UML Activity Diagrams

Hoang Huu Hanh, Hue University
hanh-at-hueuni.edu.vn

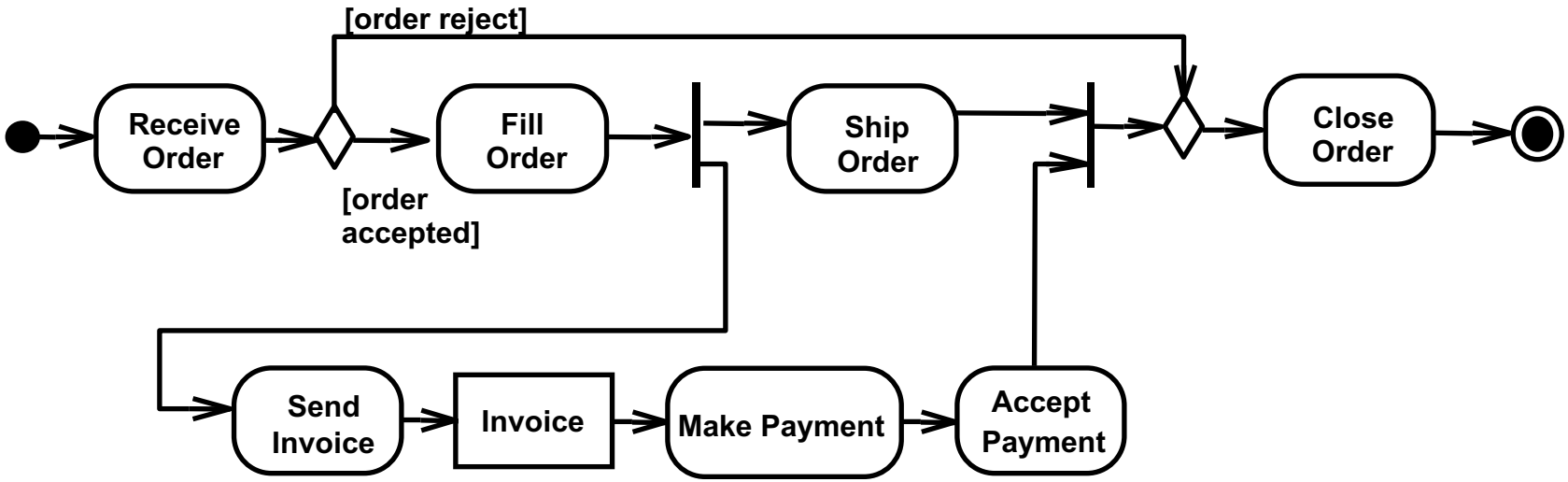
Contents

- Introduction to UML 2.0 Activity Diagrams
- Concepts of Action, Pins and Activity
- Description of activity nodes and activity edges
- New notations

Hue University

Activity diagrams

- Useful to specify software or hardware system behaviour
- Based on data flow models – a graphical representation (with a Directed Graph) of how data move around an information system



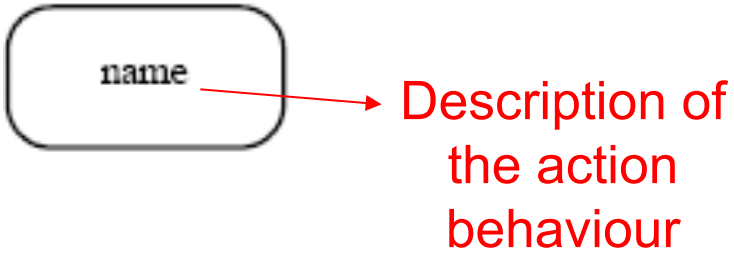
Hue University

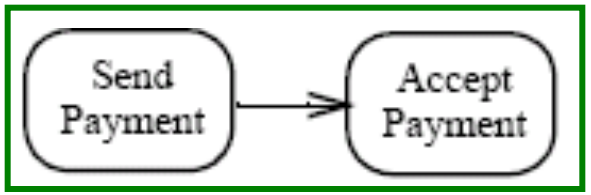
Some definitions

- ***Flow***: permits the interaction between two nodes of the activity diagram (represented by edges in activity diagram)
- ***State***: a condition or situation in the life of an object during which it satisfies some conditions, performs some activities, or waits for some events
- ***Type***: specifies a domain of objects together with the operations applicable to the objects (also none); includes primitive built-in types (such as integer and string) and enumeration types
- ***Token***: contains an object, datum, or locus of control, and is present in the activity diagram at a particular node; each token is distinct from any other, even if it contains the same value as another
- ***Value***: an element of a type domain

Actions

- The fundamental unit of executable functionality in an activity
- The execution of an action represents some transformations or processes in the modeled system (creating objects, setting attribute values, linking objects together, invoking user-defined behaviours, etc.)

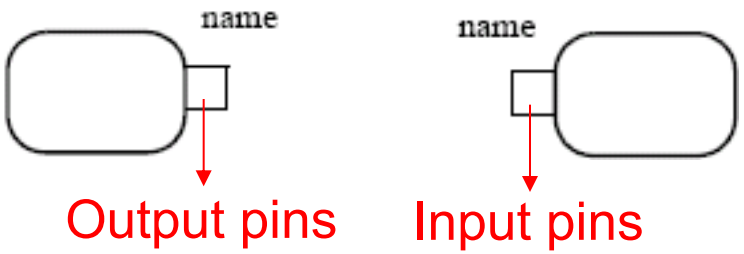


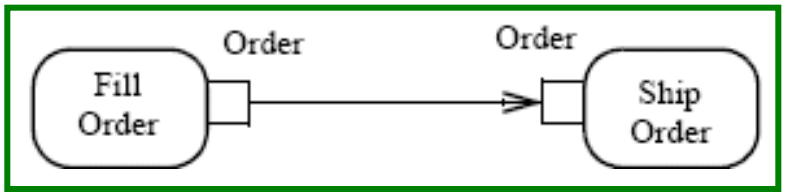


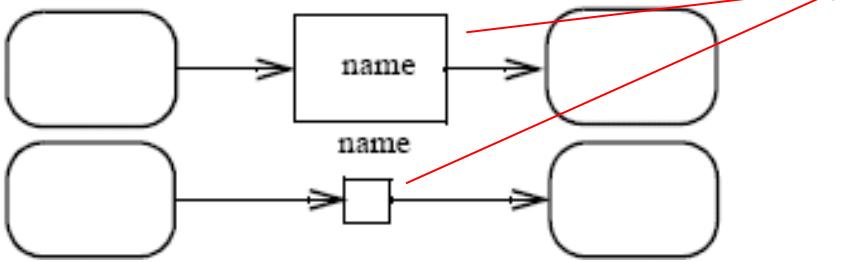
Hue University

Pins

- Actions can have inputs and outputs, through the pins
- Hold inputs to actions until the action starts, and hold the outputs of actions before the values move downstream
- The name of a pin is not restricted: generally recalls the type of objects or data that flow through the pin



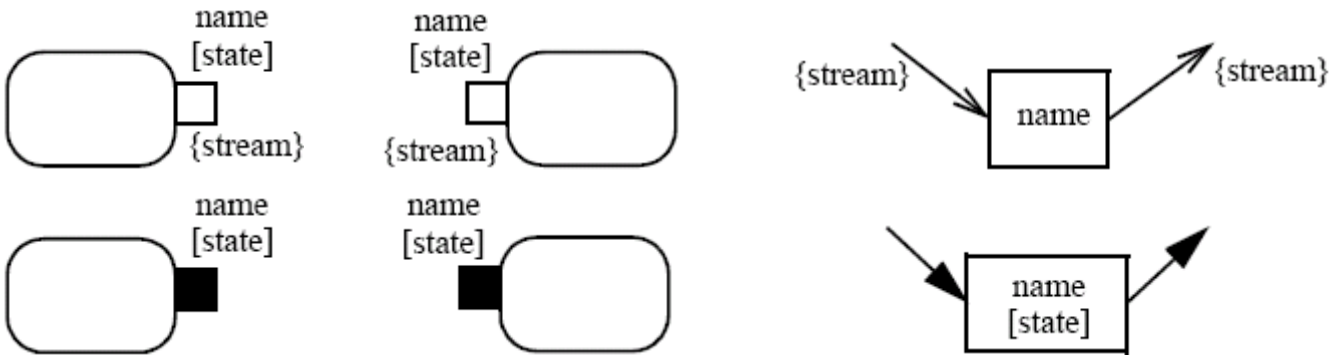




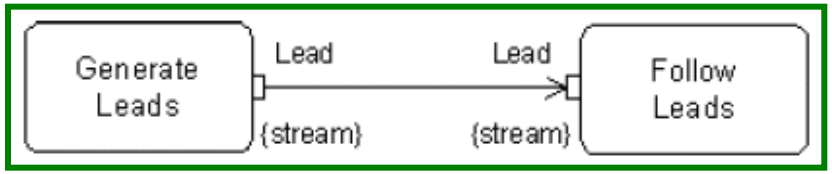
Standalone pin notations: the output pin and the input pin have the same name and same type

Special kind of pins (1)

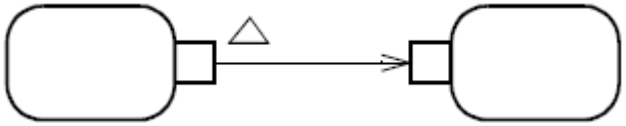
- **Streaming Parameters** (notated with {STREAM}): accept or provide one or more values while an action is executing

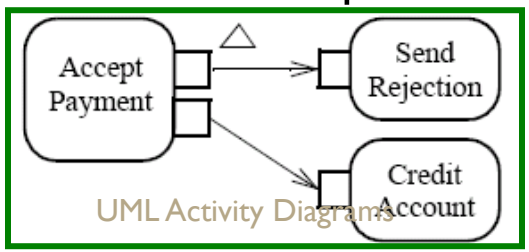


Stand-alone style



- **Exception Output Parameters** (notated with a triangle)
  - Provide values to the exclusion of any other output parameter or outgoing control of the action
  - The action *must immediately terminate*, and the output cannot quit

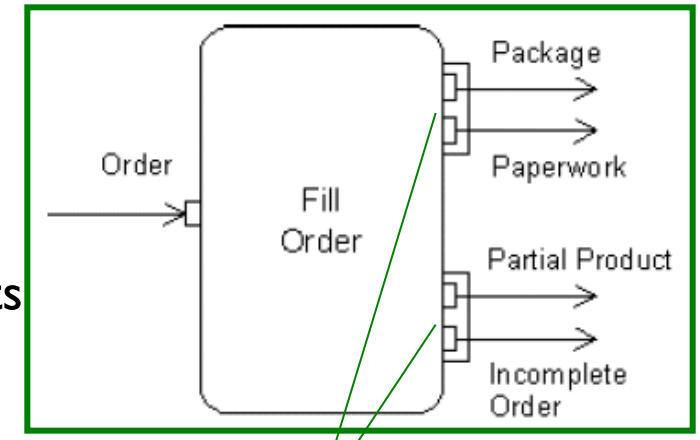




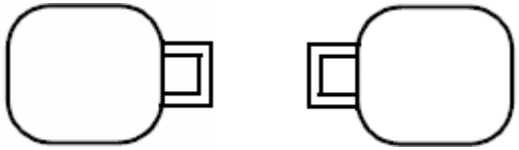
Hue University

Special kind of pins (2)

