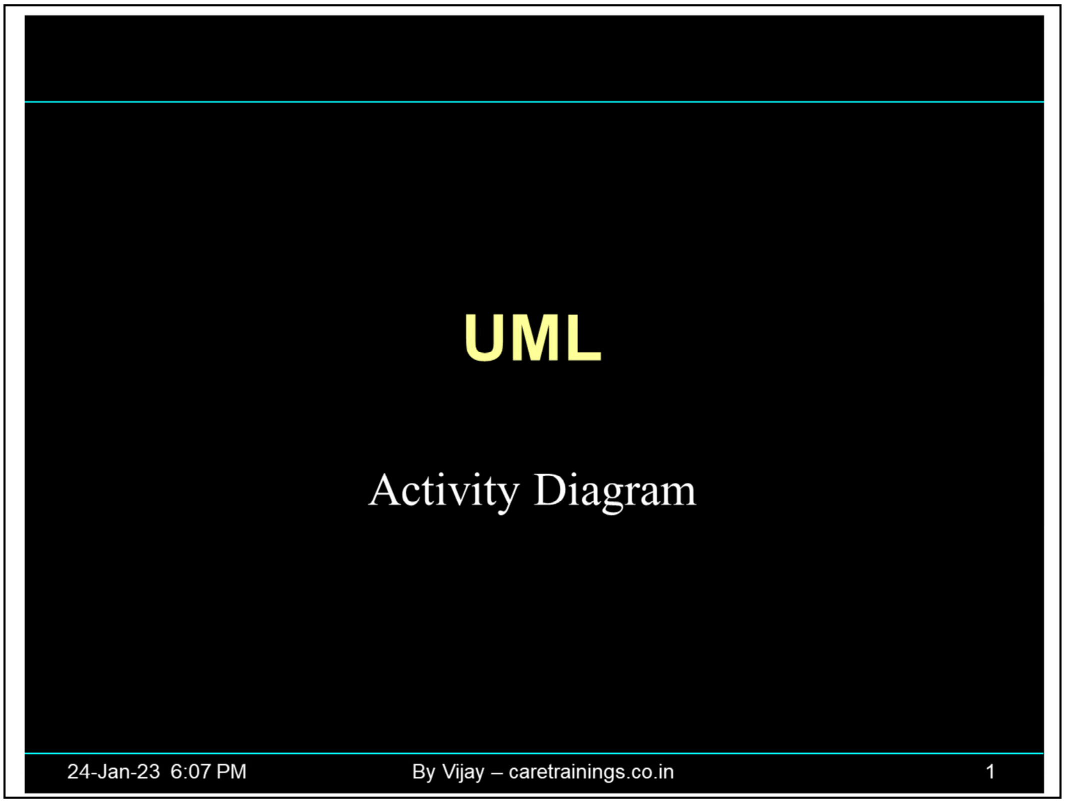
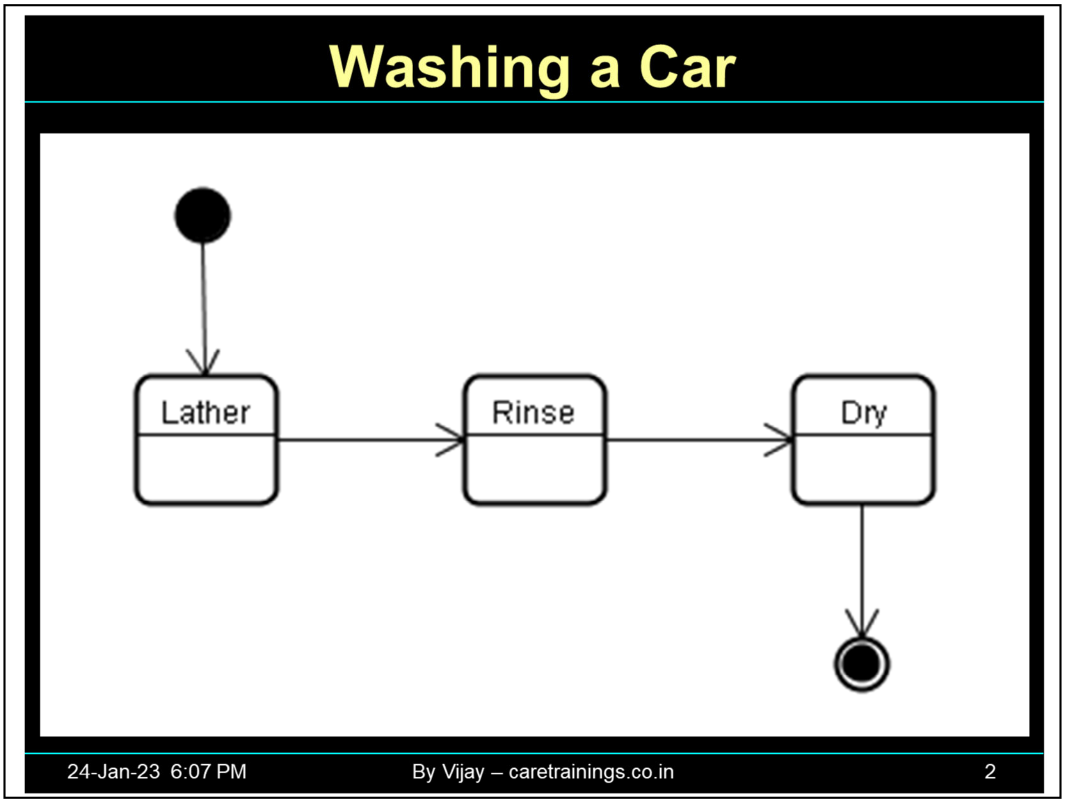


