CSC291 - Software Engineering Concepts
CSE291 – Introduction to Software Engineering (Fall2020)

Lecture 17&18

Activity Diagrams

Introduction

- Activity diagrams represent the dynamic (behavioural) view of the system.
- Activity diagrams can model a specific use case at a more detailed level.
- Activity diagrams are similar to flow charts: they describe the order of activity
- However, they **differ from traditional flow charts by allowing the representation of concurrent operations**.

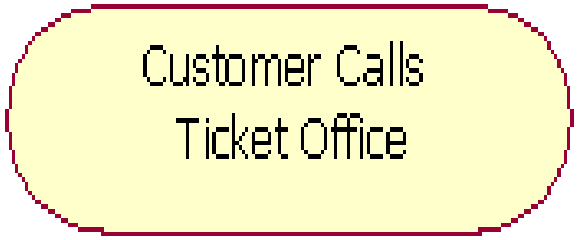
Introduction

- Activity diagrams can be used independently of use cases for modeling a business-level function, such as buying a concert ticket or registering for a college class

- Because it models procedural flow, the activity diagram focuses on the action sequence of execution and the conditions that trigger or guard those actions.

Notation Elements

- Action element in an activity diagram, whose official UML name is *action state.* usually call it either *action or activity.*
- An action is indicated on the activity diagram by a "capsule" shape – a rectangular object with semicircular left and right ends.
- The text inside it indicates the action (e.g., Customer Calls Ticket Office or Registration Office Opens).



Notation Elements

- Activity diagrams show a sequence of actions, they must indicate the **starting point** of the sequence.

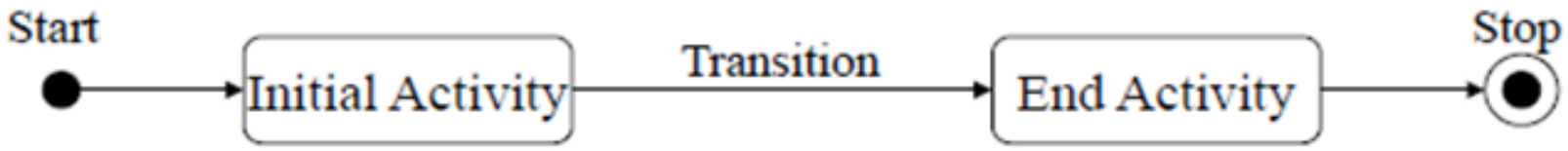
- The official UML name for the starting point on the activity diagram is ***initial state***, *and it is the point at which* you begin reading the action sequence.

- The initial state is drawn as a **solid circle** with a transition line (arrow) that connects it to the first action in the activity's sequence of actions.

