State Diagrams, Actions, and Activities

• Basic Concepts
• The State Diagram
• State
• Event
• Transition
• Additional Concepts and Notations
• Hierarchical States
• Action
• Activity
• Action Language(s) for UML
Basic Concepts

• We are now taking a deeper look at system dynamics

• This will be a low-level view
  - We will be looking inside the (classes of) objects themselves

• Some of the dynamic behavior will be specified in terms of sequencing / timing

• Some of the dynamic behavior will be specified in terms of functions (transformations / computations)
The State Diagram

• We will use the state diagram to specify the sequencing / timing behavior of objects in a class
  - States
  - Events
  - Transitions

• Generally speaking, there should be one state diagram for every class
  - But this is not prescribed by UML, it allows state diagrams to describe the system at any level
State

• State
  - A state represents a discrete, continuous segment of time wherein the object’s behavior will be stable
  - The object will stay in a state until it is stimulated to change by an event

• Notation

\[\text{Closed}\]
Event

• An event is an instant in time that may be significant to the behavior of the objects in a class
  - Events can have associated arguments

• Events tend to represent
  - Commands or requests from other objects
  - Significant times (it’s time to...)
  - Circumstances or happenings in other objects (the temperature monitor notices the temperature rising over a safety setpoint)
  - “Custodial” (creation, deletion, simple update)

• Notation
  - Events are written simply as text strings

  Open
  Deposit(Amount)
  Withdraw(Amount)
  Close
Transition

• A transition shows a valid progression in state
  - Simply, “if you were in this state and you saw this event, that’s the state you would end up in”

• Examples
  - If a Bank Account was Closed and it saw an Open event, it would end up in the Opened state
  - If the account was Opened and it saw a Close event it would end up in the Closed state

• Notation

  [Diagram of state transitions]

• As far as analysis is concerned, we can say that a transition takes place in essentially zero time regardless of how complicated actions on that transition (below) are
Additional Concepts and Notations

• Initial State
  - The initial state (there can be only one) is the state that a new object will be in immediately following its creation

• Final State
  - A final state (there can be many) is a state that represents the object going out of existence

• Self Transitions
  - Sometimes an object is required to perform some action (below) when it recognizes an event, but it ends up in the same state it started in
    * Technically, it never really leaves the state
Later, we can put some action on the self-transition.
Additional Concepts and Notations
(cont)

- **Guarded Transitions**
  - A guarded transition is a shorthand notation that says “in addition to the event happening, the guard condition must also be true for the transition to take place”
  - When the same event causes multiple transitions out of some state, the guards should be mutually exclusive

- **Unlabeled Transitions**
  - An unlabeled transition means the transition is taken when the activity (processing, below) completes
Some State
Do/Some Activity

Some Other State
Hierarchical States

- UML uses the StateChart notation originally developed by Harel [Harel87]

- Superstates
  - You may find a set of states that have a common response (transition) to a particular event

![Diagram 1](image1)

- Concurrent State Diagrams

![Diagram 2](image2)

[Harel87]
Actions and Activities

• Actions and activities are used to specify the functional (transformational / computational) behavior of objects in a class
  - Actions
  - Activities
Action

• An Action is the UML way to specify that some discrete amount of work gets done as an object makes a transition
  - The work is expected to be a one-shot computation

• Notation
  - Append “/action-name” to the “event[guard]” for every transition that has an action

• An alternative notation to self-transitions is to put event-name(arg-list)/action-name in the lower compartment of a state

• When you want to be sure that every entry into, or every exit out of, some given state has the same action then you
can put entry/action-name or exit/action-name in the lower compartment of a state
Activity

• An Activity is the UML way to specify that some relatively long-term amount of work gets done while an object is in a state
  - The work is continuous and interruptible (it stops when you exit the state)

• Notation
  - Compartmentalize the state
  - Include “do/activity-name” in the lower compartment of every state that has an activity

```
Driver Control

Cruise[Speed>30] /SaveSpeed
Resumse [Speed>30]
Brake

Cruise Control

do/MaintainSpeed

Accelerate
End Accelerate

Accelerating

do/MaintainSpeed
do/IncreaseSpeed

Brake

Accelerate
```

States, Actions, & Activities (14-Jan-01)
**Action Language(s) for UML**

- The OMG is currently involved in extending UML to provide precise, software-platform independent languages for specifying the details of actions and activities.

- The language(s) would be partially algorithmic (and thus be a step into design), but would be very high-level:
  - No traditional data structures (tables, arrays, linked lists), just values and collections
  - No traditional control structures (for-next, ...)
    functions would be applied to an entire input set

- Languages of this sort would enable
  - Executable analysis models
  - Complete code generation from analysis models
  - Formal proofs-of-correctness of analysis models

- See also [Mellor98]
[Mellor98]
Key Points

• We are now taking a deeper look at system dynamics

• This is a low-level view
  - Looking inside the (classes of) objects themselves

• State diagrams specify the sequencing / timing behavior of objects in a class
  - States
  - Events
  - Transitions

• Generally speaking, there should be one state diagram for every class

• Actions and activities specify the functional (transformational / computational) behavior of objects in a class
  - Actions
  - Activities