Experiment No. 6

Title: Modelling Behaviour - Activity and State Chart Diagram using UML

Batch: B3 **Roll No.: 1714103** **Experiment: 6**

Aim: To model Behaviour- Activity and State Chart Diagram using UML.

Resources needed: IBM Rational Rose/Open Source UML Tool

Theory

Behavior diagrams show the dynamic behavior of the objects in a system, which can bedescribed as a series of changes to the system over time.

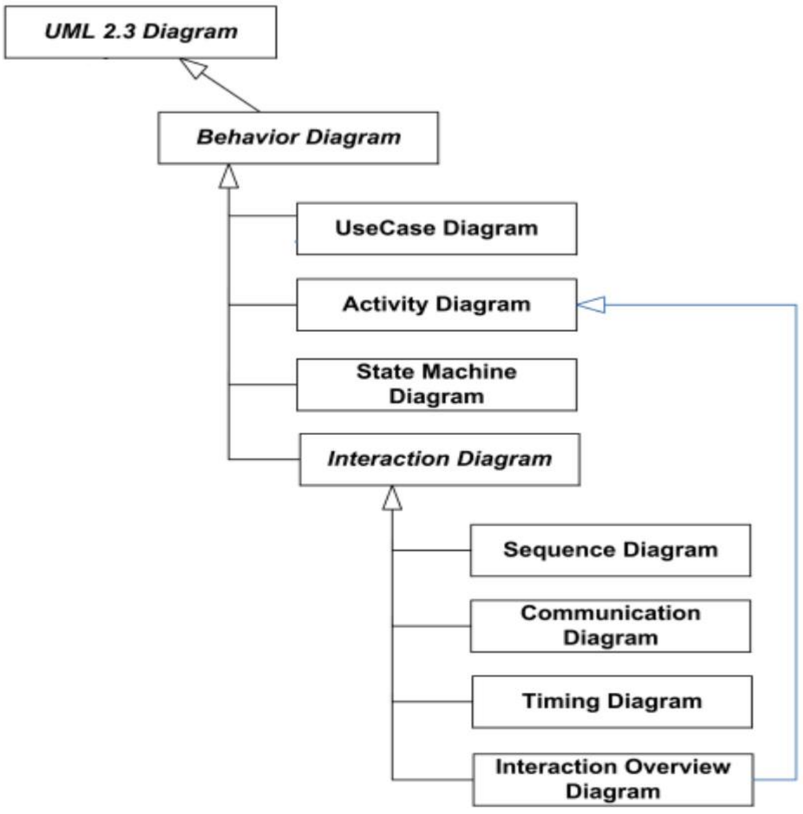


Figure 5.1 UML Behaviour Diagram

UML State machine diagram and activity diagram are both behavioural diagrams but have different emphases.

Activity diagram shows sequence and conditions for coordinating lower-level behaviors,rather than which classifiers own those behaviors. These are commonly called control flow and object flow models.

State machine diagram is used for modelling discrete behavior through finite statetransitions. In addition to expressing the behavior of a part of the system, state machines can

also be used to express the usage protocol of part of a system. These two kinds of state machines are referred to as behavioral state machines and protocol state machines.

Activity and State chart Diagrams:

State Chart Diagrams

In case of Object Oriented Analysis and Design, a system is often abstracted by a (or more) class with some well-defined behaviour and states. A state chart diagram is a pictorial representation of such a system, with all its states, and different events that lead transition from one state to another. To illustrate this, consider a computer. Some possible states that it could have are: running, shutdown, hibernate. A transition from running state to shutdown state occur when user presses the "Power off" switch, or clicks on the "Shut down" button as displayed by the OS. Here, clicking on the shutdown button, or pressing the power off switch act as external events causing the transition.

State

A state is any "distinct" stage that an object (system) passes through in its lifetime. An object remains in a given state for finite time until "something" happens, which makes it to move to another state.

