Université Nice Sophia Antipolis

Conception Orientée Objets

Activités

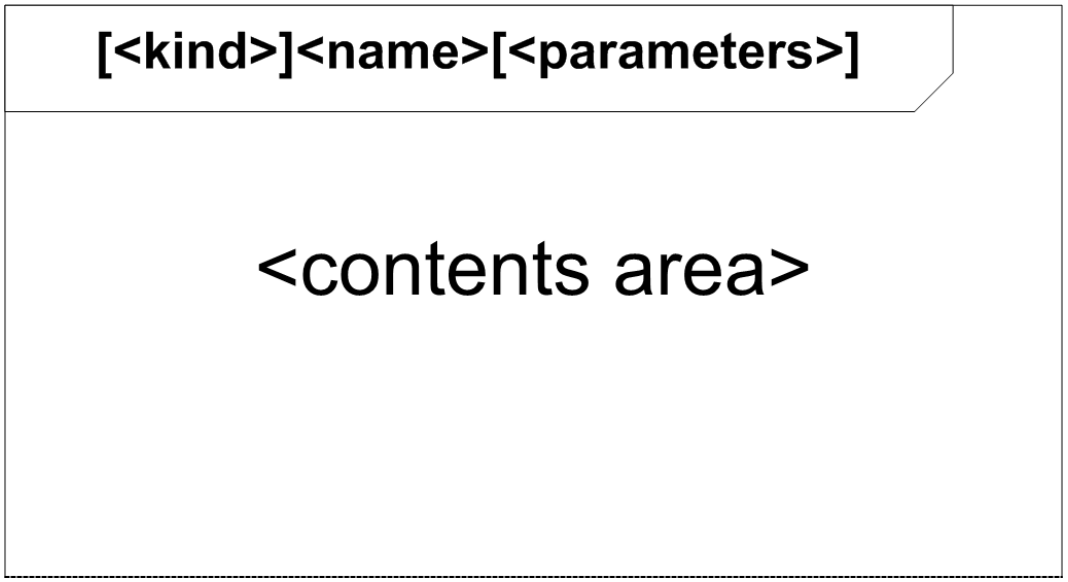
Frédéric Mallet

http://deptinfo.unice.fr/~fmallet/

Université Nice Sophia Antipolis

Diagram and frames

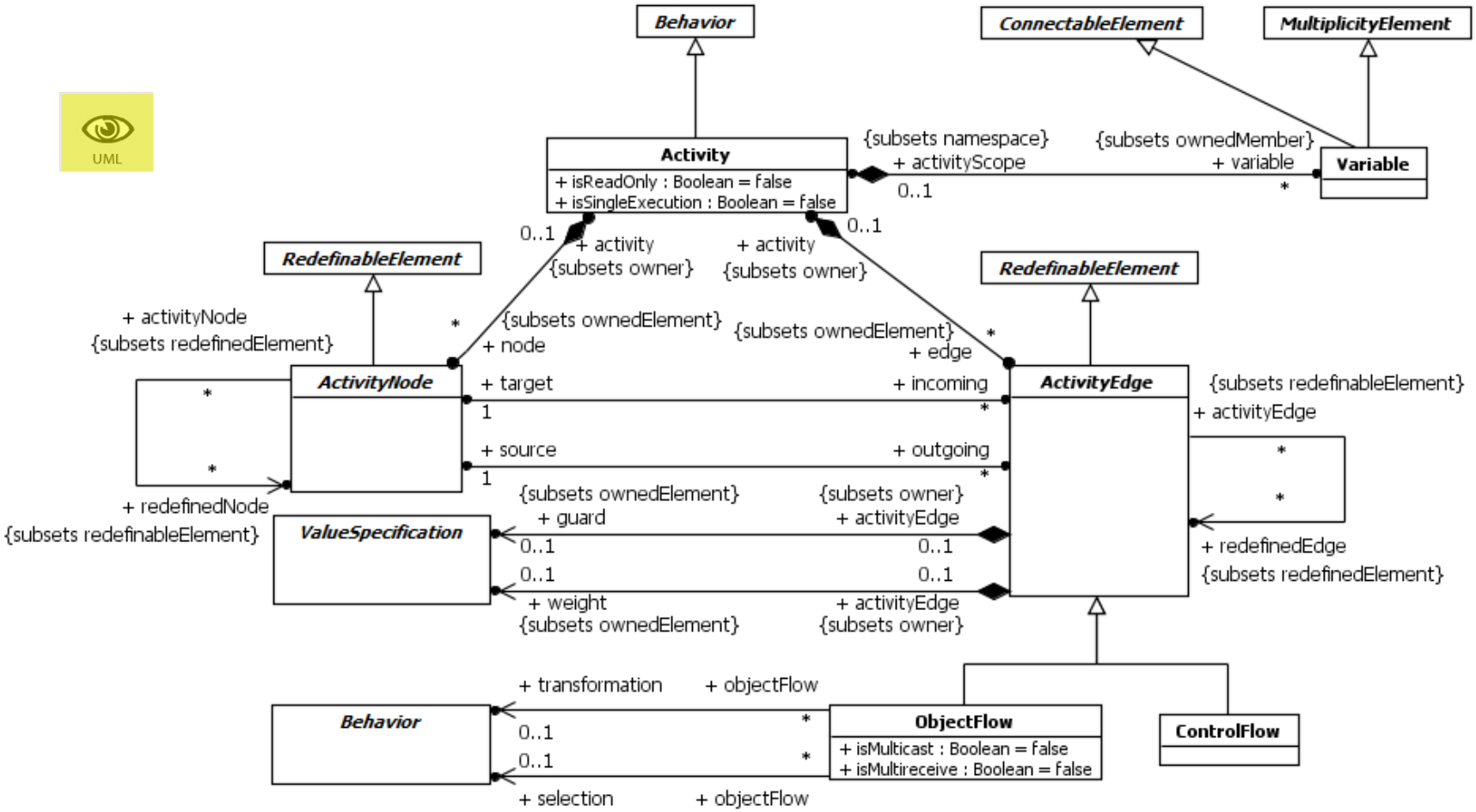
- Diagrams should be within frames
  - Heading should give a name, kind and parameters if any
    - kind ∈ { **activity**, class, component, deployment, interaction, package, state machine, use case }
    - Short form { **act**, class, cmp, dep, sd, pkg, stm, uc }