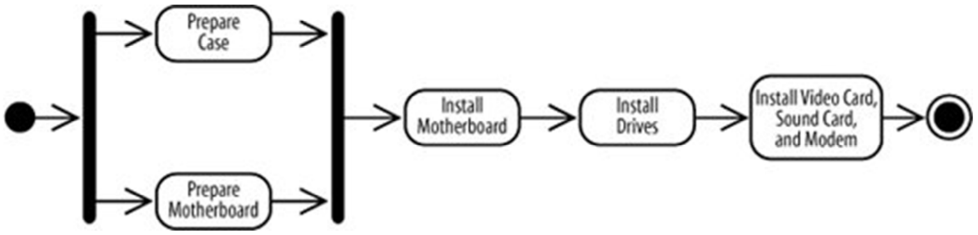
UML

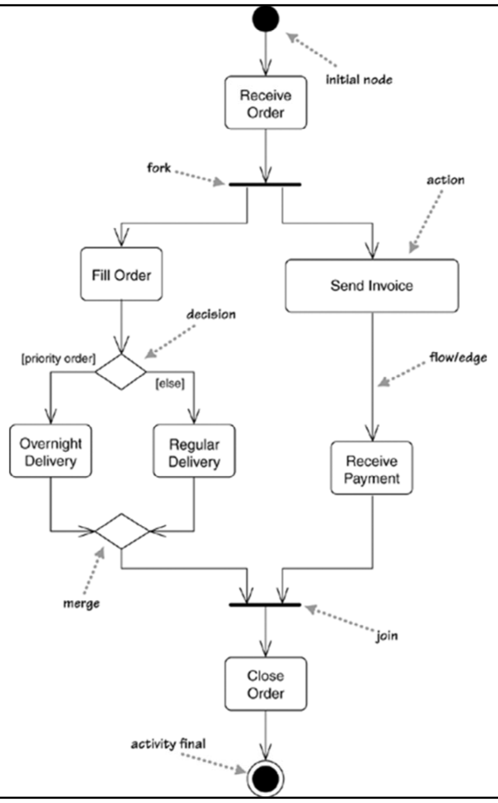
Activity Diagram

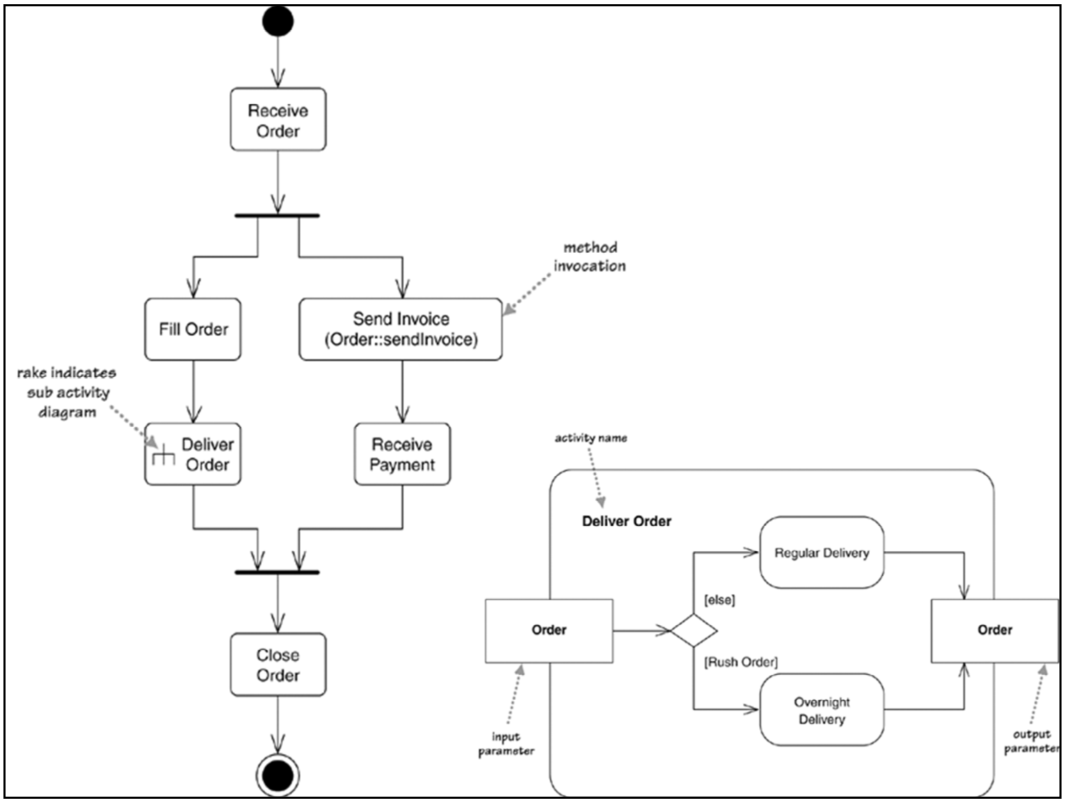


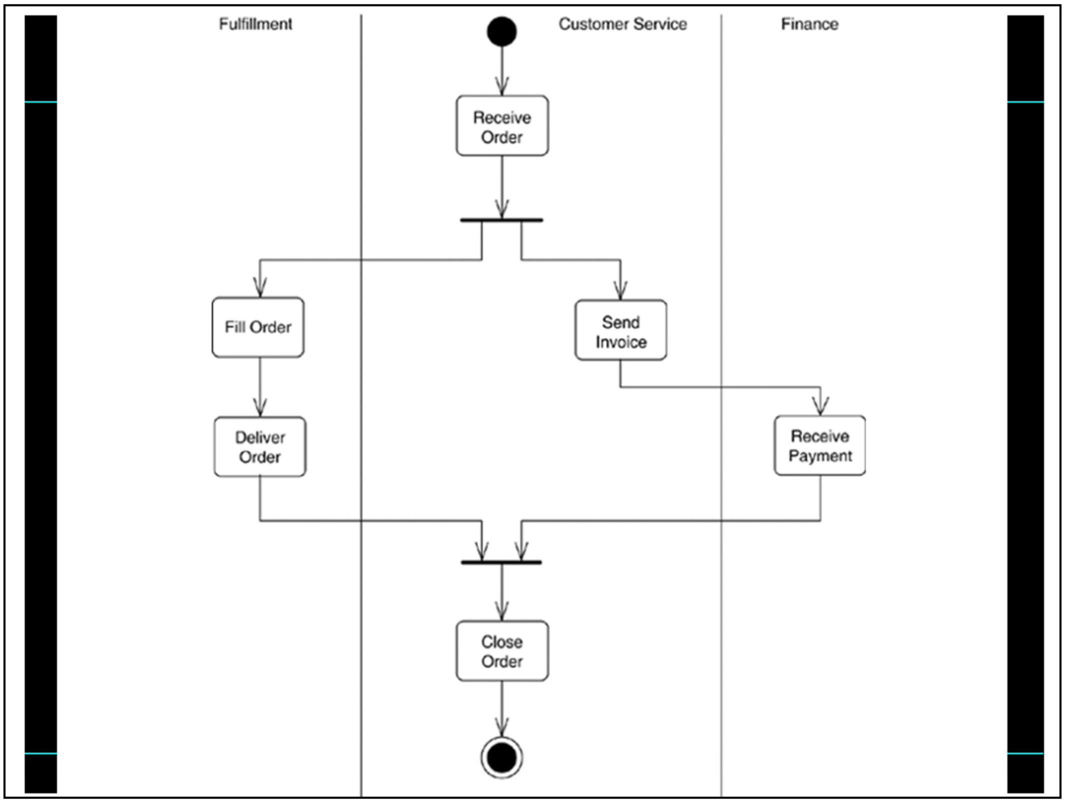
Washing a Car

Building a PC

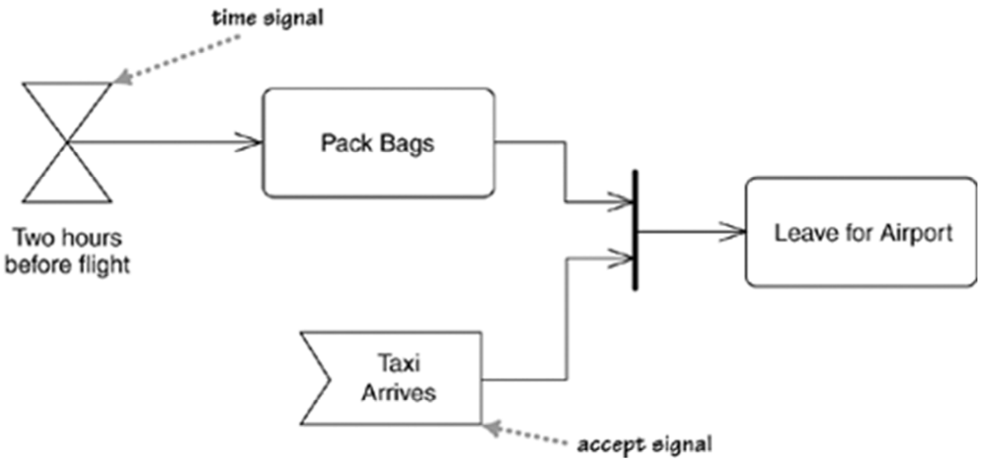



initial node
Receive Order
fork
action
Fill Order
Send Invoice
decision
[priority order]
[else]
flow/edge
Overnight Delivery
