COURSE NAME

OBJECT ORIENTED ANALYSIS AND DESIGN

CSC 2210

(UNDERGRADUATE)

CHAPTER 5

ACTIVITY DIAGRAM

VICTOR STANY ROZARIO
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE, AIUB
Web: https://cs.aiub.edu/profile/stany

RG Google Scholar LinkedIn

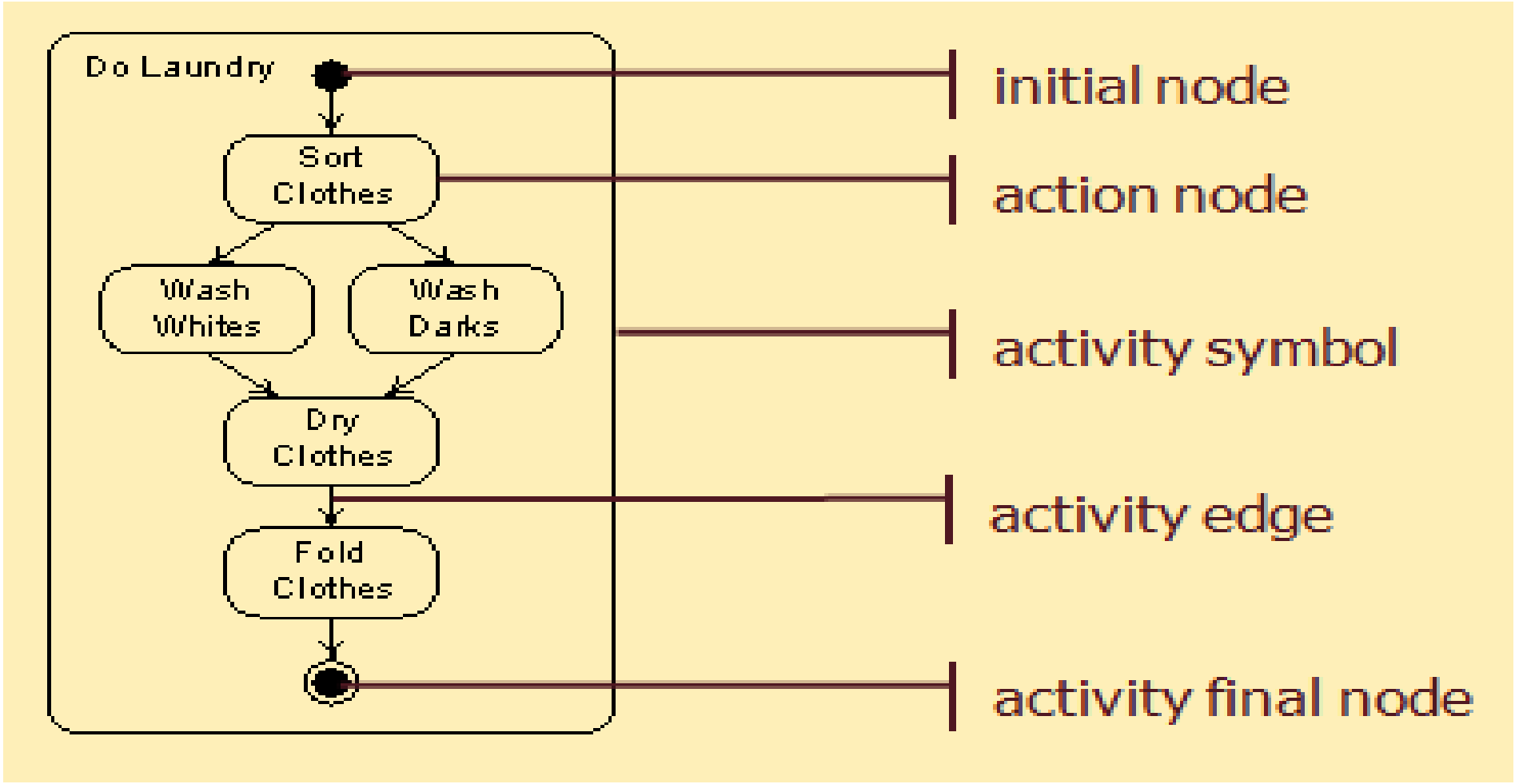
PROCESS

- A **process** is a collection of related tasks that transforms a set of inputs into a set of outputs.
- Process description notations describe **design processes** as well as **computational processes** we design.
- An **activity** is a non-atomic task or procedure decomposable into actions.
- An **action** is a task or procedure that cannot be broken into parts.