Université Nice Sophia Antipolis

Activity

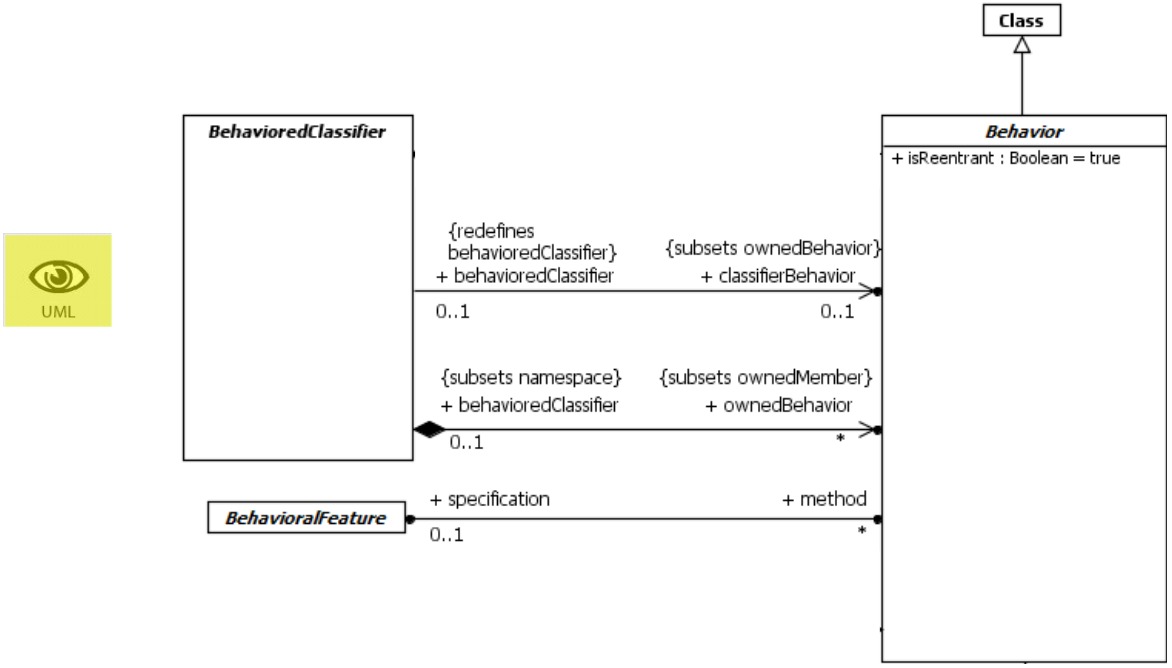
- Meta-Model (UML 2.5, ptc/2013-09-05 , Fig. 15.1)



