State Diagram

State Chart Diagram

State Transition Diagram

Introduction

- Two interaction diagrams with objects of the same class receiving the same messages may respond differently

- This is because an object’s behavior is affected by the values of its attributes

- UML State Machine Diagram records these dependencies

State Transition Diagram

- A state transition diagram is a technique to depict:

1. The states of an entity
2. The transitions of states of the entity
3. The trigger or the event that caused the transition of state of the entity

- The *entity* may be a physical device such as a light switch or a vending machine; it may be a software system or component such as a word processor or an operating system; it may be a biological system such as a cell or a human; or - - - -

This modeling technique came from a more formal area called automata theory. State transition diagram depicted a *Finite State Machine*.

Software Program View

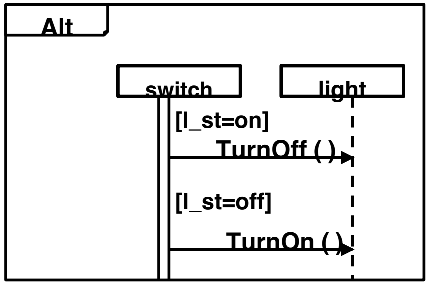
- **The end product of a software is a program which executes. In depicting the program (or an object) we can consider:**
  - **Variables which take on different values**
  - **Control structure and assignment statements (events) in the program that change the values of the variables**

1. ***Combination of values of the data (variables & constants) at any point of the program represent the program state at that point.***

2. ***The change made to the values of the variables through assignment statements represent a transition of state***

A very simple example

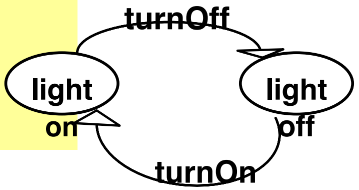
light switch



From State (light)	Event (switch)	To State (light)
on	turnOff	off
off	turnOn	on

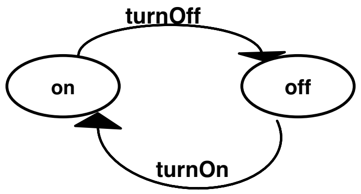
1. "Sequence diagram" *(alternative fragment) for switch and light interaction*

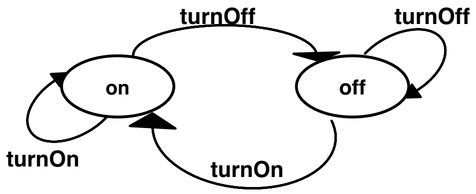
2. "State transition table" *for light with switch events*



3. "State transition diagram" *for light with switch events*

A little “more” on the light switch





What happens if we turn on a light that is already on?

state can “transition” to its current state

Using State Transition Diagram

- Model the entity at the "abstraction" level where the *number of states is* *"manageable."*

1. List (design) the *states* (should not be large)
2. List *events* that will trigger the state transition (should not be big)
3. There must be a *starting state*
4. There must be a *terminating state or states*
5. Design the *transition rules* (*the bulk of your design work is thinking through the transition rules*)

1.The above is not necessarily performed in sequence; iterate through these.

2. Even with a modest number of states and events, the *state transition* diagram, which really depicts the transition rules, can be enormous.

State Diagrams

- State diagrams are used to show possible states a single object can get into
  - shows states of an object

- How object changes state in response to events
  - shows transitions between states

Elements of State Diagram

- Start marker
- Stop marker
- States – box with rounded corners
- Transitions – shown as arrows between states
- Events – that cause transition between states
- Action – an object’s reaction to an event
- Guard

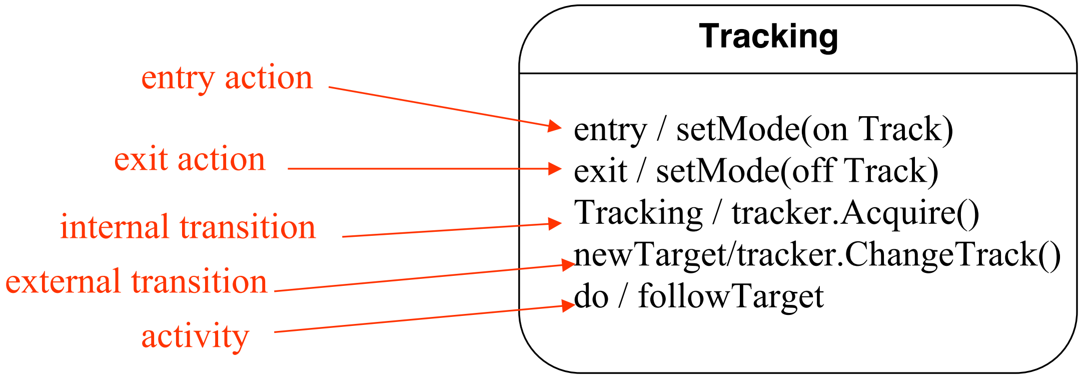
Naming Conventions for a state

- Each unique state has a unique name
- State Names are **verb** phrases
- A noun to name a state – **incorrect**

State Diagrams: States

- States are represented as rounded boxes which contain:
  - the **state name**
  - and the following optional fields
    - **entry and exit actions**: entry and exit actions are executed whenever the state is entered or exited, respectively
    - **Internal transitions**: A response to an event that causes the execution of an action but does not cause a change of state or execution of exit or entry actions.
    - **External transition**: A response to an event that causes a change of state or a self-transition, together with a specified action.
    - **Activities**: Typically, once the system enters a state it sits idle until an event triggers a transition. Activities help you to model situations where while in a state, the object does some work that will continue until it is interrupted by an event

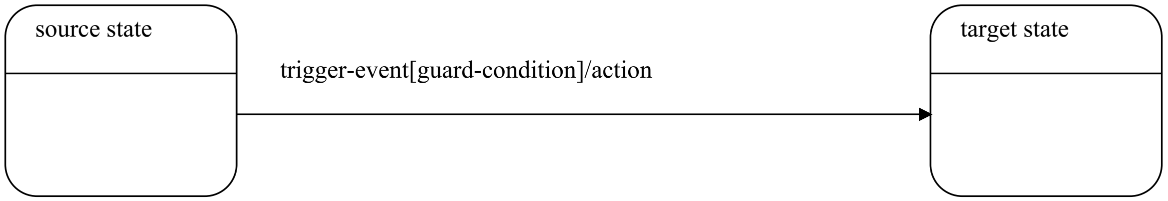
State Diagrams: States


Tracking
entry / setMode(on Track)
exit / setMode(off Track)
Tracking / tracker.Acquire()
newTarget/tracker.ChangeTrack()
do / followTarget
entry action
exit action
internal transition
external transition
activity

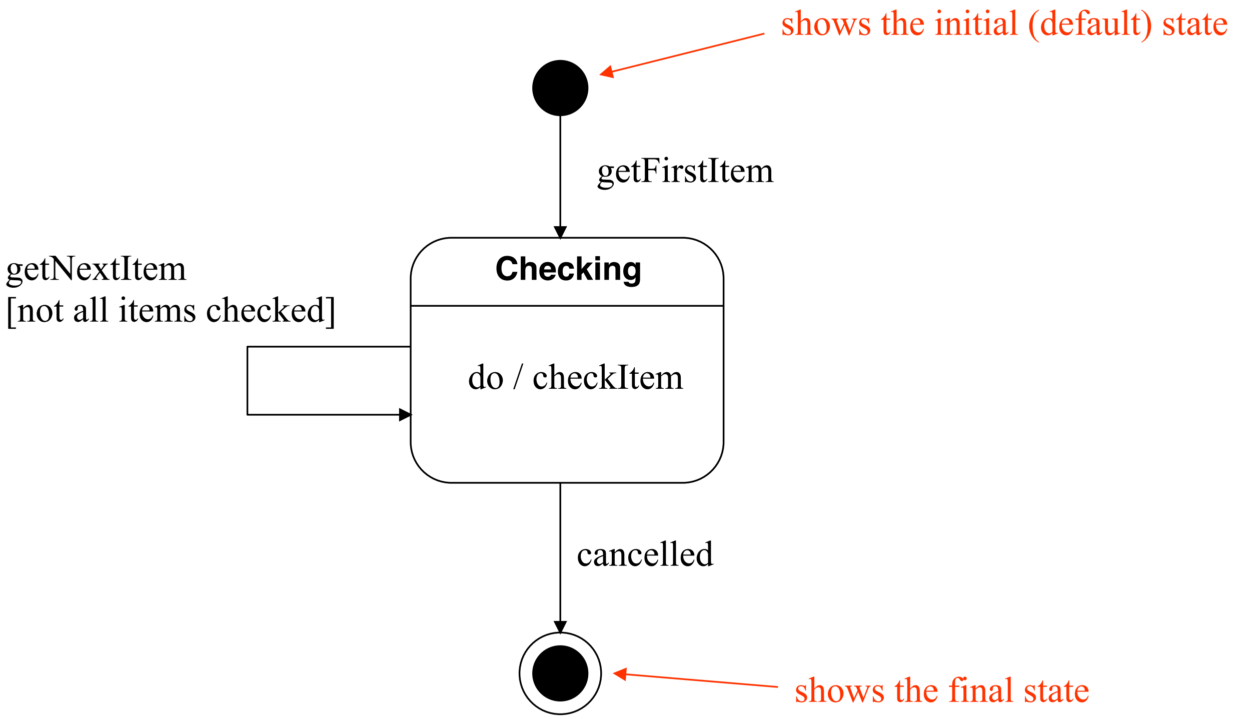
Note that, “entry”, “exit”, “do” are keywords