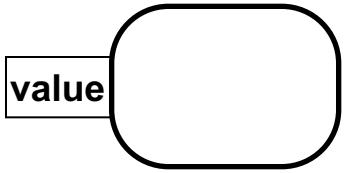
- **Parameter Sets**
  - group of parameters
  - the action can only accept inputs from the pins in one of the sets
  - the action can only provide outputs to the pins in one of the sets

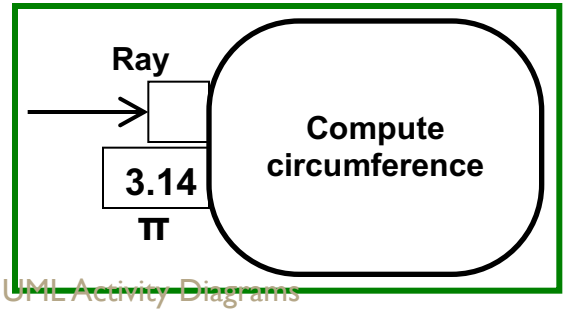


Output is provided only to the first or to the second set



- **ValuePin**: special kind of input pin defined to provide constant values (the type of specificated value must be compatible with the type of the value pin)





Hue University

Conditions to start and end actions

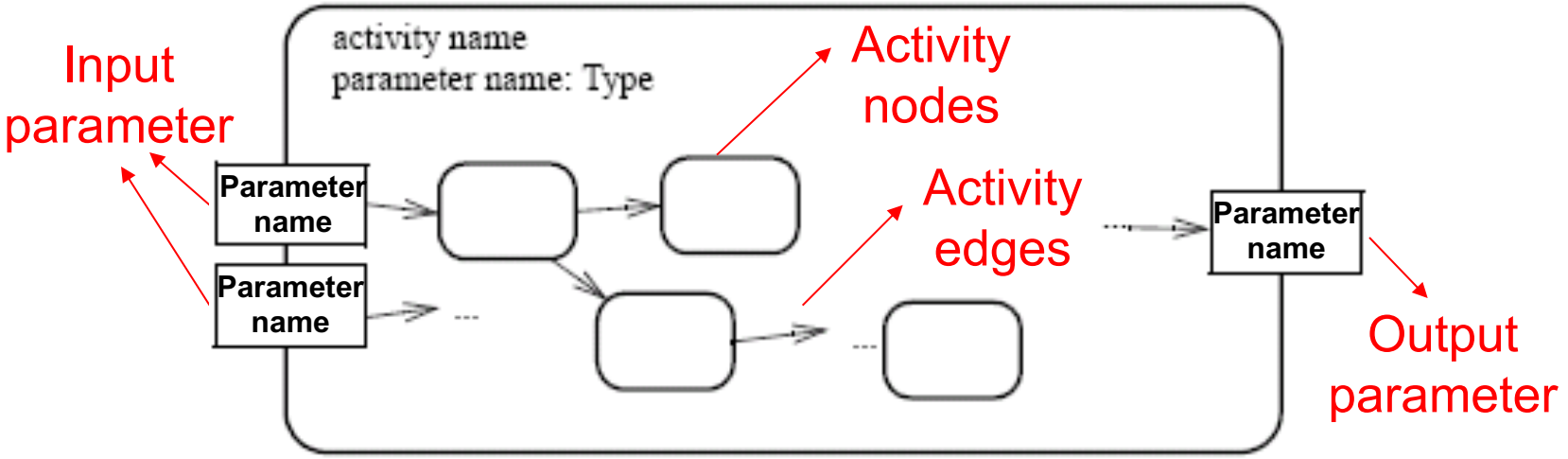
- An action can start only if:
  - all non-stream inputs have arrived (or at least one stream input if there are only stream inputs)
- The action can finish only if:
  - all inputs have arrived (streaming inputs included)
  - all non-stream and non-exception outputs (or an exception outputs) have been provided

- ***Prevent deadlock***: an input pin of an action cannot accept tokens until all the input pins of the action can accept them

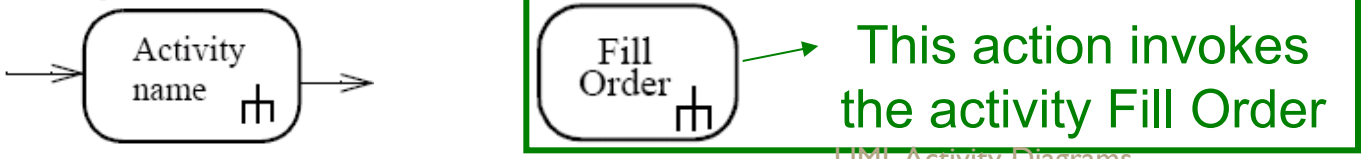
Hue University

Activities

- An activity is the specification of parameterized behaviour as the coordinated sequencing of subordinate units whose individual elements are actions
- Uses parameters to receive and provide data to the invoker



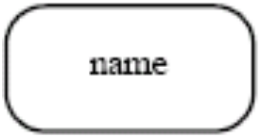
- An action can invoke an activity to describe its action more finely



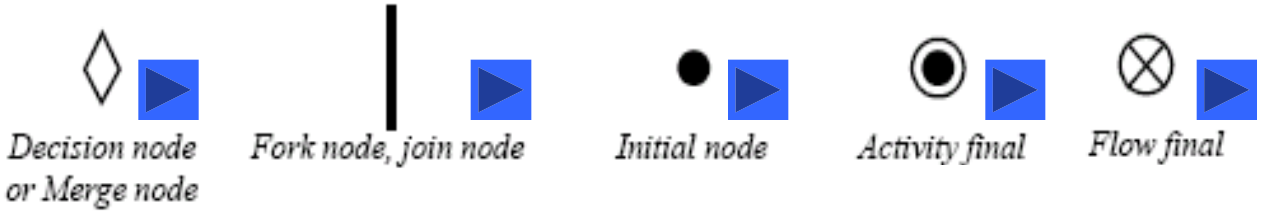
Hue University

Activity nodes

- Three type of activity nodes:
  - **Action nodes**: executable activity nodes; the execution of an action represents some transformations or processes in the modeled system (already seen)



  - **Control nodes**: coordinate flows in an activity diagram between other nodes

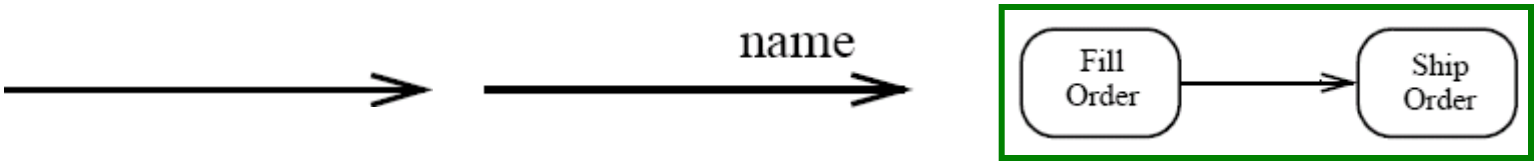


  - **Object nodes**: indicate an instance of a particular object, may be available at a particular point in the activity (i.e Pins are object nodes)

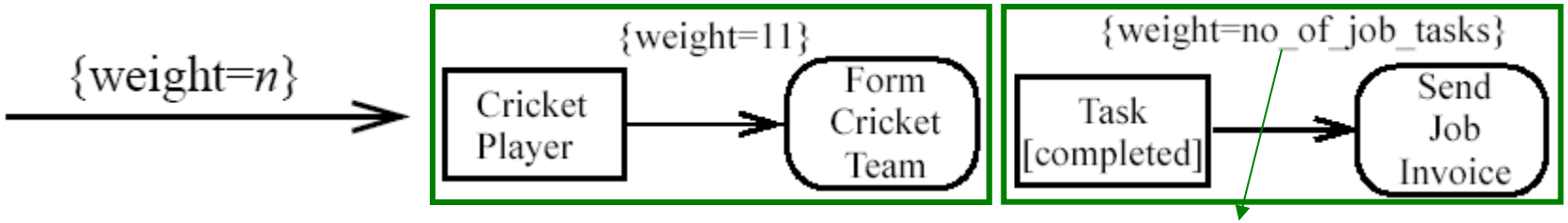
Hue University

Activity edges (1)

- Are directed connections
- They have a source and a target, along which tokens may flow



- Any number of tokens can pass along the edge, in groups at one time, or individually at different times
- ***Weight***: determines the minimum number of tokens that must traverse the edge at the same time

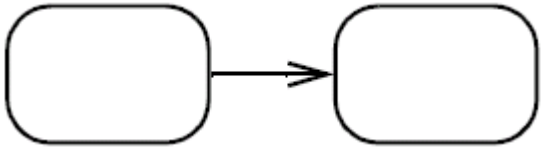


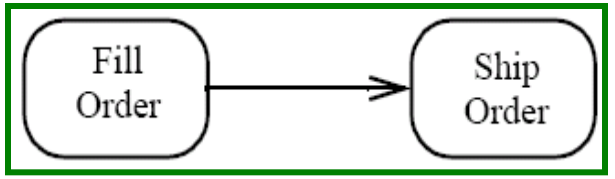
In this example we use a non-constant weight: an invoice for a particular job can only be sent when all of its tasks have been completed

Activity edges (2)

- Two kinds of edges:
  - **Control flow edge -** is an edge which starts an activity node after the completion of the previous one by passing a control token

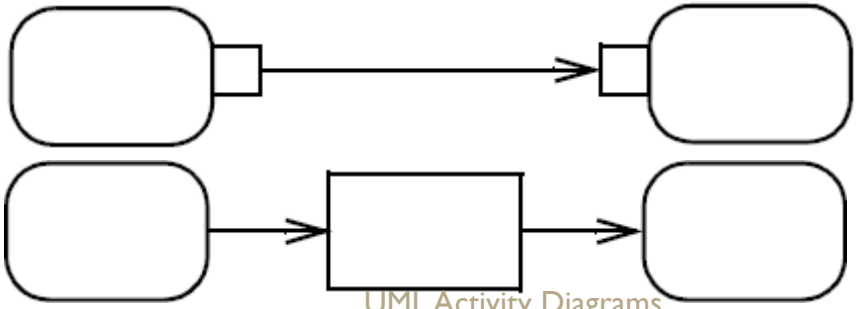
Notation without activity nodes





  - **Object flow edge -** models the flow of values to or from object nodes by passing object or data tokens

Notation without activity nodes



conclusions

- UCD: đặc tả bài toán, UC, UD => GUI với đầy đủ các thông tin
- UCD: quy trình nghiệp vụ (business processes)=> AD
- UC, UD + AD: một hệ thống giao diện hoàn chỉnh: thông tin mỗi form là đầy đủ; các form đc sắp đặt theo AD => thực hiện các chức năng <= UC (kiểm chứng)
- UCD=Use Case Diagram; UD=Use Case Description; AD=Activity Diagram; GUI=Graphical User Interface