ACTIVITY DIAGRAM

- Describes activities and flows of data or decisions between activities
- Provides a very broad view of business processes
- Can be used to break out the activities that occur within a use case
- Commonly shows many different activities that will be handled by lots of different symbols
- Good for showing parallel threads
- Activity diagrams can be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams

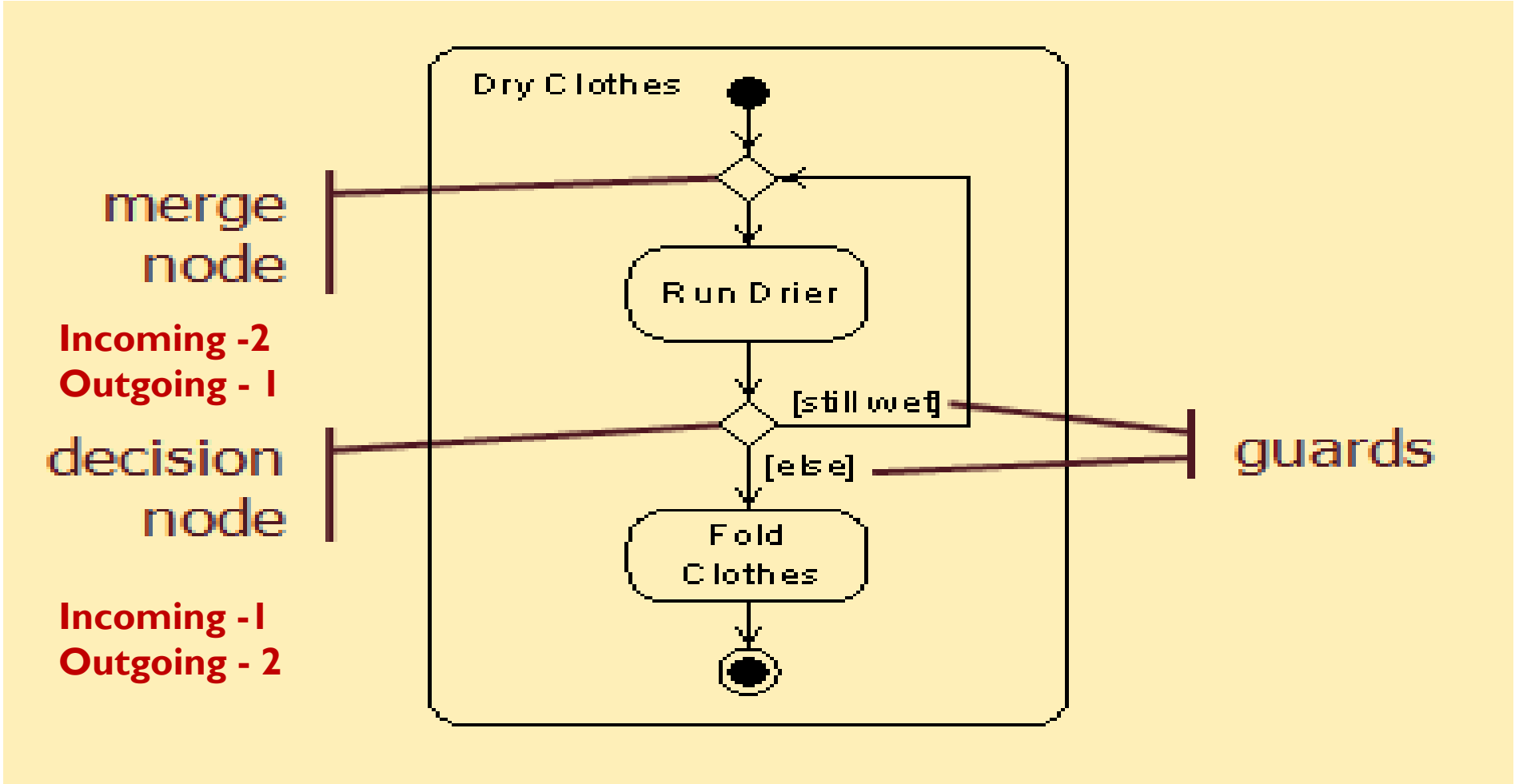
ACTIVITY GRAPH ELEMENTS



EXECUTION MODEL

- Execution is modeled by **tokens** that are produced by action nodes, travel over action edges, and are consumed by action nodes.
- When there is a token on every incoming edge of an action node, it consumes them and begins execution.