State Diagrams: Transitions

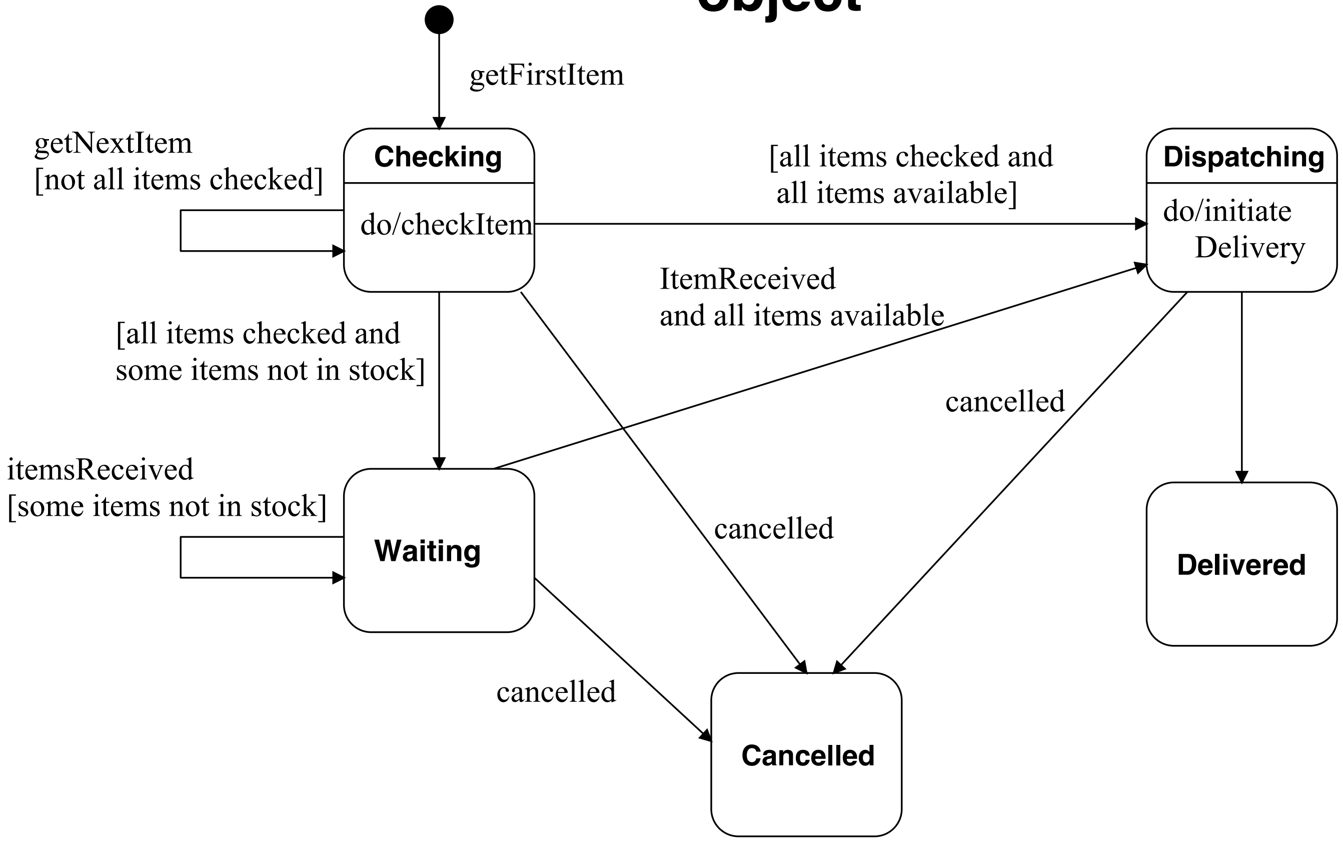
- Transitions
  - **source state** and **target state**: shown by the arrow representing the transition
  - **trigger event**: the event that makes the transition fire
  - **guard condition**: a Boolean expression that is evaluated when the trigger event occurs, the transition can fire only if the guard condition evaluates to true
  - **action**: an executable atomic computation that can directly act on the object that owns the state machine or indirectly on other objects that are visible to the object
  - **initial and final states**: shown as filled black circle and a filled black circle surrounded by an unfilled circle, respectively



State Diagrams

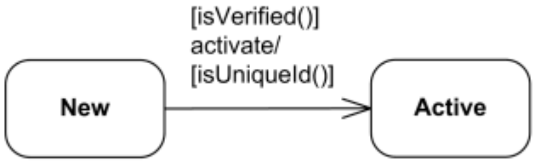


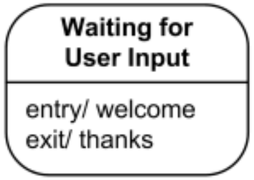
State Diagram Example: States of an Order object



Types of State Machine Diagram

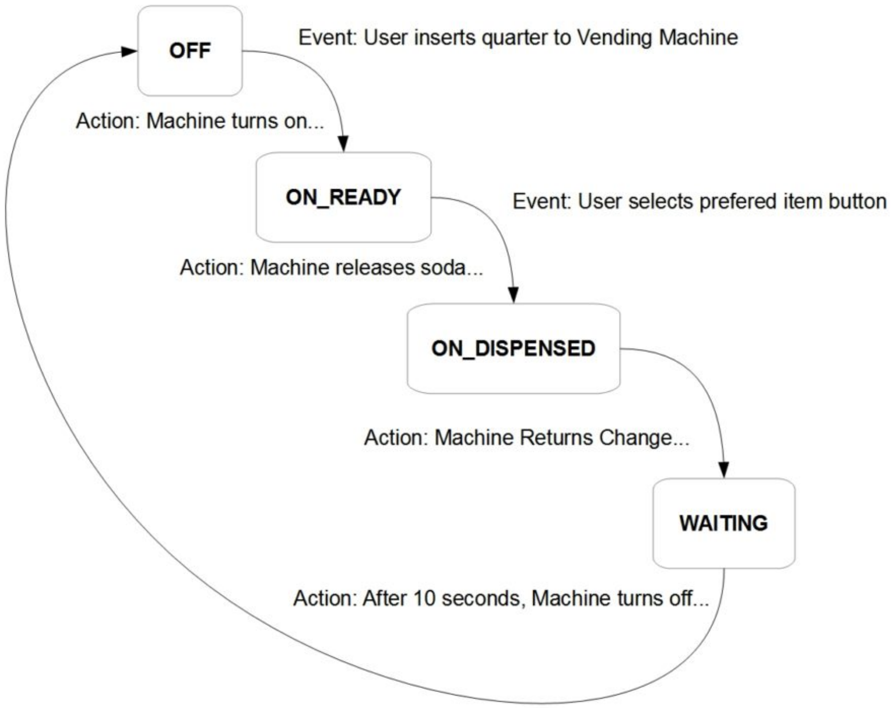
- Protocol State Machines: These are used to express a **usage protocol** or a **lifecycle** of some **classifier**. It shows which operations of the classifier may be called in each state of the classifier, under which specific conditions, and satisfying some optional postconditions after the classifier transitions to a target state.
- Behavioral State Machines: These are specialization of behavior and is used to specify discrete behavior of a part of designed system through finite state transitions.





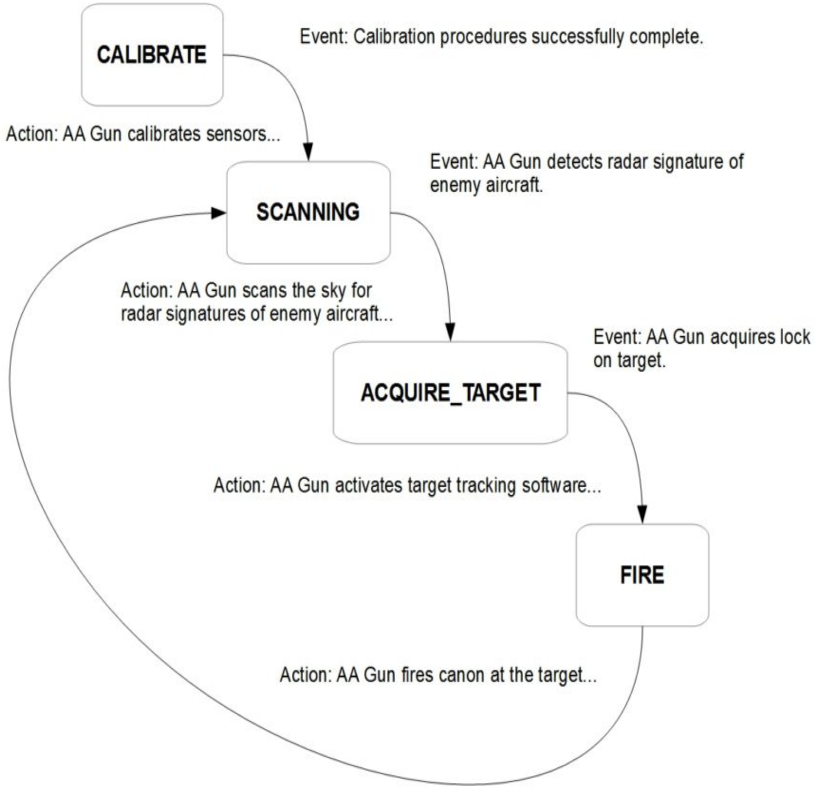
Example of Protocol State Machines

Soda Vending Machine


OFF
Event: User inserts quarter to Vending Machine
Action: Machine turns on...
ON_READY
Event: User selects prefered item button
Action: Machine releases soda...
ON_DISPENSED
Action: Machine Returns Change...
WAITING
Action: After 10 seconds, Machine turns off...