Regular Delivery
Receive Payment
merge
join
Close Order
activity final

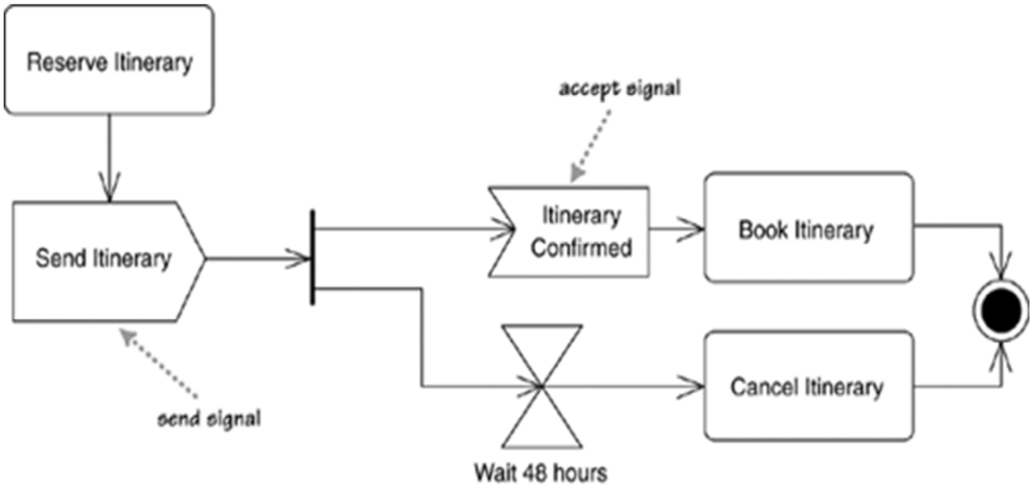

Receive Order
Fill Order
Send Invoice (Order::sendInvoice)
method invocation
rake indicates sub activity diagram
Deliver Order
Receive Payment
Close Order
activity name
Deliver Order
Regular Delivery
[else]
Order
[Rush Order]
Overnight Delivery
Order
input parameter
output parameter