Hue University

Options

- Object nodes
  - multiplicities and upperBound
  - effect and ordering
- Activity nodes
  - presentation options
  - transformation
- Selection
- Token competion

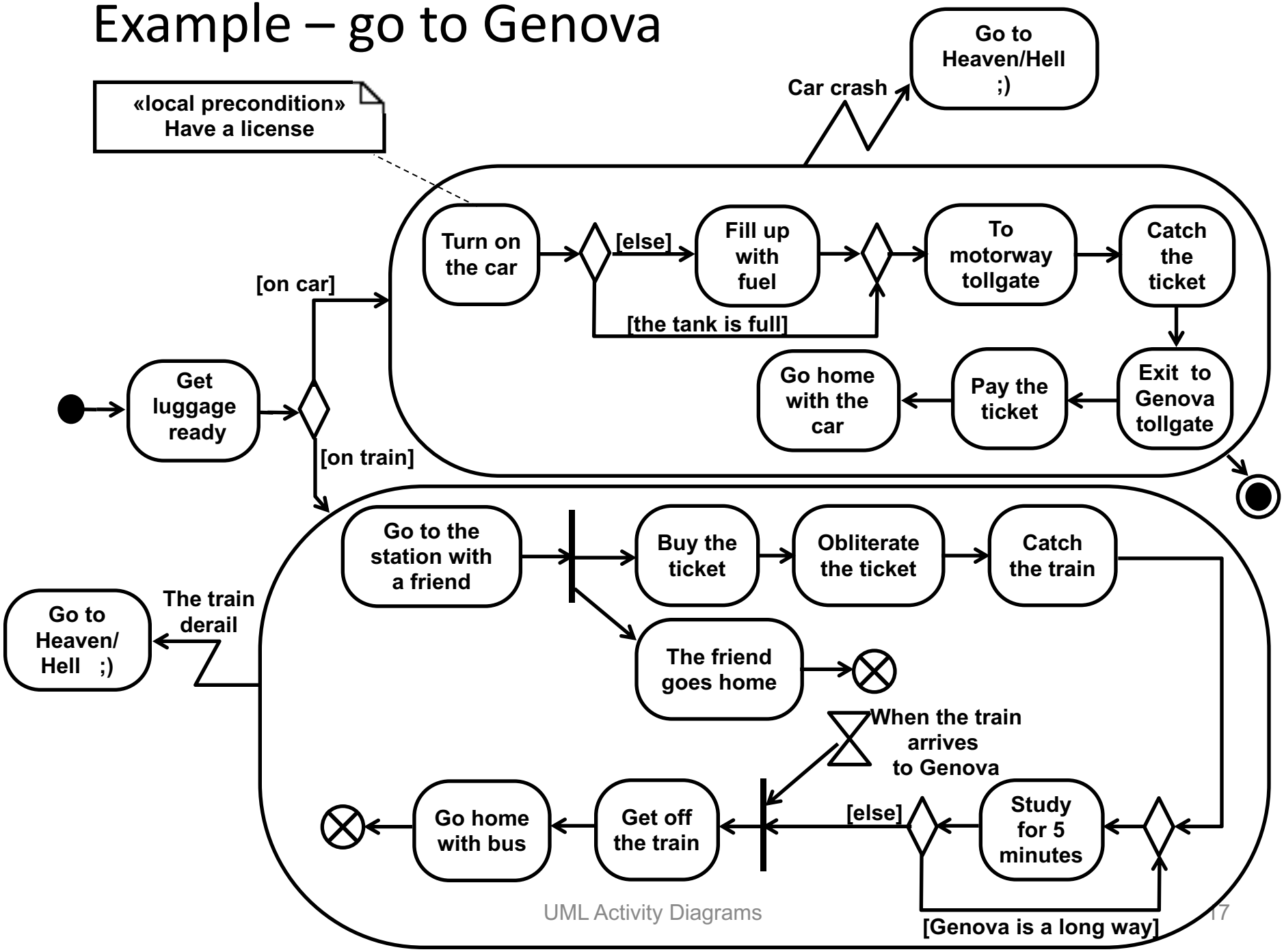
Hue University

New notations

- ActivityPartition
- Pre & post condition
- SendSignalAction
- Time triggers and Time events
- AcceptEventAction
- InterruptibleActivityRegion
- Exception
- ExpansionRegion

Hue University

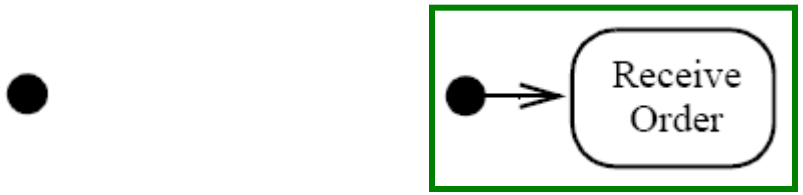
Example – go to Genova



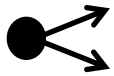
Control Nodes

Initial Nodes

- In an activity the flow starts in initial nodes, that return the control immediately along their outgoing edges



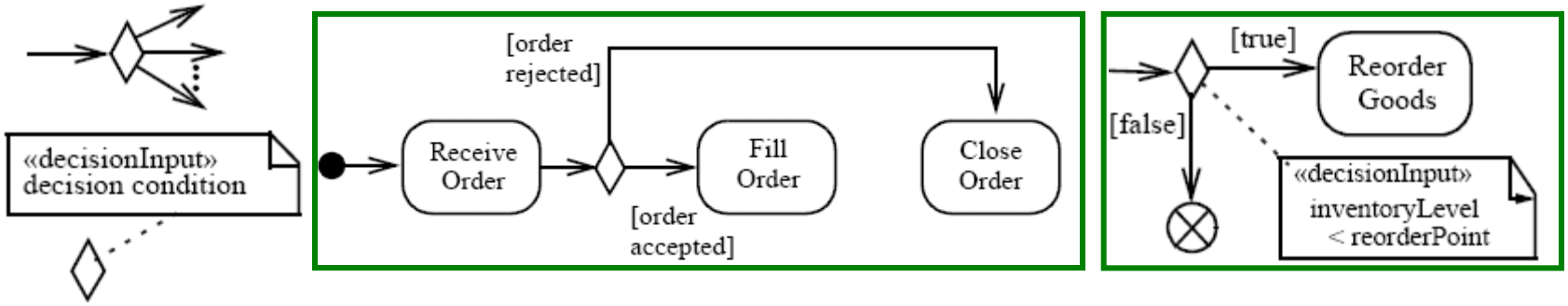
- If there are more than one initial node, a control token is placed in each initial node when the activity is started, initiating multiple flows
- If an initial node has more than one outgoing edge, only one of these edges will receive control, because initial nodes cannot duplicate tokens



A or B ?

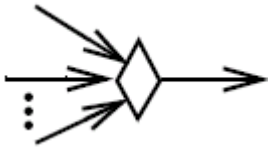
Decision Nodes

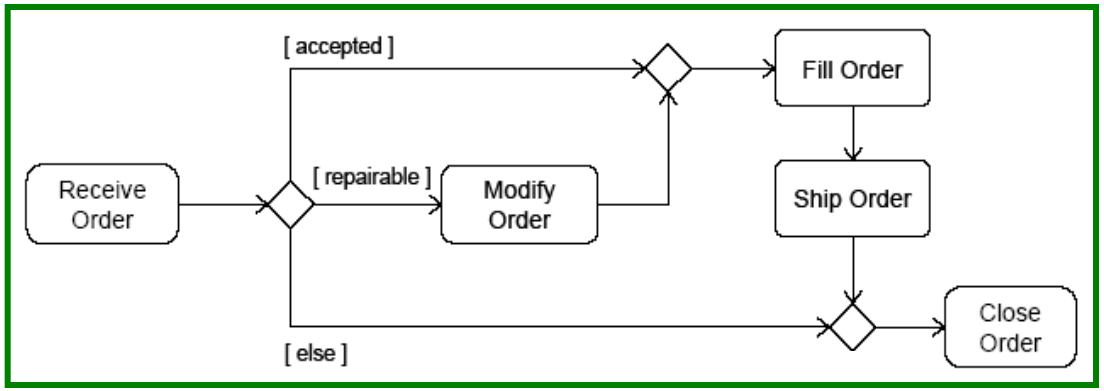
- Route the flow to one of the outgoing edges (tokens are not duplicated)
- Guards are specified on the outgoing edges or with the stereotype «decisionInput»
- There is also the predefined guard **[else],** chosen only if the token is not accepted by all the other edges
- If all the guards fail, the token remains at the source object node until one of the guards accept it



Merge Nodes

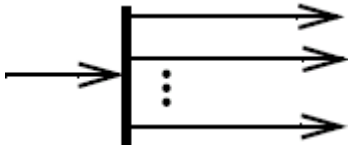
- Bring together multiple alternate flows
- All controls and data arriving at a merge node are immediately passed to the outgoing edge
- There is no synchronization of flows or joining of tokens

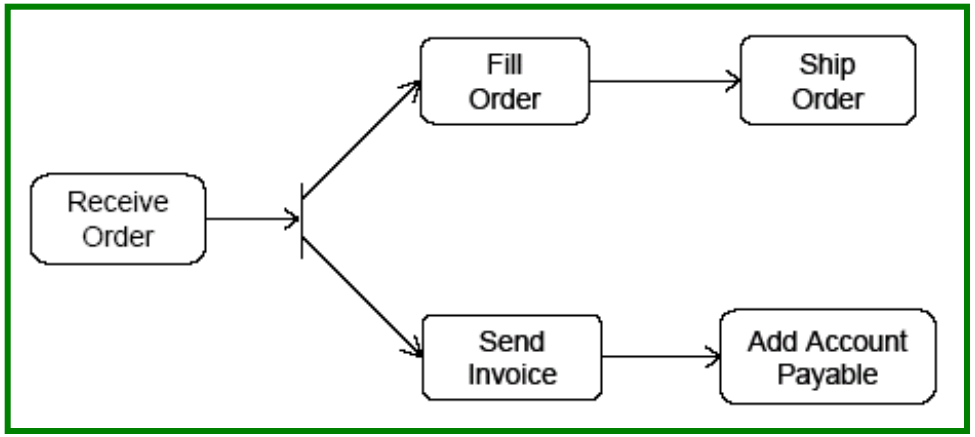




Fork Nodes

- Fork nodes split flows into multiple concurrent flows (tokens are duplicated)

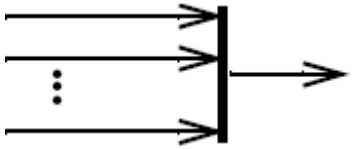


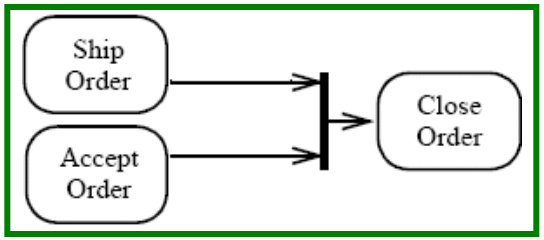


- State machine forks in UML 1.5 required synchronization between parallel flows through the state machine RTC step (it means that the first state in each branch is executed, then the second one, etc.)
- UML 2.0 activity forks model unrestricted parallelism