- When an action node completes execution, it produces tokens on each of its outgoing edges.
- An initial node produces a token on each outgoing edge when an activity begins.
- An activity final node consumes a token available on any incoming edge and terminates the activity.

BRANCHING NODES EXECUTION

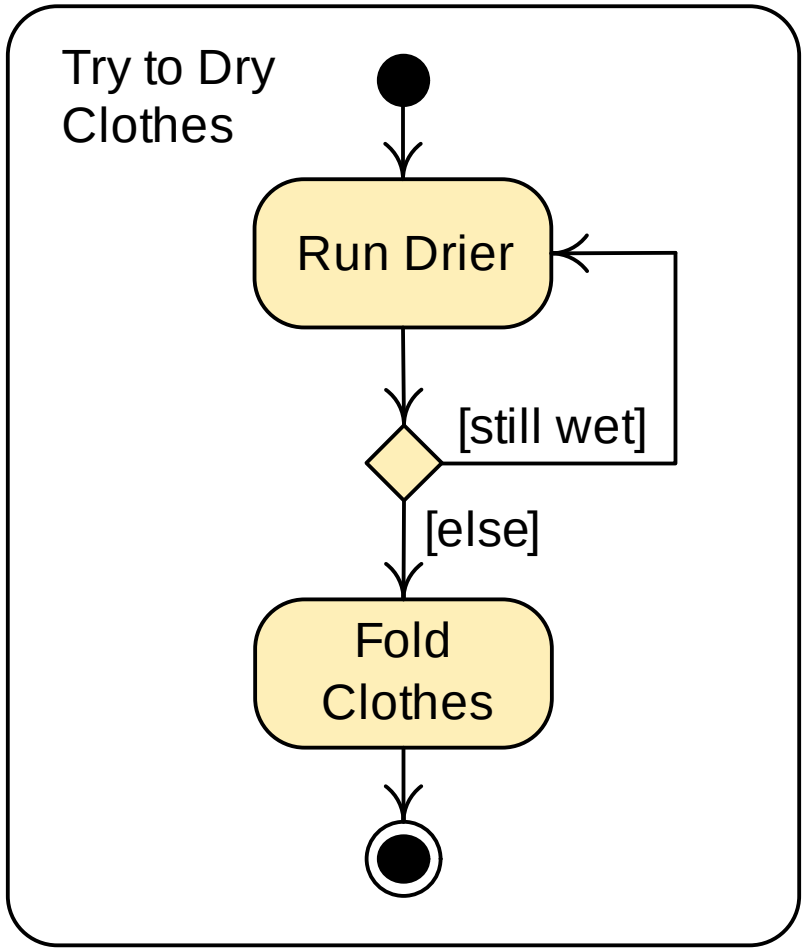


BRANCHING EXECUTION

- If a token is made available on the incoming edge of a **decision node** (join **if…else** condition), , the token is made available on the outgoing edge whose guard is true.
- Guards must be mutually exclusive (the outgoing signal is transfer to either true or false, but not both)
- If a token is available on any incoming edge of a **merge node** (join **parallel** actions), it is made available on its outgoing edge.

