CS6001 – OOAD

Module 6

State machine diagram and Modeling –State Diagram -Activity diagram

Dr.S.Neelavathy Pari
Assistant Professor (Sr. Grade)
Department of Computer Technology
Anna University, MIT Campus

- The dynamic model represents the time–dependent aspects of a system.
- It is concerned with the temporal changes in the states of the objects in a system.
- The main concepts are −
  - State, which is the situation at a particular condition during the lifetime of an object.
  - Transition, a change in the state
  - Event, an occurrence that triggers transitions
  - Action, an uninterrupted and atomic computation that occurs due to some event, and
  - Concurrency of transitions.

States and State Transitions

The state is an abstraction given by the values of the attributes that the object has at a particular time period. It is a situation occurring for a finite time period in the lifetime of an object, in which it fulfils certain conditions, performs certain activities, or waits for certain events to occur. In state transition diagrams, a state is represented by rounded rectangles.

Parts of a state

- **Name** – A string differentiates one state from another. A state may not have any name.
- **Entry/Exit Actions** – It denotes the activities performed on entering and on exiting the state.
- **Internal Transitions** – The changes within a state that do not cause a change in the state.
- **Sub–states** – States within states.

Initial and Final States

The default starting state of an object is called its initial state. The final state indicates the completion of execution of the state machine. The initial and the final states are pseudo-states, and may not have the parts of a regular state except name. In state transition diagrams, the initial state is represented by a filled black circle. The final state is represented by a filled black circle encircled within another unfilled black circle.

States and State Transitions

Transition

A transition denotes a change in the state of an object. If an object is in a certain state when an event occurs, the object may perform certain activities subject to specified conditions and change the state. In this case, a state–transition is said to have occurred. The transition gives the relationship between the first state and the new state. A transition is graphically represented by a solid directed arc from the source state to the destination state.