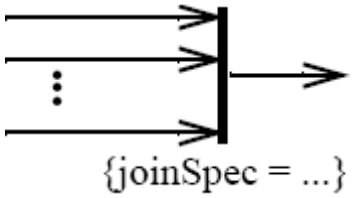
Join Nodes

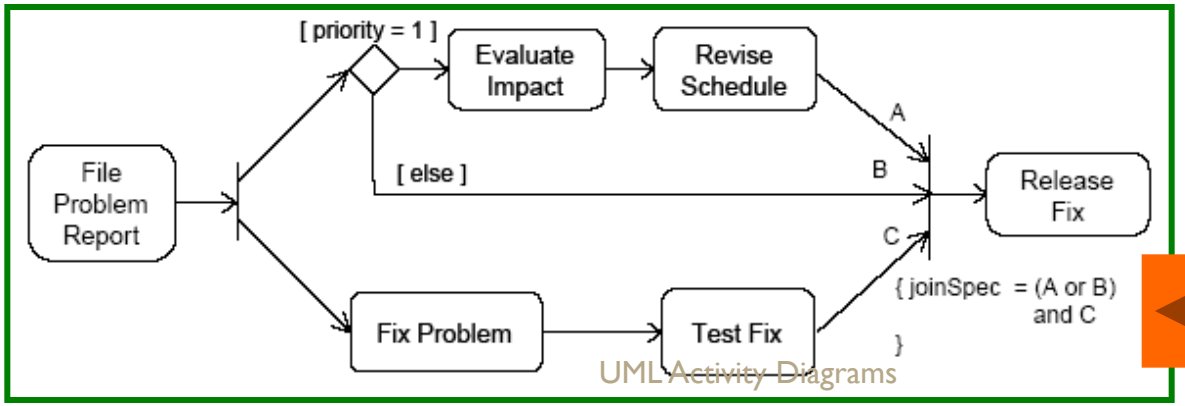
- Join nodes synchronize multiple flows





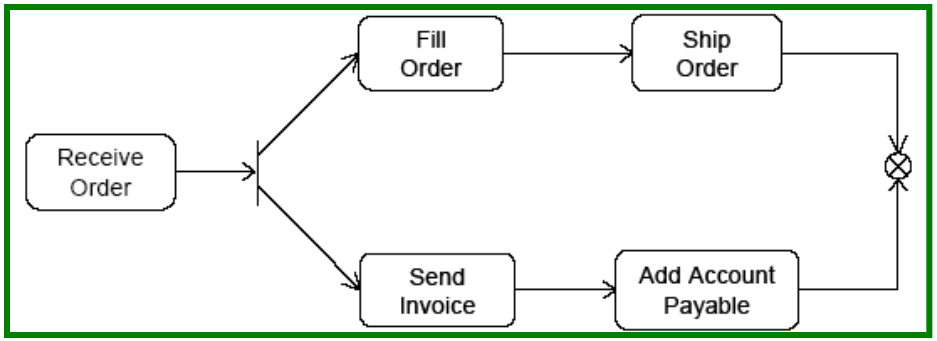
- Generally, controls or data must be available on every incoming edge in order to be passed to the outgoing edge, but user can specify different conditions under which a join accepts incoming controls and data using a join specification



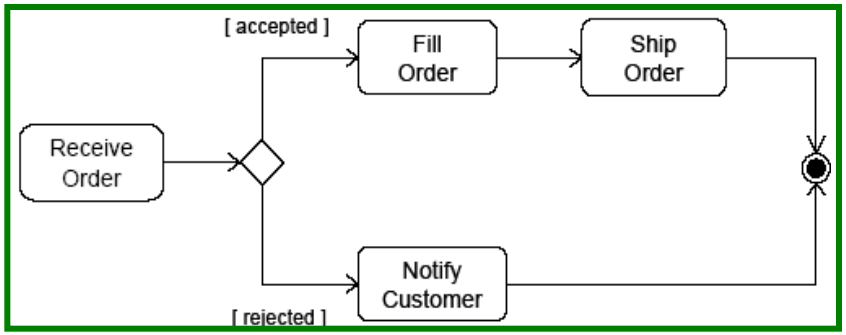


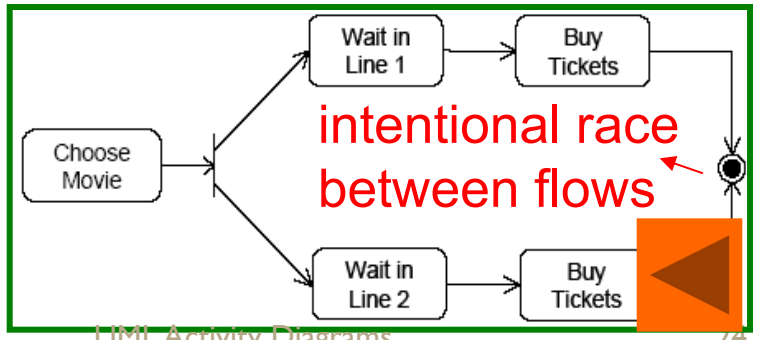
Final Nodes

- **Flow final:**
  - destroys the tokens that arrive into it
  - the activity is terminated when all tokens in the graph are destroyed



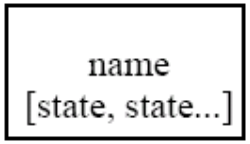
- **Final node:**
  - the activity is terminated when the first token arrives





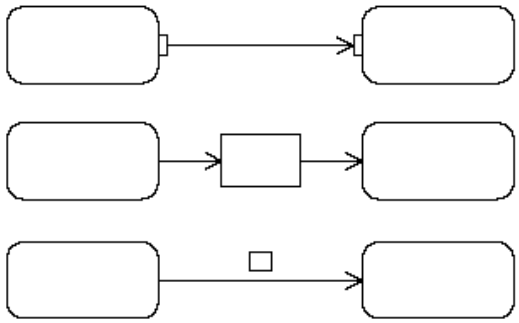
Object nodes

- Hold data temporarily while they wait to move through the graph
- Specify the type of values they can hold (if no type is specified, they can hold values of any type)
- Can also specify the state of the held objects

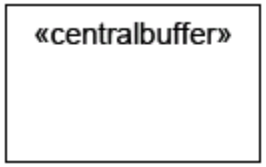


- There are four kinds of object nodes:

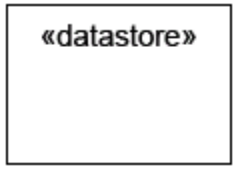
Activity Parameter Nodes



Pins
(three differents notations)



Central Buffer Nodes

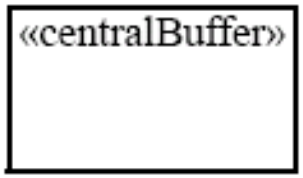


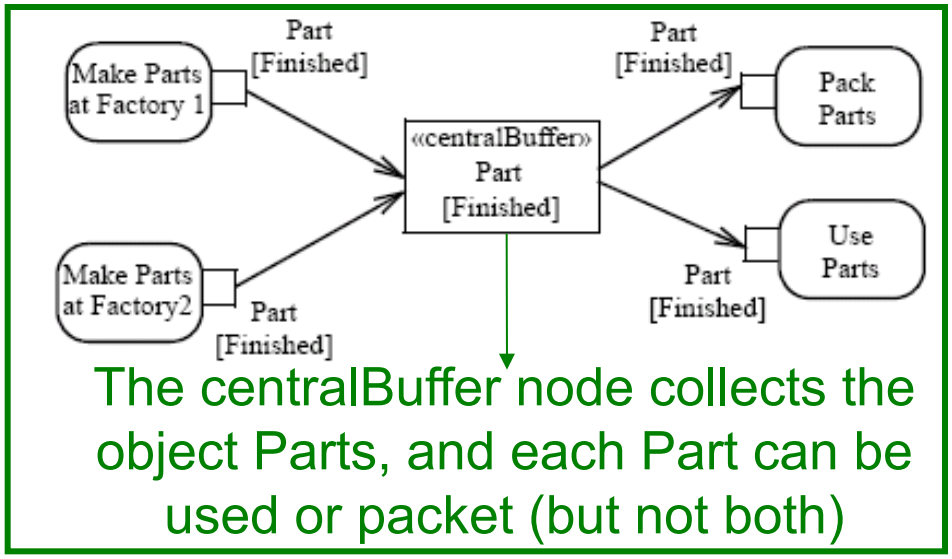
Data Store Nodes

Hue University

centralBuffer

- A central buffer node is an object node that manages flows from multiple sources and destinations (as opposed to pins and parameters)
- Acts as a buffer for multiple input flows and output flows
- Is not tied to an action like pins, or to an activity like activity parameter nodes

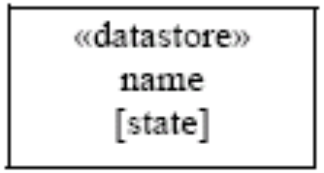


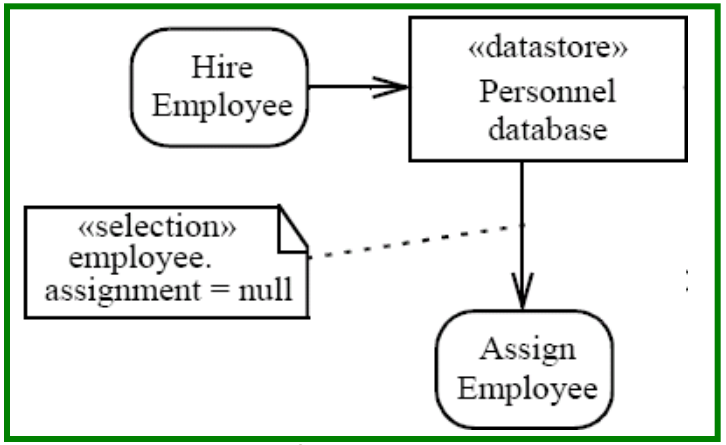


Object Nodes

datastore

- Is a specific central buffer node which stores objects persistently
- Keeps all tokens that enter into it
- Tokens chosen to move downstream are copied so that tokens never leave the data store
- If arrives a token containing an object already present in the data store, this replaces the old one
- Tokens in a data store node cannot be removed (they are removed when the activity is terminated)

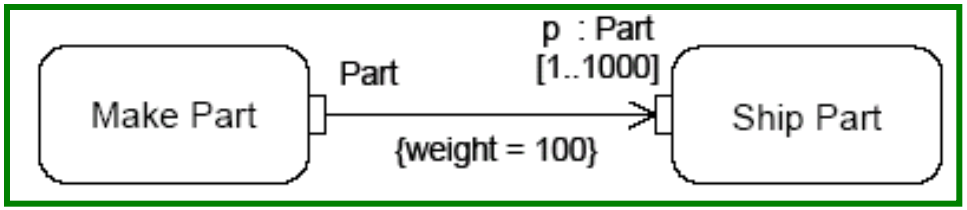




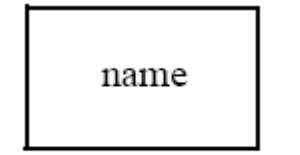
Hue University

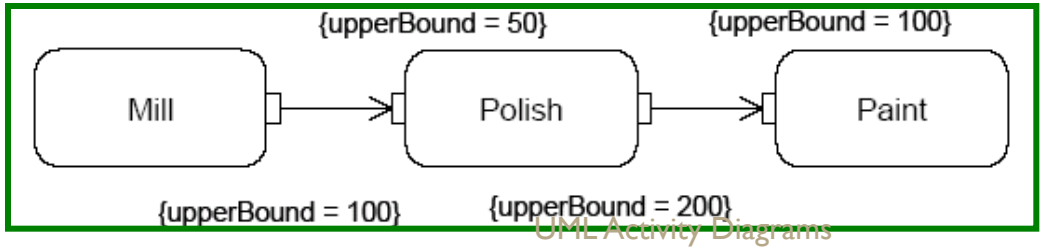
Multiplicities and UpperBound

- **Multiplicities**: specify the minimum (≥0) and maximum number of values each pin accepts or provides at each invocation of the action:
  - when is available the minimum number of values, the action can start
  - if there is more values than the maximum, the action takes only the first maximum value