There are three types of state:

Initial: The state in which an object remain when created

Final: The state from which an object do not move to any other state [optional]

Intermediate: Any state, which is neither initial, nor final

An initial state is represented by a circle filled with black. An intermediate state is depicted by a rectangle with rounded corners. A final state is represented by a unfilled circle with an inner black-filled circle.

Intermediate states usually have two compartments, separated by a horizontal line, called the name **compartment and internal transitions compartment.** They are described below:
Name compartment: Contains the name of the state, which is a short, simple, descriptive string

Internal transitions compartment: Contains a list of internal activities performed as long as the system is in this state.

The internal activities are indicated using the following syntax: **action-label / action-expression**. Action labels could be any condition indicator. There are, however, four specialaction labels:

Entry: Indicates activity performed when the system enter this state

Exit: Indicates activity performed when the system exits this state

Do: indicate any activity that is performed while the system remain in this state oruntil the action expression results in a completed computation **Include:** Indicates invocation of a sub-machine

Any other action label identifies the event (internal transition) as a result of which the corresponding action is triggered. Internal transition is almost similar to self transition, except that the former doesn't result in execution of entry and exit actions. That is, system doesn't exit or re-enter that state. Figure-3.5 shows the syntax for representing a typical (intermediate) state.

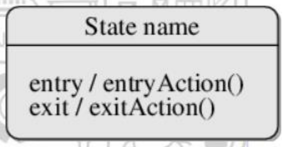


Figure-5.2: A typical state in a state chart diagram

States could again be either simple or composite (a state containing other states).

Transition

Transition is movement from one state to another state in response to an external stimulus (or any internal event). A transition is represented by a solid arrow from the current state to the next state. It is labelled by: event [guard-condition]/[action-expression], where

- Event is the what is causing the concerned transition (mandatory) -- Written in past tense
- Guard-condition is (are) precondition(s), which must be true for the transition to happen [optional]
- Action-expression indicate action(s) to be performed as a result of the transition [optional]

It may be noted that if a transition is triggered with one or more guard-condition(s), which evaluate to false, the system will continue to stay in the present state. Also, not all transitions do result in a state change. For example, if a queue is full, any further attempt to append will fail until the delete method is invoked at least once. Thus, state of the queue doesn't change in this duration.

Action

An action represents behaviour of the system. While the system is performing any action for the current event, it doesn't accept or process any new event. The order, in which different actions are executed, is given below:

- Exit actions of the present state
- Actions specified for the transition
- Entry actions of the next state

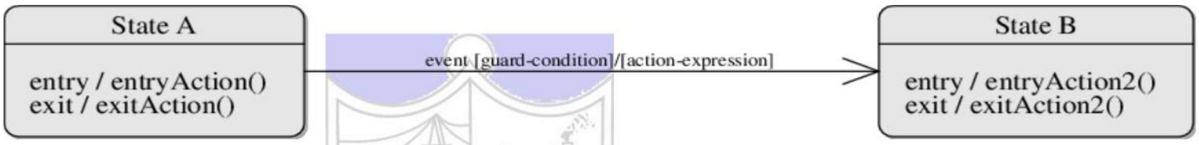


Figure-5.3 shows a typical state chart diagram with all its syntaxes.

Activity Diagram

Activity diagram is used for business process modelling, for modelling the logic captured by a single use case or usage scenario, or for modelling the detailed logic of a business rule. Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. The easiest way to visualize an activity diagram is to think of a flowchart and data flow diagram (DFDs)

Need of an Activity Diagram:

The general purpose of activity diagrams is to focus on flows driven by internal processing Vs external events. Activity diagrams are also useful for, analysing a use case by describing what actions needs to take place and when they should occur describing a complicated sequential algorithm and modelling applications with parallel processes. Elements of activity diagram:

Initial Activity: This shows the starting point or first activity of the flow and denoted by asolid circle. There can only be one initial state on a diagram. **Activity** : Activity represented by rectangle with rounded edges.