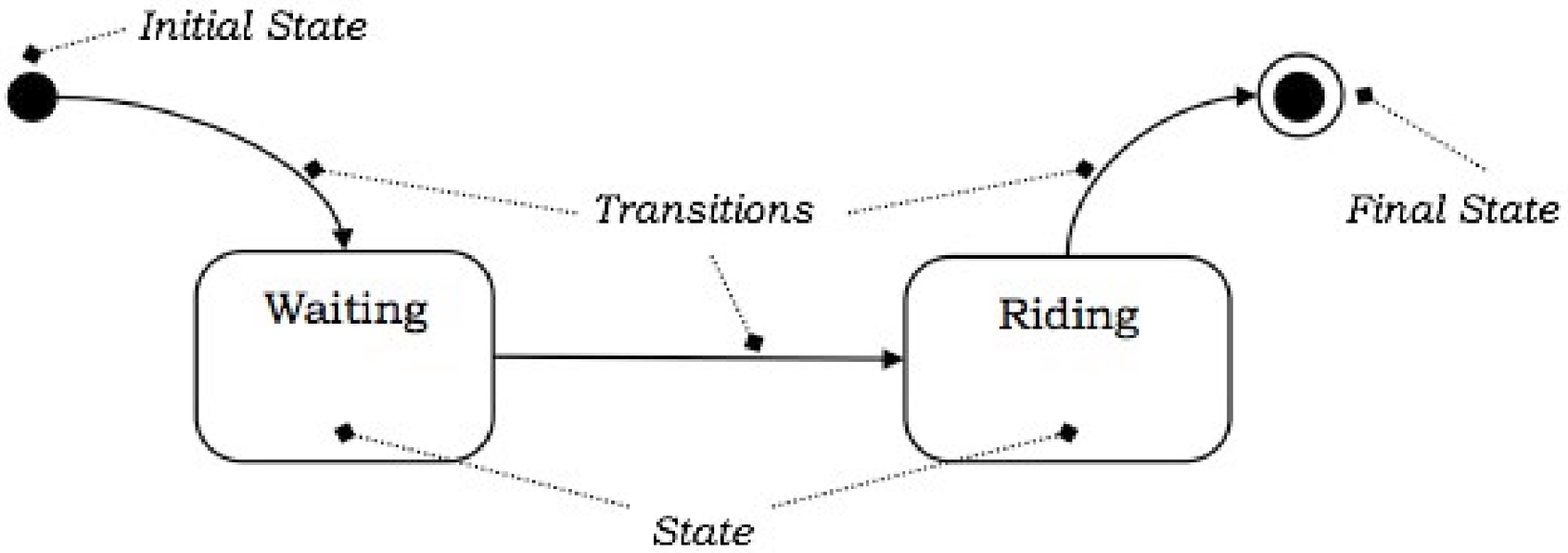
The five parts of a transition are –

- **Source State** – The state affected by the transition.
- **Event Trigger** – The occurrence due to which an object in the source state undergoes a transition if the guard condition is satisfied.
- **Guard Condition** – A Boolean expression which if True, causes a transition on receiving the event trigger.
- **Action** – An un-interruptible and atomic computation that occurs on the source object due to some event.
- **Target State** – The destination state after completion of transition.

States and State Transitions

Example

Suppose a person is taking a taxi from place X to place Y. The states of the person may be: Waiting (waiting for taxi), Riding (he has got a taxi and is travelling in it), and Reached (he has reached the destination). The following figure depicts the state transition.



Events

Events are some occurrences that can trigger state transition of an object or a group of objects. Events have a location in time and space but do not have a time period associated with it. Events are generally associated with some actions.

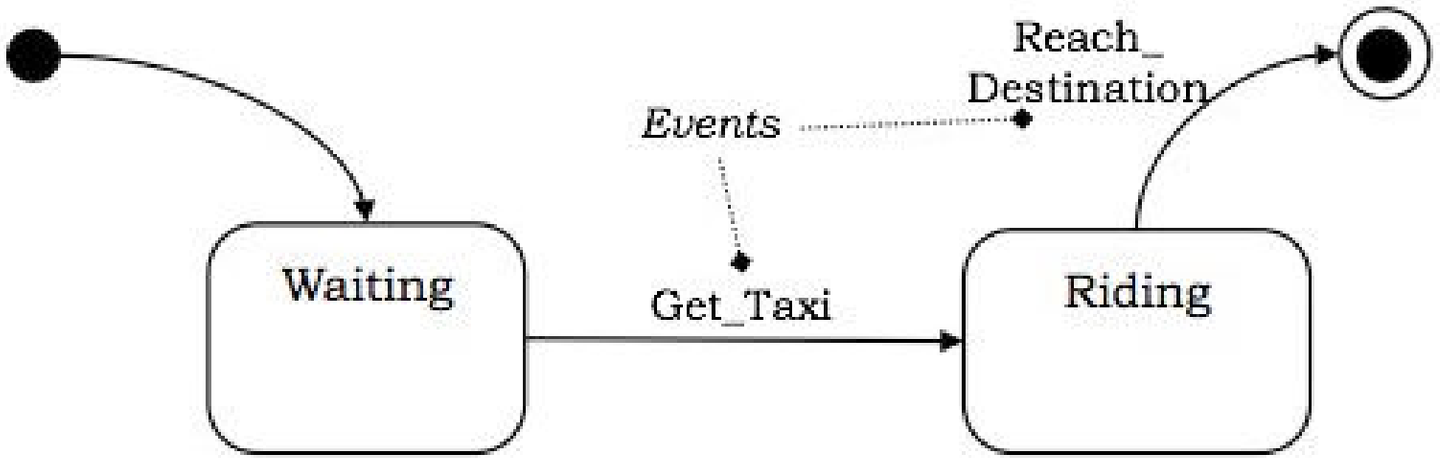
Examples of events are mouse click, key press, an interrupt, stack overflow, etc.

Events that trigger transitions are written alongside the arc of transition in state diagrams.

Events

Example

Considering the example shown in the above figure, the transition from Waiting state to Riding state takes place when the person gets a taxi. Likewise, the final state is reached, when he reaches the destination. These two occurrences can be termed as events Get_Taxi and Reach_Destination. The following figure shows the events in a state machine.



Events

External and Internal Events

External events are those events that pass from a user of the system to the objects within the system. For example, mouse click or key–press by the user are external events.

Internal events are those that pass from one object to another object within a system. For example, stack overflow, a divide error, etc.

Deferred Events

Deferred events are those which are not immediately handled by the object in the current state but are lined up in a queue so that they can be handled by the object in some other state at a later time.