- **UpperBound**: shows the maximum number of values that an object node can hold: at runtime, when the upper bound has been reached, the flow is stopped (buffering)

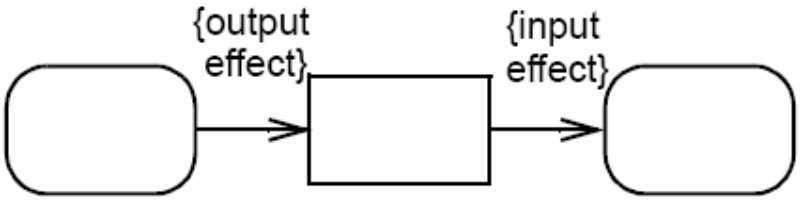


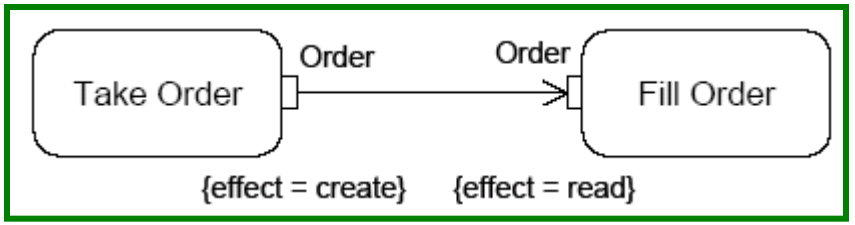


Object Nodes

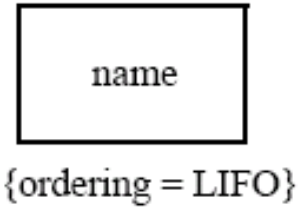
Effect and Ordering

- ***Effect***: pins can be notated with the effect that their actions have on objects that move through the pin
- The effects can be: 'create' (only on output pins), 'read', 'update' or 'delete' (only on input pins)





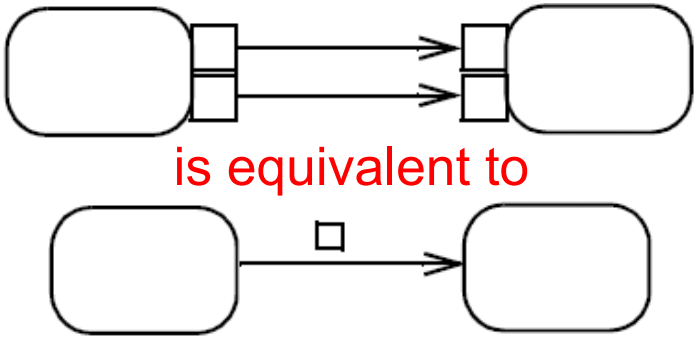
- ***Ordering***: specifies the order in which the tokens of an object node are offered to the outgoing edges (FIFO, LIFO or modeler-defined ordering)



Presentation Options

- An edge can also be notated using a connector
- Every connector with a given label must be paired with exactly one other with the same label on the same activity diagram

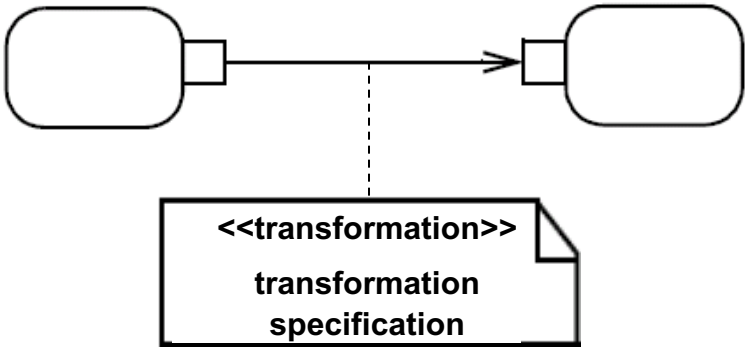
- To reduce clutter in complex diagrams, object nodes may be elided

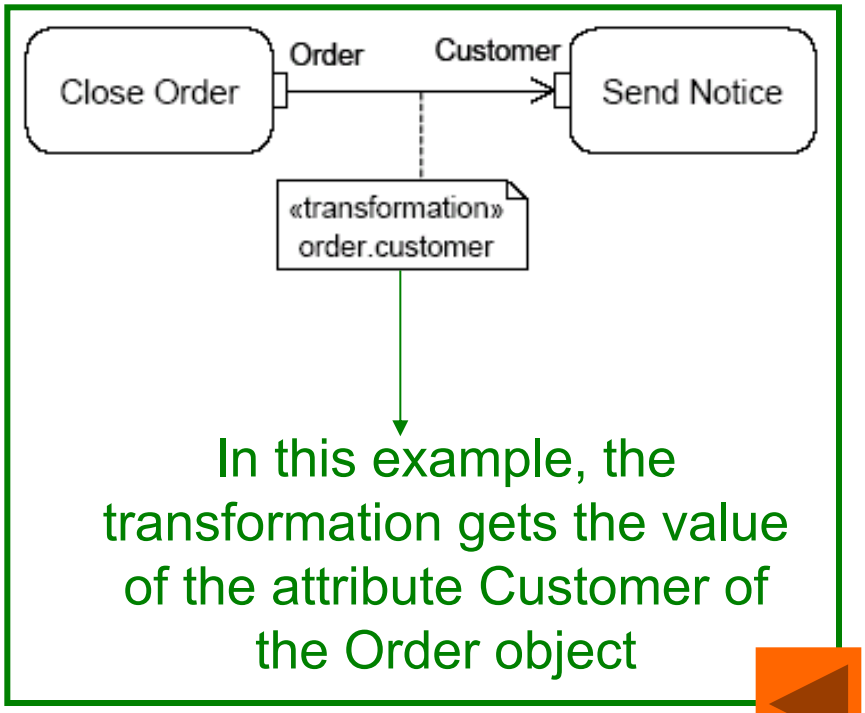


Activity Edges

Transformation

- It is possible to apply a transformation of tokens as they move across an object flow edge (each order is passed to the transformation behaviour and replaced with the result)

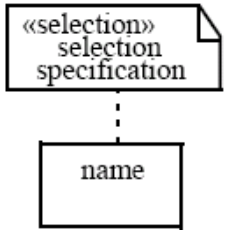


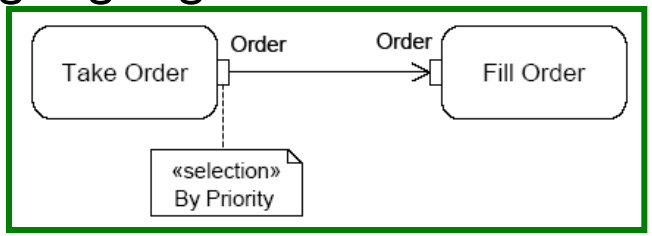


Hue University

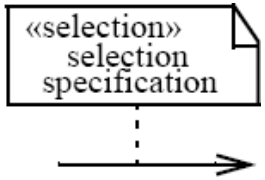
Selection

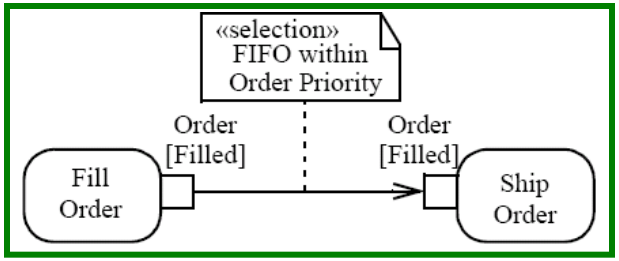
- Specifies the order (FIFO, LIFO or modeler-defined ordering) in which tokens in the node are offered to the outgoing edges
- Can be applied to:
  - ***Object node*** - specifies the object node ordering, choosing what token offers to the outgoing edge whenever it asks a token





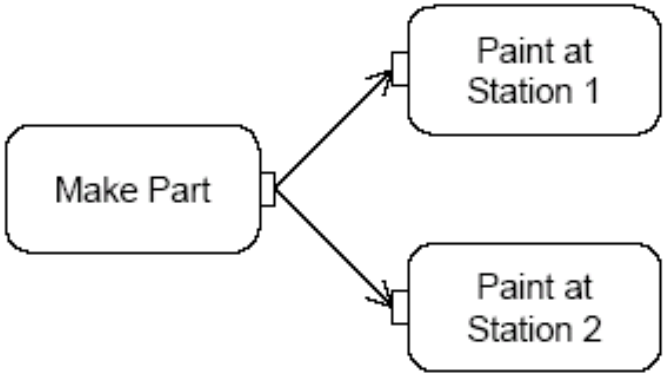
  - ***Edge*** - chooses the order on which tokens are offered from the source object node to the edge (overrides any selection present on the object node, that is object node ordering)





Token competition

- A parameter node or pin may have multiple edges coming out of it, whereupon there will be competition for its tokens, because object nodes cannot duplicate tokens while forks can
- Then there is *indeterminacy* in the movement of data in the graph



If the input pin of *Paint at Station 1* is full, the token remains at the output of *Make Part* until the traversal can be completed to one of the input pins

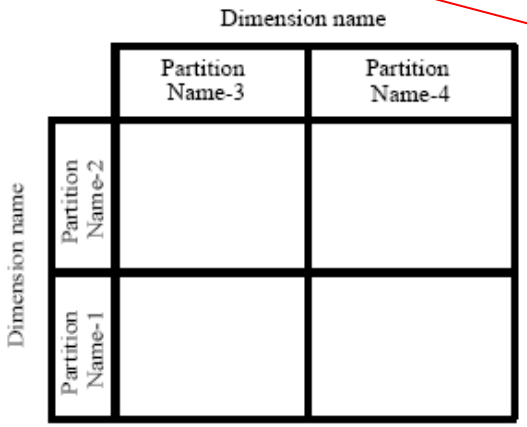
Hue University

ActivityPartition (1)

