UML Activities & Actions
Action & Object Nodes

Accept inputs, start behaviors, provide outputs

Object/Data flow

Control flow

Send Envoice → Invoice → Make Payment → Accept Payment

Invoice1234: Invoice
Invoice1234: Invoice

Output provided when action is complete
Action starts when input arrives

Sequencing: control « flows » when action is complete

Charles ANDRE - UNSA
Overview

• Activity modeling emphasizes the sequence and conditions for coordinating lower-level behaviors, rather than which classifiers own those behaviors.

• These are commonly called control flow and object flow models.

• The actions coordinated by activity models can be initiated
  – because other actions finish executing,
  – because objects and data become available,
  – or because events occur external to the flow.
Activity diagrams

- Possibly more than any other UML diagram, activity diagrams apply to much more than just software modeling. They are applicable to just about any type of behavioral modeling; for example, business processes, software processes, or workflows.
- **Software modeling**, activities typically represent a behavior invoked as a result of a method call.
- **Business modeling**, activities may be triggered by external events, such as an order being placed, or internal events, such as a timer to trigger the payroll process on Friday afternoons.
- **Activity diagrams** have undergone significant changes with UML 2.0; they have been promoted to first-class elements and no longer borrow elements from state diagrams.
- **Activity diagrams** capture activities that are made up of smaller actions.
Actions & Activities

• An **action execution** corresponds to the execution of a particular action.

• Similarly, an **activity execution** is the execution of an activity, ultimately including the executions of actions within it.

• Each action in an activity may execute zero, one, or more times for each activity execution.

• At the minimum, actions need **access** to data, they need to transform and test data, and actions may require sequencing.

• The activities specification (at the higher compliance levels) allows for several (logical) threads of control executing at once and synchronization mechanisms to ensure that activities execute in a specified order.
**Activity - Meta model**

```
Behavior

Activity

ObjectNode

Pin

Action

ActivityParameterNode

ActivityFinalNode

ControlNode

InitialNode

```

Charles ANDRE - UNSA
Flows - Meta model

![Diagram of Flows meta model]

- **ActivityNode**
  - target: 1
  - incoming: 0..*
  - source: 1
  - outgoing: 0..*

- **ActivityEdge**
  - edge: 0..*
  - activity: 0..1

- **Activity**

- **ControlFlow**

- **ObjectFlow**

Charles ANDRE - UNSA
Actions

- An action is a named element that is the fundamental unit of executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise.
- The action begins execution by taking tokens from its incoming control edges and input pins. When the execution of an action is complete, it offers tokens in its outgoing control edges and output pins, where they are accessible to other actions.
Actions (2)

- **AcceptCallAction**: an accept event action representing the receipt of a synchronous call request.
- **AcceptEventAction**: an action that waits for the occurrence of an event meeting specified condition.
- **CallBehaviorAction**: a call action that invokes a behavior directly rather than invoking a behavioral feature that, in turn, results in the invocation of that behavior.
- **CallOperationAction**: an action that transmits an operation call request to the target object, where it may cause the invocation of associated behavior.
- **BroadcastSignalAction**: an action with implementation-specific semantics.
- **OpaqueAction**: an action that creates a signal instance from its inputs, and transmits it to the target object, where it may cause the firing of a state machine transition or the execution of an activity.
- **SendSignalAction**: The action returns the result of evaluating a value specification.
- **ValueSpecificationAction**: a call action that creates a signal instance from its inputs, and transmits it to the target object, where it may cause the firing of a state machine transition or the execution of an activity.
Activity Modeling (1)

- Activity modeling emphasizes the input/output dependencies, sequencing, and conditions for coordinating other behaviors.
- Uses secondary constructs to show which classifiers are responsible for those behaviors.
- Focus is on what tasks need to be done, with *what* inputs, in what order, *rather than who/which* performs each task.
Activity Modeling (2)

Tasks and ordering …
Activity Modeling (3)

... plus resource assignments
"Flow" semantics

Activity execution defined in terms of flow of control and objects/data.

Charles ANDRE - UNSA
Actions & Object Nodes (1)

Accept inputs, start behaviors, provide outputs

Output provided when action is complete

Sequencing: control « flows » when action is complete

Action starts when input arrives
Alternate object node notation.

Must use this notation if the output type is different than the input type.
Queuing (1)

- Tokens can
  - Stack up in “in/out” boxes
  - Prevent upstream behaviors from taking new inputs
- Applicable to systems with significant resource constraints, such as physical or manual processes
Tokens can be
- Stored temporarily
- Divided between flows

Tokens cannot
- Flow in more than one direction, unless copied
Activity Parameter Nodes

Parameter nodes accept and provide values to/from whatever behavior uses this activity
Streaming Parameters

**Streaming**: values accepted and provided while action is executing

In this activity diagram, the original analog audio is streamed through a digital-to-analog converter, and the resulting digital data is sent to the encoding algorithm for processing.
Control Nodes (1)

Route objects/data

At beginning and end of activity:

Initial Node
- Gets control when containing activity starts. Flow out immediately.

Activity Final Node
- Accepts inputs, aborts containing activity

Flow Final
- Accepts inputs, does nothing
Control Nodes (2)

Route objects/data

In middle of activity:

- **Decision**: Flows out in exactly one direction.
- **Merge**: Flows through immediately. Does not combine the tokens.
- **Fork**: Copies inflow to multiple outflows.
- **Join**: Flows out when all inflows arrive. Combine tokens when possible.
Interruptible Region

Flows abort region

Diagram:

- Receive Order
- Fill Order
- Ship Order
- Close Order
- Send Invoice
- Invoice
- Make Payment
- Accept Payment
- Request Cancel
- Cancel Order

Interrupting edge
Interruptible region
Extensions for Systems Engineering

- Available in **SysML** (a UML profile for SE)
- Control as data
  - Enabling and disabling control values
  - Output from activities to turn other behaviors “on” and “off”.
- Rate of flow, on edges and streaming parameters
- Reduce buffering
  - Overwrite values already in buffer
  - Turn off buffering
- Probability on decisions, parameter sets, competing outflows from object nodes
- Behavior decomposition
Rate and buffer reduction

<controlOperator>
Monitor Temperature

{ rate < 1 per 10 minutes }

Heat Air

<controlOperator>
Monitor Temperature

{ stream }

Measure Temperature

{ stream }

[ above threshold ]

«ValueSpecificationAction»
enable

{ stream }

[ else ]

«ValueSpecificationAction»
disable

ControlValue

Measure Temperature

«continuous»

Temperature

«noBuffer»

Calculate Feels-like Temperature

Temperature

{ stream }

Receive Temperature

{ rate = 1 per second }

Humidity

Charles ANDRE - UNSA
Activity decomposition

«activity» Maintain Temperature
- maintainTemperature [1..1]
- heating [0..1]
- monitorTemperature [0..1]

«activity» Monitor Temperature
- monitorTemperature [1..1]
- calculateTraction [0..1]

«activity» Heat Air
- maintainTemperature [1..1]

«activity» Measure Temperature

«class» Temperature
- temperature [0..*]

«class» Humidity
- humidity [0..*]
References

• Conrad Bock, UML 2 Activity articles:  
  http://www.conradbock.org/#UML2.0

• UML 2 specification:  
  http://www.omg.org/docs/formal/07-02-03.pdf

• SysML specification:  
  http://www.omg.org/docs/ad/05-11-01.pdf