Transition : When an activity state is completed, processing moves to another activity state.Transitions are used to mark this movement. Transitions are modelled using arrows.
Decisions: Similar to flowcharts, a logic where a decision is to be made is depicted by adiamond, with the options written on either side of the arrows emerging from the diamond, within box brackets.

Synchronization Bar :

Activities often can be done in parallel. To split processing ("Fork") or to resume processing when multiple activities have been completed ("Join"), synchronization bars are used. These are modelled as solid rectangles, with multiple transitions going in and/ or out. Fork denotes the beginning of parallel activity. Join denotes the end of parallel processing.

Final Activity :

The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity. An activity diagram can have zero or more activity final nodes.

Swim Lanes :

Activity diagrams provide another ability, to clarify which actor performs which activity. If you wish to distinguish in an activity diagram the activities carried out by individual actors, vertical columns are first made, separated by thick vertical black lines, termed swim lanes and name each of these columns with the name of the actor involved. You place each of the activities below the actor performing these activities and then show how these activities are connected.

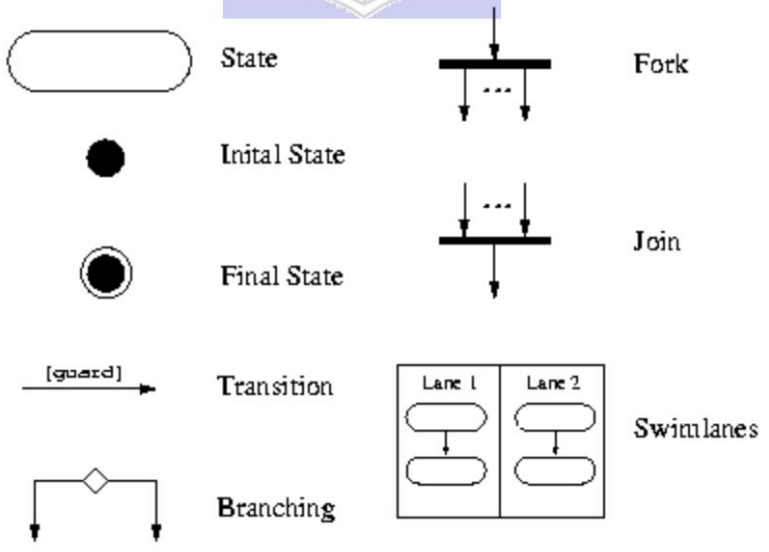


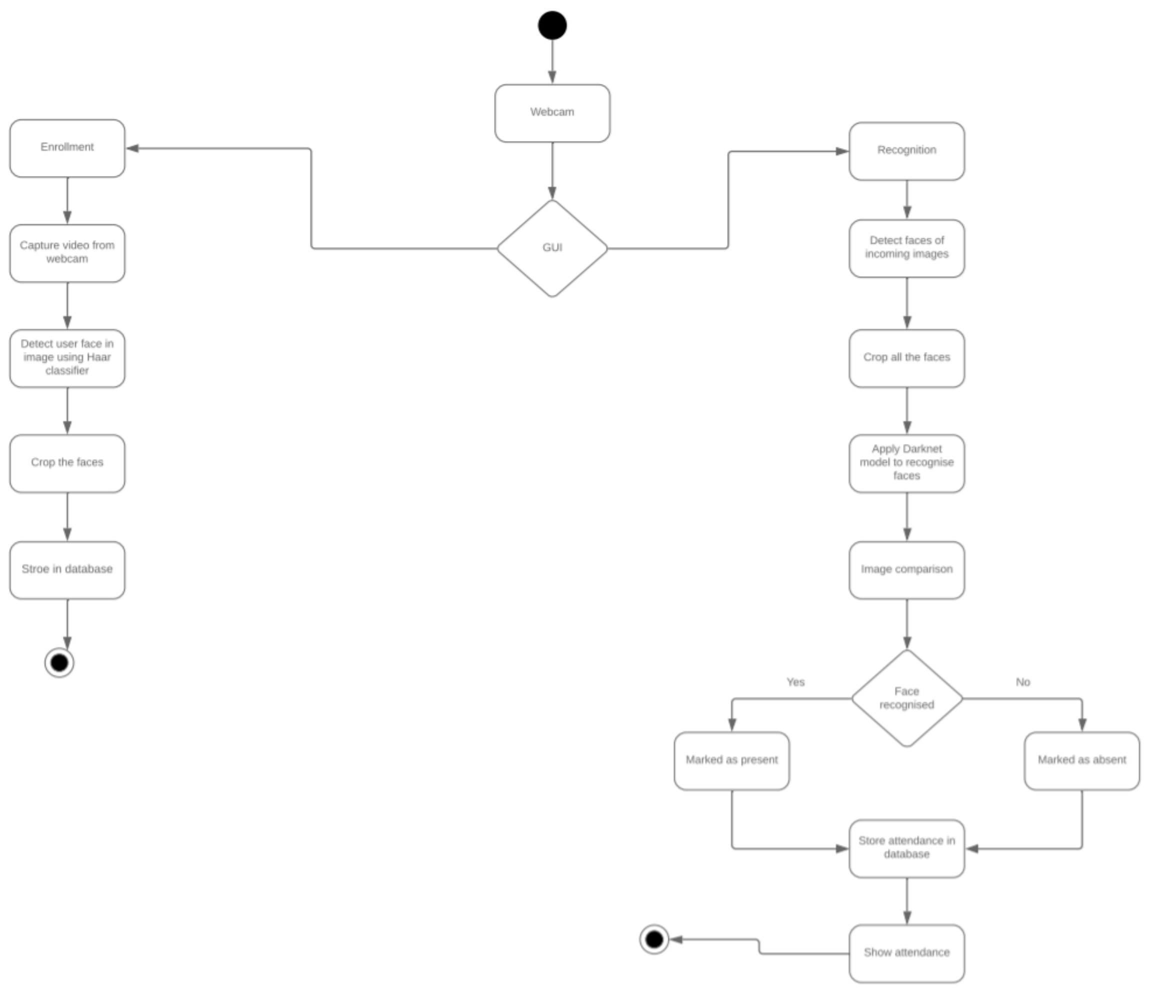
Figure-5.4 Activity Diagram Symbols

Procedure:

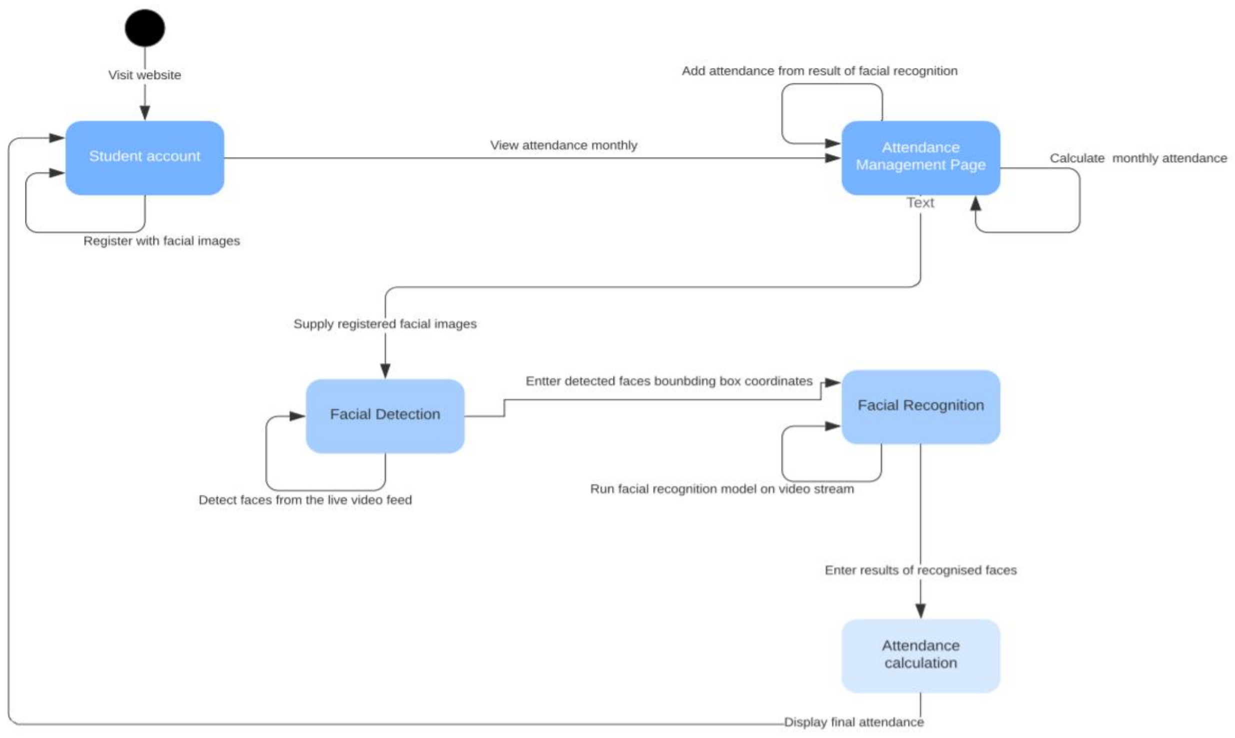
Prepare mentioned behaviour diagrams for chosen problem using Rational Rose/ any other Open Source UML tool.

