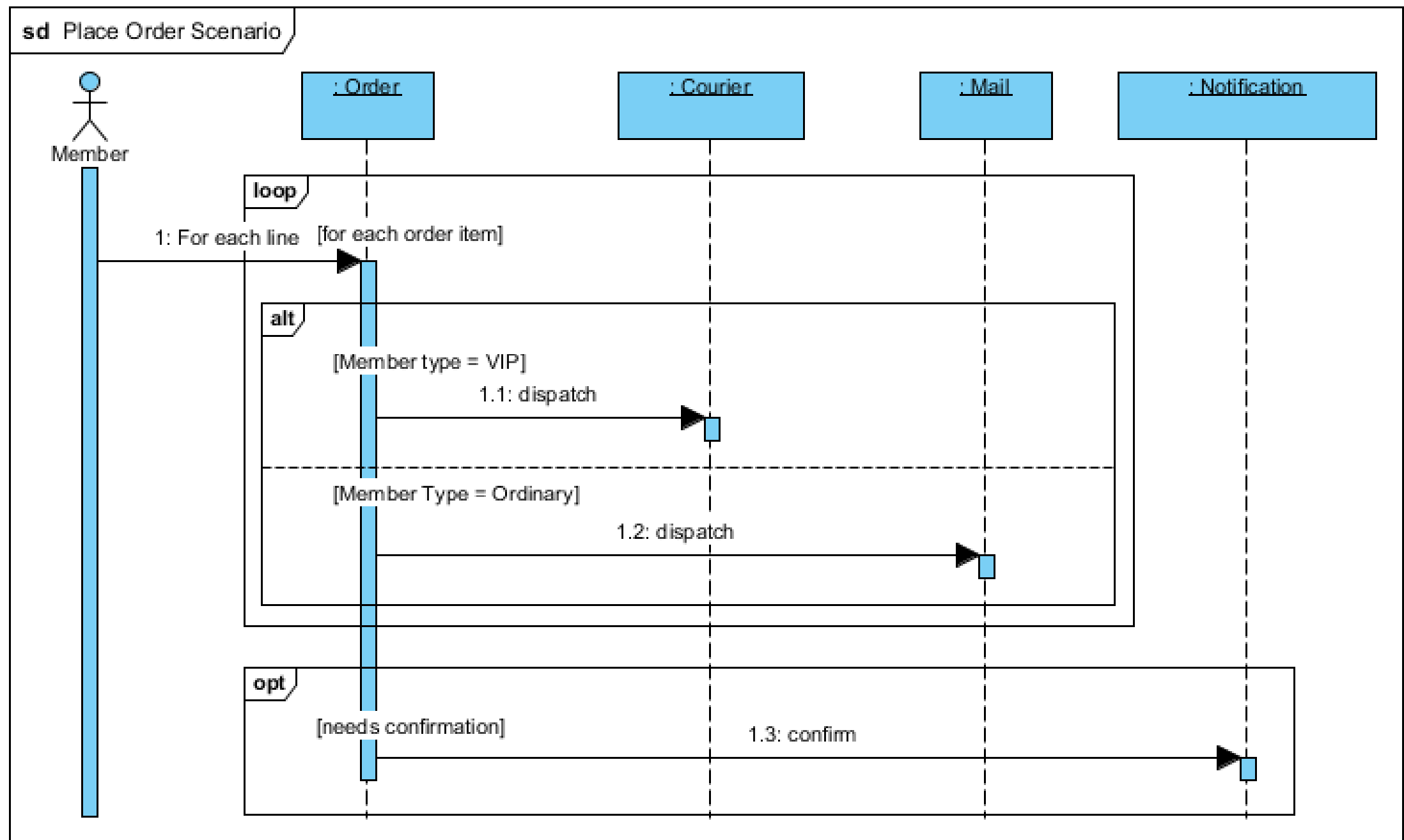
[Member Type = Ordinary]

1.2: dispatch

opt

[needs confirmation]

1.3: confirm



To Do

Scenario: Online Shopping Platform with Recommendations

Imagine you're designing a sequence diagram for an online shopping platform that provides personalized product recommendations to users based on their browsing history and preferences.

Give description of actors, sequence of events and entities



Entities:

1.User: Represents a user of the online shopping platform. Each user has a unique user ID and may have a browsing history and preferences associated with their account.

2.Product: Represents a product available for sale on the platform. Each product has a unique product ID, name, description, category, and price.

3.Recommendation Engine: Represents the system component responsible for generating personalized product recommendations for users based on their browsing history and preferences.

Sequence of Events:

- 1.User Browsing:** The sequence begins when a user browses the online shopping platform and views various products.
- 2.Recommendation Generation Request:** When the user requests product recommendations, the platform sends a request to the recommendation engine, including the user's browsing history and preferences.
- 3.Recommendation Generation:** The recommendation engine receives the request and processes the user's browsing history and preferences to generate personalized product recommendations.
- 4.Recommendation Retrieval:** Once the recommendations are generated, the recommendation engine sends the list of recommended products back to the platform.
- 5.Display Recommendations:** The platform receives the recommended products and displays them to the user, along with other relevant information such as product details and pricing.
- 6.User Interaction:** The user may interact with the recommended products by selecting them for further exploration or adding them to their shopping cart.

Actors:

- User
- Online Shopping Platform
- Recommendation Engine

Messages:

- User Browsing: ViewProduct()
- Recommendation Generation Request: GenerateRecommendations(userID, browsingHistory, preferences)
- Recommendation Generation: GenerateRecommendations()
- Recommendation Retrieval: SendRecommendations(recommendedProducts)
- Display Recommendations: DisplayRecommendedProducts(recommendedProducts)
- User Interaction: SelectProduct(productID), AddToCart(productID)



That's it