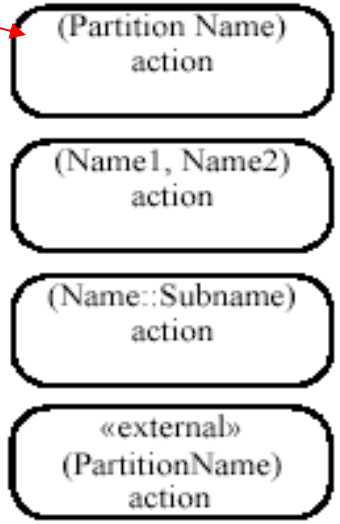
- Partitions divide the nodes and edges for identifying actions that have some characteristics in common
- They often correspond to organizational units in a business model
- Partitions can be hierarchical and multidimensional
- Additional notation is provided: placing the partition name in parenthesis above the activity name

a) Partition using a swimlane notation

b) Partition using a hierarchical swimlane notation

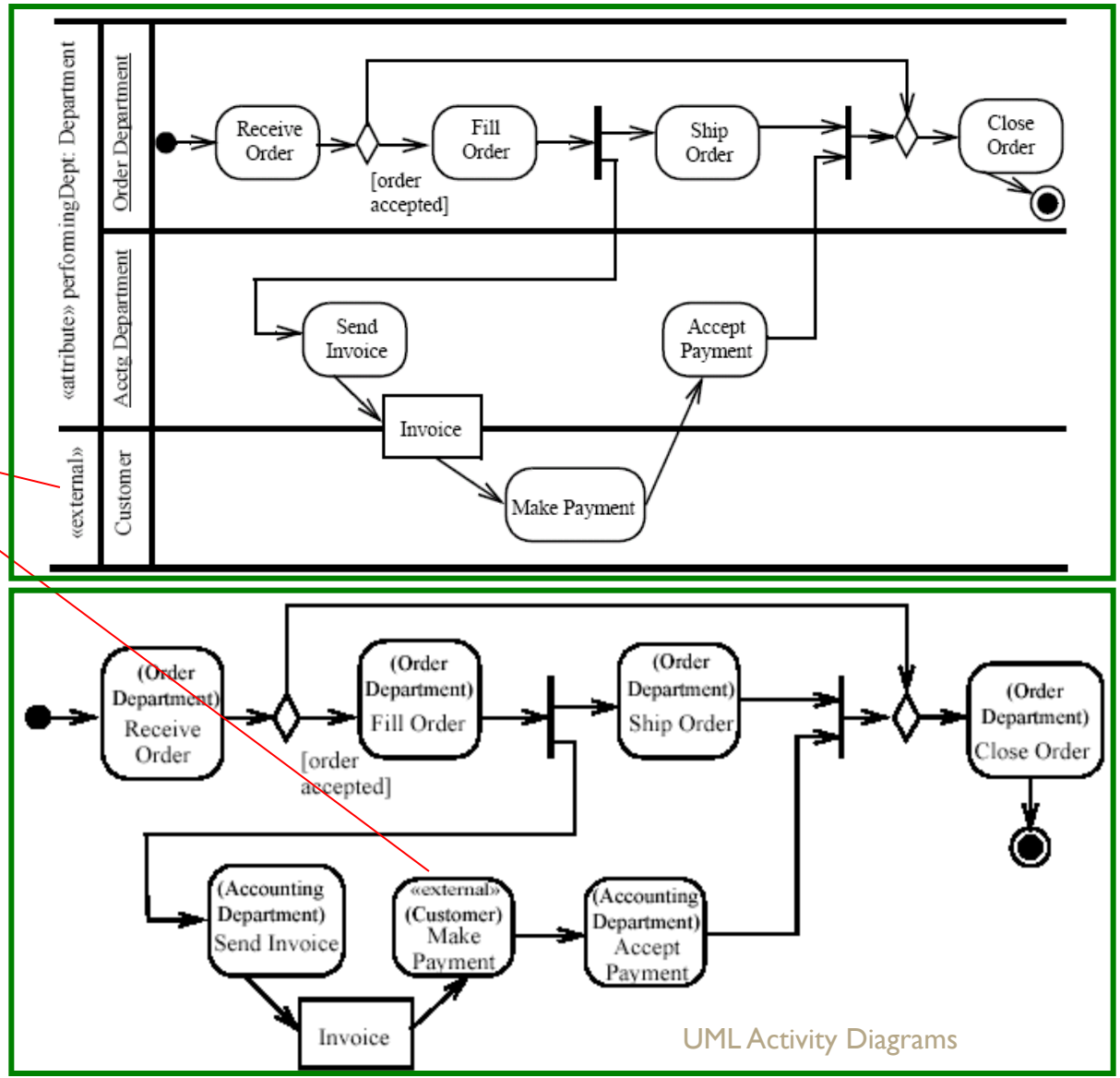


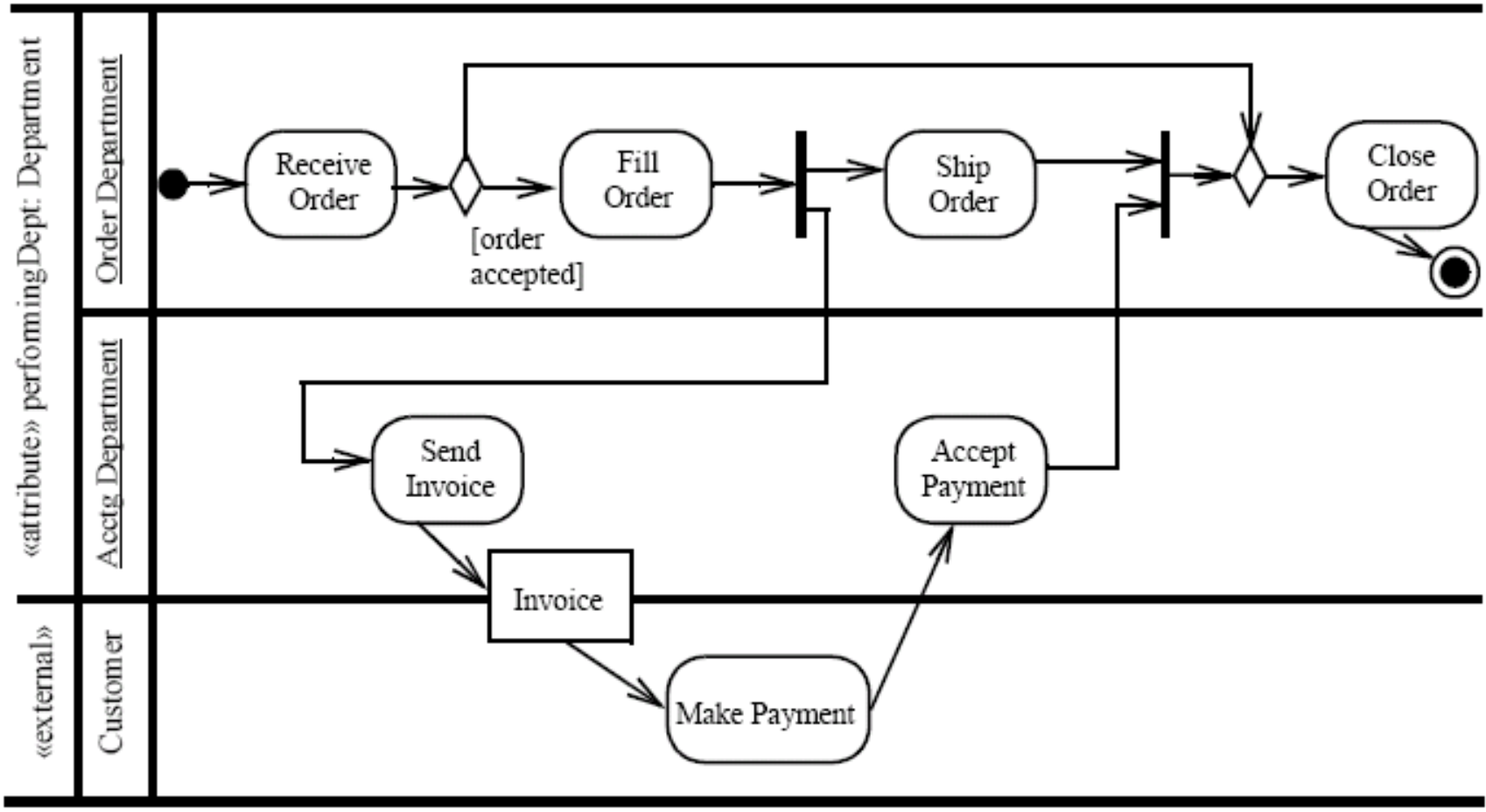
c) Partition using a multidimensional hierarchical swimlane notation



ActivityPartition (2)

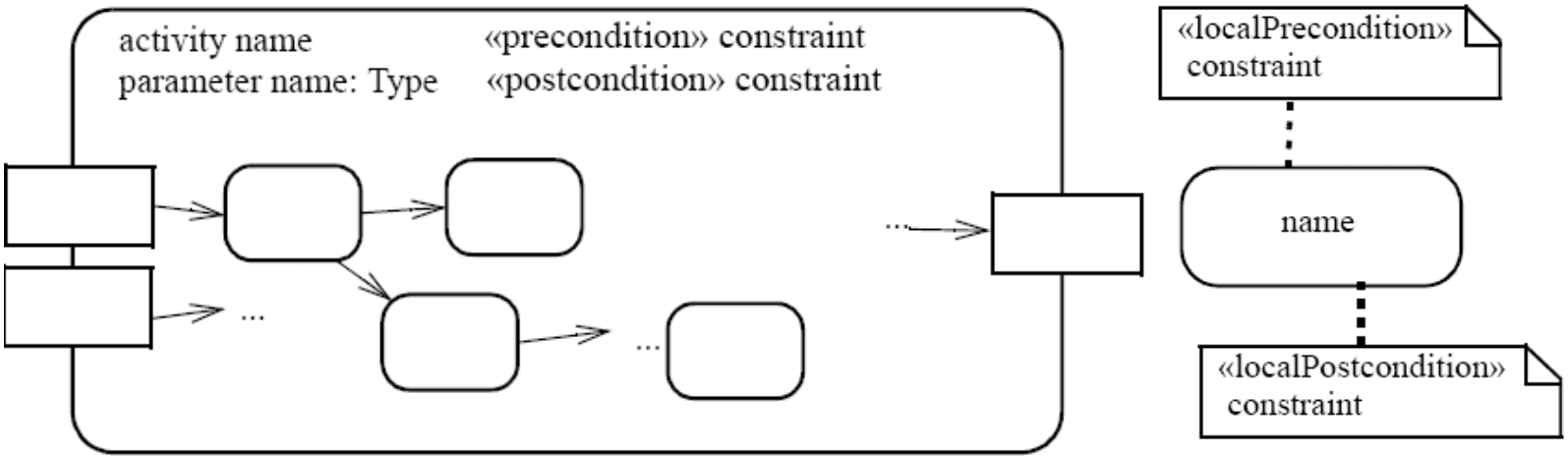
Partition notated to occur outside the primary concern of the model




«attribute» performingDept: Department
Order Department
Acctg Department
«external»
Customer
Receive Order
[order accepted]
Fill Order
Ship Order
Close Order
Send Invoice
Invoice
Make Payment
Accept Payment

