UNIT 5

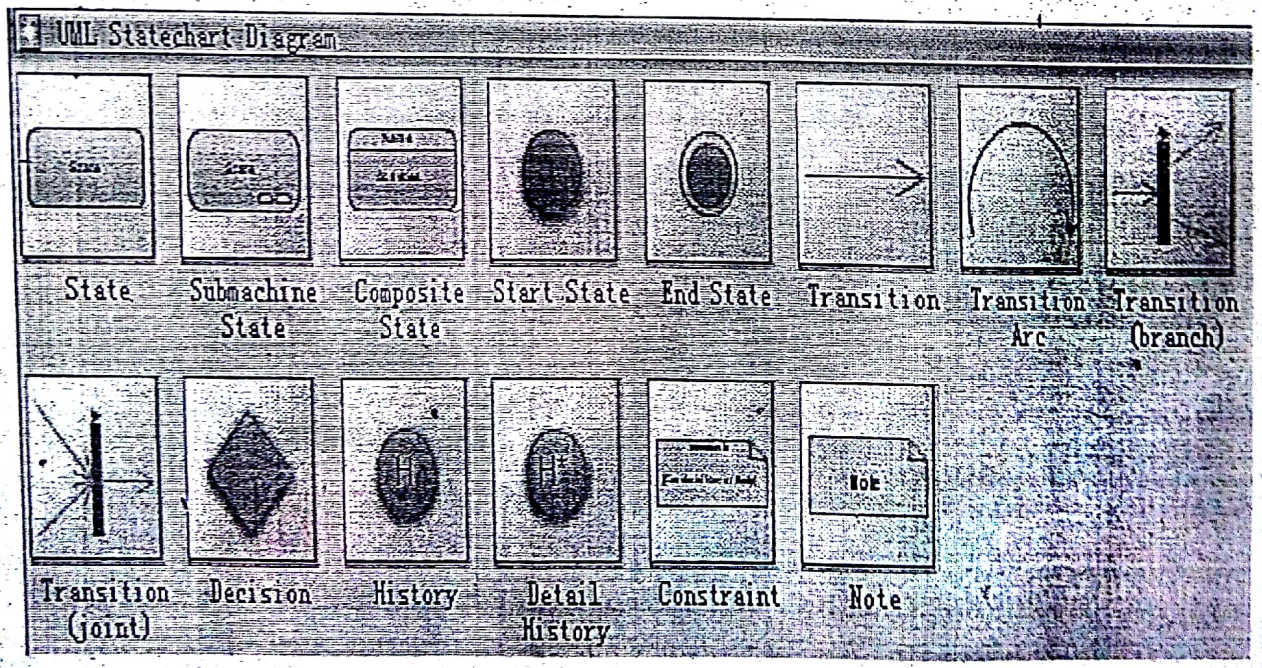
STATE CHART DIAGRAMS

- State chart diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of the systems.
- State chart diagram shows a state machine.
- An activity diagram is a special case of a state chart diagram.
- Both activity and state chart diagrams are useful in modeling the life type of an object.
- Activity diagram shows flow of control from activity to activity.
- State chart diagram shows flow of control from state to state.