Initial State 

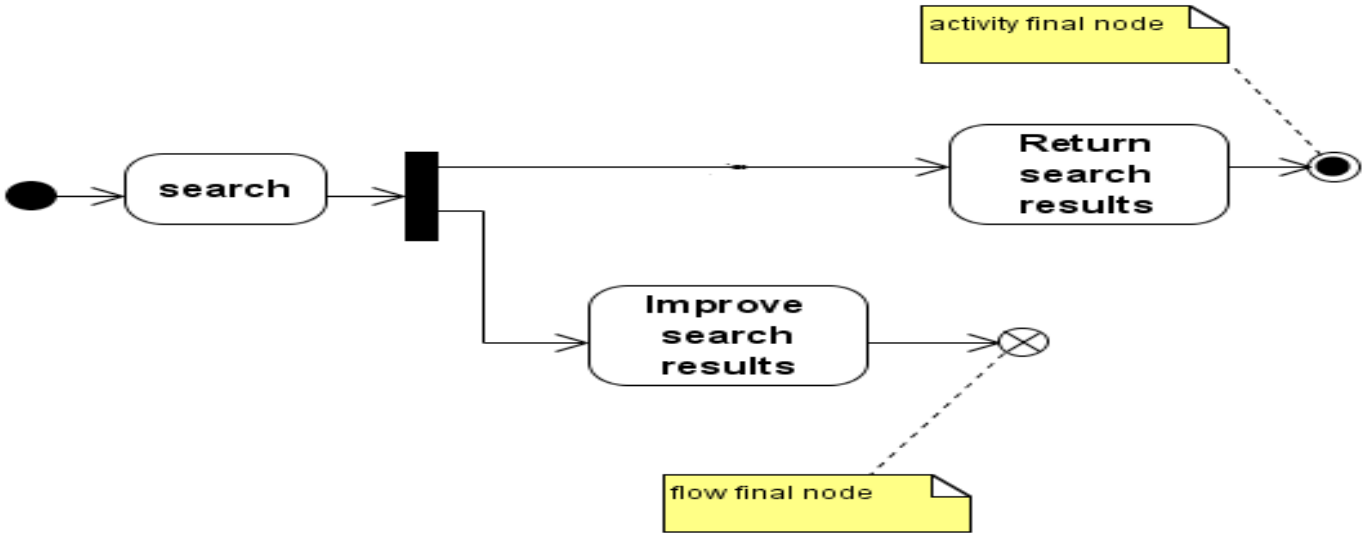
Notation Elements

- With **arrows** indicating direction, the transition lines on an activity diagram show the sequential flow of actions in the modeled activity.
- The arrow will always point to the next action in the activity's sequence.
- The activity's flow terminates when the transition line of the last action in the sequence connects to a **"final state"** symbol, which is a **bulls eye** (a circle surrounding a smaller solid circle).



Notation Elements

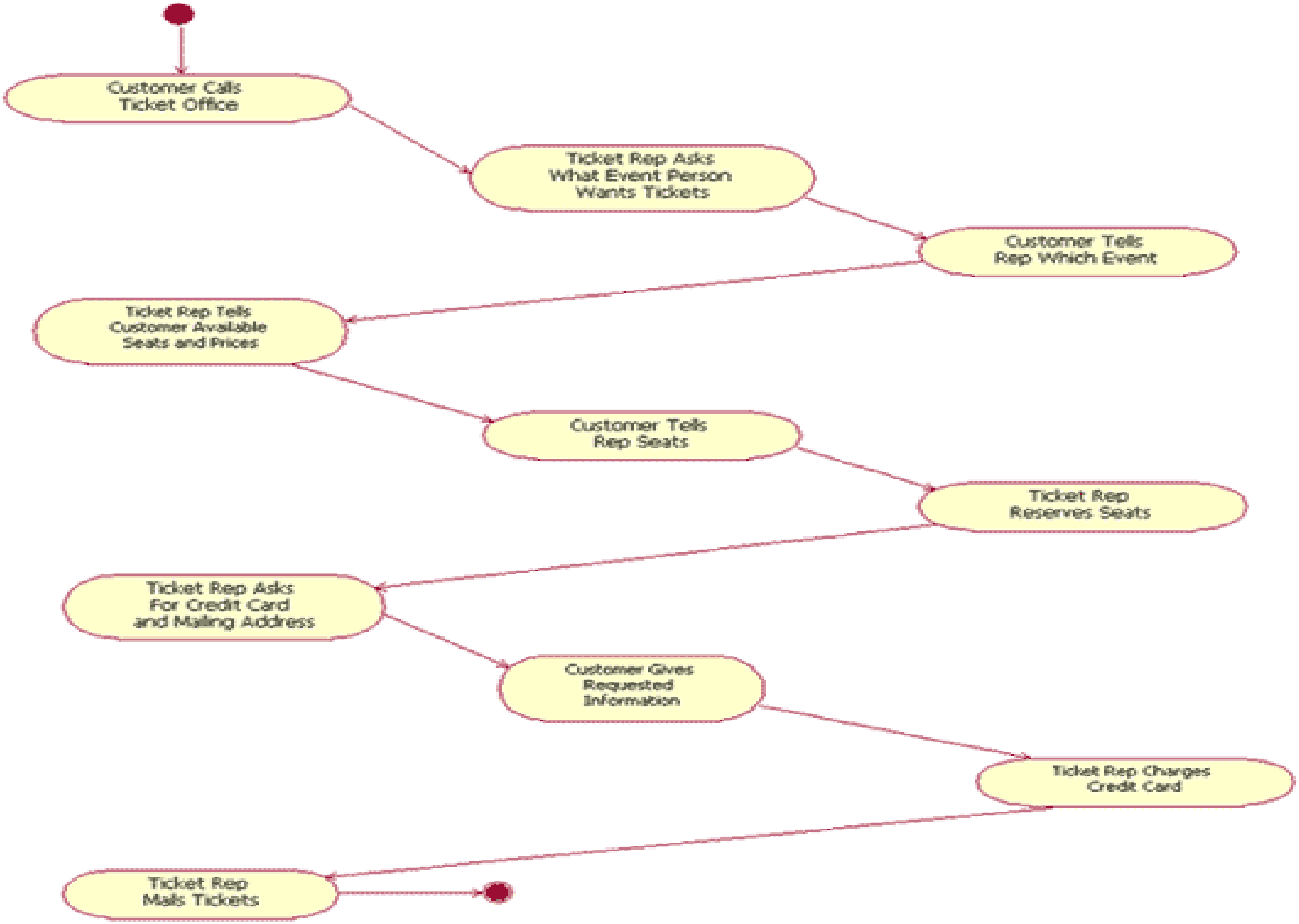
- There are two types of Final nodes: **Activity Final** and **Flow final**
- An activity final node terminates the entire activity
- A flow final node terminates a path through an activity, but not the entire activity