Fulfillment
Customer Service
Finance
Receive Order
Fill Order
Send Invoice
Deliver Order
Receive Payment
Close Order

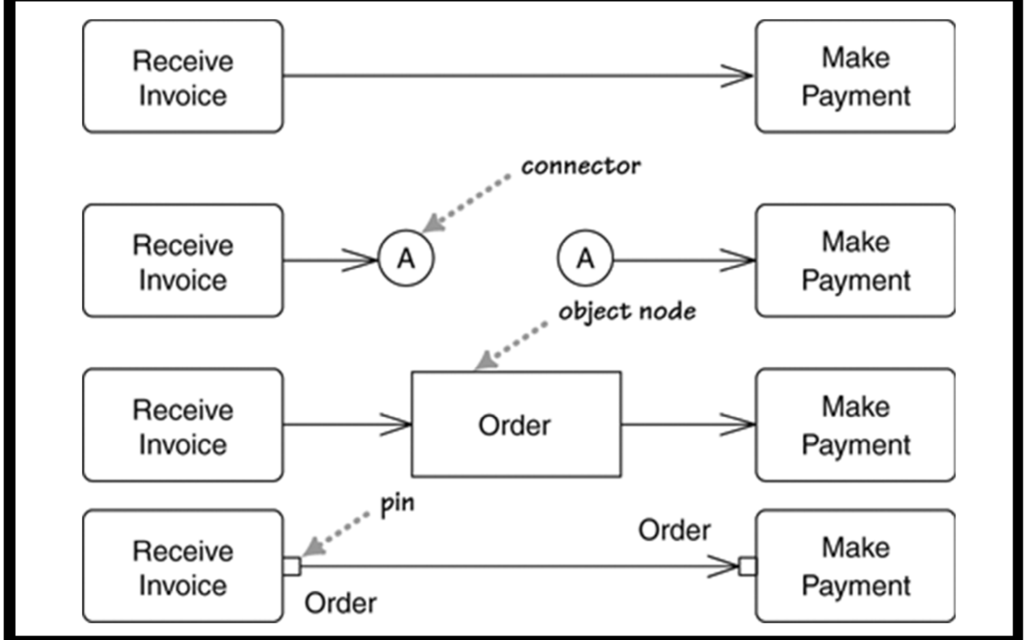
Time and Signal

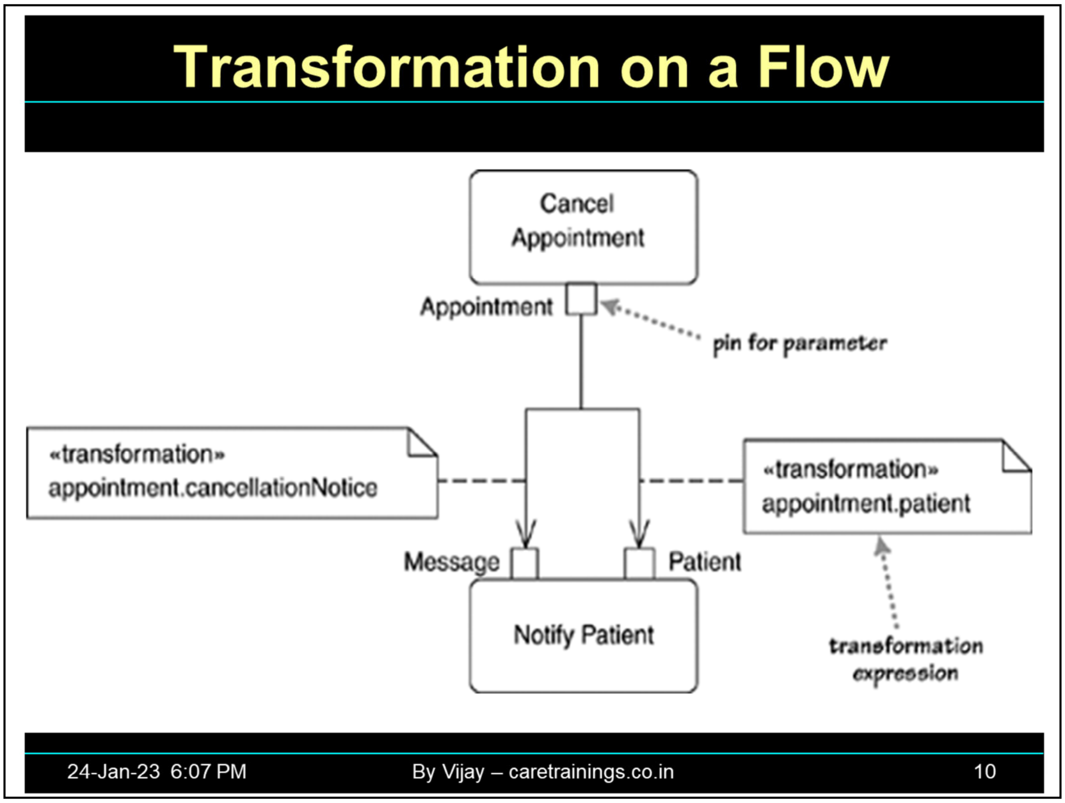


Sending / Receiving Signals



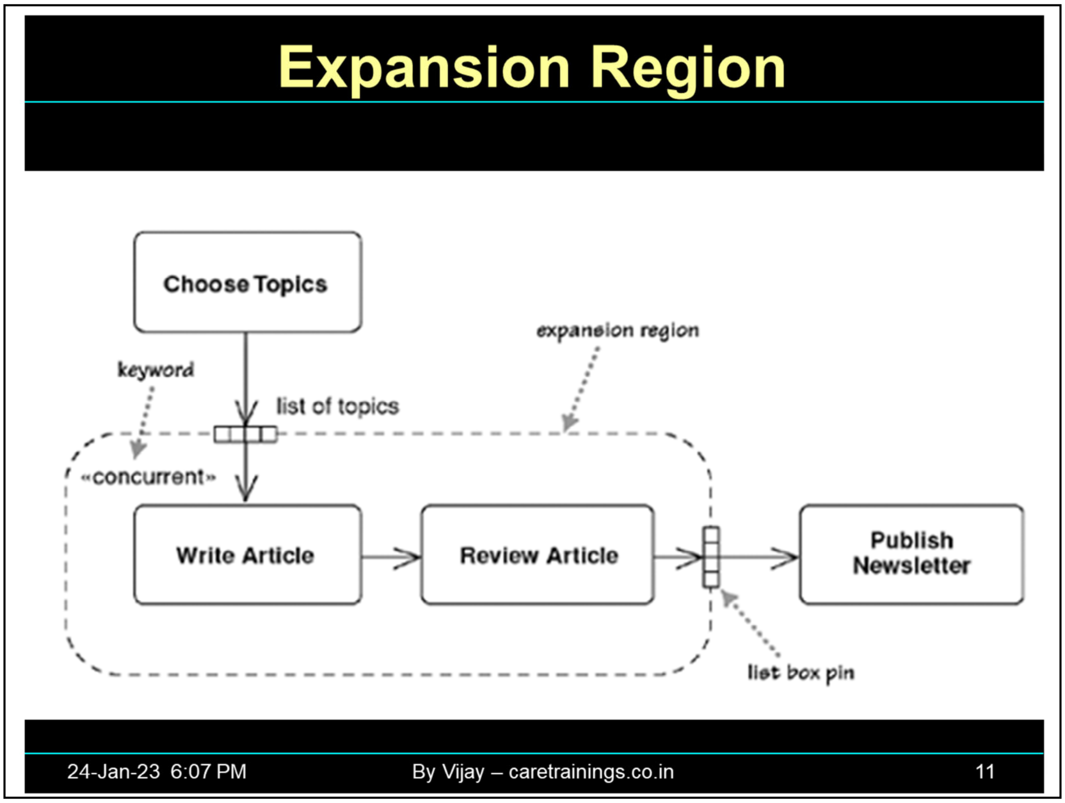
Equivalent Notations





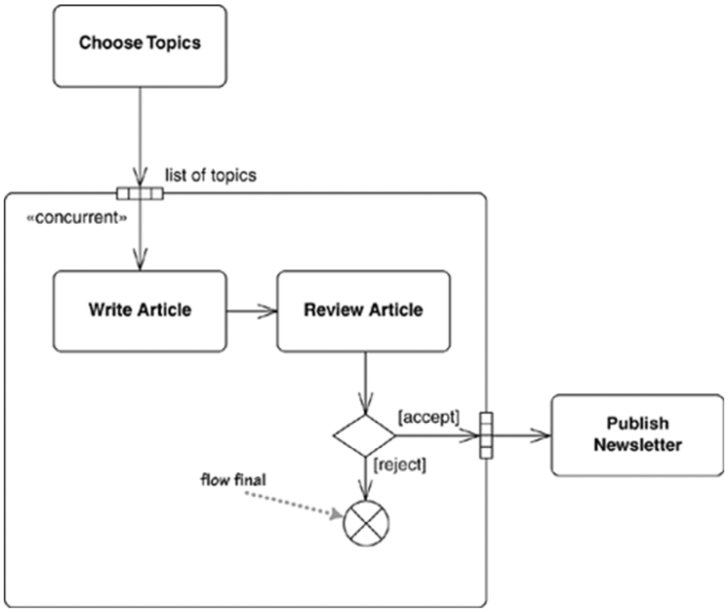
Transformation on a Flow

Cancel Appointment
Appointment
pin for parameter
«transformation» appointment.cancellationNotice
«transformation» appointment.patient
Message
Patient
Notify Patient
transformation expression



Expansion Region

Flow Final


Choose Topics
list of topics
«concurrent»
Write Article
Review Article
[accept]
[reject]
flow final
Publish Newsletter

Introduction to Behavior Models

- How and when things happen in a system defines the behavior of the system. It refers to how:
  - The various objects in the domain model interact with each other over time
  - Their state changes during this interaction
- The events, message flows, and states of the objects during the interaction need to be modeled additionally to model the behavior of the system.
- The dynamic domain model consists of the following types of diagrams:
  - Activity diagrams
  - Interaction diagrams
  - State diagrams

