Behavioral Modeling in UML (part II)

Vadim Malvone
vadim.malvone@telecom-paris.fr

(theses slides are inspired by tutorials given at the OMG and the material of prof. Élie Najm)
Simple State Machine Diagram

- **Initial pseudostate**
- **Transition**
- **Final state**
- **State**
- **Trigger**
- **Guard**
- **Action**

**Diagram Elements:**
- **Ready**
  - entry/a1
- **Done**
  - exit/a3
- **Stop**
  - t1 \( [g_1 > 0] \) / a2

**Labels:**
- **Initial pseudostate**
- **Transition**
- **Final state**
- **State**
- **Trigger**
- **Guard**
- **Action**
Hierarchical State Machines

- States decomposed into state machines
- Reduced complexity
- It does not increase the expressive power
Hierarchical State Machines

Transition of a group of states

LampOff
entry/lamp.off()

off/

on/

flash/

LampOn
entry/lamp.on()

LampFlashling

FlashOn
entry/lamp.on()

tm(1)/

FlashOff
entry/lamp.off()

tm(1)/

transition to internal initial state
Hierarchical State Machines

LampOff
entry/lamp.off()

flash/
off/

LampOn
entry/lamp.on()

LampFlashOn
entry/lamp.on()

flashOff/
tm(1)/

LampFlashOff
entry/lamp.off()

tm(1)/
Completion transitions

Completion transition (no trigger)
Triggering rules

Two (or more) transitions can have the same event trigger:
- Precedence to innermost transition
Order of Actions: Complex Case

Sequence of actions performed:

exS11 \(\Rightarrow\) exS1 \(\Rightarrow\) actE \(\Rightarrow\) enS2 \(\Rightarrow\) initS2 \(\Rightarrow\) enS21
State history

It allows to return to the last visited state

suspend resume

suspend/

... resume/

\[ H^* \]
Orthogonality
Multiple simultaneous perspectives on the same entity

age

Child
Adult
Retiree

financialStatus

Poor
Rich
Orthogonal Regions

State → Pair of contiguous states (Cartesian product)
Semantics of Orthogonal Regions (1)

Each region can detect and execute an event independently of other regions
Semantics of Orthogonal Regions (2)

- All regions can detect and execute the same event simultaneously.
- The execution is interleaved (not necessarily deterministic).

Diagram:

- Legal status: LawAbiding (Outlaw) and Poor (Rich)
- Financial status: robBank/ (Outlaw) and robBank/ (Rich)
Internal Transitions and Orthogonality

S

on/print(on);

S1

off/

S2
Internal Transitions and Orthogonality
Fork and join transitions

For in/out transitions of orthogonal regions:

```
<table>
<thead>
<tr>
<th>Staff Member</th>
<th>Adult</th>
<th>Retiree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Child</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>f/</td>
<td>g/</td>
</tr>
<tr>
<td></td>
<td>p/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>g/</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Manager</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k/</td>
</tr>
</tbody>
</table>
```

```
e/  r/  
```
Misuse of Orthogonality

Using regions to model independent objects
## Behavior, state machines, and classes

Classes:
- With or without a State machine
- With or without a thread control

<table>
<thead>
<tr>
<th>State machine</th>
<th>without</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>without</strong></td>
<td><img src="image1" alt="attributes" /></td>
<td><img src="image2" alt="attributes" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="operations" /></td>
<td><img src="image4" alt="operations" /></td>
</tr>
<tr>
<td></td>
<td>Passives</td>
<td>Reactive</td>
</tr>
<tr>
<td><strong>with</strong></td>
<td><img src="image5" alt="attributes" /></td>
<td><img src="image6" alt="attributes" /></td>
</tr>
<tr>
<td></td>
<td><img src="image7" alt="operations" /></td>
<td><img src="image8" alt="operations" /></td>
</tr>
<tr>
<td></td>
<td>Active</td>
<td></td>
</tr>
</tbody>
</table>
Properties of Reactive Classes

An instance object of a reactive class:

- Starts and becomes available at the moment of its creation
- Remains reactive until:
  - The behavior reaches the final state
  - It is terminated by another object
- Does not have a thread control to collect signals
- 2 types of operations:
  - Primitives: the operation (and its body) is declared in the class
  - Triggered: the operation is a trigger of a state machine transition
Behavior of a Reactive Object

- Initialize Object
- Wait for Request
- Handle Request
- Terminate Object

Lamp On
- on
- on/print(on)

Lamp Off
- off
- off
- stop
Properties of Active Classes

An instance object of an active class:

- Starts and becomes available at the moment of its creation
- Remains active until:
  - The behavior reaches the final state
  - It is terminated by another object
- **Has a thread control:**
  - has a mailbox to receive signals
  - the signals are placed in the mailbox
  - each event is processed until completion before considering the next event
- 2 types of operations:
  - Primitives: the operation (and its body) is declared in the class
  - Triggered: the operation is a trigger of a state machine transition
Behavior of an Active Object (I)

“Run-to-completion” semantics:

- One event is admitted at a time

Benefits:

- It eliminates internal concurrency
- Minimal context switching overhead
Behavior of an Active Object (II)

An active object performs the following steps:
- Saves the events in its queue
- Dispatches (chooses) the event
- Deals with the event

1. **Queue**
   - => Saves the event in the mailbox

2. **Dispatcher**
   - => Chooses the event to process

3. **Processor**
   - => Consumes and processes the chosen event

FIFO most commonly used in tools
Postpone an event (I)

Pending events are deleted if there is no corresponding transition.

```
on  off
```

Diagram:
```
LampOff
entry/lamp.off()

LampOn
entry/lamp.on()
```
Postpone an event (II)

Unless they are declared as "deferred" in the state
Priority of objects

An active object can be suspended by an event triggering of another active object of higher priority.
Concurrent access

- Encapsulation does not protect against concurrent access
- Explicit synchronization is necessary
State machines: some problems and/or restrictions

- Restriction of event extraction mechanisms
  - We cannot extract several events together
- The mechanism for choosing events is not specified
  - Most common (simple) = FIFO (First In First Out)
  - Use of "deferred" possible (implemented in some tools)
- Problem of "infinite" loop of transitions
  - Activation of a transition by the absence of an event