Object-Oriented Software Analysis and Design

School of Computer Science
University of Windsor

UML State Machine Diagrams

- A UML state machine diagram, illustrates the interesting **events** and **states** of an object, and the **behavior** of an object in reaction to an event.
- **Transitions** are shown as arrows, labeled with their event.
- **States** are shown in rounded rectangles.
- It is common to include an **initial pseudo-state**, which automatically transitions to another state when the instance is created.

UML State Machine Diagrams (contd.)

- A state machine diagram shows the lifecycle of an object:
  - what **events** it experiences,
  - its **transitions**, and
  - the **states** it is in between these events.
- *It need not illustrate every possible event*; if an event arises that is not represented in the diagram, the event is ignored as far as the state machine diagram is concerned.
- Therefore, we can create a state machine diagram that describes the lifecycle of an object at arbitrarily **simple or complex levels of detail, depending on our needs**.

UML State Machine Diagrams: Example

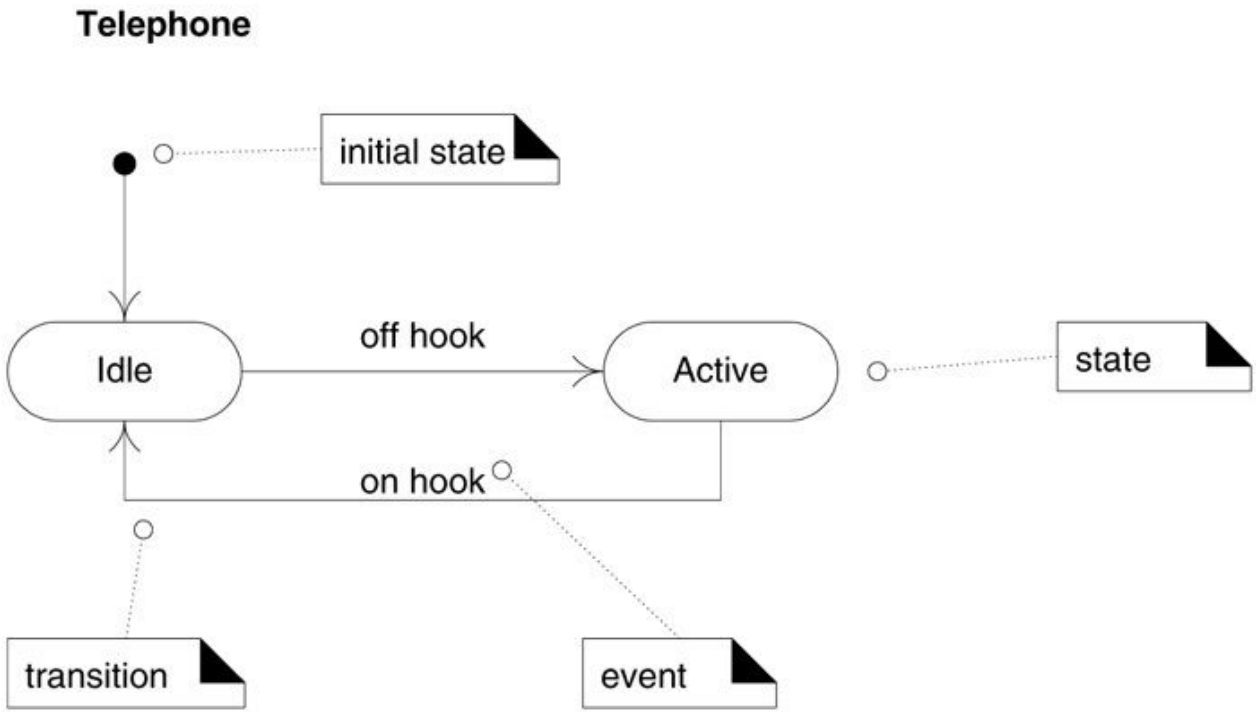

Figure: *State machine diagram for a telephone.*

UML State Machine Diagrams: Definitions (1)

- An **event** is a significant or noteworthy occurrence.

  For example,

  A telephone receiver is taken off the hook.

UML State Machine Diagrams: Definitions (2)

- A **state** is the condition of an object at a moment in time—the time between events.

  For example,

  A telephone is in the state of being "idle" after the receiver is placed on the hook and until it is taken off the hook.

UML State Machine Diagrams: Definitions (3)

- A **transition** is a relationship between two states that indicates that when an event occurs, the object moves from the prior state to the subsequent state.

  For example,

  When the event "off hook" occurs, transition the telephone from the "idle" to "active" state.

How to Apply State Machine Diagrams? State-Independent Objects

- If an object always responds the same way to an event, then it is considered state-independent (or modeless) with respect to that event.