Activity Diagrams

- The most intuitive way of modeling the behavior of any system is to show the workflow (data and control flow) of the business processes as a flowchart.
- In OOAD, you can do this by using activity diagrams.
- Activity diagrams describe flow of control among steps (actions) in an activity.
- They model the logic behind an activity in the problem domain by showing various control flows, including:
  - Sequential
  - Parallel
  - Branched
  - Concurrent

Activity Diagrams (Cont'd)

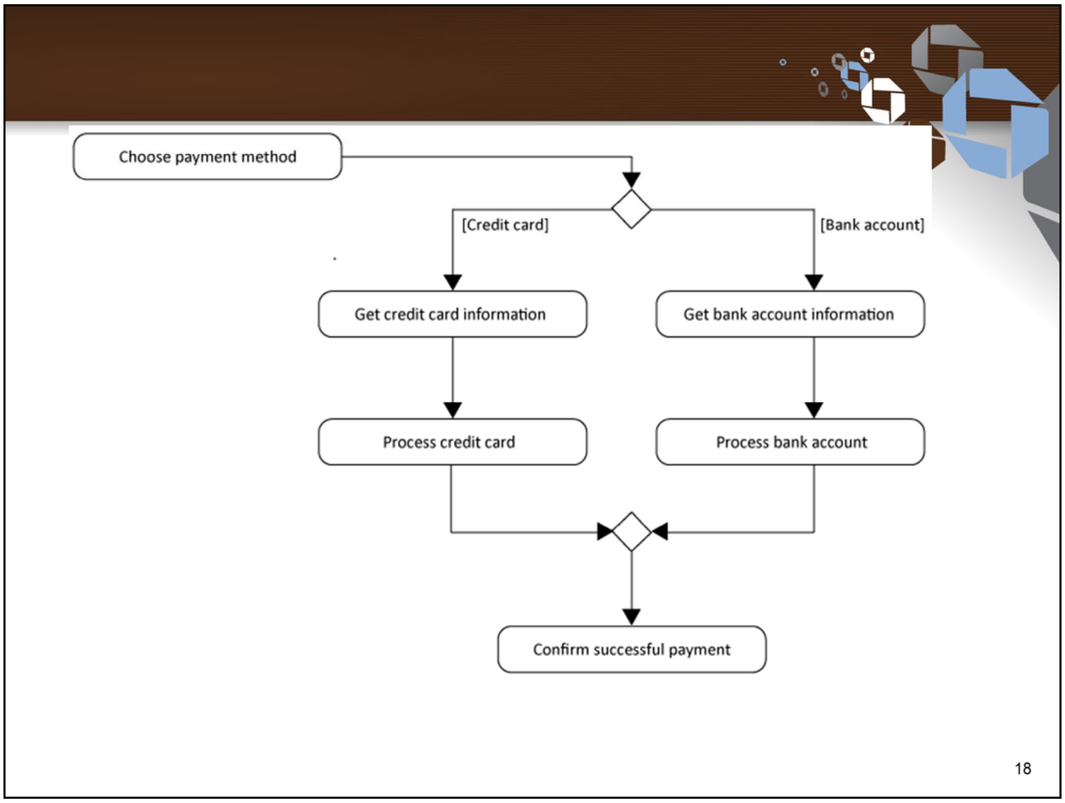
- The activity can be any of the following artifacts:
  - A single business rule
  - A business process
  - A single use case
  - Series of multiple use cases
- An activity diagram shows, the following aspects for the preceding artifacts:
  - Decision-making points
  - Actions that need to be executed repeatedly
  - Actions that need to be executed concurrently

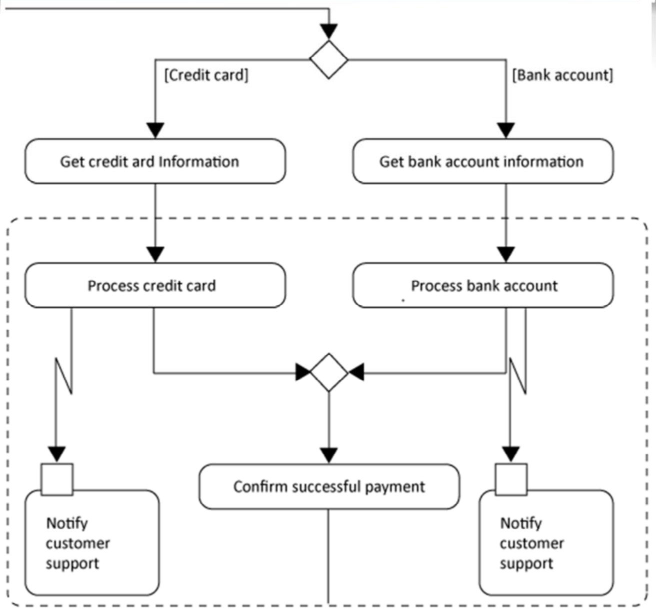
Activity Diagrams (Cont’d)

- As part of domain modeling, you should create activity diagrams for all:
  - Primary use cases
  - Business processes/rules identified during requirements capturing
- The tasks involved in creating an activity diagram are:
  - Define the behavior that you are modeling.
  - Identify the steps comprising the activity.
  - Identify the control and data flow between steps.
  - Identify the expansion regions.
  - Group actions into partitions.
  - Identify exceptions in the workflow.

Defining the Behavior

- You can start creating an activity diagram by clearly defining the behavior for which you are creating the activity.
- The activity node in the activity diagram represents a behavior. It can contain other elements, including:
  - Actions (steps)
  - Control flow constructs
  - Objects in a particular state
- These are represented as rounded rectangles. The name of the activity is specified in the top-left corner of the rounded rectangle.