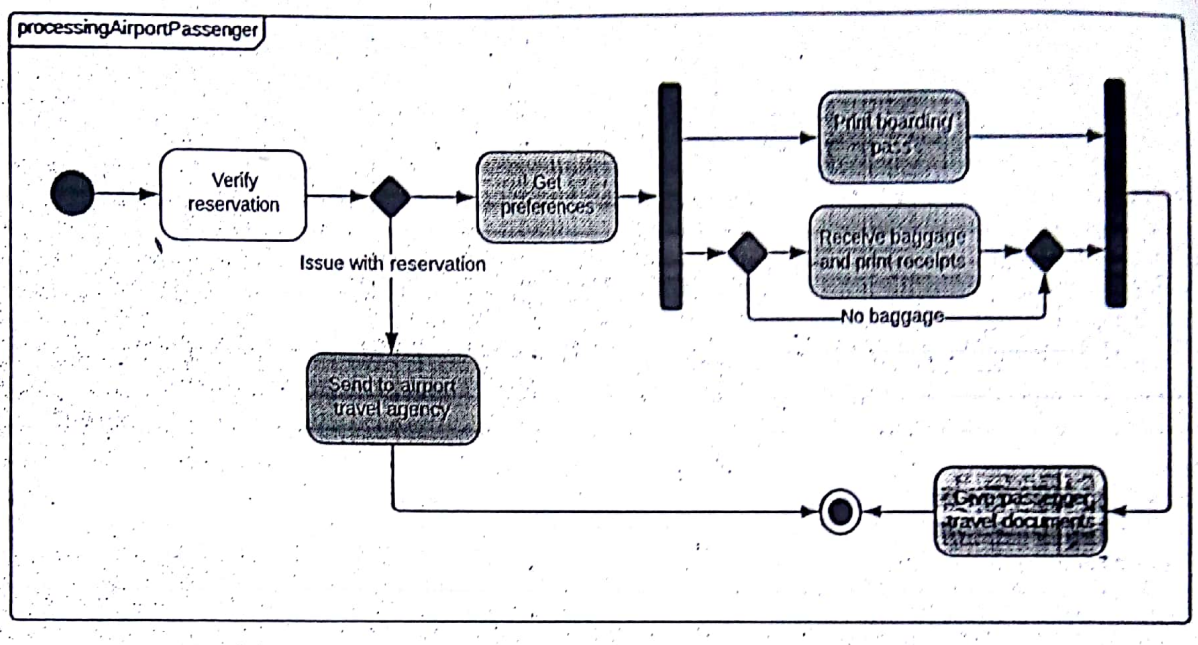
Contents:

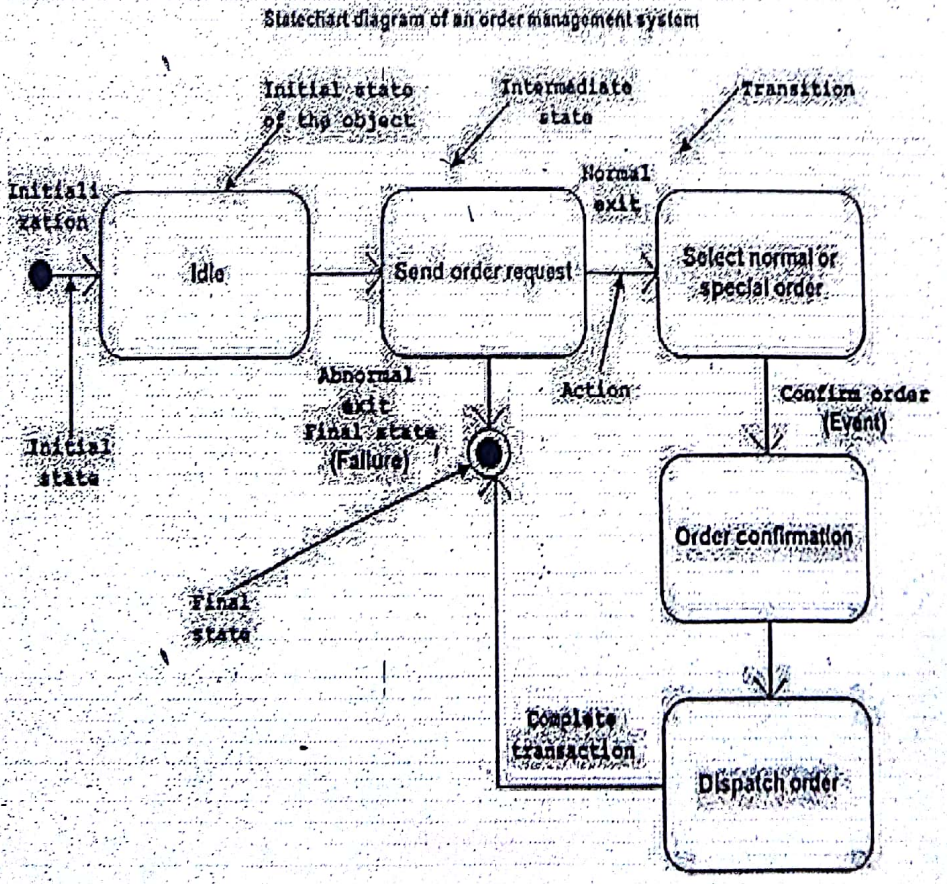
- State chart diagrams commonly contain
  - » Set of states & Complex states
  - » Transitions including events and actions

