

# Object- Oriented Programming & Design

## Part X: UML State Diagrams

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

## State

---

The composite values of all attributes and links held by an object represents the object's *identity* (as every object has a unique identity, we may also have to consider it's physical address).

An object's *state* is derived from the same set of attributes and links.

- Ignore attributes that do not affect the object's behavior.
- No two distinct states of a single object can have identical responses to all events.
- Lump together sets of values that have same response to events.
- External events cause changes in state:
  - State transitions can trigger actions.
  - Transitions are essentially instantaneous.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 2

## State-full Objects

---

- If an object has state-dependent behavior, it may be useful to model its states.
  - Much easier to understand and debug state diagrams than code.
  - Code can be written mechanically from a good state diagram.
- The more an object can be regarded as a pure server, the better it is to make it *stateless*, whereas clients are more likely to have state.
- Common state-full objects:
  - Commands and transactions:
    - » A *WriteCD* command must *readSource*, *writePacket*, *writeTOC*, etc.
  - Controllers:
    - » A *Clerk* in the video store may be *busy* or *free*.
  - Devices:
    - » A *Modem* object could be *dialing*, *sending*, *receiving*, etc.
  - Mutators (objects that change state or role)
    - » A *RentalVideo* is *rented*, *inStore*, or *overDue*

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 3

## State Diagrams

---

- Model the *states* of an object and how it moves from state to state for its entire *lifetime*.
- A class has its own state diagram *if* it has interesting dynamic behavior.

Example: The state of a String class is the ASCII value of the String; this probably doesn't need to be modeled.

Example: The states of a Telephone Connection class (dial tone, dialing, ringing, connected, hung up) is probably interesting enough to model.

- One class' state diagram may refer to the state of another class.
- The complete state diagram of a system is a collection of sub-diagrams that interact by sending events to each other.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 4

## State Diagrams (cont.)

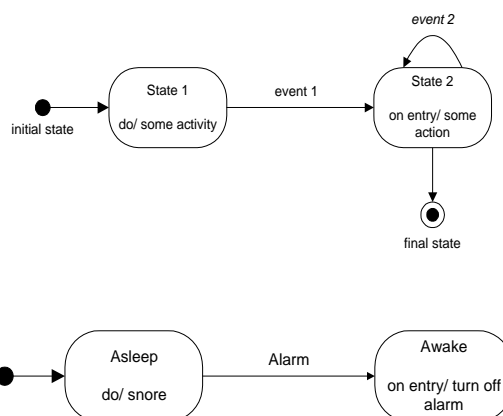
- A type of finite state machine.
- Best for modeling one class whose behavior depends on its state.
- States are represented by ovals.
  - A state may or may not have a name.
- Directed arcs between states represent transitions associated with events.
- The *source* of the event is not specified.
- Not ideal for situations involving many collaborating objects.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 5

## State Diagram Symbols



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 6

## Events

---

- Stimuli that affect an object.
- Drive objects from one state to another.
- Primary means of communication between objects.
- Events correspond to operations and/or the modification of attributes.

User interfaces are “event driven” where the “event” in this case is a key on the keyboard being pressed, or a mouse click. It is often useful to make a state diagram for the user interface.

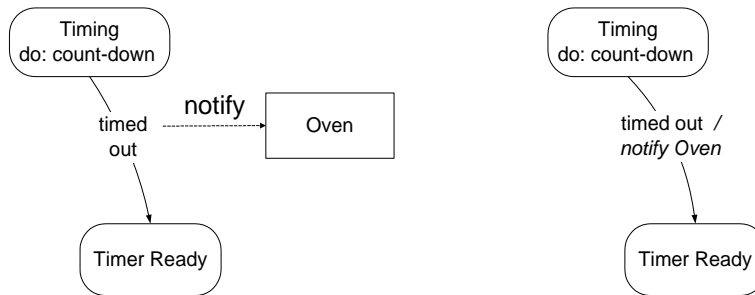
## Activities and Actions

---

- Activities are operations that take time:
  - Generating microwaves.
  - Writing to a disk.
  - Can often be modeled as nested state diagrams.
- Actions are of very short duration:
  - Beep.
  - Display a menu.
  - Set a flag.
- Both can occur within a state (as a side effect).