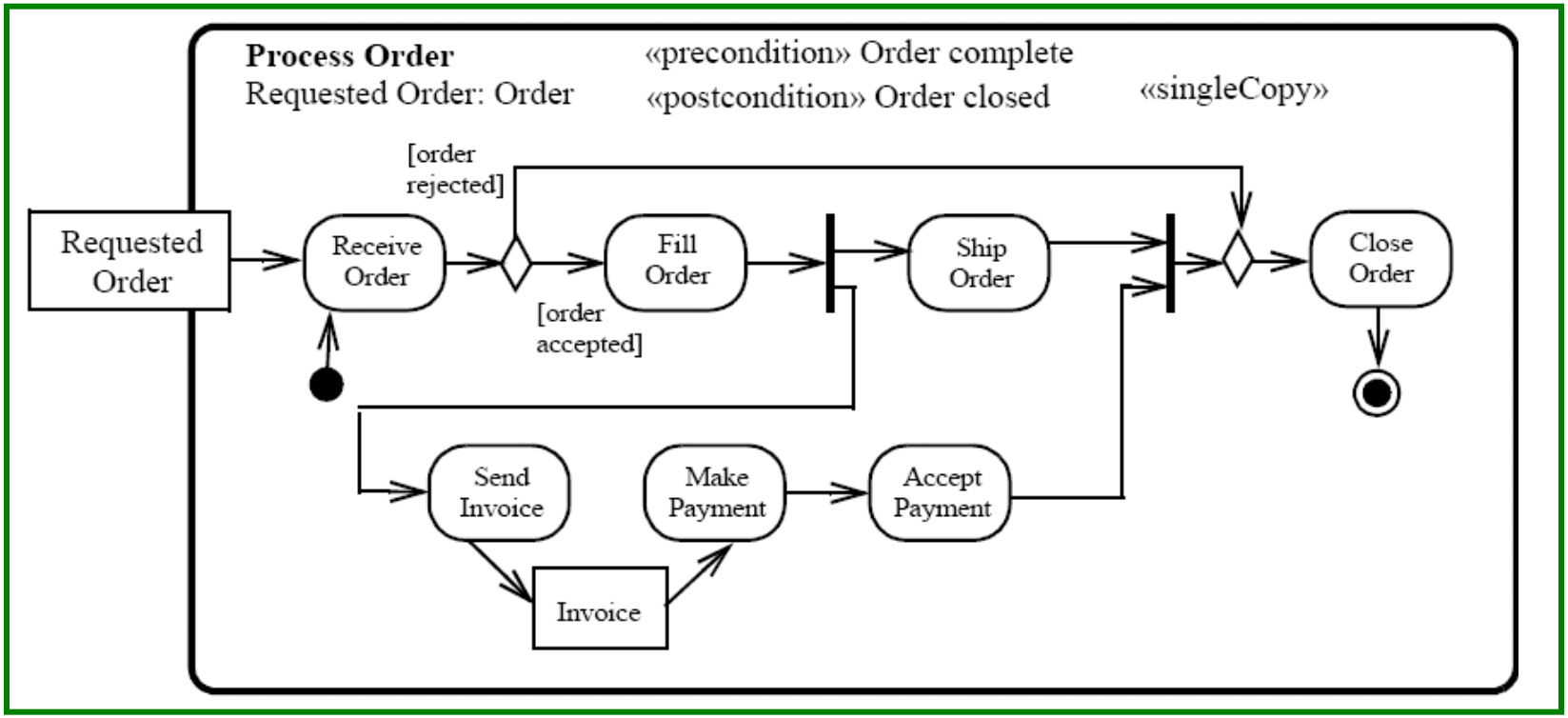
Pre & post condition (1)

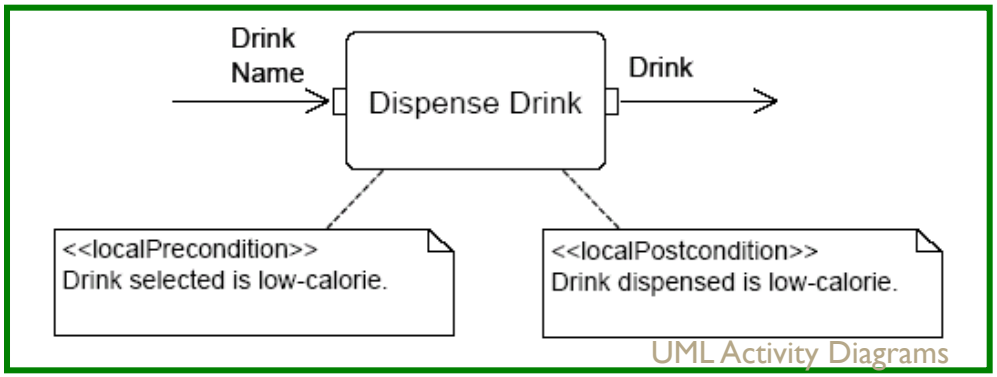
- Can be referred to an activity or to an action (local condition)



- UML intentionally does not specify when or whether pre/post conditions are tested (design time, runtime, etc.)
- UML also does not define what the runtime effect of a failed pre/post condition should be (error that stops execution, warning, no action)

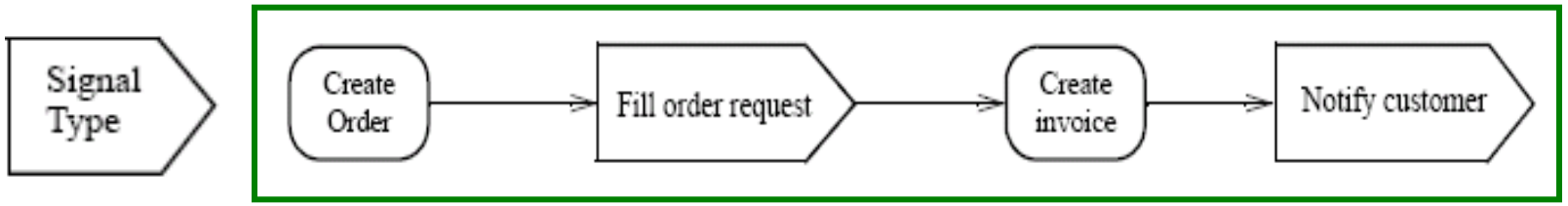
Pre & post condition (2)





SendSignalAction

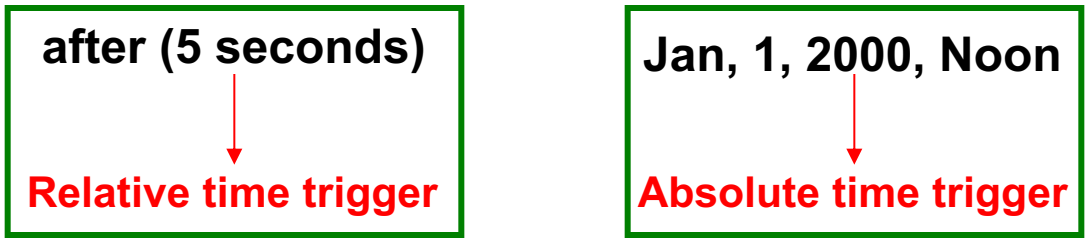
- Creates a signal instance from its inputs, and transmits it to the target object (local or remote)
- A signal is an asynchronous stimulus that triggers a reaction in the receiver in an asynchronous way and without a reply
- Any reply message is ignored



Hue University

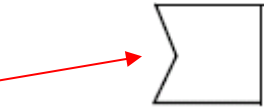
Time triggers and Time events

- A *Time trigger* is a trigger that specifies when a time event will be generated
- *Time events* occur at the instant when a specified point in time has transpired
- This time may be relative or absolute
  - ***Relative time trigger***: is specified with the keyword 'after' followed by an expression that evaluates to a time value
  - ***Absolute time trigger***: is specified as an expression that evaluates to a time value



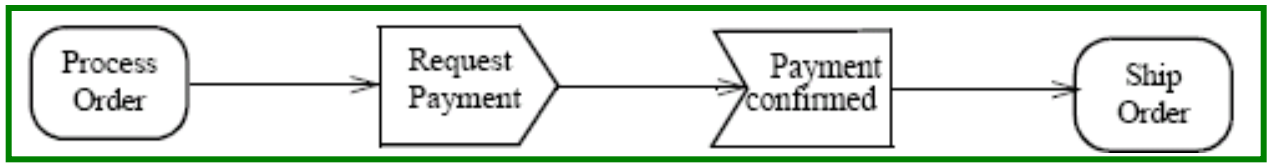
AcceptEventAction

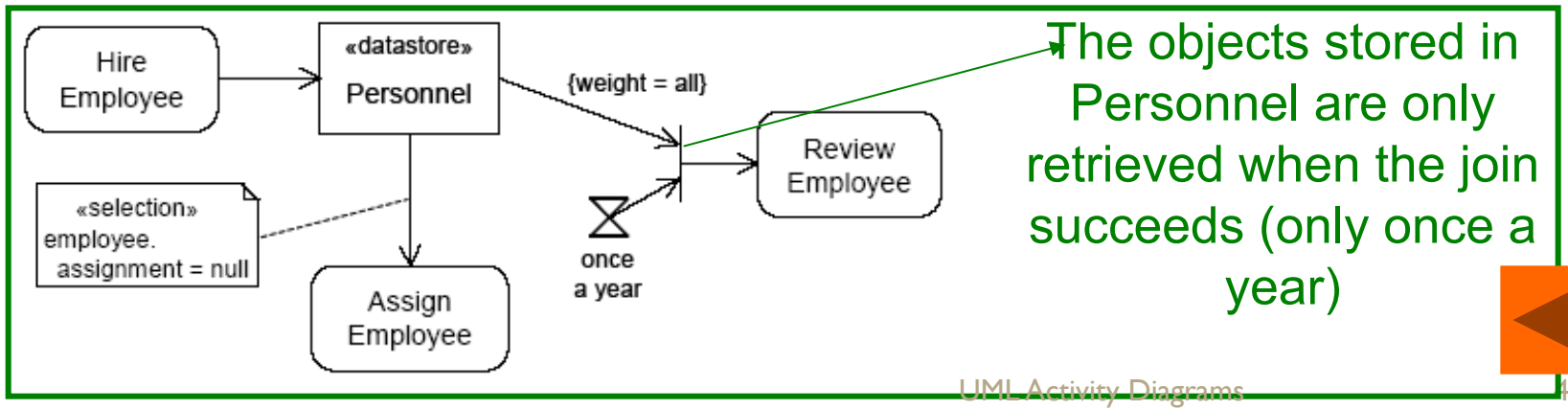
- Waits for the occurrence of an event meeting specified conditions
- Two kinds of AcceptEventAction:
  - **Accept event action** – accepts signal events generated by a SendSignalAction
  - **Wait time action** – accepts time events