  For example,

  if an object receives a message, and the responding method always does the same thing. The object is state-independent with respect to that message.

How to Apply State Machine Diagrams? State-Dependent Objects

- By contrast, state-dependent objects react differently to events depending on their state or mode.

  For example,

  a telephone is very state-dependent. The phone's reaction to pushing a particular button (generating an event) depends on the current mode of the phone off hook, engaged, in a configuration subsystem, and so forth.

- It's for these kind of complex state-dependent problems that a state machine diagram may add value to either understand or document something.

How to Apply State Machine Diagrams? State-Independent and State-Dependent Objects: Guideline:

- Consider state machines for state-dependent objects with complex behavior, not for state-independent objects.
- In general, business information systems have few complex state-dependent classes. It is seldom helpful to apply state machine modeling.
- By contrast, process control, device control, protocol handlers, and telecommunication domains often have many state-dependent objects. If you work in these domains, definitely know and consider state machine modeling.

How to Apply State Machine Diagrams? Modeling State-dependent Objects (1)

- Broadly, state machines are applied in two ways:
  1. To model the behavior of a complex reactive object in response to events.
  2. To model legal sequences of operations protocol or language specifications.
     - This approach may be considered a specialization of #1, if the "object" is a language, protocol, or process. A formal grammar for a context-free language is a kind of state machine.

How to Apply State Machine Diagrams? Modeling State-dependent Objects (2)

- The following is a list of common objects which are often state-dependent, and for which it may be useful to create a state machine diagram:
  - Complex Reactive Objects
  - Protocols and Legal Sequences

How to Apply State Machine Diagrams? Modeling State-dependent Objects- Complex Reactive Objects

- Physical Devices controlled by software
- Transactions and related Business Objects
- Role Mutators

How to Apply State Machine Diagrams? Modeling State-dependent Objects- Protocols and Legal Sequences

- Communication Protocols
- UI Page/Window Flow or Navigation
- UI Flow Controllers or Sessions
- Use Case System Operations
- Individual UI Window Event Handling

Transition Actions and Guards

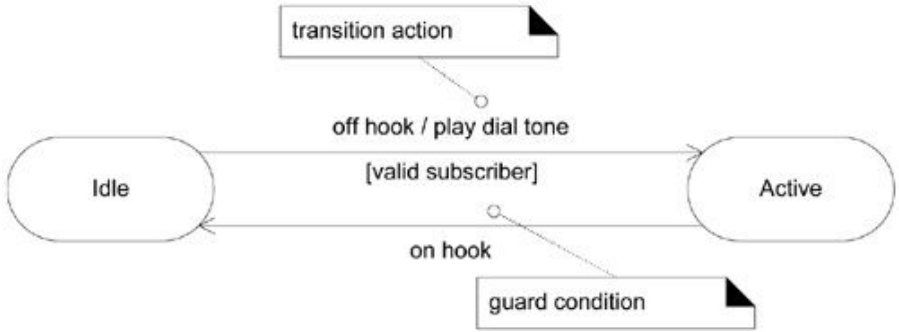

Figure: *Transition action and guard notation.*

- A transition can cause an action to fire. In a software implementation, this may represent the invocation of a method of the class of the state machine diagram.
- A transition may also have a conditional guard or boolean test. The transition only occurs if the test passes.

Nested States

- A state allows nesting to contain substates; a **substate** inherits the transitions of its **superstate** (the enclosing state).

- Substates may be graphically shown by nesting them in a superstate box.