## Action on a Transition

Example from Microwave Timer:



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 9

## Model Syntax

The general syntax for *arc adornments*:

- **event1 ( attributes ) [ conditions ] / actions**
- IF event1 occurs AND IF the conditions are true THEN make the state transition specified by the arc AND spawn the specified action(s).

Keywords used within a state:

- Actions can be spawned on *entry* and *exit* to states:
  - » entry / entry-action(s)
  - » exit / exit-action(s)
- Activities may be indicated:
  - » do / activity(ies)
- Internal events may be indicated (instead of self-directed arcs):
  - » internal-event1 / action(s)

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 10

## Refinements

If an object is an aggregation of other objects, it is possible to have concurrent state machines, one for each of the aggregate objects. Arc conditionals can refer to the states of the other aggregate objects.

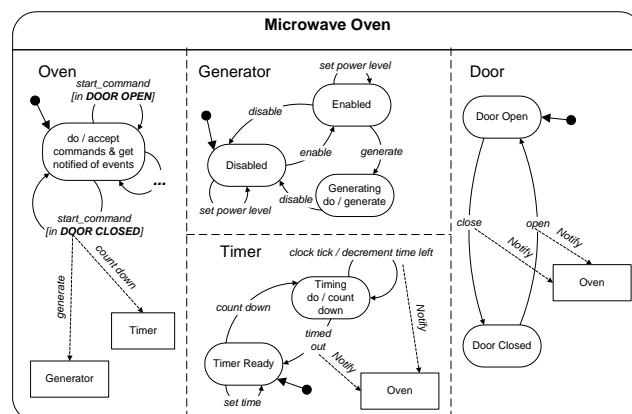
- For example, a microwave oven can have concurrent state models for the timer, the generator, and the door. The “push start button” event will have no effect if the state of the door is *open*; if the door is *closed*, however, then a message is sent to both the timer and the generator.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 11

## Example: Microwave Aggregation



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 12

## Refinements (cont.)

Sometimes an object must perform two or more activities concurrently, both of which must complete before the next state can be reached; this is called “splitting and synchronization”.

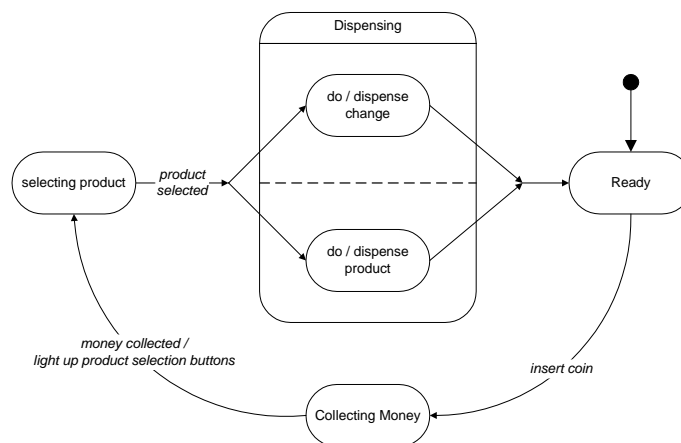
- For example, a vending machine might have to dispense both change and product before being ready for its next transaction.
- In the model on the next slide, note how the two concurrent “dispense” states can be combined into a “superstate” (also known as a generalization relationship).

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 13

## Splitting and Synchronization



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 14

## Refinements (cont.)

- It is possible to decrease the visual complexity of a model by generalizing activities, actions, events and states as higher-level, more abstract elements, and then showing them broken down in separate diagrams.
- For example, the vending machine state “collecting money” is really more complex than it appears from the previous slide...
- This is also useful when the same event(s) cause the same action(s) for multiple states... it becomes possible to generalize.

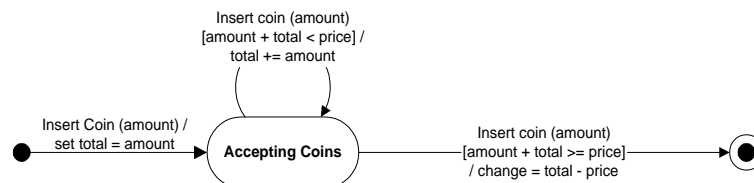
Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 15

## Example: Vending Machine

### Collecting Money



Consider also the Sticks Game Referee...