Event Classes

Event class indicates a group of events with common structure and behavior. As with classes of objects, event classes may also be organized in a hierarchical structure. Event classes may have attributes associated with them, time being an implicit attribute. For example, we can consider the events of departure of a flight of an airline, which we can group into the following class –

Flight_Departs (Flight_No, From_City, To_City, Route)

Concurrency of Events

System Concurrency

Here, concurrency is modelled in the system level. The overall system is modelled as the aggregation of state machines, where each state machine executes concurrently with others.

Concurrency within an Object

Here, an object can issue concurrent events. An object may have states that are composed of sub-states, and concurrent events may occur in each of the sub-states.

Concepts related to concurrency within an object are as follows –

Simple and Composite States

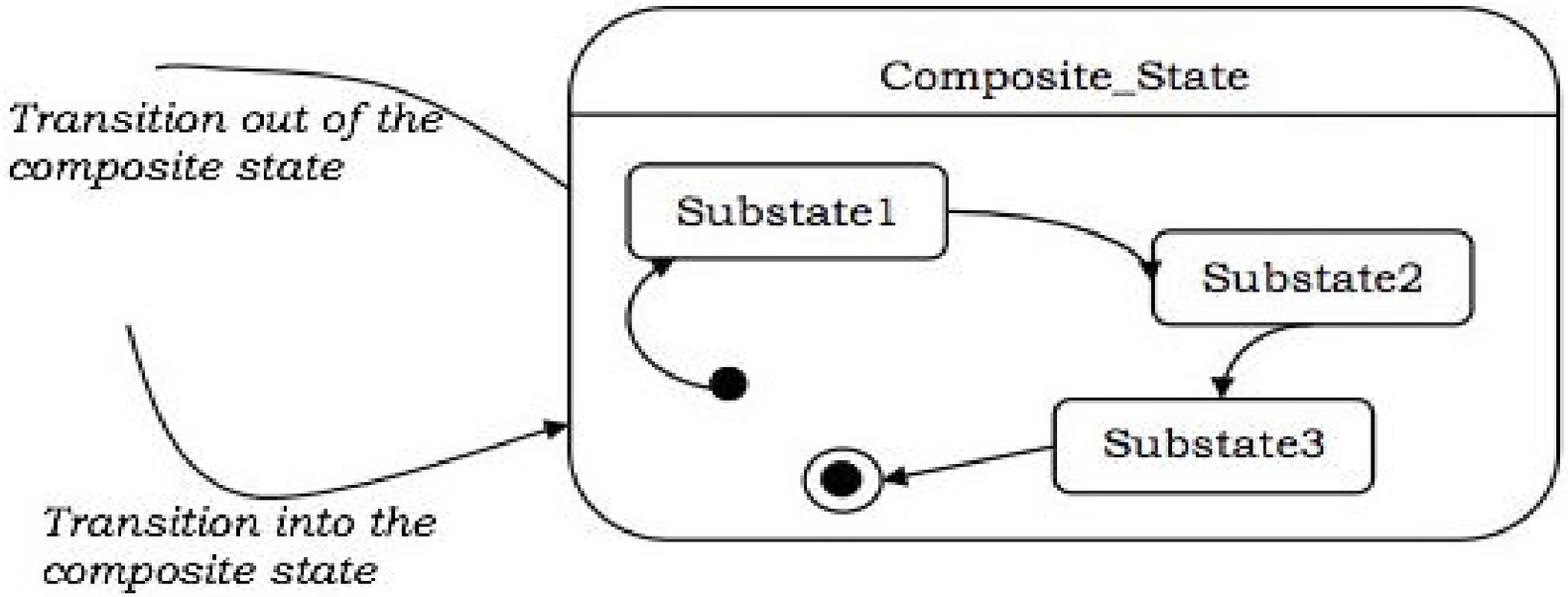
A simple state has no sub-structure. A state that has simpler states nested inside it is called a composite state. A sub-state is a state that is nested inside another state. It is generally used to reduce the complexity of a state machine. Sub-states can be nested to any number of levels.

Composite states may have either sequential sub-states or concurrent sub-states.

Sequential Sub-states

In sequential sub-states, the control of execution passes from one sub-state to another sub-state one after another in a sequential manner. There is at most one initial state and one final state in these state machines.

The following figure illustrates the concept of sequential sub-states.