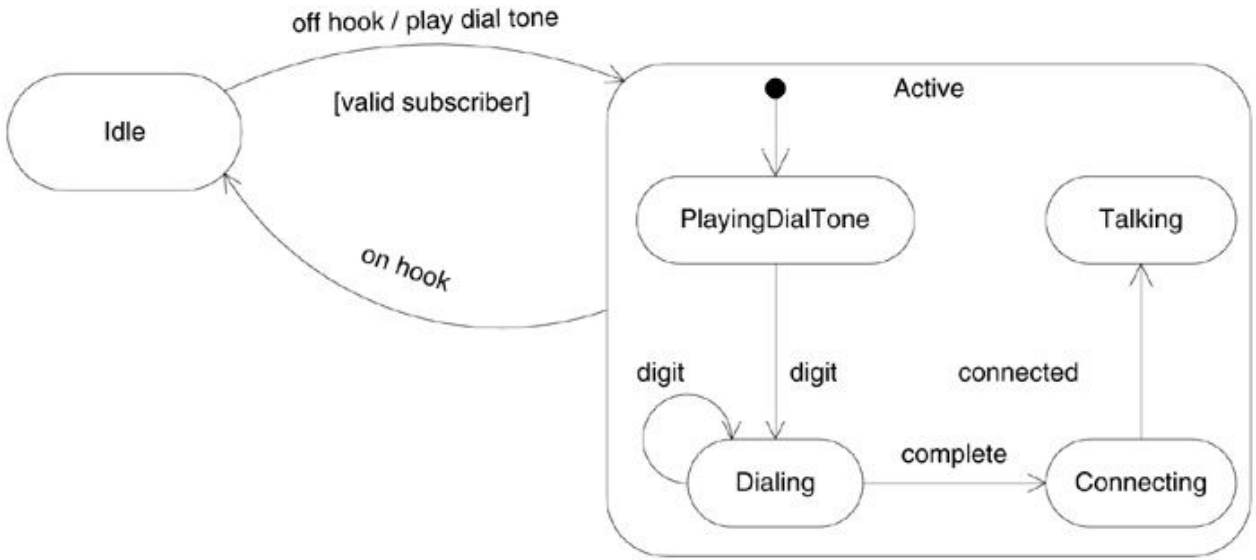

Figure: *Nested states.*

UML State Machine Diagrams: Example (UI Navigation Modeling)

- Some UI applications, especially Web UI applications, have complex page flows.

- State machines are a great way to document that, for understanding, and a great way to model page flows, during creative design

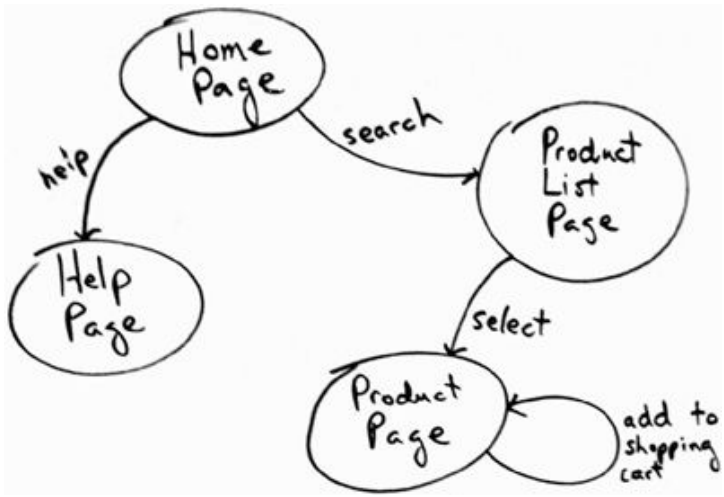

Figure: *Applying a state machine to Web page navigation modeling.*

UML State Machine Diagrams: Example (NextGen POS Use Case)

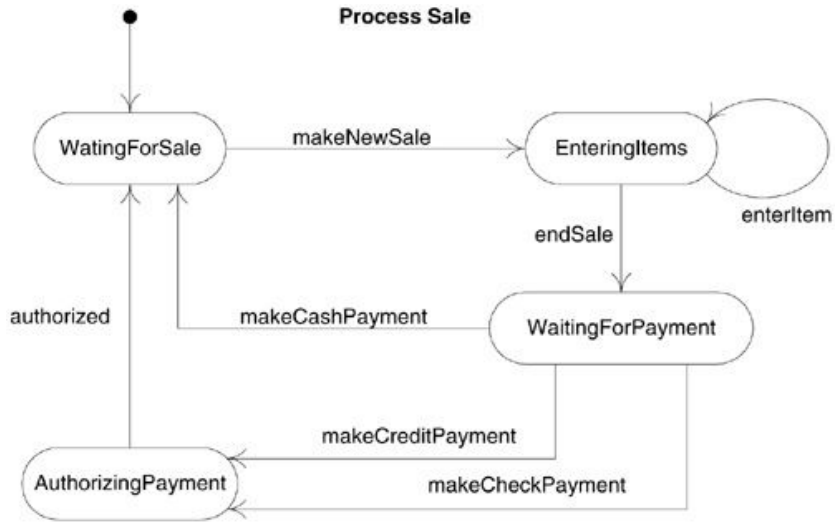

Figure: *A sample state machine for legal sequence of use case operations.*

It's Quiz Time

1. A ..... is a relationship that represents the movement of an object from one state to another state.
   1.1 event
   1.2 state
   1.3 transition

2. An ..... is something that takes place at a certain point in time and changes a value or values that describe an object, which, in turn, changes the object's state.
   2.1 event
   2.2 state
   2.3 transition

3. A transition may also have a conditional guard or boolean test. The transition only occurs if the test passes. (True or False)