Object Oriented Methodology

Week – *8*, Lecture – *1*

Creating State Diagrams

SAURABH SRIVASTAVA

VISITING FACULTY

IIIT LUCKNOW

Structural vs Behavioural Modelling

Recall that a system's details have two dimensions – Structural and Behavioural

The Class Diagram describes certain structural details of the system …

- … i.e., the classes that constitute the system, and how they connect to each other !!

When the system gets implemented, the Classes just become templates

- The actual execution involves operating over data – transforming it from one form to another
- The data, in turn, is encapsulated in objects if the system is built over OO principles
- One way to define the expected behaviour of the system is to provide details of data transformations

A State Diagram can be helpful in showing these transformations

- They show how particular objects transform when the system becomes operational

States of an Object

What moods could you be at any point of time?

- Happy, Sad, Angry, Relaxed etc.
- Q. Does your mood affect your behaviour?
- A. Probably yes :D

An object too, could be in one of the many possible *states*

- The state of an object is just the collection of the values of its fields
- The actual number of possible states, thus, is a Cartesian Product of all the fields of the object

Note that not all the states may be "valid" for an object, from the perspective of the domain

- For instance, consider the example of an object of type `Transaction`, signifying a banking transaction
- Assume that it has three fields – `old_balance`, `new_balance` and `transaction_amount`
- Now, any state where `new_balance` is not equal to `old_balance + transaction_amount` …
- … can be considered as an example of "invalid" state for the `Transaction` object

Transitions and Events

A *transition* represents a change in the state of an object

The state of an object may change due to multiple reasons

- For instance, it may be because the values were changed because of a user input …
- … or, because an operation scheduled at a certain point in time gets triggered

Overall, a transition is brought upon the occurrence of an *event*

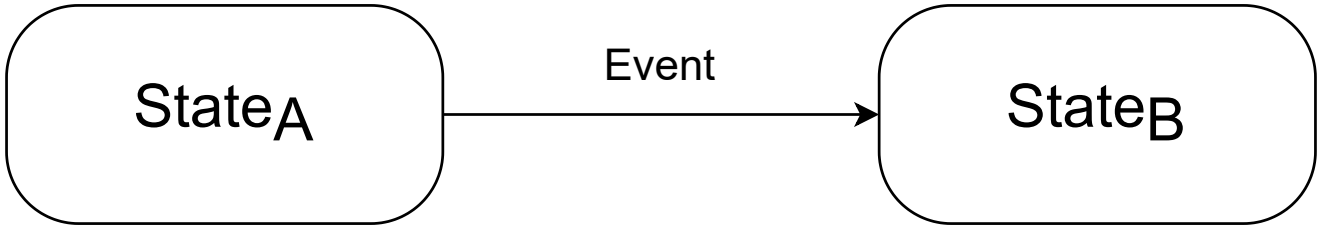
- Examples of events could be user clicking on a button …
- … or, an alarm ticking off at a pre-defined time

To summarise, for a particular object, we can say that

- … the *state* of an object sees a *transition* to another state on the occurrence of some *event* …

A State Diagram shows these transitions for specific objects in the system

Showing Simple State Transitions



A transition in a State Diagram (the arrow represents the transition)

Guard Conditions and Effects

An event may be something that is not directly associated with a single object

- Their occurrence may affect multiple objects

Sometimes, transitions may be ”conditional” in nature

- For example, for an object of type `Account`, a relevant event could be `request_debit`
- However, the debit request may only change the state of the account, if the debit amount is valid
- Otherwise, the event is simply ignored

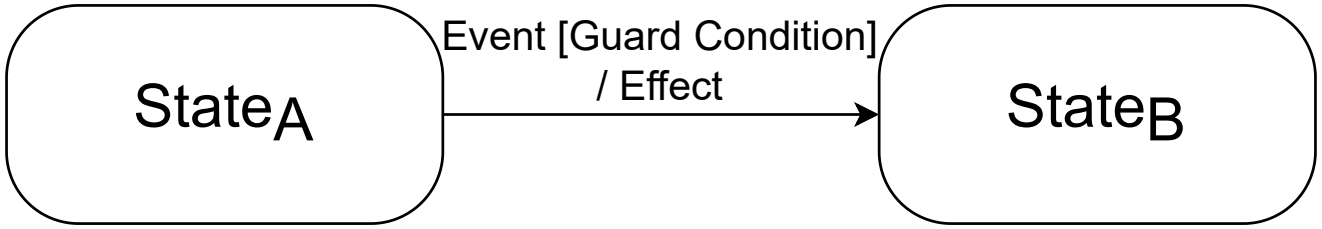
A *guard condition* can be specified along an event to signify that the transition is conditional

- It is provided in the form of a Boolean expression in square brackets, next to the event
- The transition happens when the event occurs, as well as the guard condition is true

Almost all transitions have a meaning, with respect to the domain

- For example, a state change on a debit request signifies deduction of some amount from the Account
- Such information can be indicated via an *effect*, written after the event, separated by a forward slash

More Complex Transitions



A transition with a guarded condition and an effect

Initial and Final States

Every object goes through a lifecycle

- It starts with the call to the constructor, and ends with the call to the destructor

When an object is created, all the fields get some starting values …

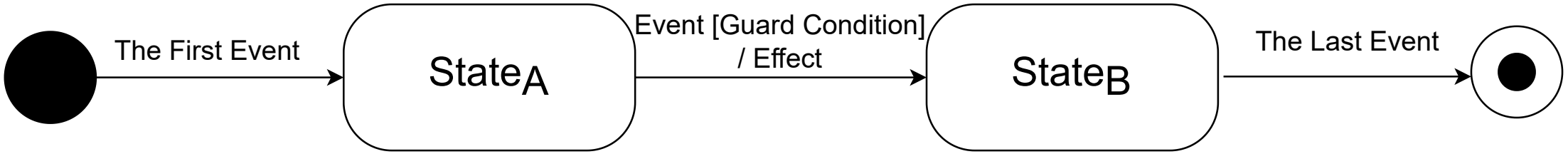
- This usually happens inside some constructor

This state may or may not be of relevance with respect to the domain

- For instance, there may be a process which assigns some values to the fields, prior to its usage
- However, to show this assignment event, you need at least two states (as the ends of the transition)
- In such a case, you may use a special symbol to signify the *initial state* of the object

Similarly, if a transition takes an object to a state, after which it doesn't change …

- … e.g., right before the call to the destructor; this state can be considered as the *final state* for the object
- There is a special symbol for this state too



Transitions with initial and final states

State Diagrams

If you decide to use State Diagrams in your project, you may have to …

- … pick a subset of objects over others – based on the severity of their task on the domain, and
- … decide upon the states of an object that is of importance

There is no rule-of-thumb for picking such objects – it is specific for a project

Similarly, an object may be in one of the many possible states

- You should ideally show only those states in a State Diagram which are of higher importance

UML State Diagrams represent these transitions in detail

- You can compose a State Diagram for a particular object

However, one State Diagram, shows the effect of the environment on one single object

- It does not, for example, shows the interaction between different objects
- We will see how to use the Interaction Diagrams later

Homework !!

Gain more knowledge of the State Diagrams

- Also, you can now watch another part of this tutorial https://www.youtube.com/watch?v=WnMQ8HlmeXc
- See the section on State Machine Diagram

Prepare State Diagrams for important objects of your project