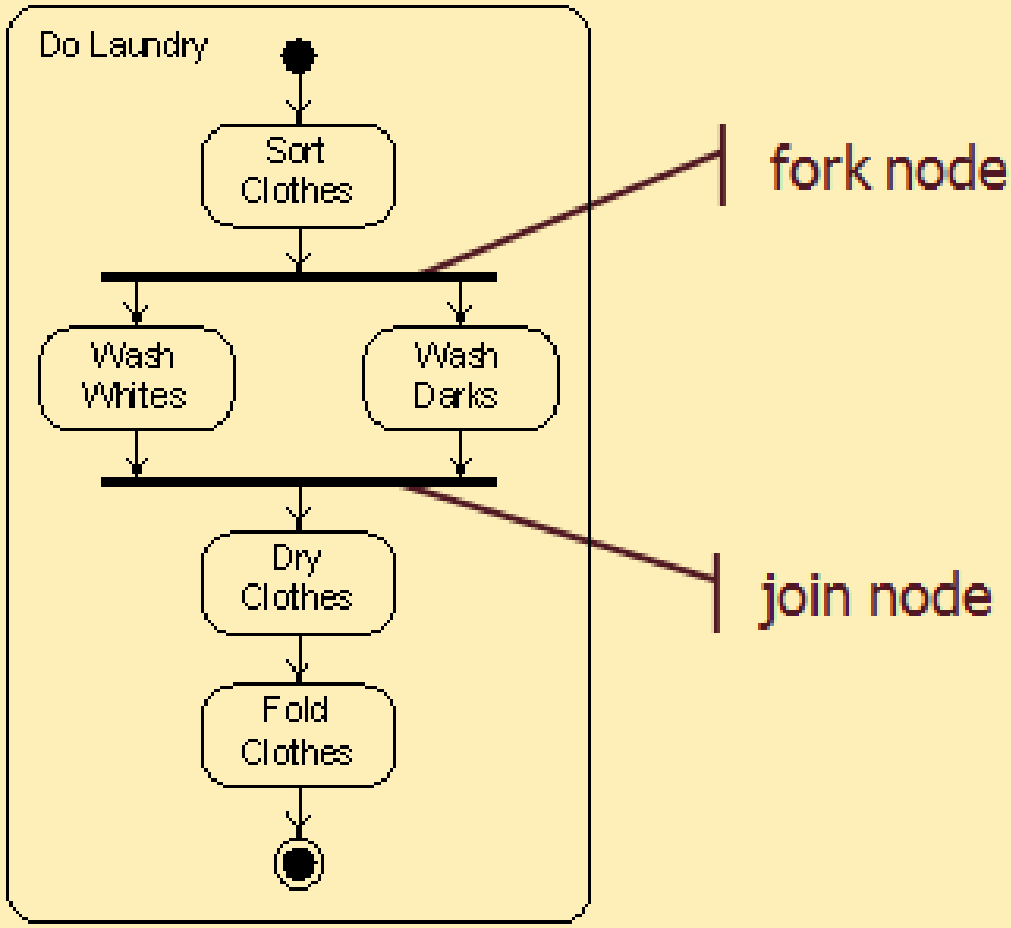
DEADLOCK

RunDrier cannot execute: when the activity begins, there is a token on the edge from the initial node but not on the other incoming edge.