There could be a high level “Running Game” state...

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

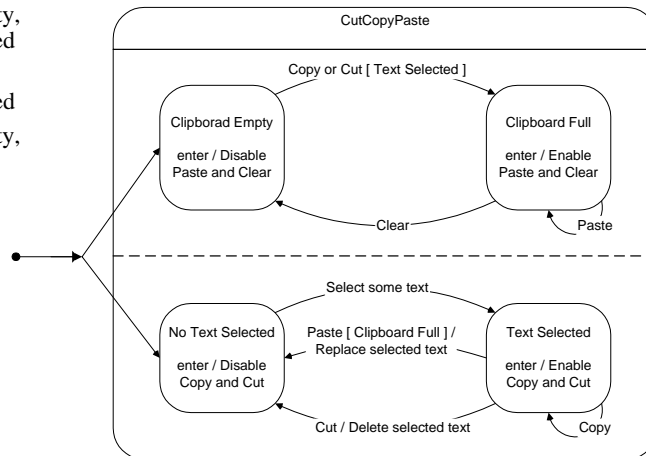
( X ) UML State Diagrams - 16



## Cut, Copy, and Paste (cont.)

This can also be done using the states:

- Clipboard Empty, No Text Selected
- Clipboard Full, No Text Selected
- Clipboard Empty, Text Selected
- Clipboard Full, Text Selected



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 19

## Implementing States

- Tables
  - Very common procedural approach.
  - Compact and efficient.
  - Good for large state machines.
  - Can be difficult to maintain.
  - Requires good documentation.
- Switch statements
  - Easier to understand than tables.
  - Tends to be self-documenting.
  - Can be difficult to maintain sometimes.
- *State* design pattern
  - Easy to extend and modify.

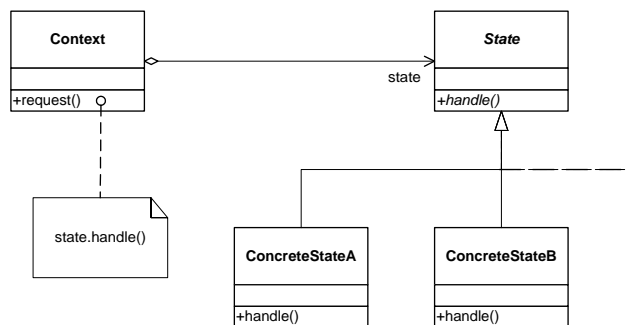
Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 20

## Design Pattern: *State*

**Intent:** Allow an object to alter its behavior when its internal state changes. The object will appear to change its class from the point of view of the calling client (meaning: its behavior will change).



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 21

## *State Design Pattern (cont.)*

- How does a *ConcreteState* know what state to go to on a transition?
  - Each class can have its own table or switch statement, or a hash table of transitions keyed by their trigger Events (and guard conditions).
  - Consider using *State*, *Action*, *Event* and *Transition* classes.
  - Note: The *Action* class might be implemented using the *Command* pattern.



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 22

## Example: Customer Account

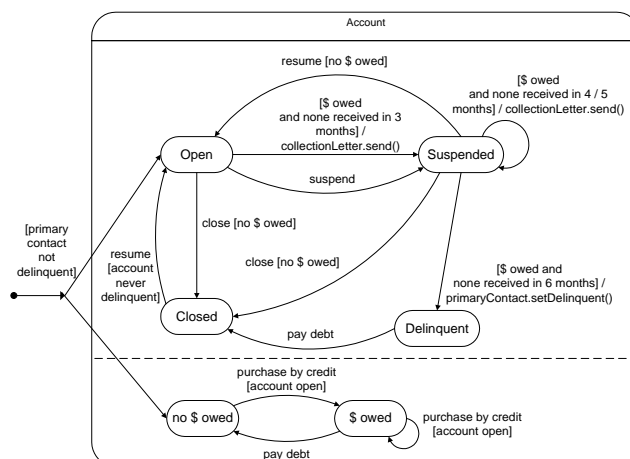
- A “customer care” application is used by Customer Service Representatives (CSRs) who talk to customers over the telephone. A given call consists of a conversation between a CSR and a contact person representing a customer account, creating “service requests”, such as: purchase goods on credit, receive payment, and “change account status” requests to open, close, suspend, or resume an account. Purchases may only be made on open accounts. An account cannot be closed if there is money owed. A suspended account may have service resumed only if there is no money owed. The only valid service request for a closed account is resume. An account will automatically get suspended if money is owed and none has been received for 3 months, in which case a collection letter will also be sent. More collection letters will be sent if no money has been received after 4 & 5 months; after 6 months, the account becomes delinquent, and may never be reopened, but of course the customer can pay his or her debt to close the account; furthermore, the primary contact for a delinquent account may never again open an account.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 23