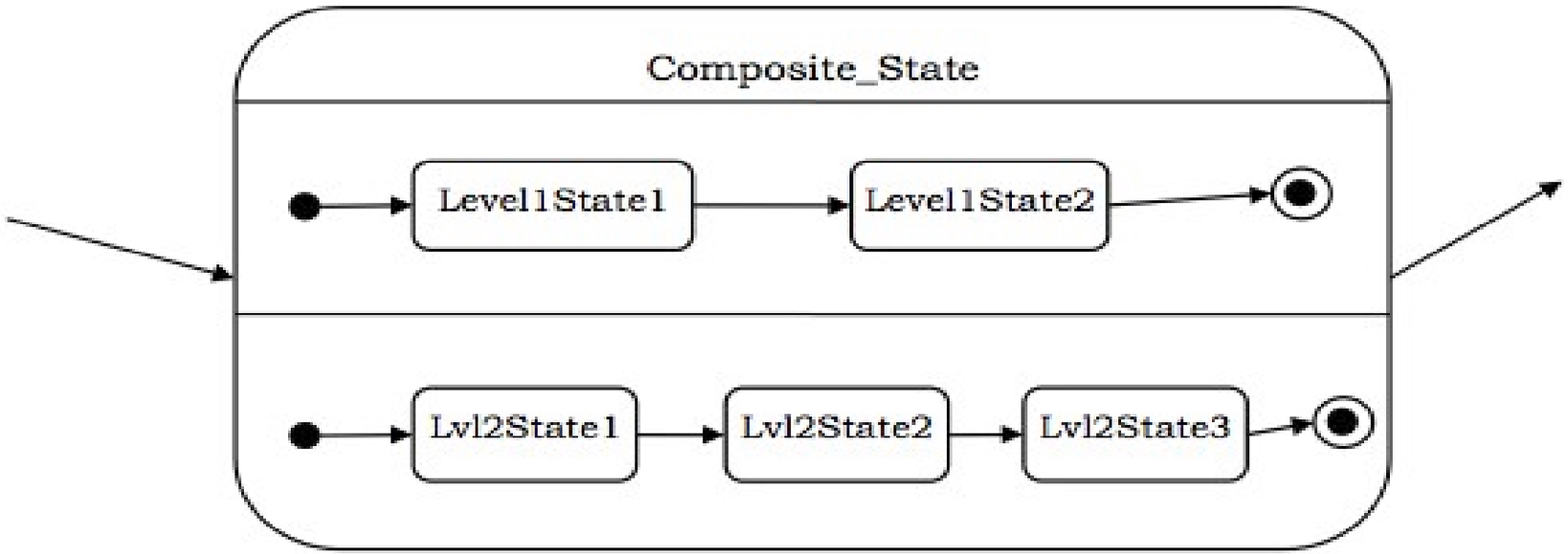


Concurrent Sub-states

Concurrent Sub-states

In concurrent sub-states, the sub-states execute in parallel, or in other words, each state has concurrently executing state machines within it. Each of the state machines has its own initial and final states. If one concurrent sub-state reaches its final state before the other, control waits at its final state. When all the nested state machines reach their final states, the sub-states join back to a single flow.

The following figure shows the concept of concurrent sub-states.



Activity Diagram

- Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

Purpose of Activity Diagrams

- Activity is a particular operation of the system.
- Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques.
- The only missing thing in the activity diagram is the message part.

How to Draw an Activity Diagram?