Example of Behavioral State Machines

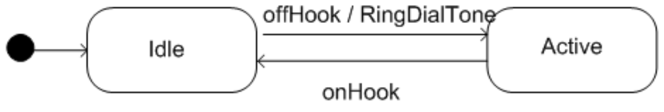
Anti-Aircraft (AA) Gun

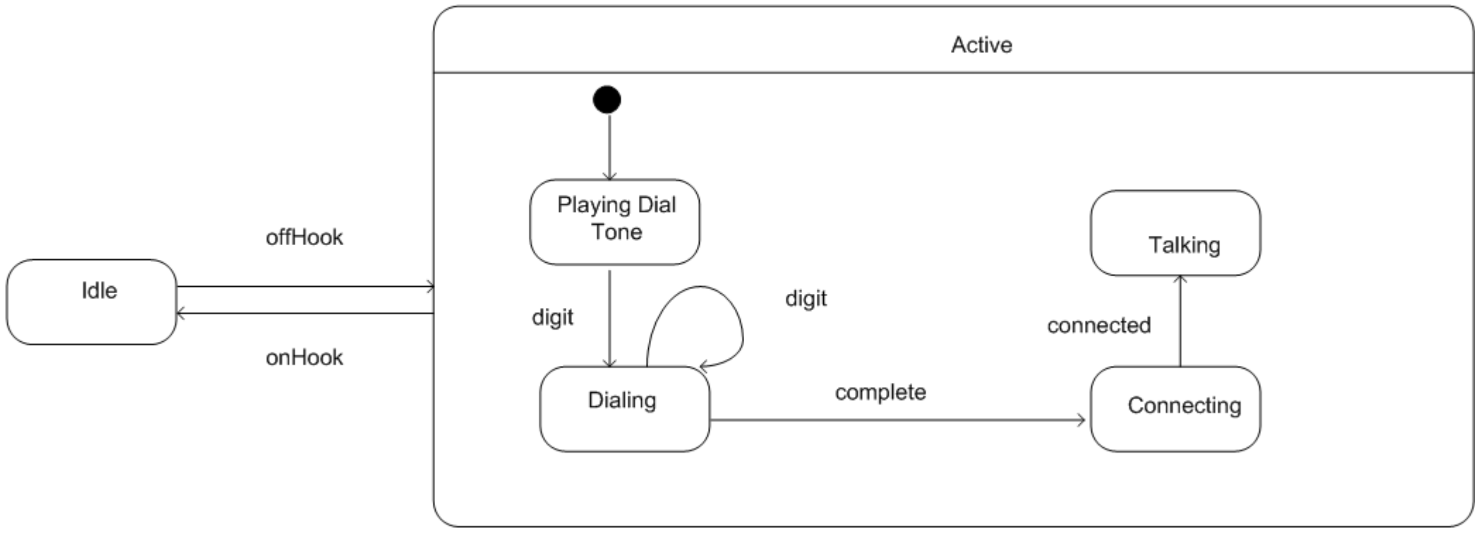

CALIBRATE
Event: Calibration procedures successfully complete.
Action: AA Gun calibrates sensors...
SCANNING
Event: AA Gun detects radar signature of enemy aircraft.
Action: AA Gun scans the sky for radar signatures of enemy aircraft...
ACQUIRE_TARGET
Event: AA Gun acquires lock on target.
Action: AA Gun activates target tracking software...
FIRE
Action: AA Gun fires canon at the target...

Advanced State Machine Modeling

- Nested states
- Concurrent states

Nested State Machine- Telephone

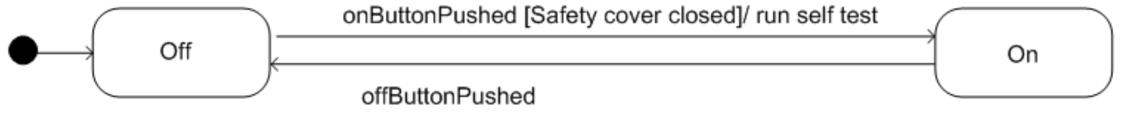


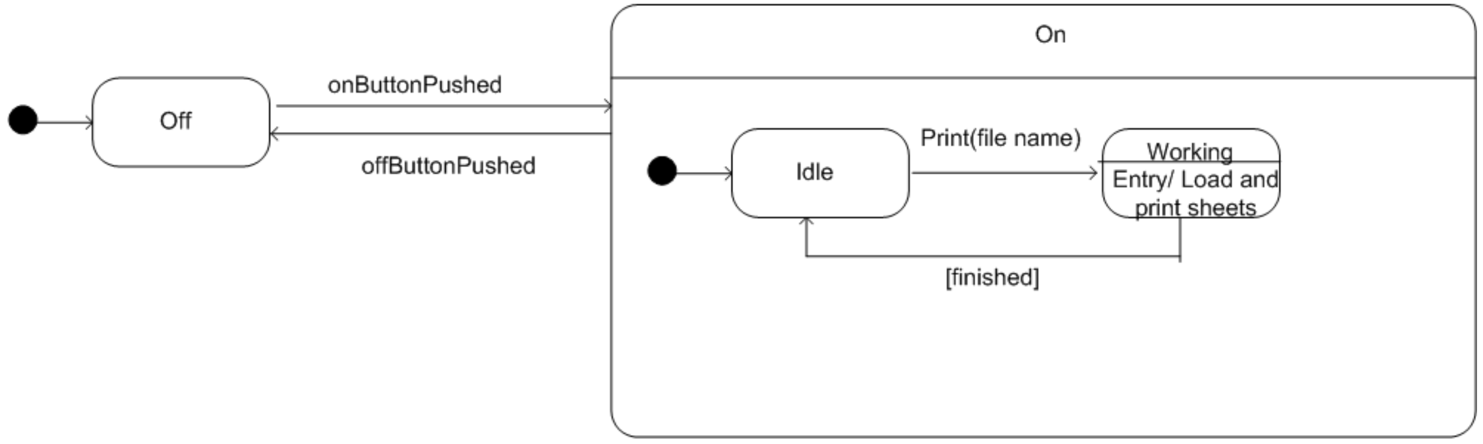


Nested State Machine- Printer

- Example: When the printer is in On state, it may also be in Idle or Working
- To show these two states, draw lower level state diagrams within On state
- When the printer is on, it begins at Idle state, so printer is in both On and Idle state
- When print message is received, it moves to working state and also remains in On state

Nested State Machine- Printer

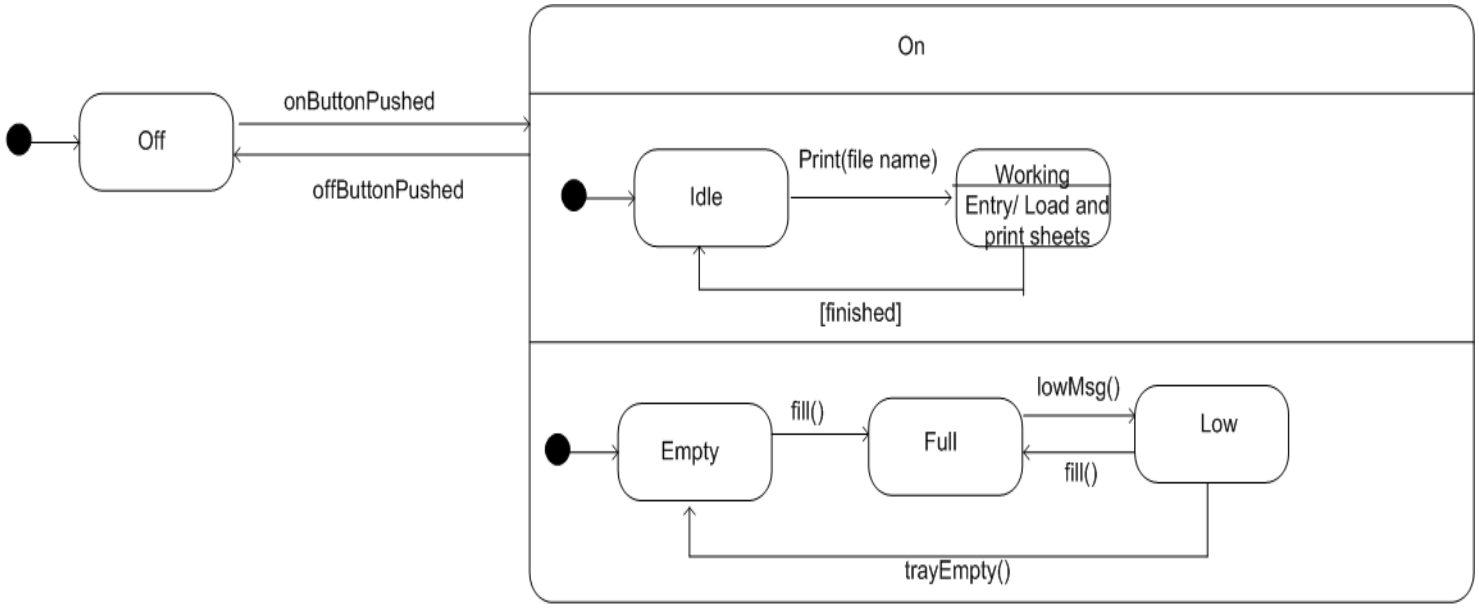




Concurrent states

- A Printer object cycles between two separate paths. The two independent paths are;
  - Representing states of the work cycle.
  - Representing states of the input paper tray