Example – Booking a ticket

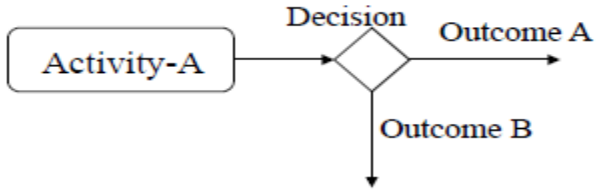
The activity "Booking a Concert Ticket," with actions in the following order:

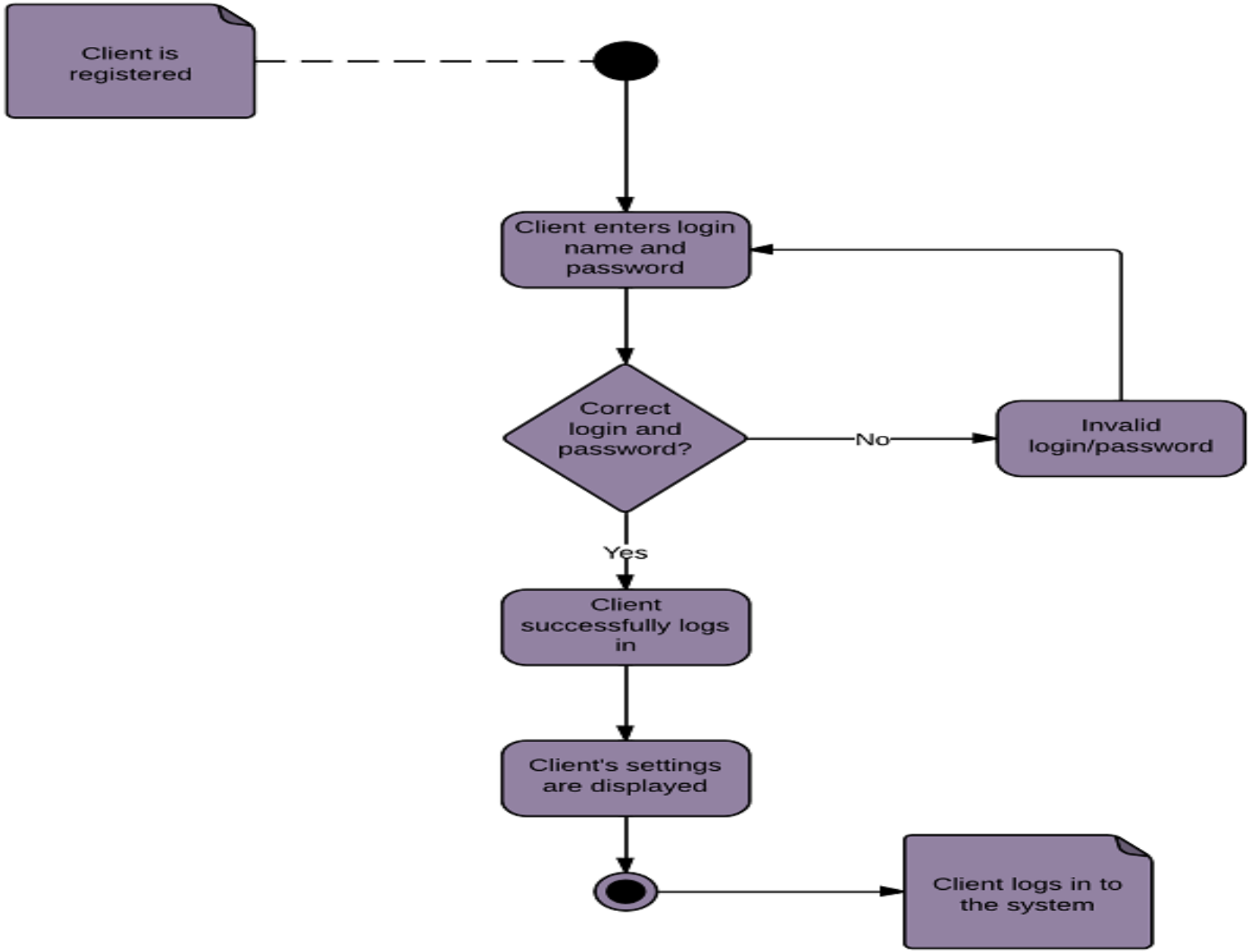
1. Customer calls ticket office.
2. Ticket representative asks what event person wants tickets for.
3. Customer tells rep event choice.
4. Ticket rep tells customer available seats and prices.
5. Customer tells rep seating choice.
6. Ticket rep reserves seats.
7. Ticket rep asks for credit card and billing address.
8. Customer gives requested information.
9. Ticket rep charges credit card.
10. Ticket rep mails tickets.