Université Nice Sophia Antipolis

UML Activities

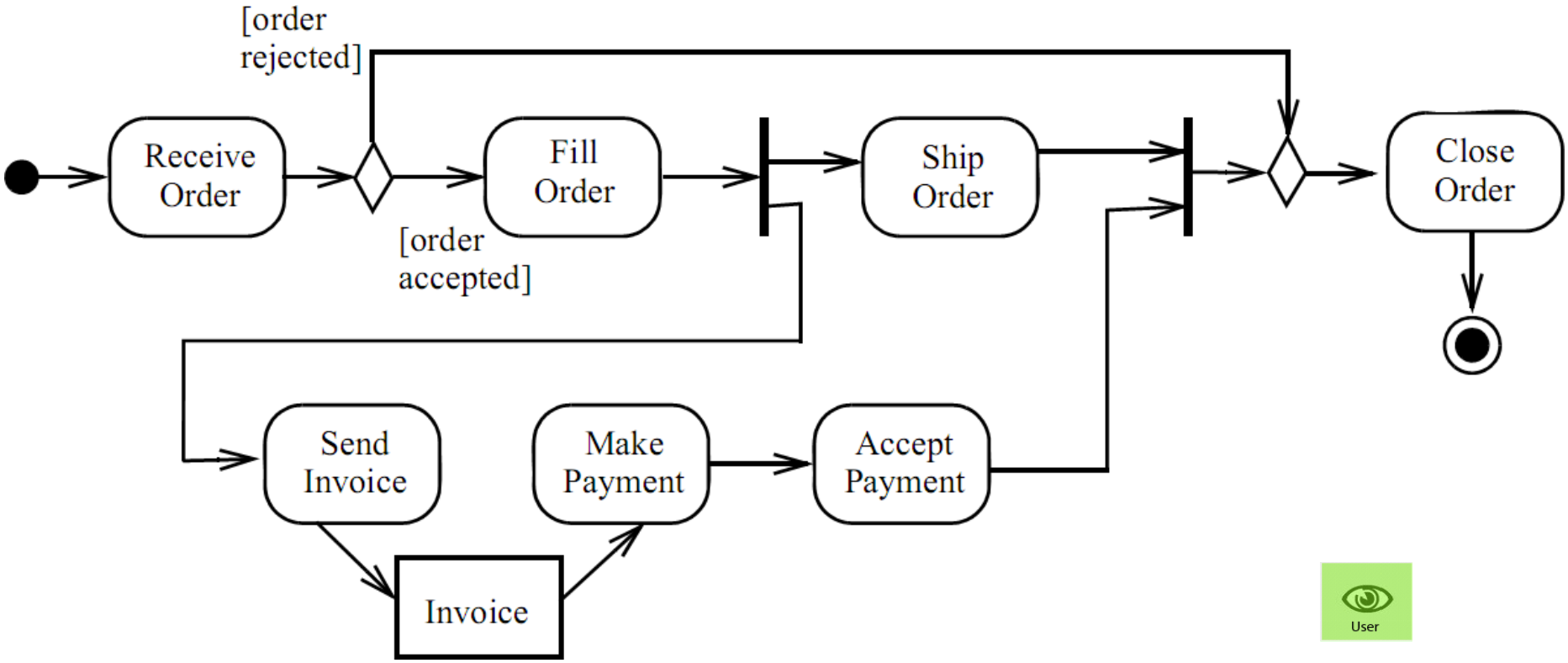
- Usually one classifier owns several behaviors
  - All the state machines are executed concurrently
  - One behavior describes THE behavior



Université Nice Sophia Antipolis

Activity: example

- Token flow semantics (Petri Net)