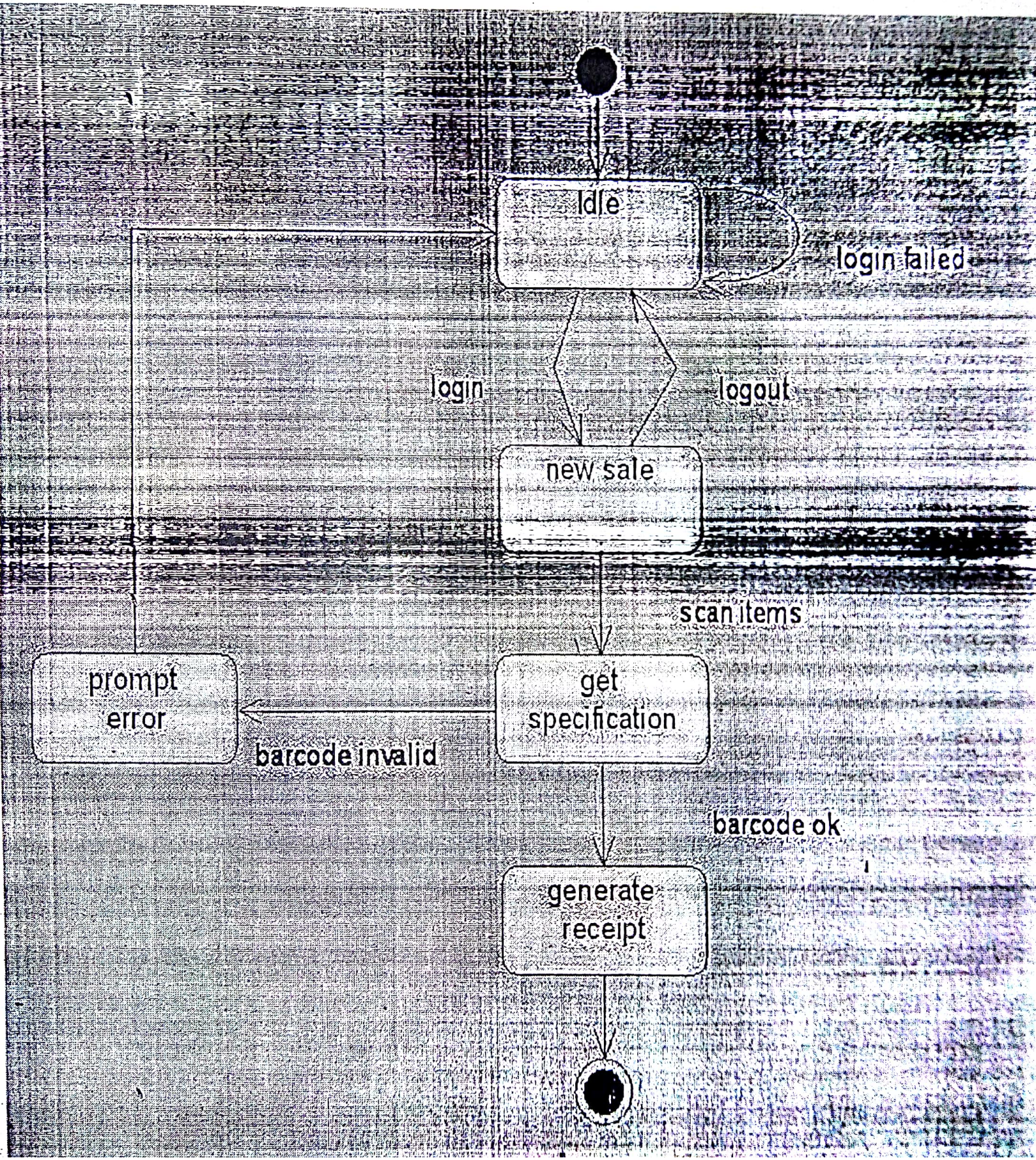
State chart Diagram Symbols:



Examples:

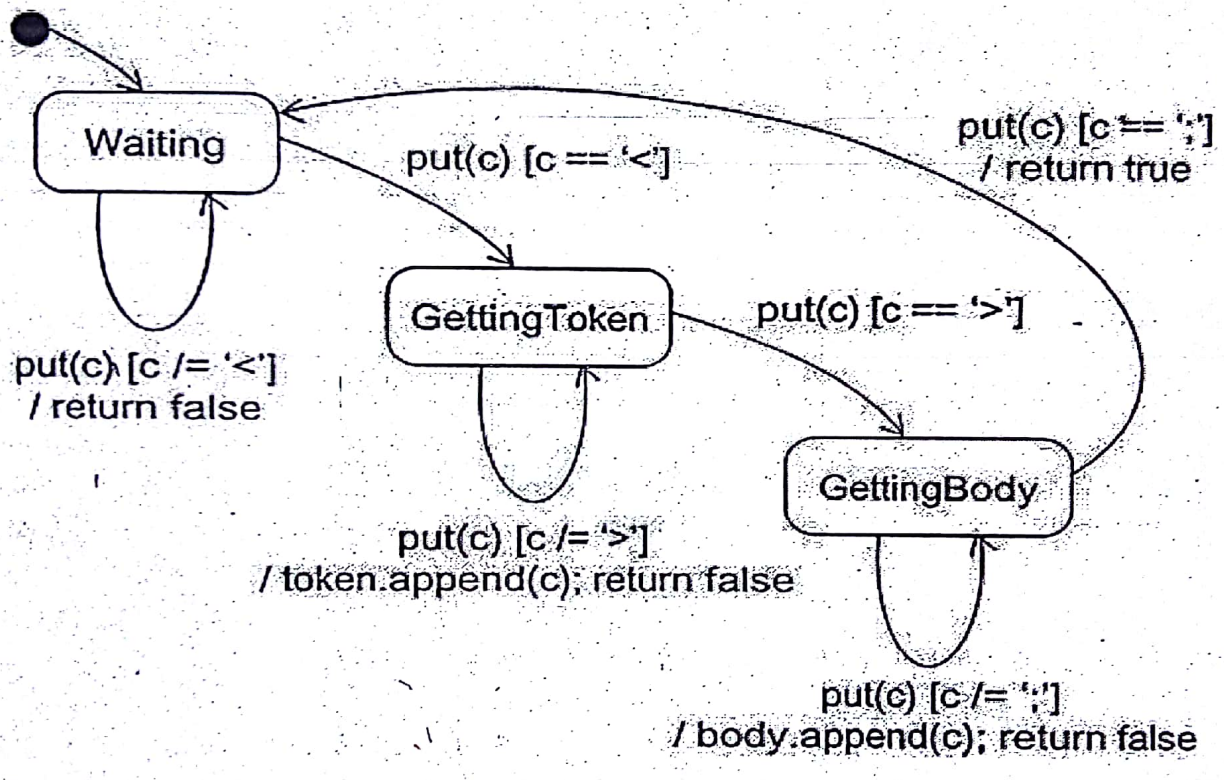

processingAirportPassenger
Verify reservation
Issue with reservation
Get preferences
Print boarding pass
Receive baggage and print receipts
No baggage
Send to airport travel agency


Statechart diagram of an order management system
Initial state of the object
Intermediate state
Transition
Initialization
Idle
Send order request
Normal exit
Select normal or special order
Abnormal exit
Final state (Failure)
Action
Confirm order (Event)
Initial state
Order confirmation
Final state
Complete transaction
Dispatch order


idle
login failed
login
logout
new sale
scan items
prompt error
get specification
barcode invalid
barcode ok
generate receipt

Modeling Reactive Objects:

- To model a reactive object
- For example the state chart diagram for parsing a simple context-free language message : '<'string '>'string ';'
- The first string represents a tag, the second string represents the body of the message.