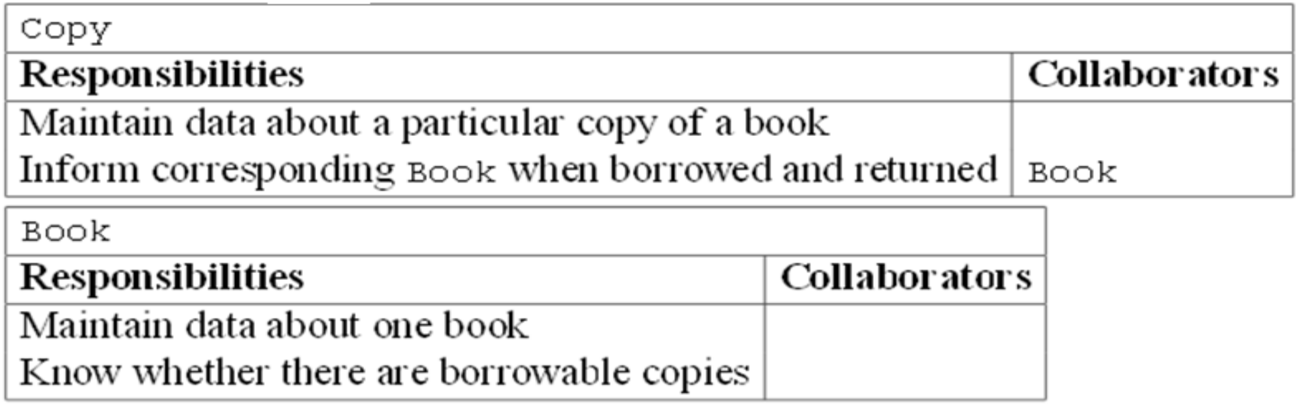
Concurrent State Machine- Printer



State Diagrams Importants

- Use them to show the behavior of a single object not many objects
  - for many objects use interaction diagrams
- Do not try to draw state diagrams for every class in the system, use them to show interesting behavior and increase understanding

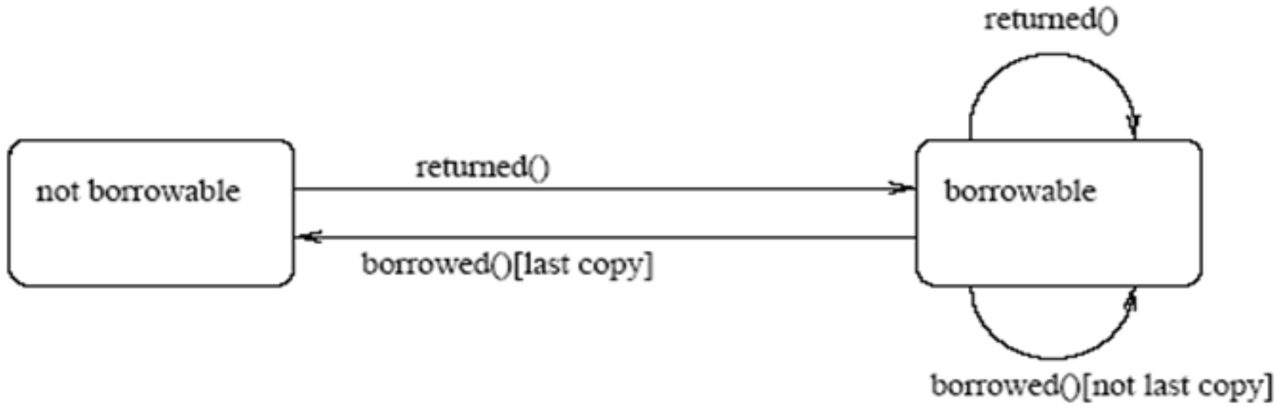
CRC for Copy & Book



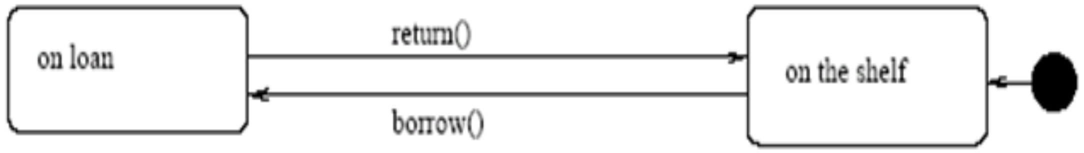
Copy	
Responsibilities	**Collaborators**
Maintain data about a particular copy of a book	
Inform corresponding `Book` when borrowed and returned	`Book`

Book	
Responsibilities	**Collaborators**
Maintain data about one book	
Know whether there are borrowable copies	

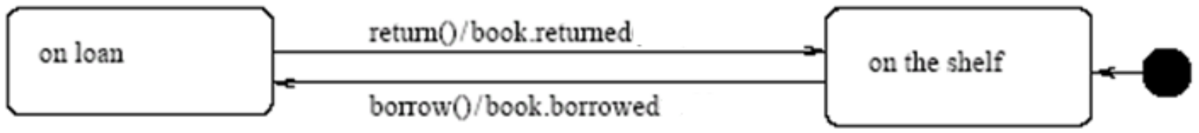
State Machine Diagram for class *Book*



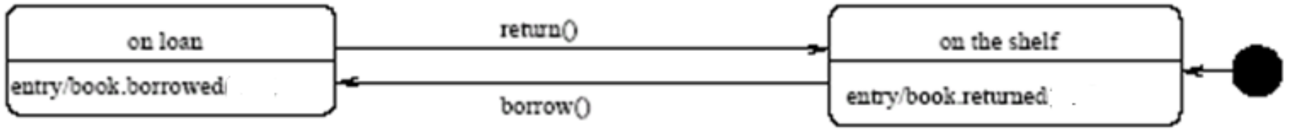
State Machine Diagram for class *Copy*



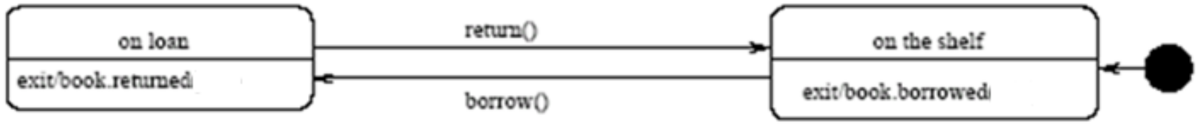
State Machine Diagram for class *Copy* with actions



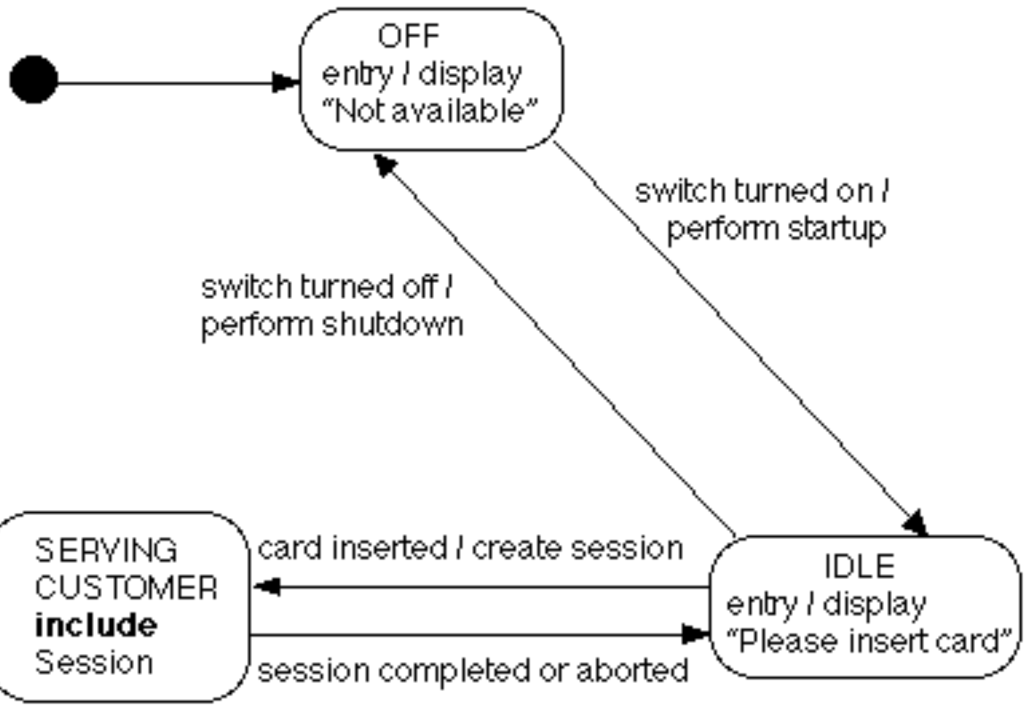
State Machine Diagram for class *Copy* with entry actions