Customer Calls Ticket Office
Ticket Rep Asks What Event Person Wants Tickets
Customer Tells Rep Which Event
Ticket Rep Tells Customer Available Seats and Prices
Customer Tells Rep Seats
Ticket Rep Reserves Seats
Ticket Rep Asks For Credit Card and Mailing Address
Customer Gives Requested Information
Ticket Rep Charges Credit Card
Ticket Rep Mails Tickets

Decision Points

- Activities modeled for real software development projects include **decision points** that control what actions take place.
- Each transition line involved in a decision point must be labeled with text above it to indicate **"guard conditions,"** commonly abbreviated as ***guards.***
- Guard condition text is always placed in brackets -- for example, [guard condition text].
- A guard condition explicitly tells when to follow a transition line to the next action.




Client is registered
Client enters login name and password
Correct login and password?
No
Invalid login/password
Yes
Client successfully logs in
Client's settings are displayed
Client logs in to the system

Components in an Activity Diagram

Fork

- A black bar (horizontal/vertical) with one flow going into it and several leaving it. This denotes the beginning of parallel activities

Join

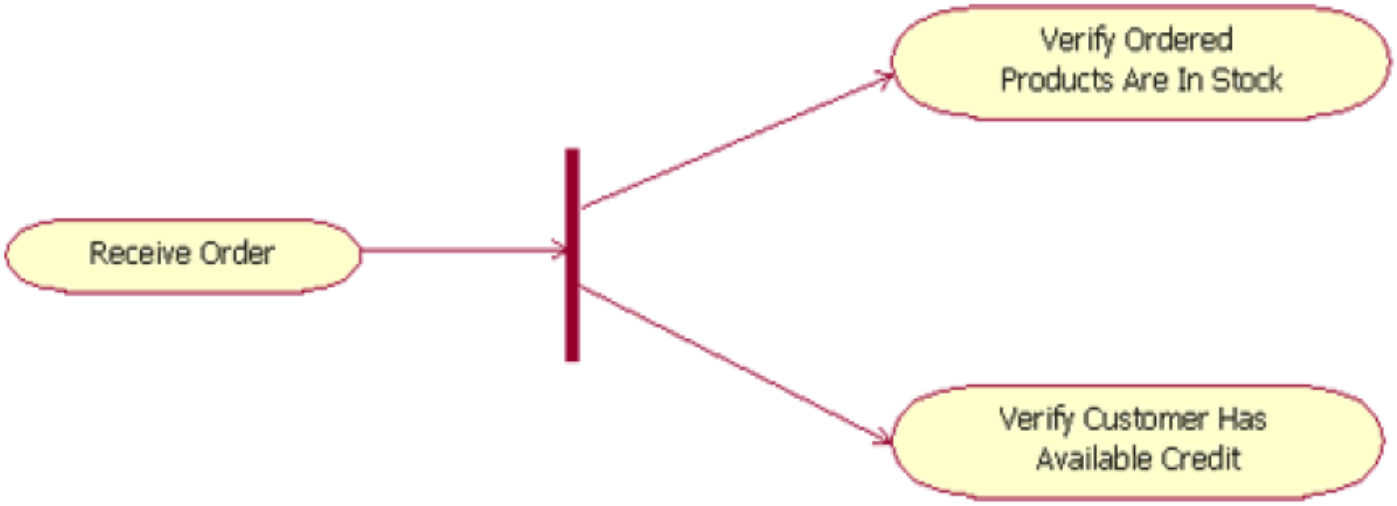
- A black bar with several flows entering it and one leaving it. this denotes the end of parallel activities.