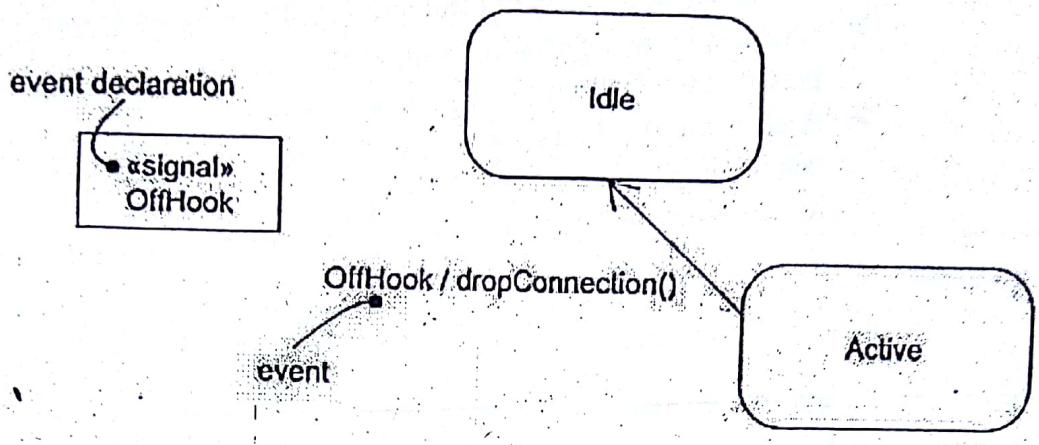
Events

- An event is the specification of a significant occurrence that has a location in time and space.
- Anything that happens is modeled as an event in UML.
- In the context of state machines, an event is an occurrence of a stimulus that can trigger a state transition
- Four kinds of events – signals, calls, the passing of time, and a change in state.

Figure 1: Events

- Events may be external or internal and asynchronous or synchronous.
- Asynchronous events are events that can happen at arbitrary times eg:- signal, the passing of time, and a change of state. synchronous events, represents the invocation of an operation eg:- Calls
- External events are those that pass between the system and its actors. Internal events are those that pass among the objects that live inside the system.

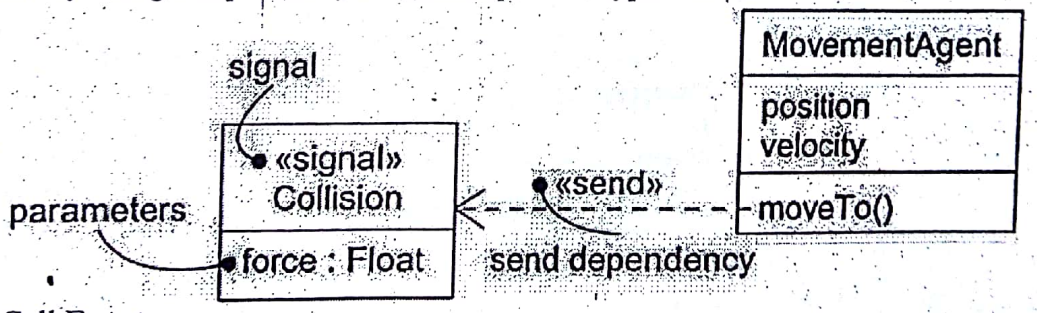
- A signal is an event that represents the specification of an asynchronous stimulus communicated between instances.



Kinds of events

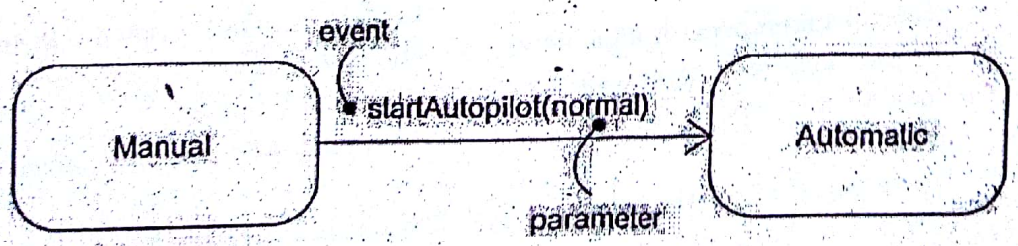
Signal Event

- a signal event represents a named object that is dispatched (thrown) asynchronously by one object and then received (caught) by another. Exceptions are an example of internal signal
- a signal event is an asynchronous event
- Signal events may have instances, generalization relationships, attributes and operations. Attributes of a signal serve as its parameters
- A signal event may be sent as the action of a state transition in a state machine or the sending of a message in an interaction
- signals are modeled as stereotyped classes and the relationship between an operation and the events by using a dependency relationship, stereotyped as send



Call Event:

- a call event represents the dispatch of an operation
- a call event is a synchronous event