Choose payment method
[Credit card]
[Bank account]
Get credit card information
Get bank account information
Process credit card
Process bank account
Confirm successful payment
18



[Credit card]

[Bank account]

Get credit ard Information

Get bank account information

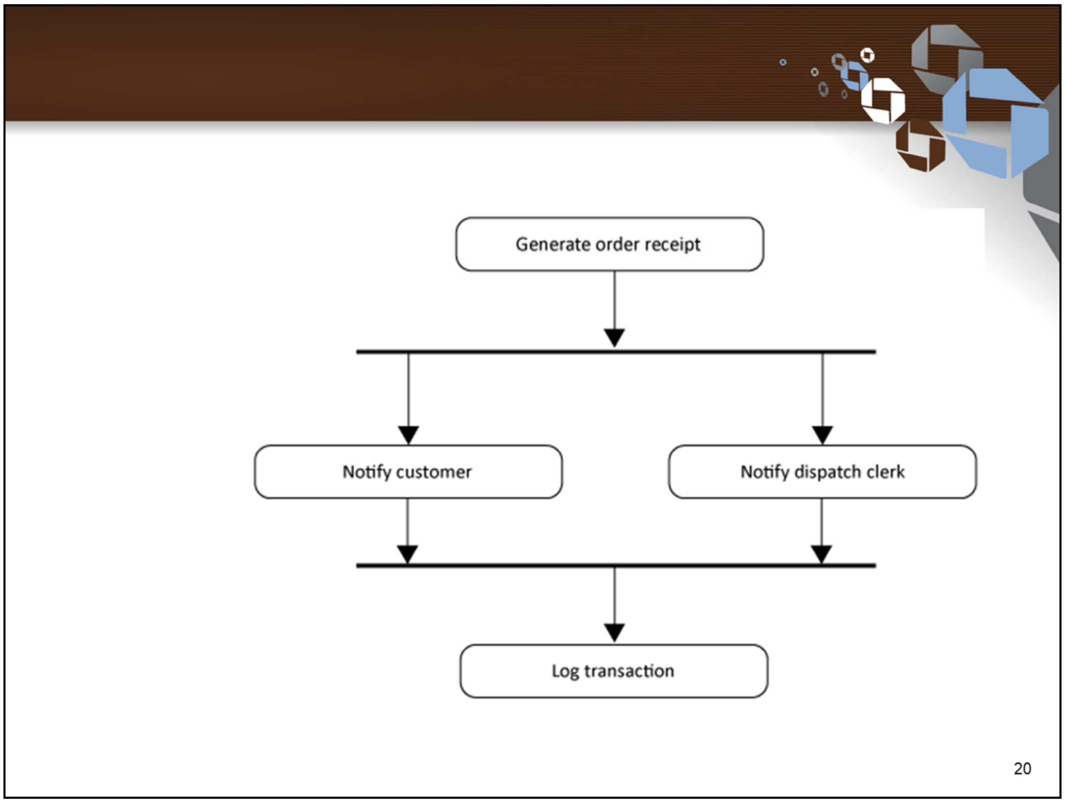
Process credit card

Process bank account

Confirm successful payment

Notify customer support

Notify customer support

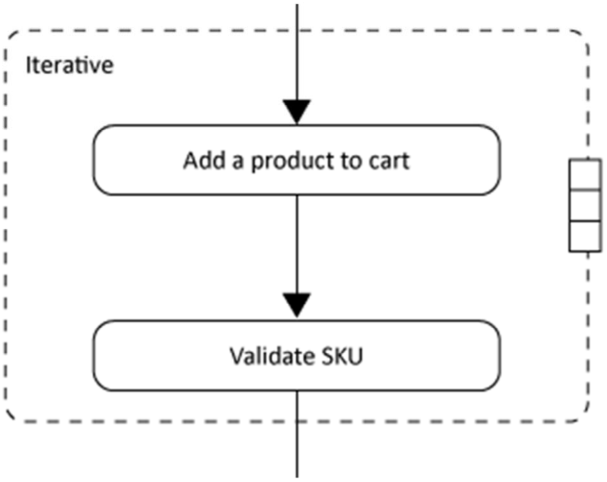


Generate order receipt

Notify customer

Notify dispatch clerk

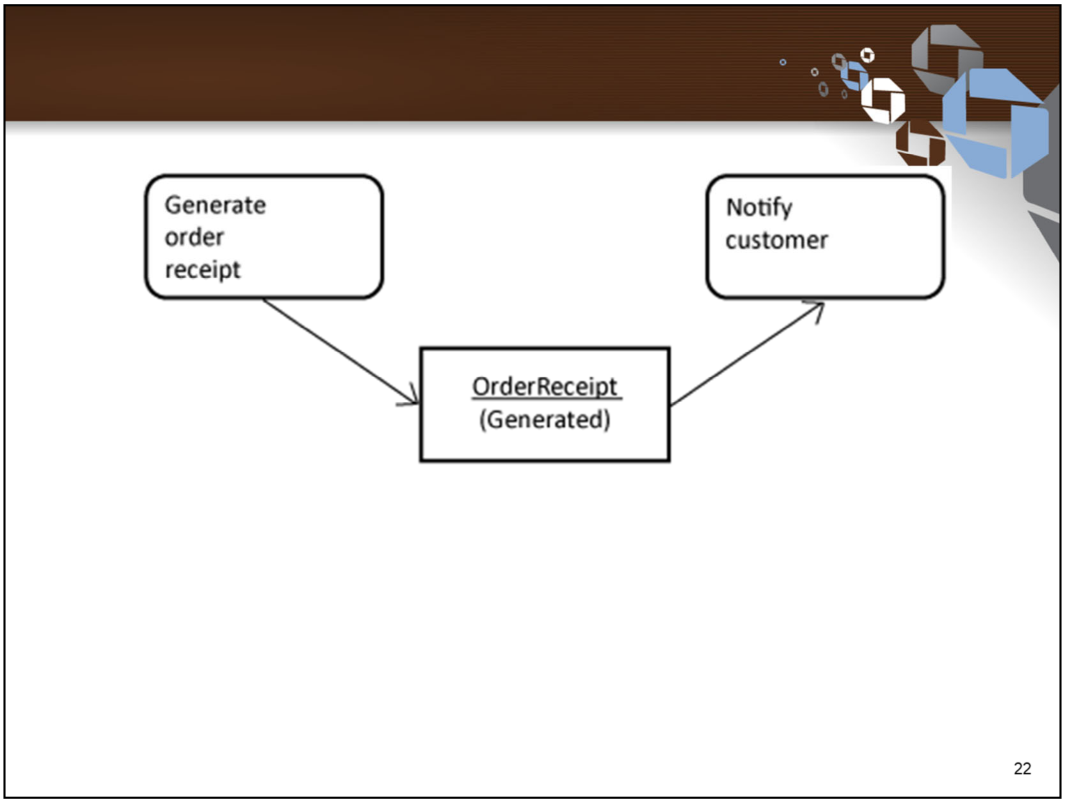
Log transaction



Iterative

Add a product to cart

Validate SKU



Generate order receipt

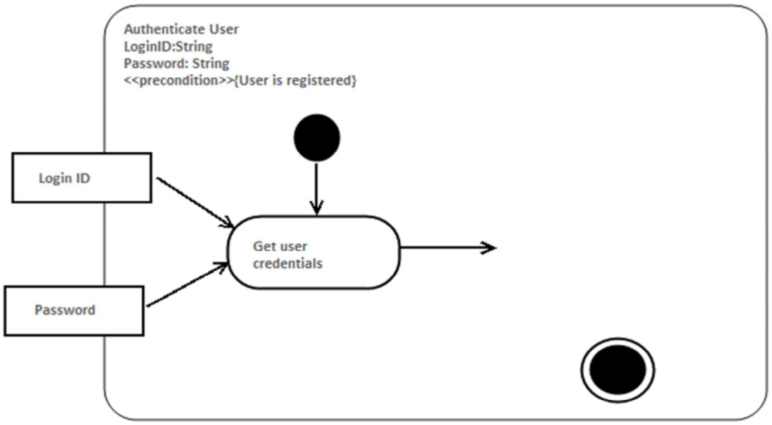
OrderReceipt
(Generated)

Notify customer

Defining the Behavior (Cont'd)

- An activity can have the following information associated with it:
  - Parameter –
    - Is the data that an activity requires to exhibit the desired behavior
    - Is specified below the activity name in the format, **Parameter name: Parameter type**
  - Pre-conditions –
    - Are pre-conditions associated with the use case or business rule/process being modeled through the activity diagram
    - Are specified under the activity name with the stereotype <<precondition>>
  - Post-conditions –
    - Are the post-conditions associated with the use case or business rule/process being modeled through the activity diagram
    - Are specified under the activity name with the stereotype <<postcondition>>