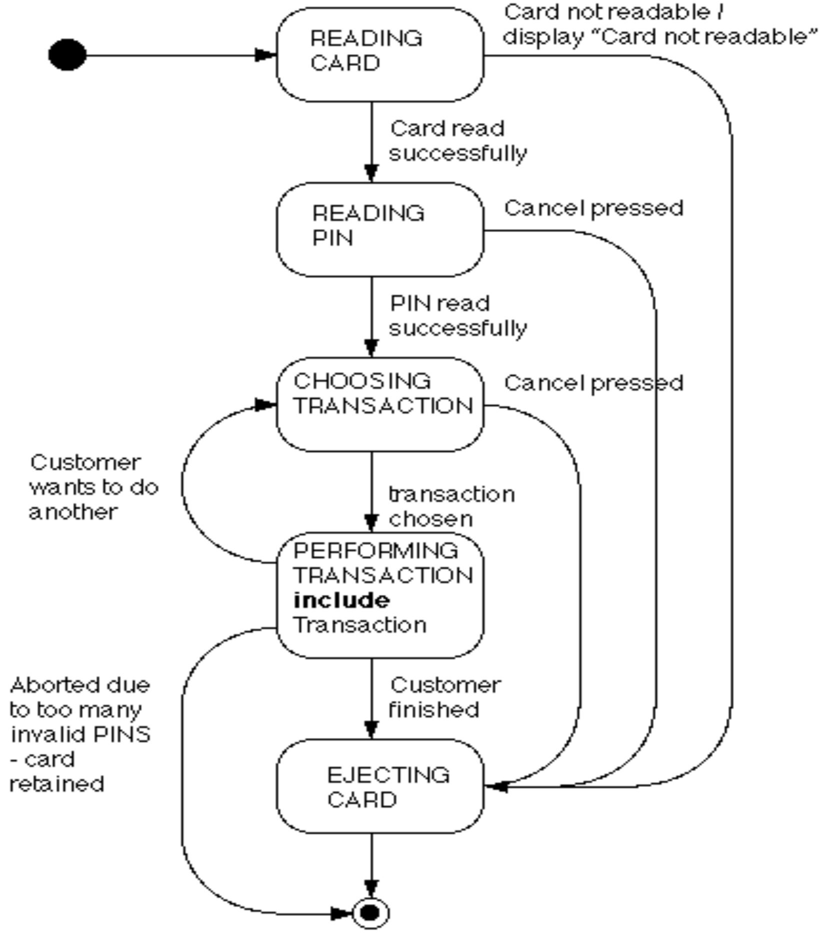
State Machine Diagram for class *Copy* with exit actions



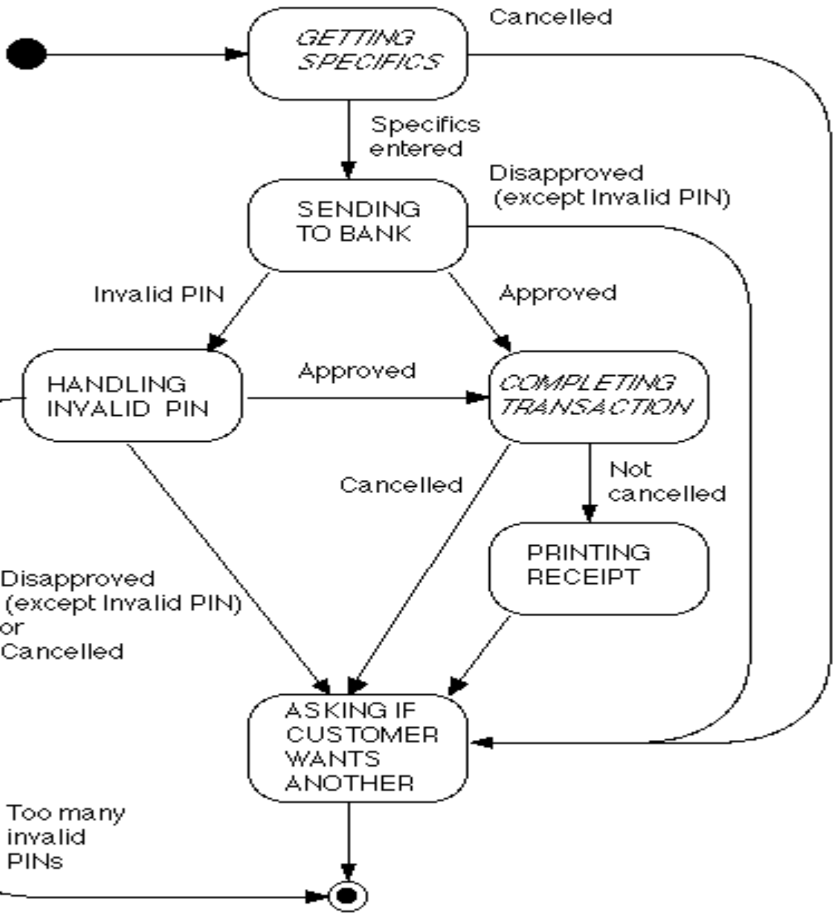
State-Chart for Overall ATM (includes System Startup and System Shutdown Use Cases)



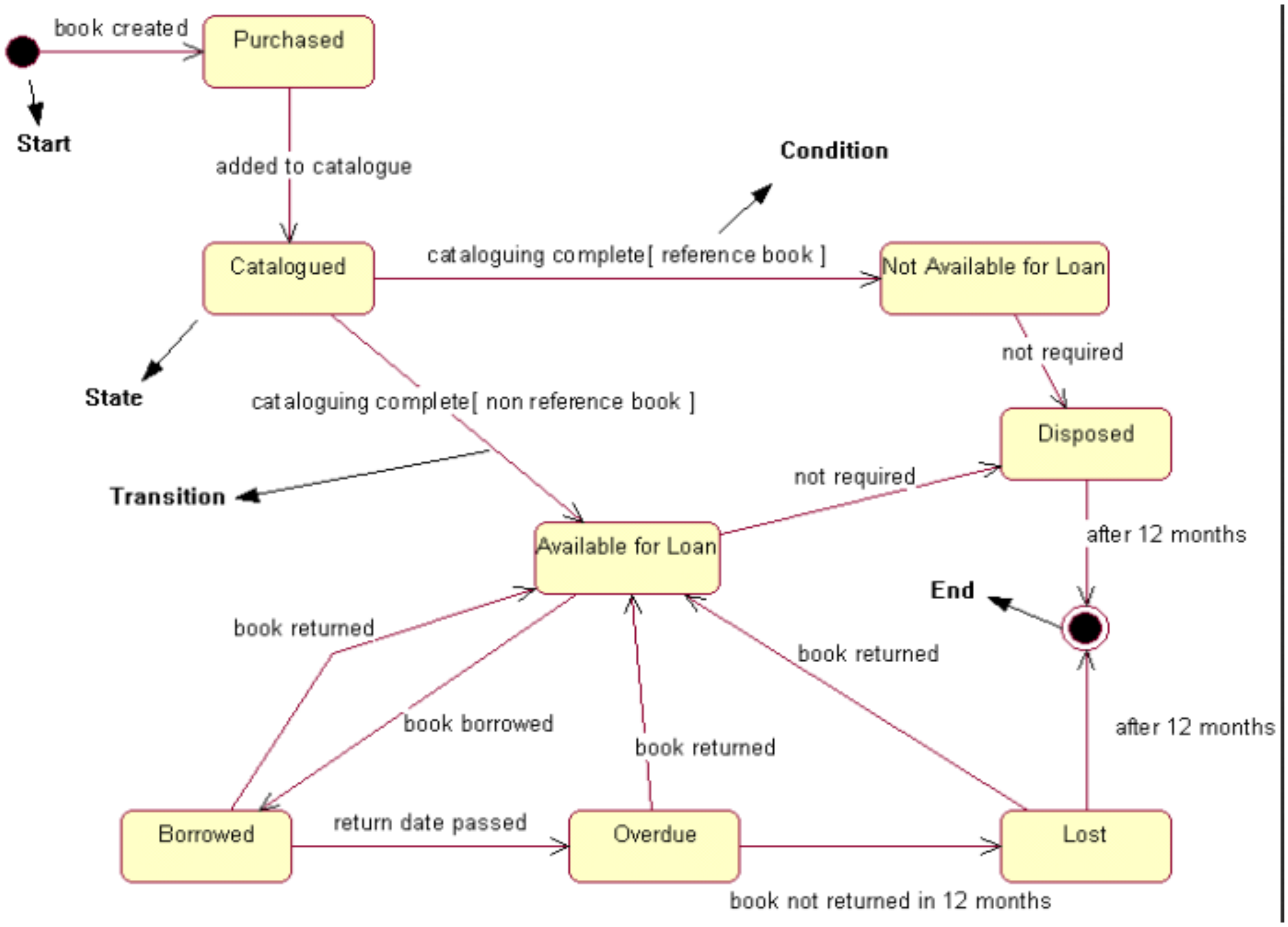
State-Chart for One Session


READING CARD
Card not readable / display "Card not readable"
Card read successfully
READING PIN
Cancel pressed
PIN read successfully
CHOOSING TRANSACTION
Cancel pressed
Customer wants to do another
transaction chosen
PERFORMING TRANSACTION **include** Transaction
Aborted due to too many invalid PINS - card retained
Customer finished
EJECTING CARD