Merge

- A diamond icon with several flows entering and one leaving. **Merge** should not be used to synchronize concurrent flows

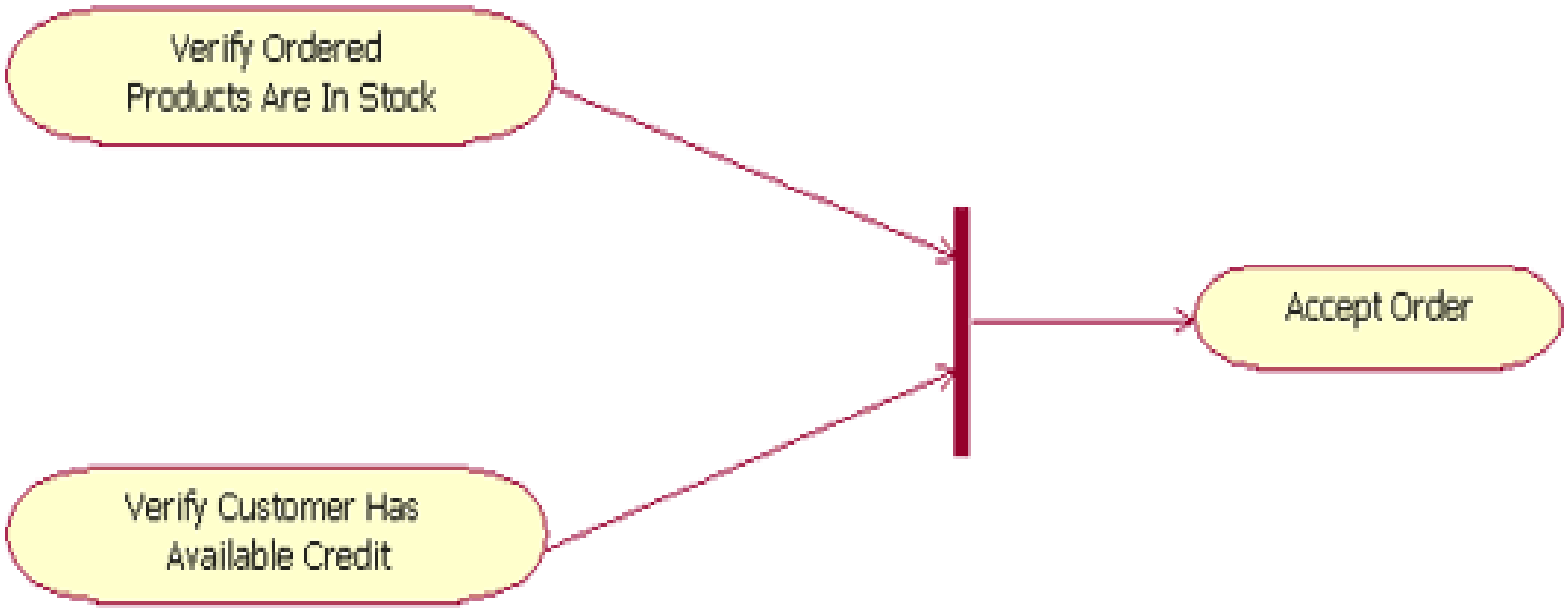
Example: Fork

- In the example , after the action "Receive Order" is completed, two threads are kicked off in parallel.
- This allows the system to process both the "Verify Ordered Products Are In Stock" and the "Verify Customer Has Available Credit" actions at the same time.