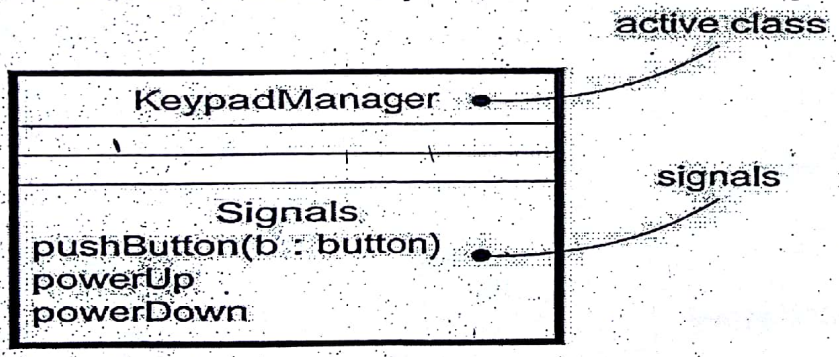


Time and Change Events:

- A time event is an event that represents the passage of time.
- Modeled by using the keyword 'after' followed by some expression that evaluates to a period of time which can be simple or complex.
- A change event is an event that represents a change in state or the satisfaction of some condition.
- Modeled by using the keyword 'when' followed by some Boolean expression.

Sending and Receiving Events

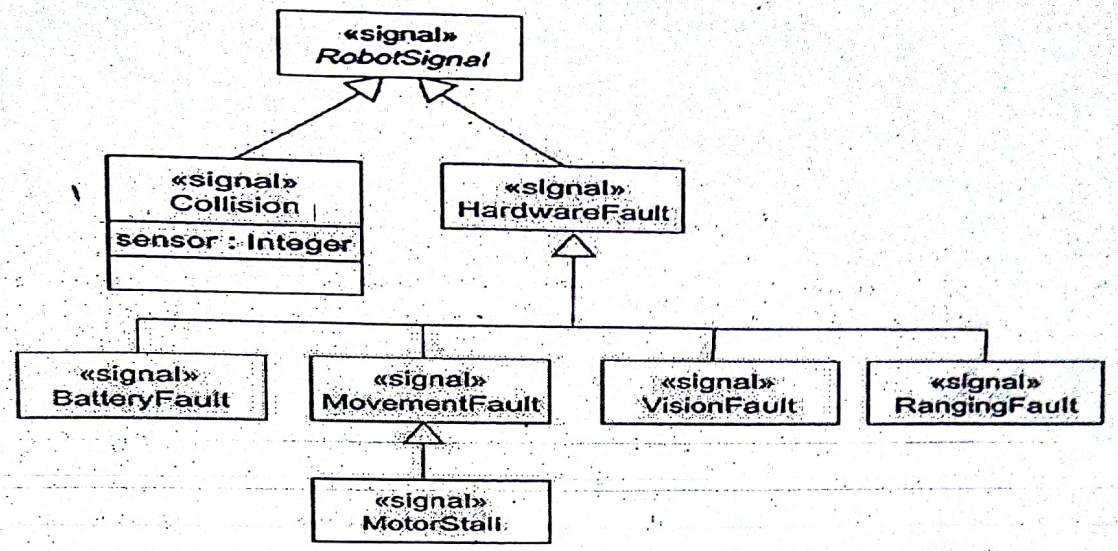
- For synchronous events (Sending or Receiving) like call event, the sender and the receiver are in a rendezvous (the sender dispatches the signal and wait for a response from the receiver) for the duration of the operation.
- when an object calls an operation, the sender dispatches the operation and then waits for the receiver.
- For asynchronous events (Sending or Receiving) like signal event, the sender and receiver do not rendezvous either sender dispatches the signal but does not wait for a response from the receiver.
- When an object sends a signal, the sender dispatches the signal and then continues along its flow of control, not waiting for any return from the receiver.
- call events can be modeled as operations on the class of the object.



Modeling family of signals:
To model a family of signals

- Consider all the different kinds of signals to which a given set of active objects may respond.
- Look for the common kinds of signals and place them in a generalization/specialization hierarchy using inheritance.
- Elevate more general ones and lower more specialized ones.
- Look for the opportunity for polymorphism in the state machines of these active objects.
- Where you find polymorphism, adjust the hierarchy as necessary by introducing intermediate abstract signals.

- 6 models a family of signals that may be handled by an autonomous robot.



Modeling Exceptions:

To model exceptions

- For each class and interface, and for each operation of such elements, consider the exceptional conditions that may be raised.
- Arrange these exceptions in a hierarchy. Elevate general ones, lower specialized ones, and introduce intermediate exceptions, as necessary.
- For each operation, specify the exceptions that it may raise.
- You can do so explicitly (by showing send dependencies from an operation to its exceptions) or you can put this in the operation's specification.