State-Chart for One Transaction (italicized operations are unique to each particular type of transaction)



Librarians categorize the library books into loanable and non-loanable books. The non-loanable books are the reference books. However, the loanable books are the non-reference books. After cataloguing the books, the books are available for loan. Students who borrow the library books should return them back before the due date. Books that are 12 months over the due date would be considered as a lost state. However, if those books are found in the future, they must be returned back to the library. When the books are found not required in the library or have been damaged, the book would be disposed.


book created
Purchased
Start
added to catalogue
Condition
Catalogued
cataloguing complete[reference book]
Not Available for Loan
State
not required
cataloguing complete[non reference book]
Disposed
Transition
not required
after 12 months
Available for Loan
End
book returned
book returned
book borrowed
book returned
after 12 months
Borrowed
return date passed
Overdue
Lost
book not returned in 12 months

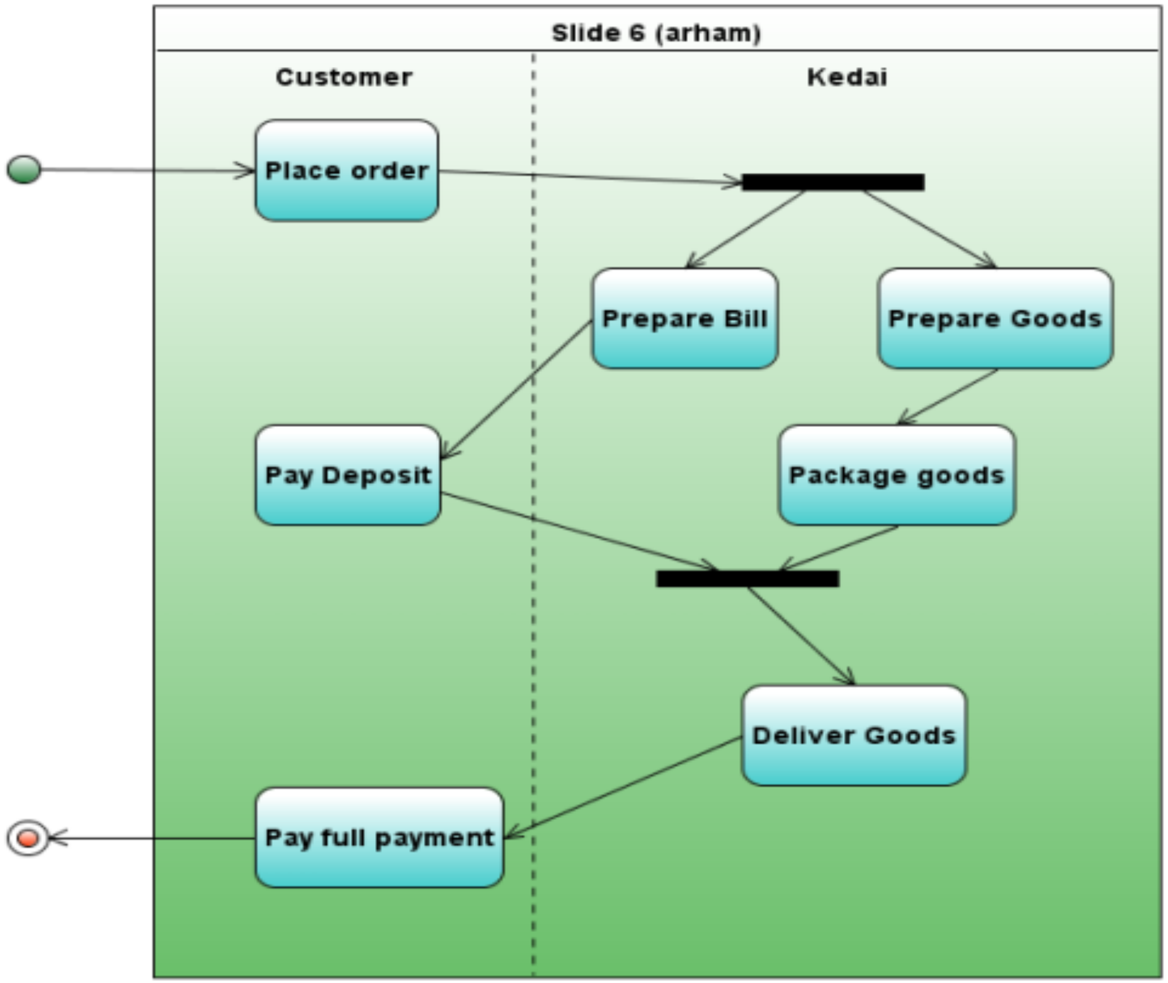
Difference between Activity and State Chart Diagram

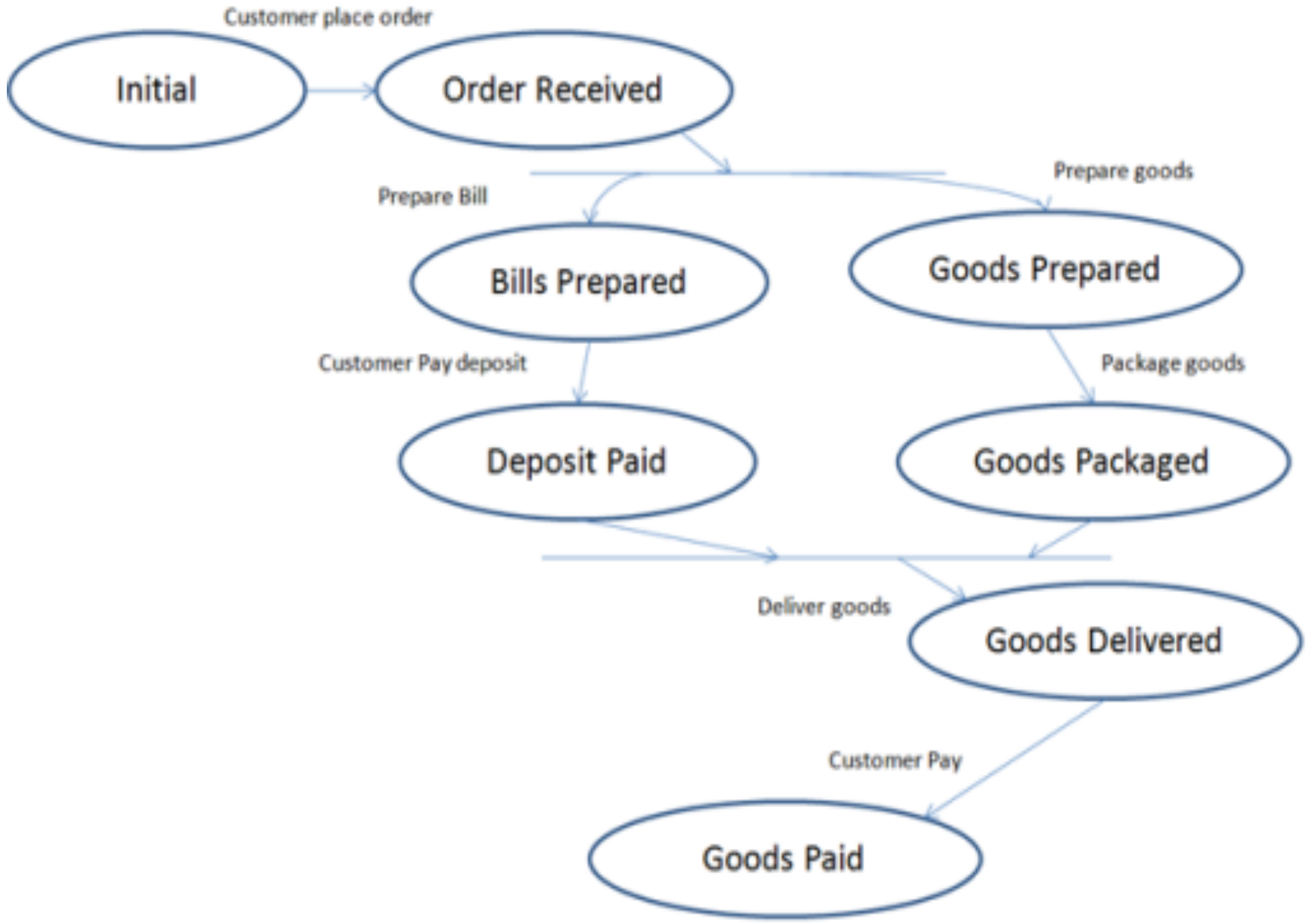
State diagram shows the object undergoing a process. It gives a clear picture of the changes in the object's state in this process.
e.g: ATM withdraw
Card object state: Checking, Approving, Rejecting

Activity diagram is a fancy flow chart which shows the flow of activity of a process.

