S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 6

Aim: Construct the State Chart Diagram for the given problem definition.

Name of Student	
Roll No	
Semester/Year	
Academic Session	
Date of Performance	
Date of Submission	

AIM: Construct the State Chart Diagram for the given problem definition.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

- Identify the distinct states a system have
- Identify the events causing transitions from one state to another
- Represent the above information pictorially using simple states

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement

- Processor : Dual Core
- RAM : 1GB
- Hard Disk Drive : > 80 GB

Software Requirement

- Operating System – Windows

THEORY

Capturing the dynamic view of a system is very important for a developer to develop the logic for a system. State chart diagrams and activity diagrams are two popular UML diagram to visualize the dynamic behavior of an information system.

In this experiment, we will learn about the different components of activity diagram and state chart diagram and how these can be used to represent the dynamic nature of an information system.

State chart Diagrams

In case of Object Oriented Analysis and Design, a system is often abstracted by one or more classes with some well-defined behaviour and states. A statechart diagram is a pictorial representation of such a system, with all it's 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.

Statechart diagrams are normally drawn to model the behaviour of a complex system. For simple systems this is optional.

Building Blocks of a Statechart Diagram

State

A state is any "distinct" stage that an object (system) passes through in it's lifetime. An object remains in a given state for finite time until "something" happens, which makes it to move to another state. All such states can be broadly categorized into following three types:

- **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

As shown in figure-01, 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.

Figure-01: Representation of initial, intermediate, and final states of a statechart diagram

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 special action 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 or until the action expression results in a completed computation
- **Include:** Indicates invocation of a sub-machine

Any other action label identify 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-02 shows the syntax for representing a typical (intermediate) state.

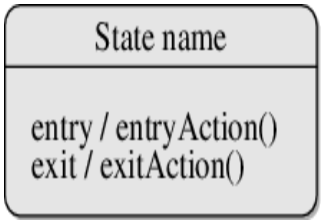


Figure-02: A typical state in a statechart diagram

States could again be either simple or composite (a state congaing other states). Here, however, we will deal only with simple 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 labeled by: event [guard-condition]/[action-expression], where

- **Event** is the what is causing the concerned transition (mandatory) -- Written in past tense (iii)
- **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

As mentioned in [ii], actions 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:

1. Exit actions of the present state
2. Actions specified for the transition
3. Entry actions of the next state

Figure-03 shows a typical statechart diagram with all it's syntaxes.

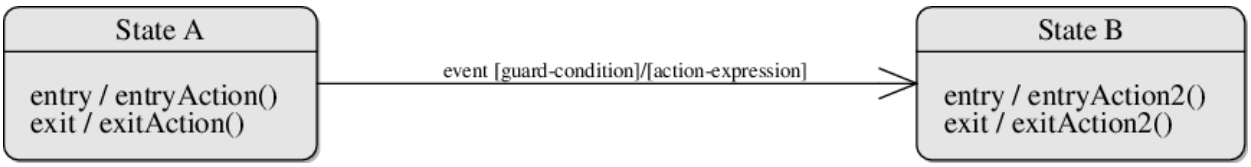


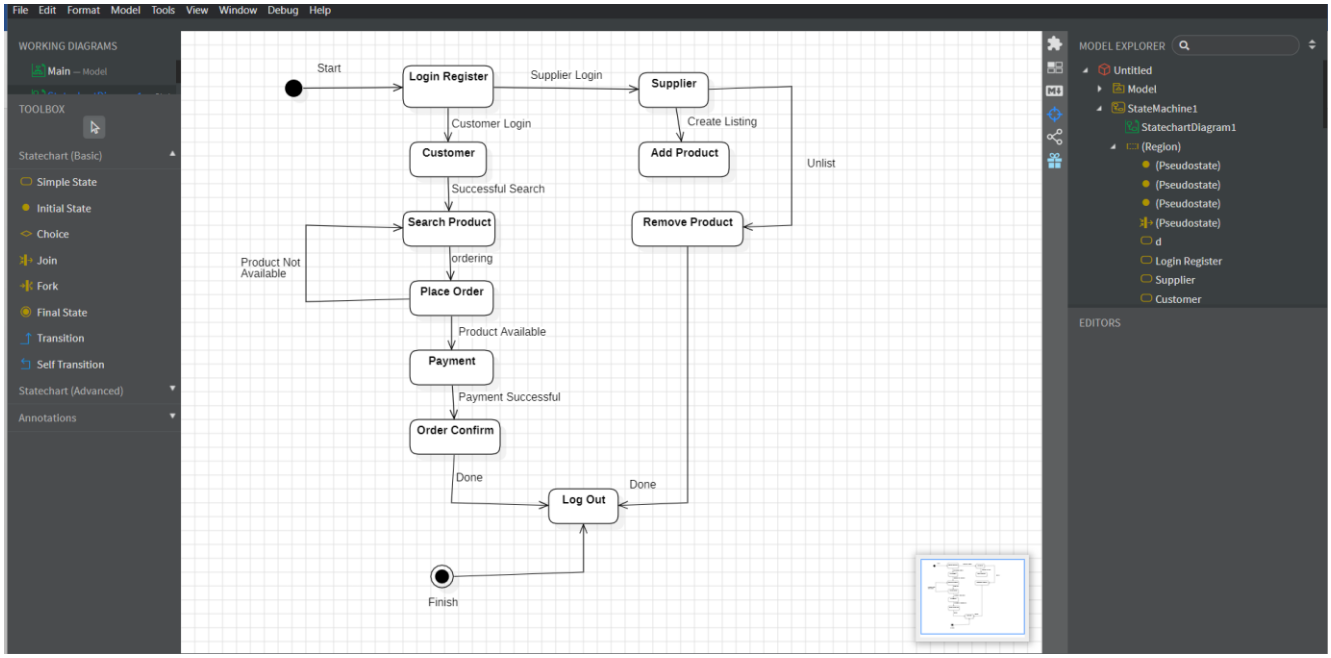
Figure-03: A statechart diagram showing transition from state A to B

Guidelines for drawing Statechart Diagrams

Following steps could be followed, as suggested in [i] to draw a statechart diagram:

- For the system to developed, identify the distinct states that it passes through
- Identify the events (and any precondition) that cause the state transitions. Often these would be the methods of a class as identified in a class diagram.
- Identify what activities are performed while the system remains in a given state

OBSERVATION (Students should attach screenshot of assigned problem statement):



CONCLUSION:

DISCUSSION QUESTIONS?

1. What is the purpose of a state chart?
2. What is state chart modeling?
3. What are state charts?

4. What is the rule of state diagram?

5. What are the symbols used in a state chart diagram?

REFERENCES:

- http://vlabs.iitkgp.ernet.in/se/1/
- https://sites.google.com/view/ait-se/Home/practicals
- https://www.javatpoint.com/state+chart+diagram