## Example: Customer Account (cont.)



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 24

# Activity Diagram

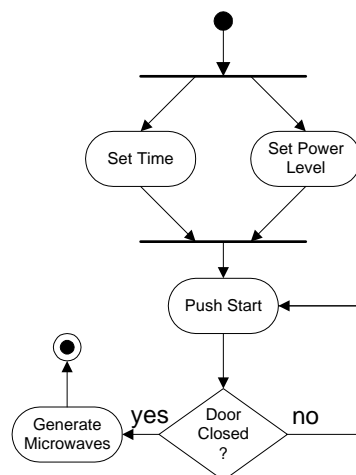
- Similar to a State Diagram except that the states in an Activity Diagram are activities representing the performance of operations. A transition from one state to another is triggered by the completion of the operation.
- Activity Diagrams enable visualization of an activity splitting into parallel activities with subsequent synchronization.
- Use Activity Diagrams to model process flows. In fact, the UML's Activity Diagram is quite similar to the well known "flow chart".

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 25

# Activity Diagram for Microwave Oven



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 26

## Example: E-Commerce Web Site

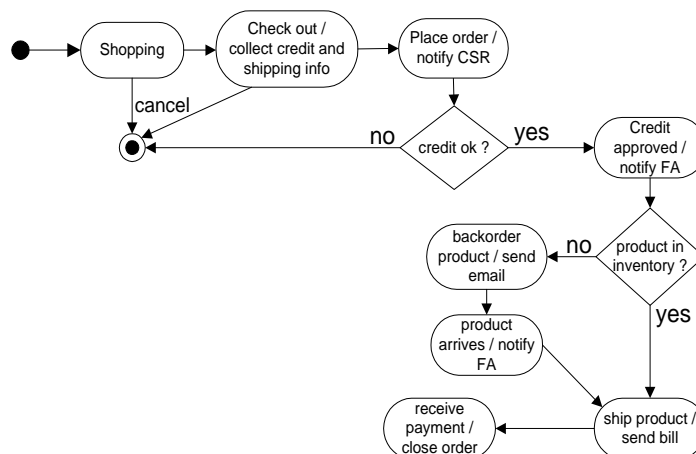
- An e-commerce web site allows users to browse a catalog, add and remove selected items from a shopping cart, then purchase the items with a credit card. Once the order has been placed, it goes to a customer service representative (CSR) who verifies that the credit card information is accurate and has enough credit to cover the purchase. If the charges are authorized, the order goes to a fulfillment agent (FA) who checks the order against available inventory. If the inventory is available, the order is shipped and billed. If not, the order is backordered, and the FA will get notified when the product finally arrives, and the product will then get shipped. When payment is received for a billed order, the order is closed. Note: if a customer only browses the site and leaves, without ever placing an order, no computer record of the customer is maintained.
- Draw a UML Activity Diagram for this process. Note: this diagram might evolve into a State Diagram for the Order class.

Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 27

## E-Commerce Web Site (cont.)



Copyright © 1996 - 2008

David Leberknight & Ron LeMaster. All rights reserved.

( X ) UML State Diagrams - 28

# Summary

---

**State Diagrams** - Ideal for one class when it has interesting dynamic behavior depending on its state.

**Activity Diagrams** - Like a flowchart except better. Activity Diagrams are excellent for modeling business workflows. These diagrams enable visualization of parallel activities and their sequencing and synchronization.

**Code** - With comments also may be used to document stateful behavior.