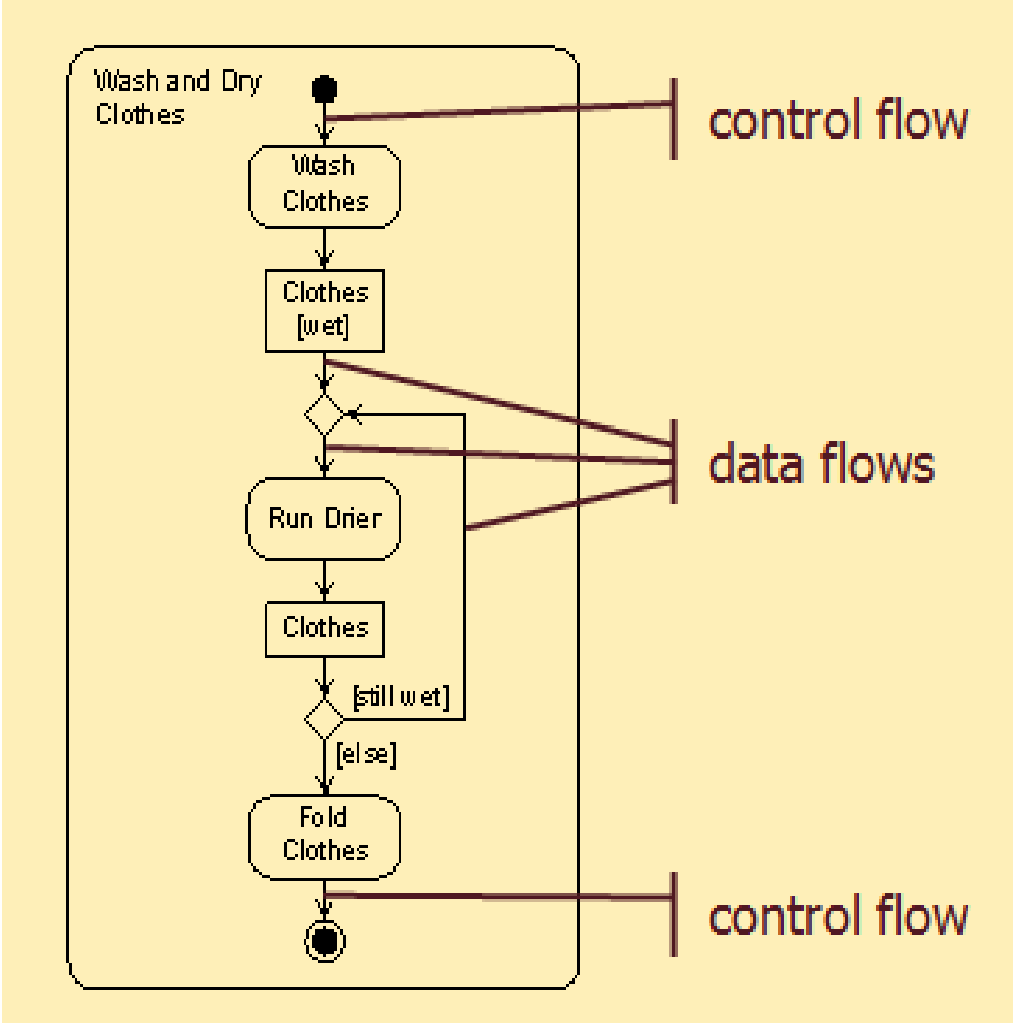
FORKING & JOINING NODES

- A token available on the incoming edge of a fork node is reproduced and made available on all its outgoing edges.
- When tokens are available on every incoming edge of a join node, a token is made available on its outgoing edge.
- Concurrency is modeled of these nodes.