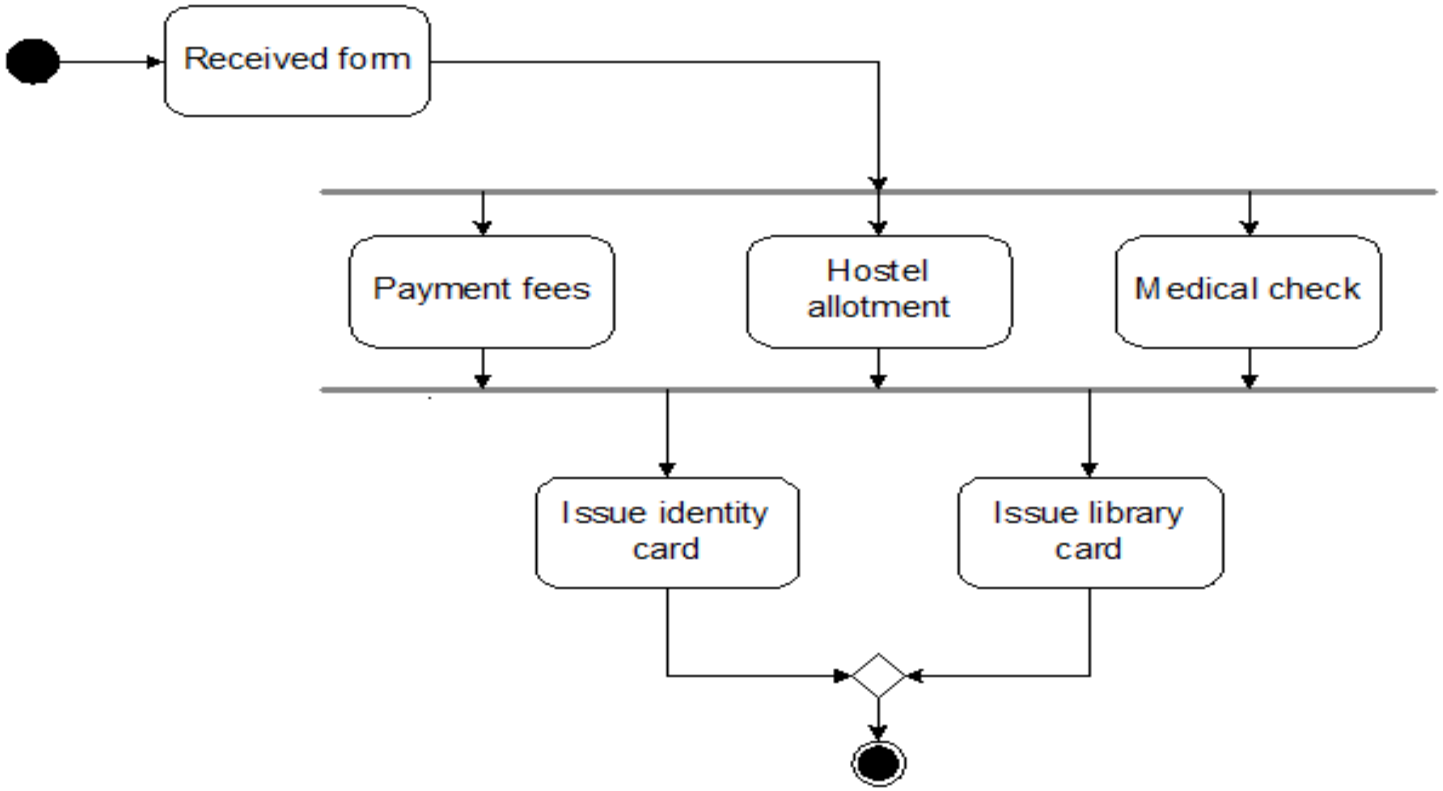


Example: Join

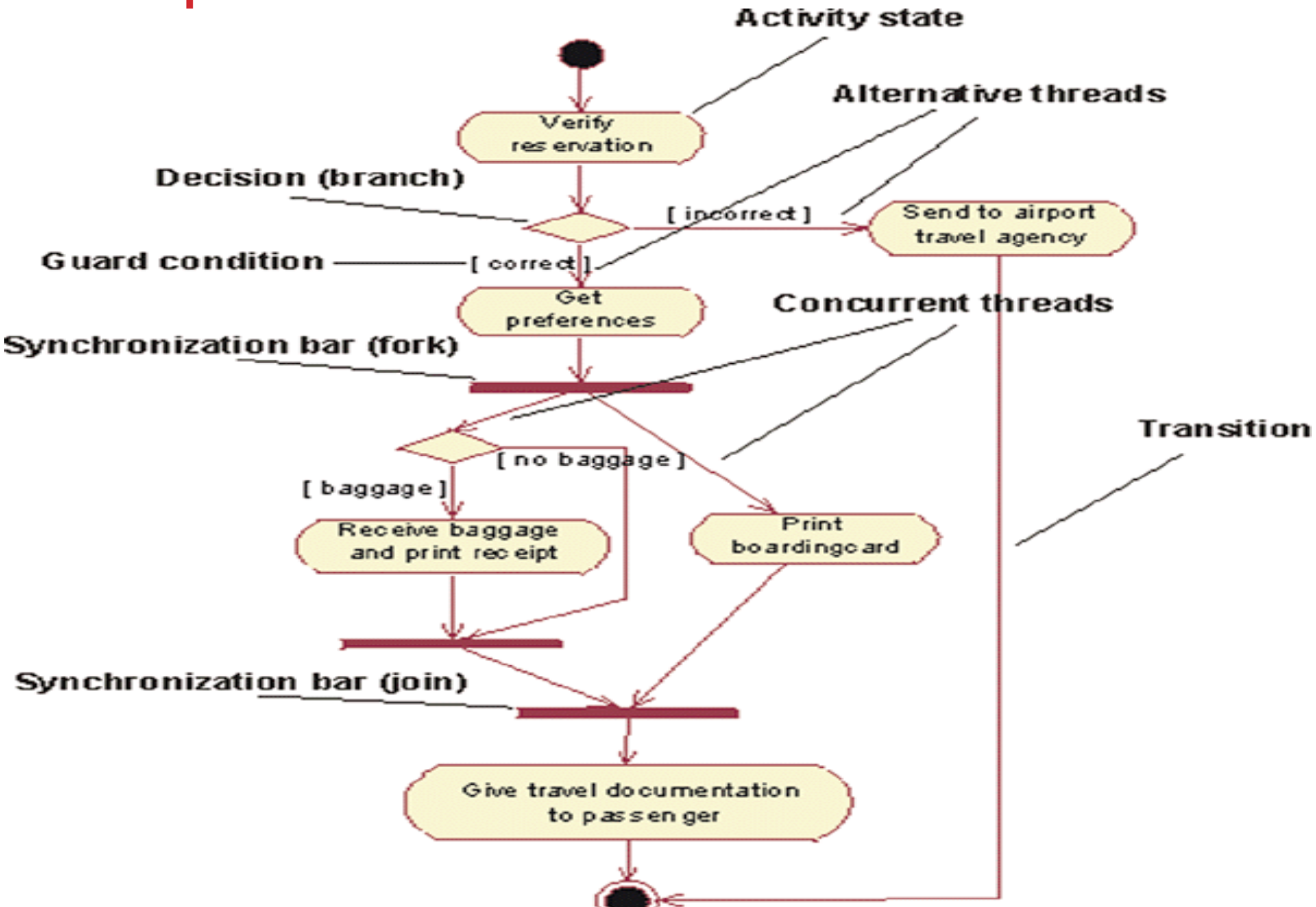
- When parallel action sequences terminate, a synch state (thick line) is used to indicate that the multiple threads are joined back into a single thread.



Example: Merge



Example


Activity state
Alternative threads
Verify reservation
Decision (branch)
[incorrect]
Send to airport travel agency
Guard condition
[correct]
Get preferences
Concurrent threads
Synchronization bar (fork)
Transition
[no baggage]
[baggage]
Receive baggage and print receipt
Print boardingcard
Synchronization bar (join)
Give travel documentation to passenger

Practice Case Study

- The case study is about the process of reserving a flight. First, you enter the dates. Once you submit your desired flight plan, you can enter your personal information and at the same time the system could be searching availability. The system flow then joins back into one and you can select the specific flight on the dates you want to fly. After this in the system there are two different paths dependent on whether you are using reward points. In case of reward points you have to enter points and at the same time the system holds your reservation. After that payment information is entered, the system performs two processes at the same time. One is the system marks seat as taken and the other is the system processes payment. The system after that sends out a confirmation email.

Chapter Reading

- ***Chapter 5*, System Modeling,**
  Software Engineering by Ian Sommerville

Course Material Link:
https://sites.google.com/a/cuilahore.edu.pk/se/home/lectures