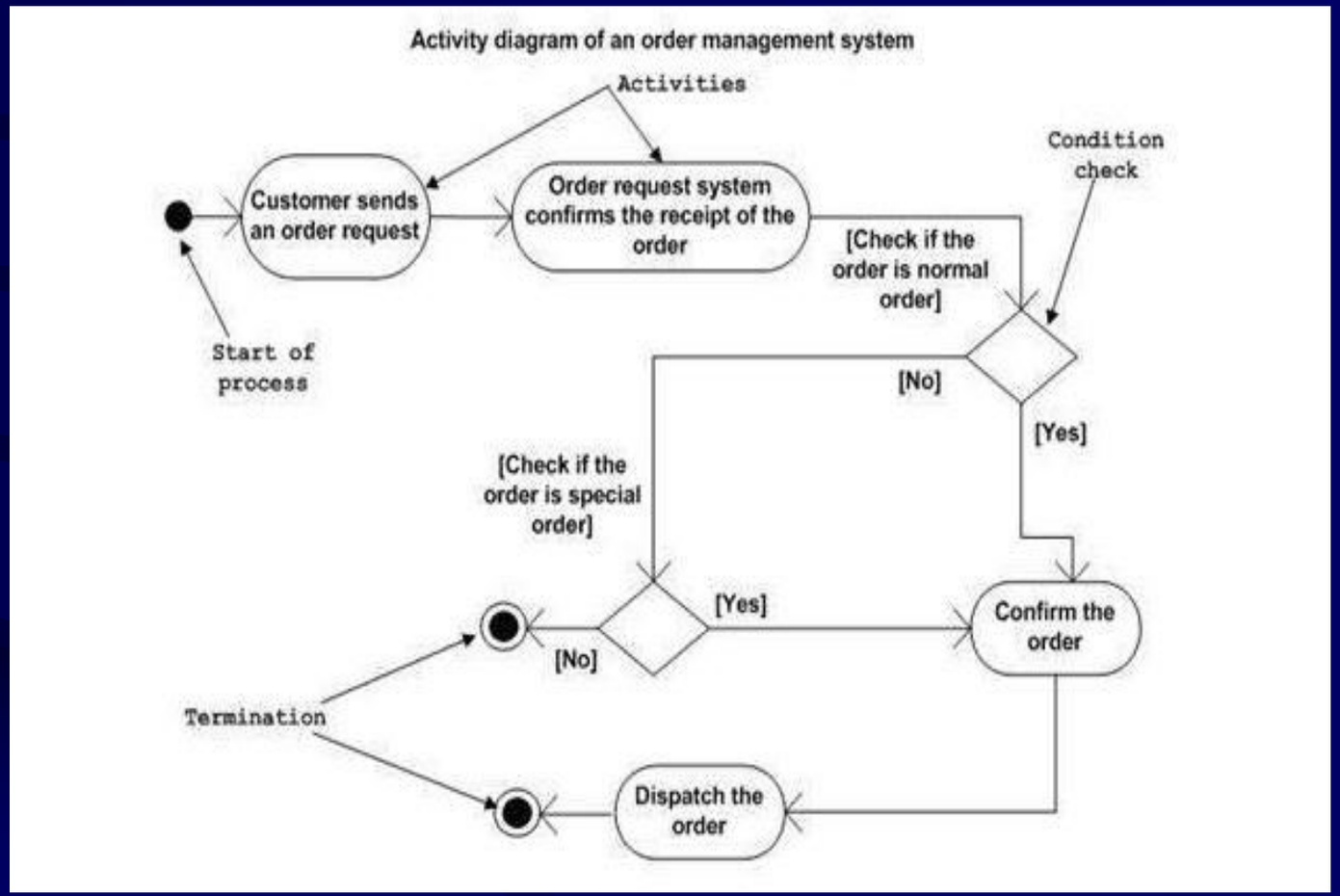
- Activity diagrams are mainly used as a flowchart that consists of activities performed by the system.
- Activity diagrams are not exactly flowcharts as they have some additional capabilities.
- These additional capabilities include branching, parallel flow, swimlane, etc.

Elements used in the activity diagram

Activities , Association, Conditions , Constraints

Example :activity diagram order management system.

- Following diagram is drawn with the four main activities
  - Send order by the customer
  - Receipt of the order
  - Confirm the order
  - Dispatch the order