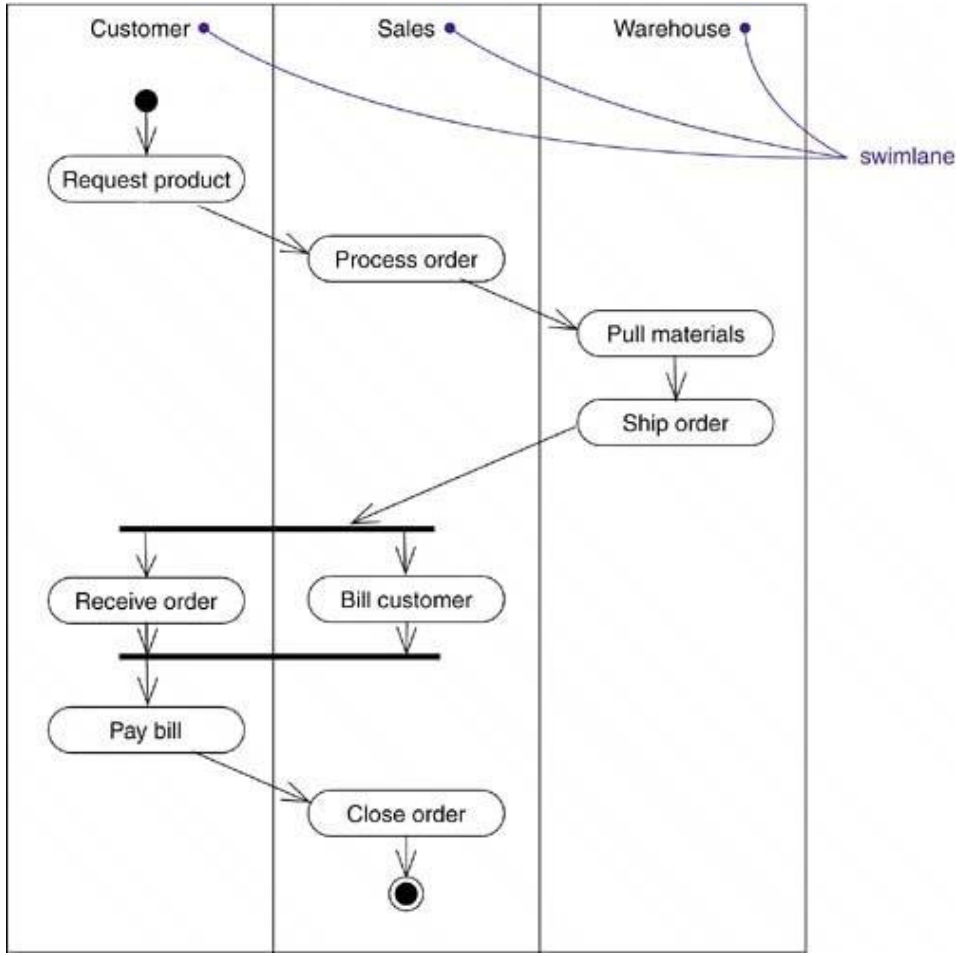
OBJECT NODES

- Data and objects are shown as object nodes (represented by noun and its state in a rectangular box)
- Control tokens do not contain data, data tokens do.
- A control flow is an activity edge that is a channel for control tokens.
- A data flow is an activity edge that is a channel for data tokens.
- Rules for token flow through nodes apply to both control and data tokens, except that data is extracted from consumed tokens and added to produced tokens.