Results: Printout of mentioned behavior diagrams

ACTIVITY DIAGRAM:


Webcam
GUI
Enrollment
Capture video from webcam
Detect user face in image using Haar classifier
Crop the faces
Stroe in database
Recognition
Detect faces of incoming images
Crop all the faces
Apply Darknet model to recognise faces
Image comparison
Face recognised
Yes
No
Marked as present
Marked as absent
Store attendance in database
Show attendance

STATE CHART DIAGRAM:


Visit website
Student account
Register with facial images
View attendance monthly
Add attendance from result of facial recognition
Attendance Management Page
Calculate monthly attendance
Text
Supply registered facial images
Facial Detection
Detect faces from the live video feed
Entter detected faces bounbding box coordinates
Facial Recognition
Run facial recognition model on video stream
Enter results of recognised faces
Attendance calculation
Display final attendance

Questions:

1. Explain the difference between activity diagram and state chart diagram.

In UML semantics, Activity Diagrams are reducible to State Machines with some additional notations that the vertices represent the carrying out of an activity and the edges represent the transition on the completion of one collection of activities to the commencement of a new collection of activities. Activity Diagrams capture high level activities aspects. In particular, it is possible to represent concurrency and coordination in Activity Diagrams.

Take a look at the Activity Diagram which models the flow of actions for an incident. Such an Activity Diagram focuses on the flow of data within a system.

In State Machines the vertices represent states of an object in a class and edges represent occurrences of events. The additional notations capture how activities are coordinated. Objects have behaviors and states. The state of an object depends on its current activity or condition. A State Machine Diagrams shows the possible states of the object and the transitions that cause a change in state.

Take a look at the State Machine Diagram below. It models the transitioning of states for an incident. Such a state diagram focuses on a set of attributes of a single abstraction (object, system).

Outcomes:Model the requirements using UML

Conclusion: Hence, we successfully were able to model the activity and state chart diagrams requirements using UML.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Website:

1.Michael Blaha, James Rumbaugh, "Object-Oriented Modeling and Design with UML", Prentice-Hall of India, 2nd Edition

2. Mahesh P. Matha, “Object-Oriented Analysis and Design using UML”, Prentice-Hall of India

3. Timothy C Lethbridge, Robert Laganiere, “Object-Oriented Software Engineering – A practical software development using UML and Java”, Tata McGraw-Hill, New Delhi.

4. http://www.uml-diagrams.org/uml-23-diagrams.html

5. http://vlabs.iitkgp.ernet.in/se/