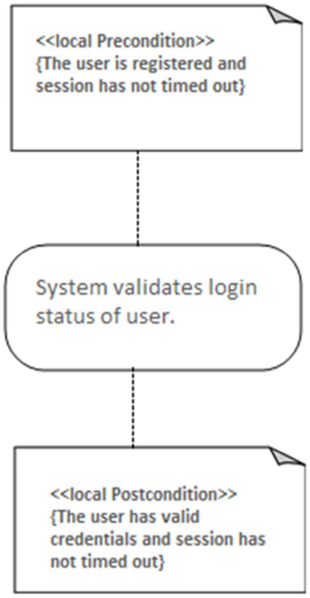
Defining the Behavior (Cont'd)



Identifying Steps

- After defining the activity node, you need to identify the individual steps that comprise the activity.
- An action represents an atomic step in the activity and is represented as a rounded rectangle with the name of the action written inside it.
- An action can also have pre-conditions and post-conditions associated with it. These conditions:
  - Have a local scope
  - Are limited to the action
  - Do not impact other actions or elements of an activity diagram

Identifying Steps (Cont’d)



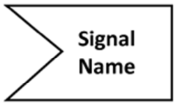
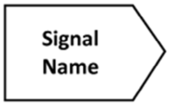
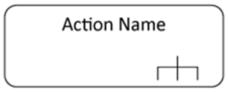
<<local Precondition>>
{The user is registered and session has not timed out}

System validates login status of user.

<<local Postcondition>>
{The user has valid credentials and session has not timed out}

Identifying Steps (Cont'd)

- Some crucial types of actions and their notations are shown in the following table.

Input signal action	A step can be created to receive a signal or message. Such a step is shown as a concave polygon instead of a rounded rectangle.	 Signal Name
Output signal action	A step can be created to send a signal or message. Such a step is shown as a convex polygon instead of a rounded rectangle.	 Signal Name
Call Activity action	A step can cause another activity to be started. This is represented by placing a hierarchy symbol at the top-right corner of the action rectangle.	 Action Name