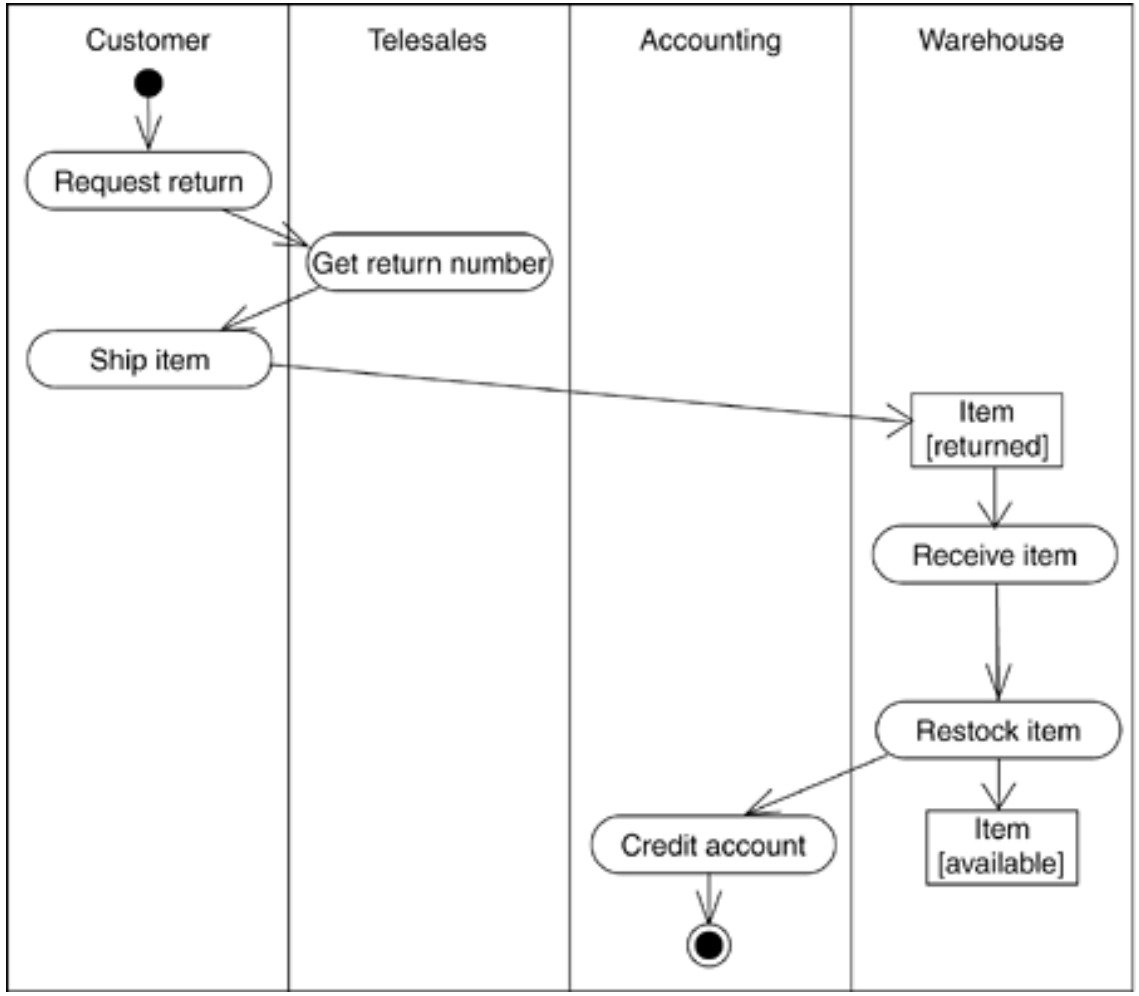
SWIMLANE

- Modeling workflows of business processes, to partition the activity states on an activity diagram into groups, each group representing the business organization responsible for those activities. Each swim lane has a name unique within its diagram. A swim lane really has no deep semantics, except that it may represent some real-world entity, such as an organizational unit of a company.



OBJECT FLOW

- Objects may be involved in the control flow associated with an activity. Usually how the state of object changes is shown with object flow.



ACTIVITY DIAGRAM HEURISTICS

- Flow of control and objects goes down the page and from left to right.
- Name activities and actions with verb phrases.
- Name object nodes with noun phrases.
- Use the [else] guard at every branch.
- **When to Use Activity Diagram**
  - When making a dynamic model of any process.
    - Design processes (what designers do)
    - Designed processes (what designers create)
      - During analysis
      - During resolution (decision)

CASE STUDIES

- **Case 1**

In an online movie ticket booking system a customer books ticket using an interface. After the customer places a request for current movies he waits for the information. While the customer is waiting the server creates a Session object which then requests a list of currently running movies from the movie database. The list is then sent to customer. Customer then selects the desired movie and the Session object retrieves the available timing of the selected movie. Before displaying the time to the customer each scheduled time of the movie is checked in the Booking Database for the availability of the seats. Only the time schedules in which seats are available are displayed to the customer. The customer then selects the number of seats he wants to book. The rate of the ticket is taken from the Movie Database and the due amount is calculated by the Session object. Then the customer enters his credit card detail for the calculated amount which is verified by a credit Card Agent. If the card is verified the session object sends a receipt to the customer, writes in the booking database and update the movie database at the same time. If the card is not verified the customer request is denied. The server finally destroys the session object.

CASE STUDIES

- **Case 2**

In an online airlines ticket booking system, a customer places a request to the system selecting the departure and arrival destinations and also the date of departure. The system then gets all the carrier names and their time schedule from the flight database. Once the carrier names and their time schedules are received the system displays the options of the carriers and the schedule to the customer. If the customer likes any one of the options, he selects the option. But, if the customer doesn't like any option he can go back and select a new date of departure or he can close the application. After selecting an option, the system asks for customer details and credit card information. The system verifies the credit card. If the credit card is valid, the system writes all the booking information in the booking database and sends customer information to the carrier simultaneously. If the verification of the credit card fails, the system sends an error message to the customer and requests for credit card information again. The verification process is done again. If the credit card verification fails for three times the system cancels the booking process and sends a warning message to the credit card agent at the same time. Once the booking is done the system passes the message to the interface and the interface generates a bill and sends it to the customer.

CASE STUDIES

- **Case 3**

In an ATM machine a customer starts a withdrawal transaction by inserting the card. Then he enters the pin, which is verified by the bank. If the pin is incorrect the machine requests for the pin again and the customer enters pin number. The verification repeated for 3 times for an incorrect pin number. If the pin is incorrect even in 4th attempt, the card is seized by the machine and the transaction is closed. If the pin is correct customer enters the amount he wishes to withdraw. The bank then checks the account balance of the customer. If the balance is greater than or equals to the withdrawal amount then money is dispatched through the machine and the information is written in the log concurrently. If the money is taken form the slot within 5 seconds, the account is debited and a transaction receipt is printed simultaneously. If money is not taken in 5 seconds, it is taken back by the machine. Then the balance is shown to the customer. The card and the receipt are then ejected at the same time, and the transaction is completed.

REFERENCES

- Booch, G., Rumbaugh, J. & Jacobson, I. (2005). *The unified modeling language user guide*. Pearson Education India.