State Machines and Statechart Diagrams

Hour 8

Events
“Things that happen”
- An event is a significant occurrence in space and time
  - Calls
  - Signals
  - Passing of time
  - Change of state
- Event characteristics
  - In relation to the system (space)
    - Internal pass among the objects composing the system
    - External pass between the system & its actors (often users)
  - In relation to locus of control (time or processes and threads)
    - Synchronous sender waits for receiver to return
    - Single task (process, thread, etc.)
    - Asynchronous sender continues running without waiting for the receiver
    - Multitasking (multiple processes, threads, etc.)

Calls
Invoking an operation
- A call is the dispatch of an operation (one object calls or invokes the operation – function or method – of another)
- Operation is declared in the class diagram and is handled by a function or method
- Synchronous calls (most typical)
  - Control passes from the sender to the receiver
  - Receiver executes an operation, may transition to a new state
  - Control returns to the sender
- Asynchronous calls start new processes or threads
  - Sender does not wait for the receiver
  - Sender and receiver may be any object
  - No visual difference between call and signal

Signals
Denoting an occurrence with an object
- A signal is an object dispatched (thrown) asynchronously by one object and received (caught) by another
- Sender continues running (does not stop or wait for receiver)
- Handled by receiver’s state machine
- An exception is an internal signal
- A Java control event (e.g., ActionEvent) is an external signal
- Signals are a specialized classes
  - May be instantiated
  - May participate in inheritance or generalization
  - May have attributes and operations (a signal’s attributes are denoted by its parameters – when a signal is sent, you may specify a value for its attributes as parameters)
- Sender and receiver may be any object

Time and Change Events
Triggering one-object transitions
- A time event represents the page of time
  - after (expression)
  - Time starts (unless otherwise specified) when entering the state
- A change event represents an object changing state or the satisfaction of some condition
  - when (time) or when (condition)
  - Although it models a continuously tested condition, it can usually be implemented with periodic testing
- Time and change events are asynchronous
- Involves only one object – the receiver

Modeling Exceptions
Sending signals
- Model as stereotyped classes
- Specify the kinds of exceptions an object may throw through its operations
Modeling Signals

Receiving signals

- Model as stereotyped classes
- Specify the kinds of signals an active object may receive

State Machines

Definitions

- State machine: models the conditions in which an object may be and how it changes conditions
- State: a condition of an object during which it satisfies a condition, performs an activity, or waits for an event
- Event: some occurrence that triggers a state transition
- Transition: a relation between two states indicating that an object in the first state will perform certain actions and enter the second state when a specified event occurs and specified conditions are satisfied
- Activity: ongoing nonatomic execution
- Action: executable atomic computation or execution

States

Definitions

- Name
- Entry and/or exit actions
  - Actions (atomic operations—indivisible and instantaneous) executed on entering or exiting the state
- Internal transitions
  - Transitions that do not cause a state change
- Substates
  - State nested within a state
- Sequential
- Concurrent
- Deferred events
  - List of events not handled in that state; queued for handling by the object in another state

Transitions

Definitions

- Source state: active at the beginning of the transition
- Event trigger: the event that initiates the transition
- Guard condition: boolean-valued expression evaluated upon receipt of an event trigger; if the expression is true, the transition may fire; if false it may not
- Action: instantaneous executable atomic operation; acts on the object owning the state machine and on objects reachable through class relationships
- Target state: active after the transition is complete

State Machine

Example 1

Example 2
### Advanced States

**Definitions and examples**
- Entry & exit actions occur on state entry or exit
  - Useful when there are multiple transition in to or out of a state
- Internal transitions do not fire entry and exit transitions
  - If there are no entry or exit transitions, then internal = self
- Activities execute while an object is in a state
  - Non-atomic and non-instantaneous
- Deferred events are remembered and processed later

### Substates

**AKA nested states**
- Simple states have no substructure
- Composite states have nested or substates
  - Sequential (or disjoint) substates are like “normal” states but are nested inside of another state
  - Concurrent (or orthogonal) substates denote multitasking (multiprocessing or multithreading)
- Transitions into a composite state may target
  - a state
  - the composite state itself, which must define an initial state
- Transitions from a composite state may leave from
  - a state
  - the composite state itself, which interrupts or cuts short the activity of the nested state machine

### Composite States

**Visio version**

### Sequential Substates

**UML standard notation**

### Concurrent Substates

**UML standard notation**
History States
A "bookmark" in a process
- Transitions into a composite state (that don't directly target a substate) normally go the initial state
- History states allow an object to remember in which substate it was last
  - H shallow history remembers only the outermost nested state
  - H* deep history remembers the innermost nested state to any depth

Statechart and Activity Diagrams
Diagrams base on state machines
- A statechart shows a state machine
  - Models reactive objects
    - Object whose behavior is characterized as a response to external events
    - Its current behavior (i.e., its response) depends on previous events
  - Models objects, use cases, and systems
- An activity diagram is a special case of a statechart in which most or all of the states are activity states and most or all of the transitions are triggered by the completion of activities in the source state
  - Shows flow of control from activity to activity
  - Model a workflow or an operation over time (like a flowchart)