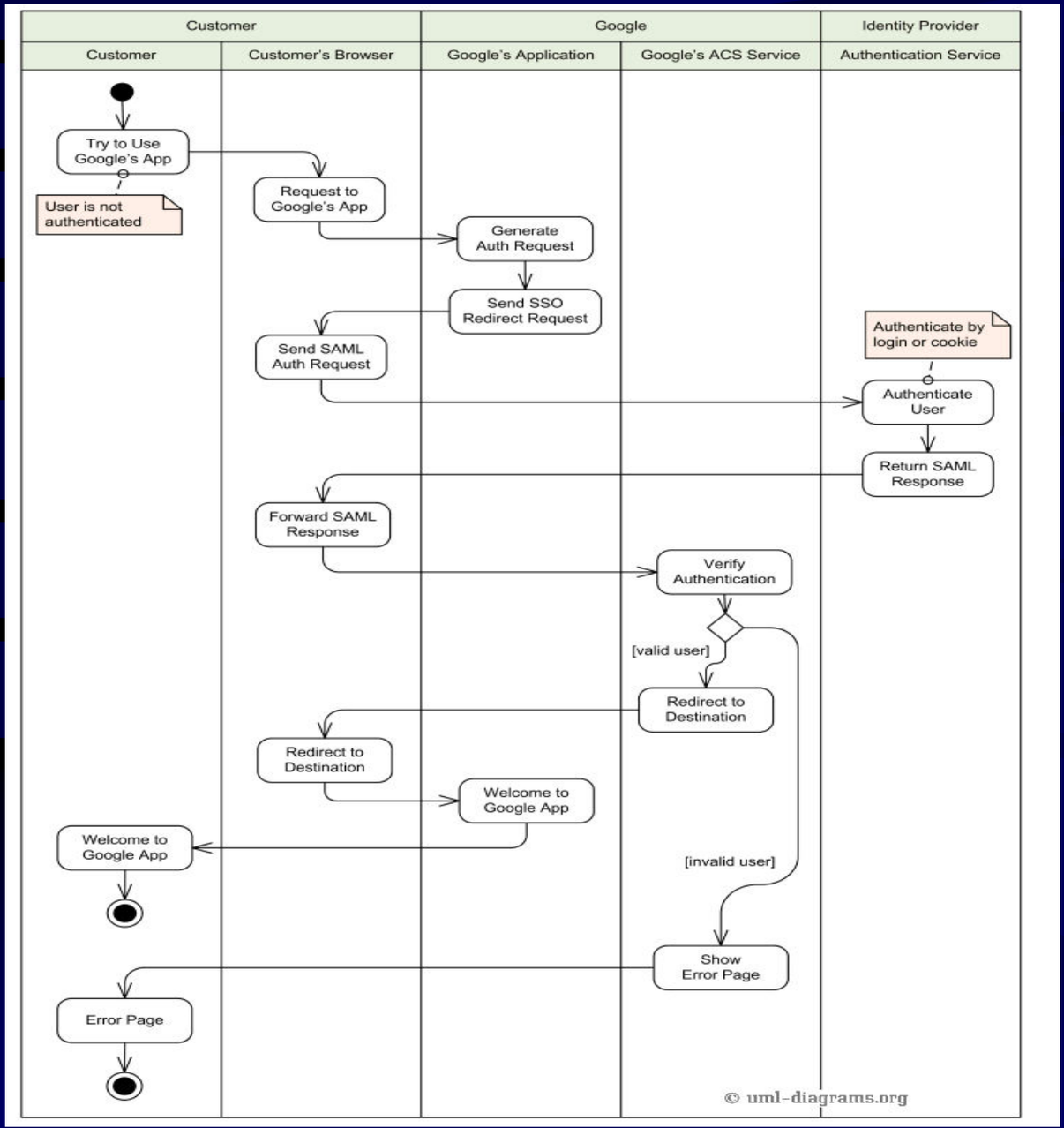


Where to Use Activity Diagrams?

- Activity diagram can be used for −
  - Modeling work flow by using activities.
  - Modeling business requirements.
  - High level understanding of the system's functionalities.
  - Investigating business requirements at a later stage.

Single Sign-On (SSO) for Google Apps

- To interact with partner companies Google uses single sign-on based on OASIS **SAML 2.0** protocol. Google acts as service provider with services such as Gmail or Start Pages.
- Partner companies act as identity providers and control user names, passwords, and other information used to identify, authenticate and authorize users for web applications that Google hosts.
- Each partner provides Google with the URL of its SSO service as well as the public key that Google will use to verify SAML responses.



- When a user attempts to use some hosted Google application, such as Gmail, Google generates a SAML authentication request and sends redirect request back to the user's browser. Redirect points to the specific identity provider. SAML authentication request contains the encoded URL of the Google application that the user is trying to reach.
- The partner identity provider authenticates the user by either asking for valid login credentials or by checking for its own valid authentication cookies. The partner generates a SAML response and digitally signs it. The response is forwarded to **Google's Assertion Consumer Service** (ACS).
- Google's ACS verifies the SAML response using the partner's public key. If the response is valid and user identity was confirmed by identity provider, ACS redirects the user to the destination URL. Otherwise user will see error message.