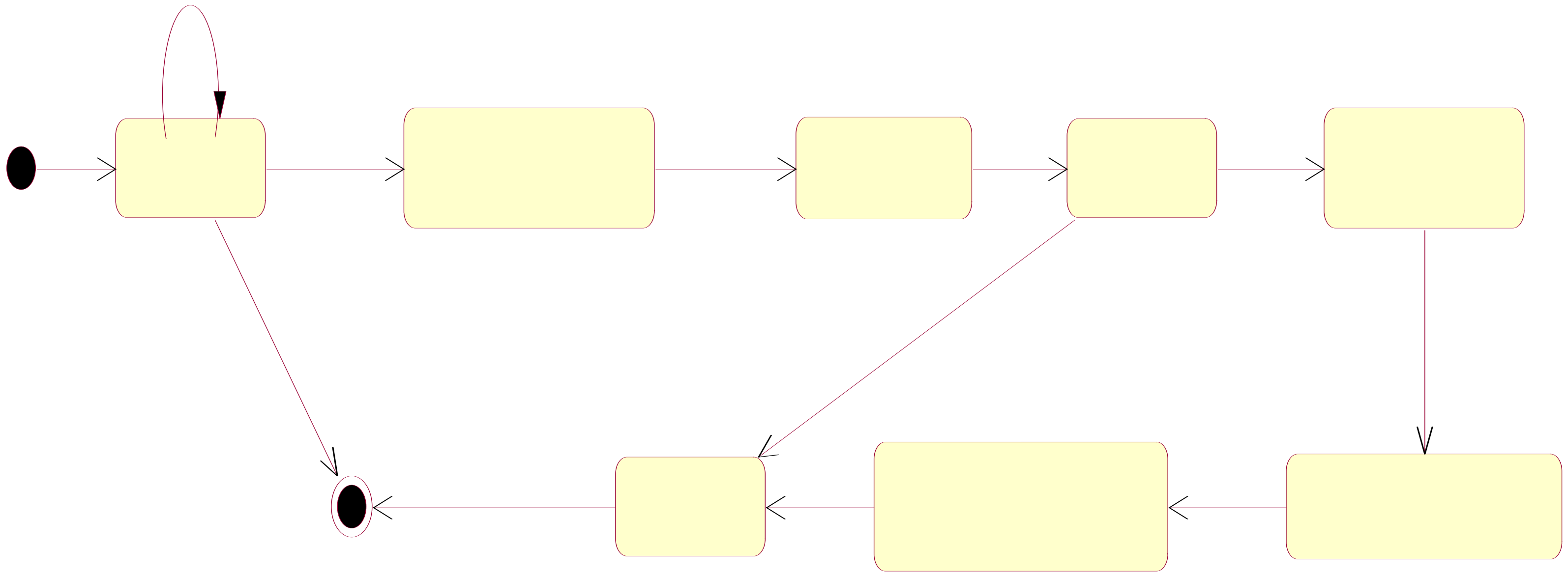
e.g: ATM withdraw
Withdraw activity: Insert Card, Enter PIN, Check balance, with draw money, get card

State chart shows the dynamic behavior of an object.
Activity diagram shows the workflow behavior of an operation as set of actions

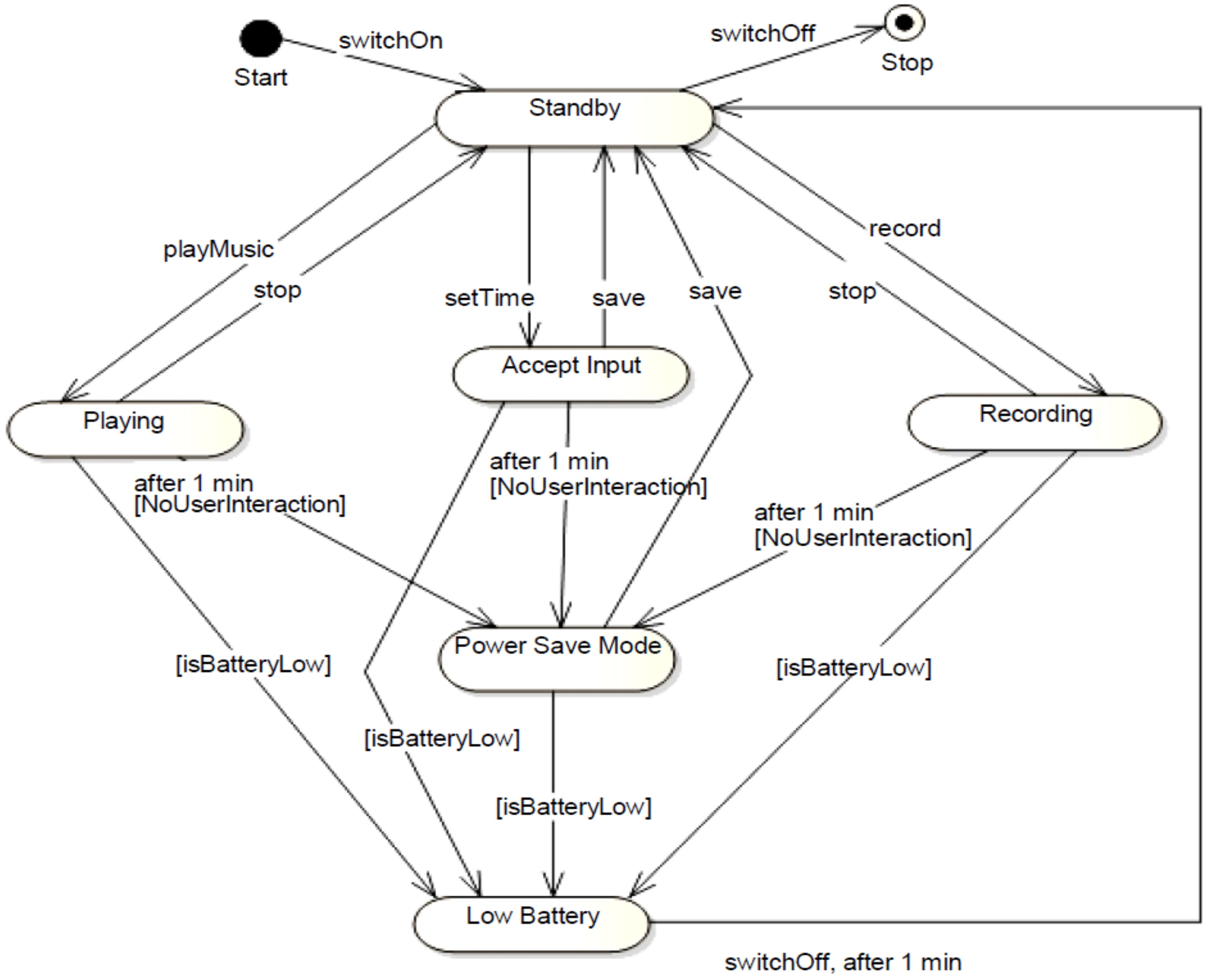

Slide 6 (arham)
Customer
Kedai
Place order
Prepare Bill
Prepare Goods
Pay Deposit
Package goods
Deliver Goods
Pay full payment


Customer place order
Initial
Order Received
Prepare Bill
Prepare goods
Bills Prepared
Goods Prepared
Customer Pay deposit
Package goods
Deposit Paid
Goods Packaged
Deliver goods
Goods Delivered
Customer Pay
Goods Paid