Accept event action

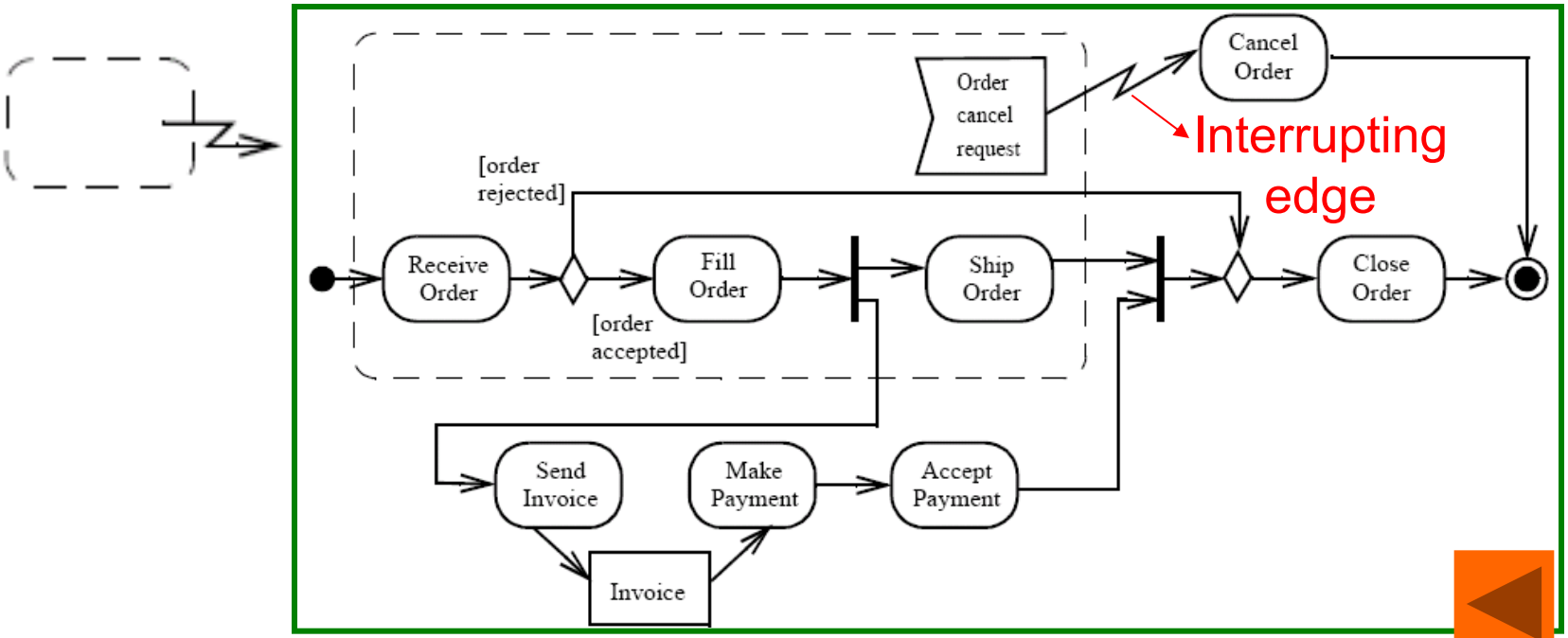
Wait time action





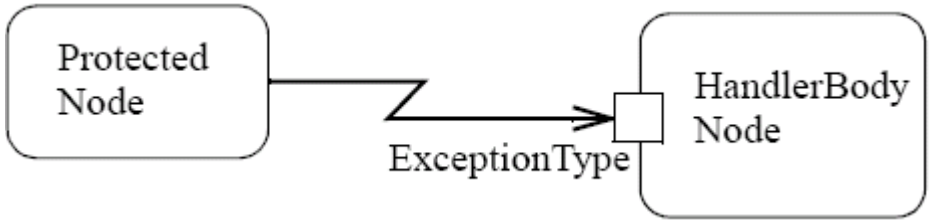
InterruptibleActivityRegion

- Is an activity group (sets of nodes and edges) that supports termination of tokens flowing into it
- When a token leaves an interruptible region via interrupting edges, all tokens and behaviours in the region are terminated
- Token transfer is still atomic: a token transition is never partial; it is either complete or it does not happen at all (also for internal stream)



Exceptions (1)

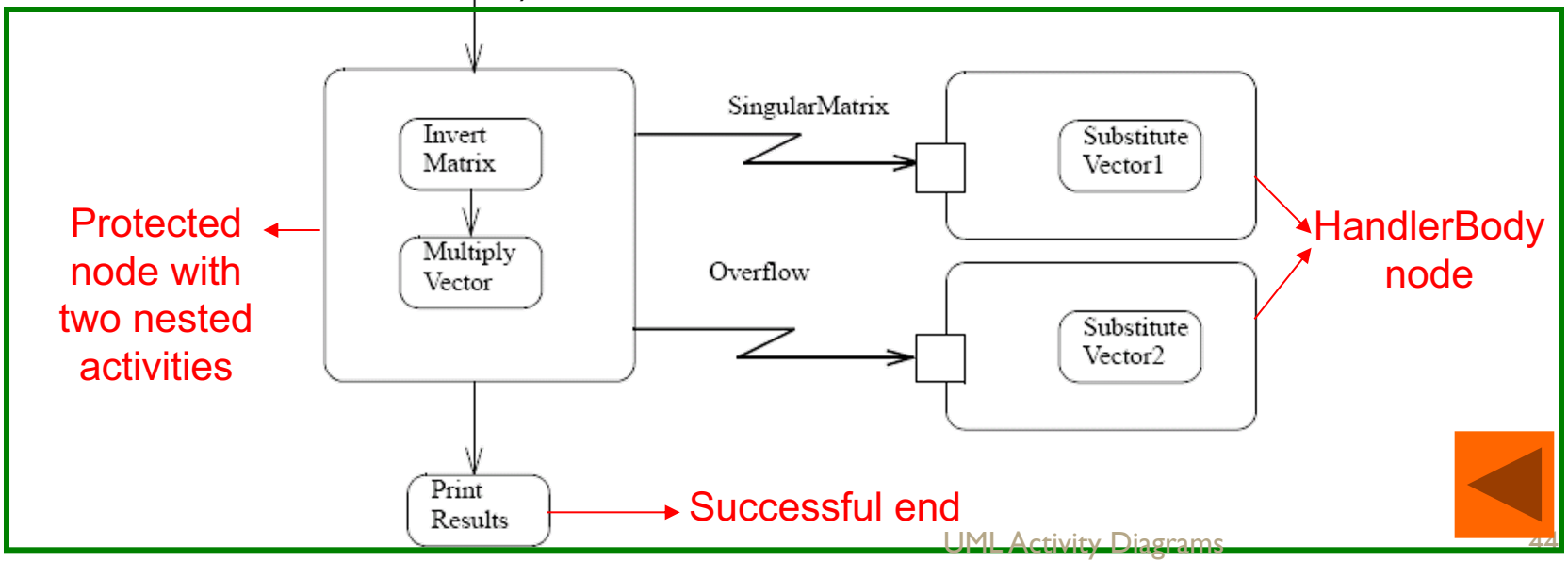
- Exception handler - specifies the code to be executed when the specified exception occurs during the execution of the protected node
- When an exception occurs the set of execution handlers on the action is examined to look for a handler that matches (*catches*) the exception
- If the exception is not caught, it is propagated to the enclosing protected node, if one exists
- If the exception propagates to the topmost level of the system and is not caught, the behaviour of the system is unspecified; profiles may specify what happens in such cases



Hue University

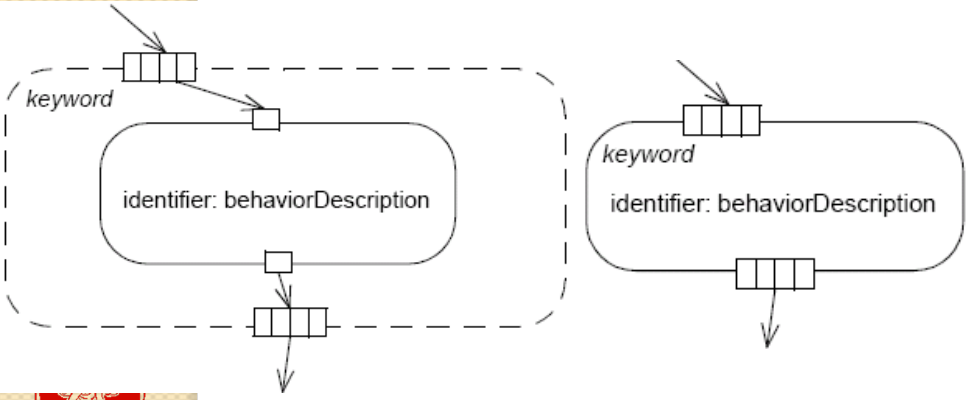
Exceptions (2)

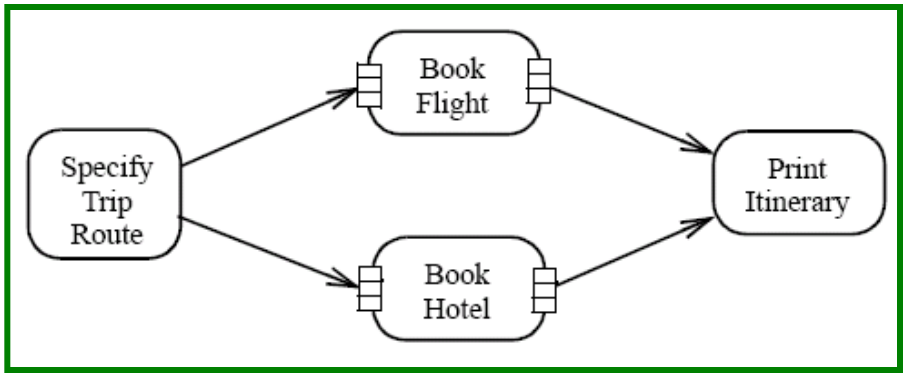
- When an exception is caught, is executed the exception body instead of the protected node, and then the token is passed to all the edges that go out from that protected node
- The exception body has no explicit input or output edges
- Exception body can resolve the problems that have caused the exception or can abort the program
- We can put any activities nested in a protected node (in UML 2.0, nesting activities is allowed)



ExpansionRegion (1)

- Nested region of an activity in which each input is a collection of values
- The expansion region is executed once for each element (or position) in the input collection
- On each execution of the region, an output value from the region is inserted into an output collection at the same position as the input elements





ExpansionRegion (2)

- There are three ways of interaction between the executions:
  - **Parallel** (concurrent): all the interactions are independent
  - **Iterative**: the interactions occur in the order of the elements (the executions of the region must happen in sequence, with one finishing before another can begin)
  - **Stream** (streaming): there is a single execution of the region, where the values in the input collection are extracted and placed into the execution of the expansion region as a stream (in order if the collection is ordered)

Hue University

Activity Diagram Heuristics

- Flow control and objects down the page and left to right.
- Name activities and actions with verb phrases.
- Name object nodes with noun phrases.
- Don't use both control and data flows when a data flow alone can do the job.
- Make sure that all nodes entering an action node can provide tokens concurrently.
- Use the [else] guard at every branch.

Hue University

When to Use Activity Diagrams

When making a dynamic model of any process.

- Design processes (what designers do)
- Designed processes (what designers create)
  - During analysis
  - During resolution

Hue University

Summary

- A process is a collection of related tasks that transforms a set of inputs to a set of outputs.
- UML activity diagrams model processes by depicting actions and the flow of control and data between them.
- Activity symbols contain activity graphs consisting of
  - action nodes
  - action edges
  - data nodes
  - special nodes for starting and stopping activities, branching, forking, and joining
- Activity diagrams can represent any process and are useful throughout software design.

Hue University

That's all for today

thank you!

Hue University