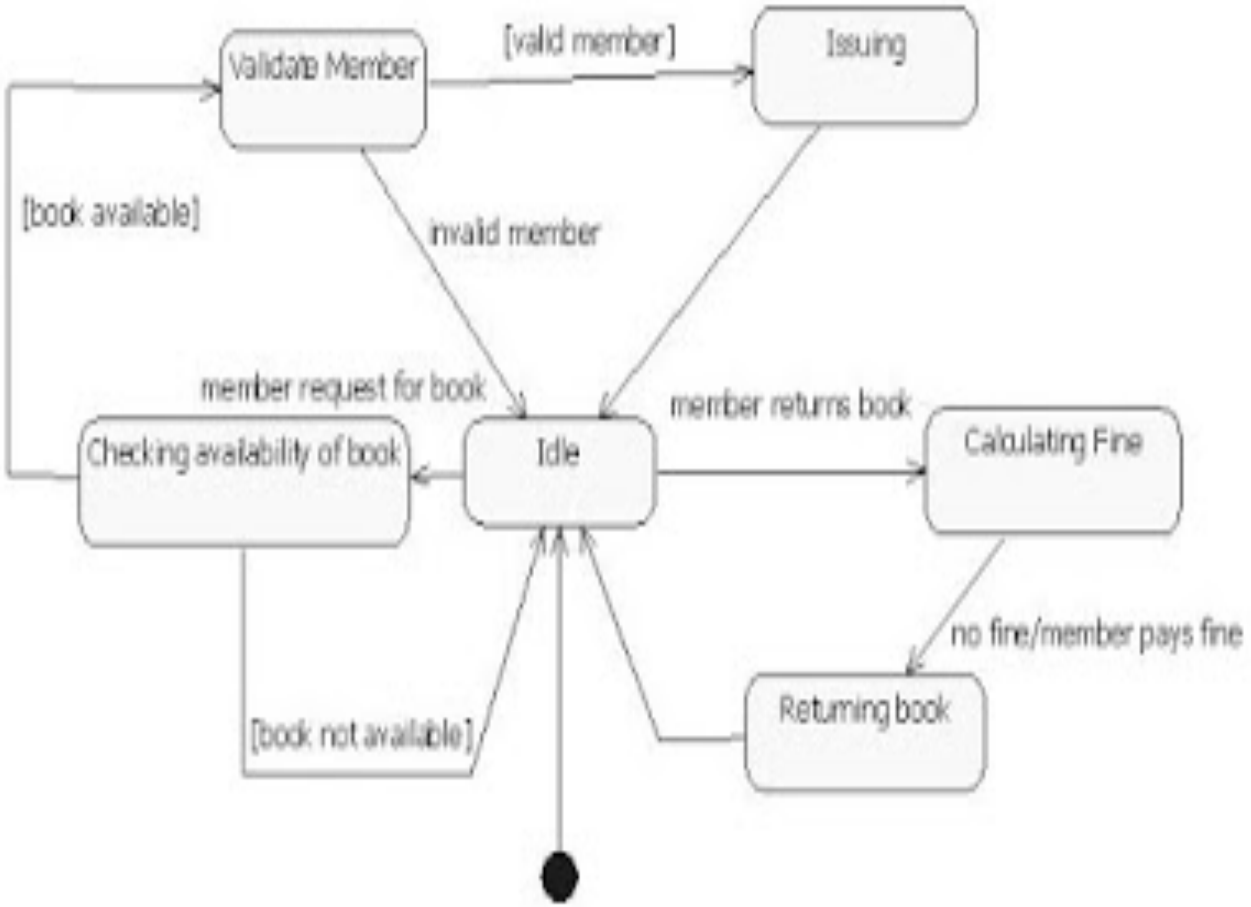
Online Shopping System



Music Player System



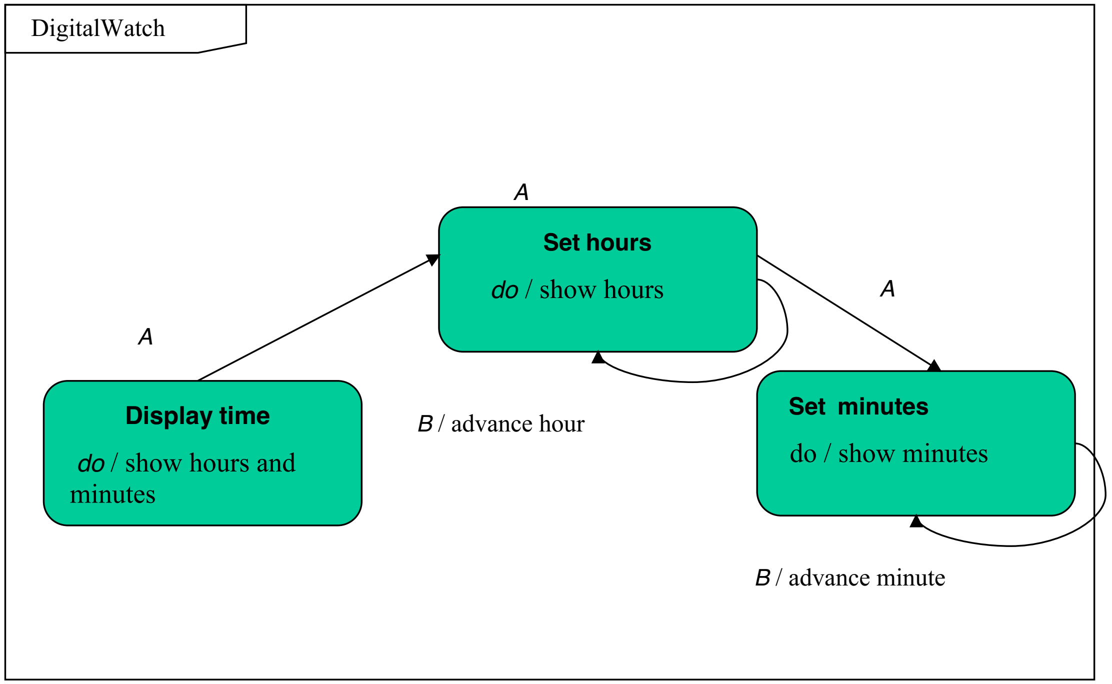
Library Management System



Problem # 1

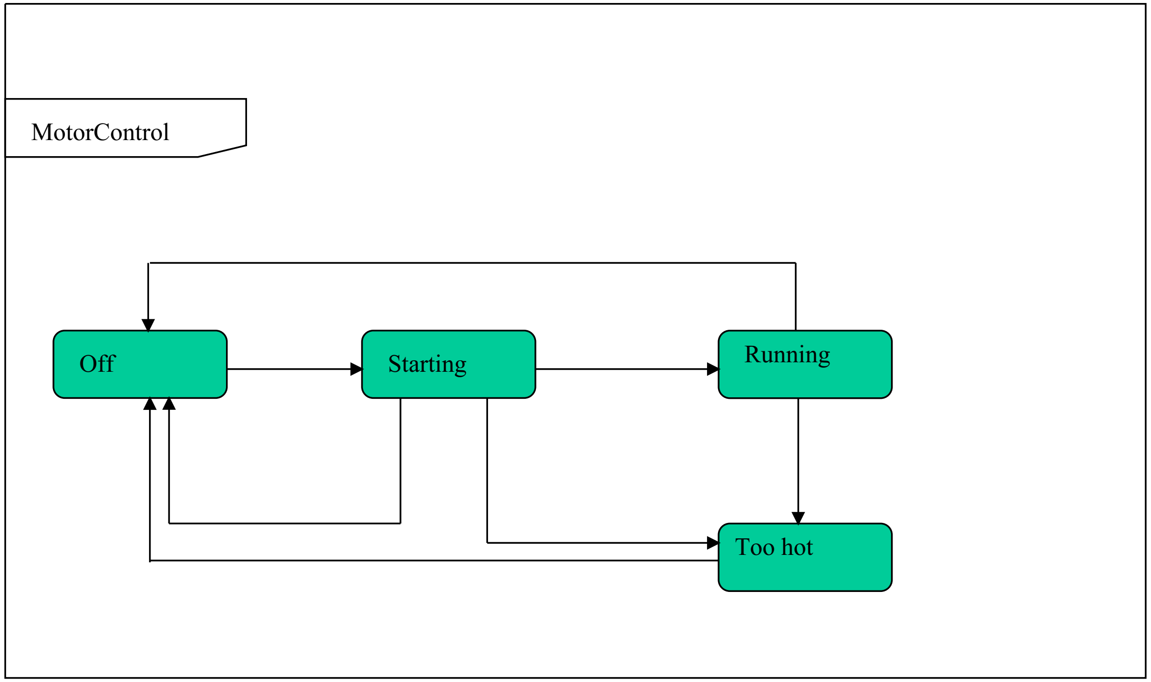
- A simple digital watch has a display and two buttons to set it, the A button and the B button. The watch has two modes of operation, display time and set time.
- In the display time mode, the watch displays hours and minutes, separated by a flashing colon.
- The set time mode has two submodes, set hours and set minutes. The A button selects modes. Each time it is pressed, the mode advances in the sequence: display, set hours, set minutes, display, etc.
- Within the submodes, the B button advances the hours or minutes once each time it is pressed. Buttons must be released before they can generate another event.
- Prepare a state diagram of the watch.

Solution of Problem # 1



Problem # 2

- A separate *appliance control* determines when the motor should be on and continuously asserts on as an input to the motor control when the motor should be running.
- When on is asserted, the motor control should start and run the motor.
- The motor starts by applying power to both the start and the run windings. A sensor, called a starting relay, determines when the motor has started, at which point the start winding is turned off, leaving only the run winding powered. Both winding are shut off when on is not asserted.
- Appliance motors could be damaged by overheating if they are overloaded or fail to start. To protect against thermal damage, the motor control often includes an over-temperature sensor. If the motor becomes too hot, the motor control removes power from both windings and ignores any on assertion until a reset button is pressed and the motor has cooled off.
- **Add the following to the diagram** .
  - *Activities*: apply power to run winding, apply power to start winding.
  - *Events*: motor is overheated, on is asserted, on is no longer asserted, motor is running, reset.
  - *Condition*: motor is not overheated.


MotorControl
Off
Starting
Running
Too hot