Université Nice Sophia Antipolis

Activities

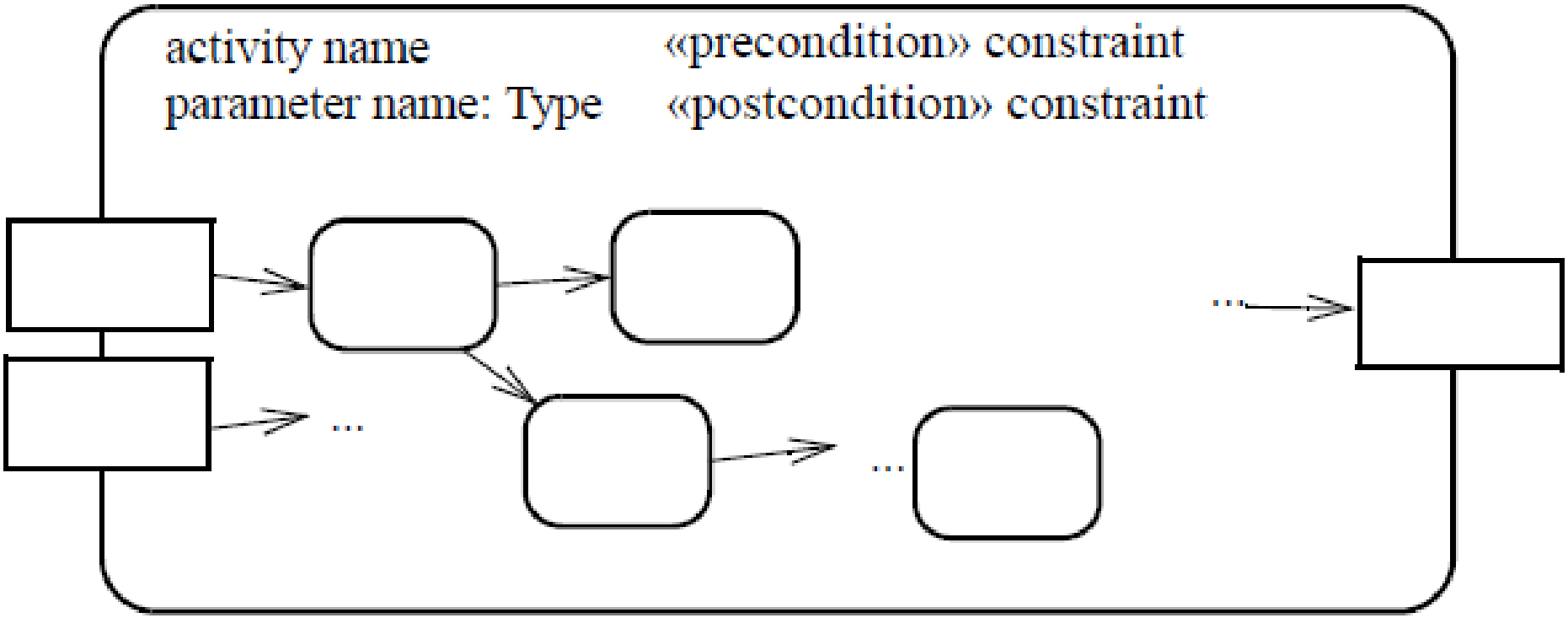
- Activities
  - Represent sequence of actions;
  - Combine data-flow and control-flow:
    - SDF, simulink, pipeline, streaming, signal processing, workflow, business process, BPMN, queueing networks
  - Token flow semantics, like Petri Nets.

- Can describe
  - The behavior of ONE Use Case: what should happen
    - Not necessarily meaning that the actual realization should behave exactly like that
  - A general behavior to be realized by several classes;
  - The specific algorithm implemented by ONE method.

Université Nice Sophia Antipolis

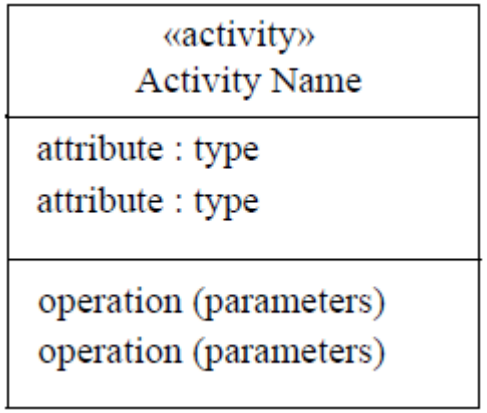
Activity

- A graph: flow of tokens



Université Nice Sophia Antipolis

Activities are classes

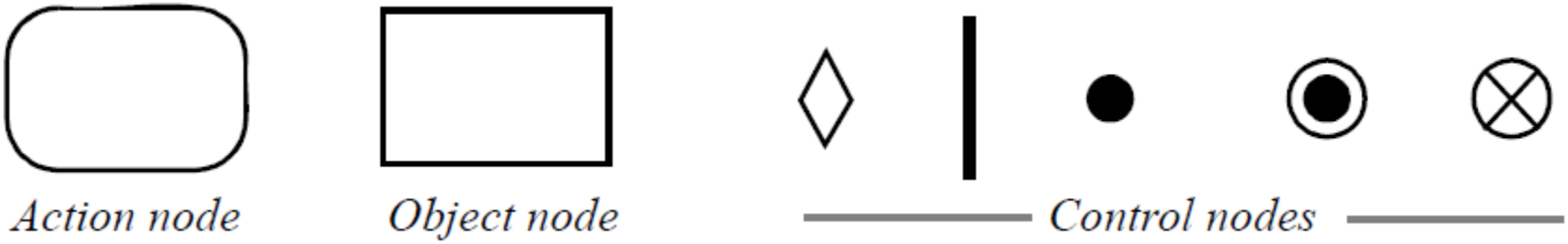


- They may have properties
  - How long has the process been running ?
  - What is its cost?
- They may have operations:
  - To manage their execution (called by the action nodes);
  - E.g., starting, stopping, aborting.
- They may have state machines:
  - To describe their internal states.

Université Nice Sophia Antipolis

Activity Nodes

- Three kinds of activity nodes
  - Executable Nodes (Actions)
    - Where the actions take place.
  - Control Nodes
    - Act as “traffic switches”;
    - Tokens cannot rest at control nodes (exception initial and fork).
  - Object nodes
    - Hold object tokens from object flows.



Université Nice Sophia Antipolis

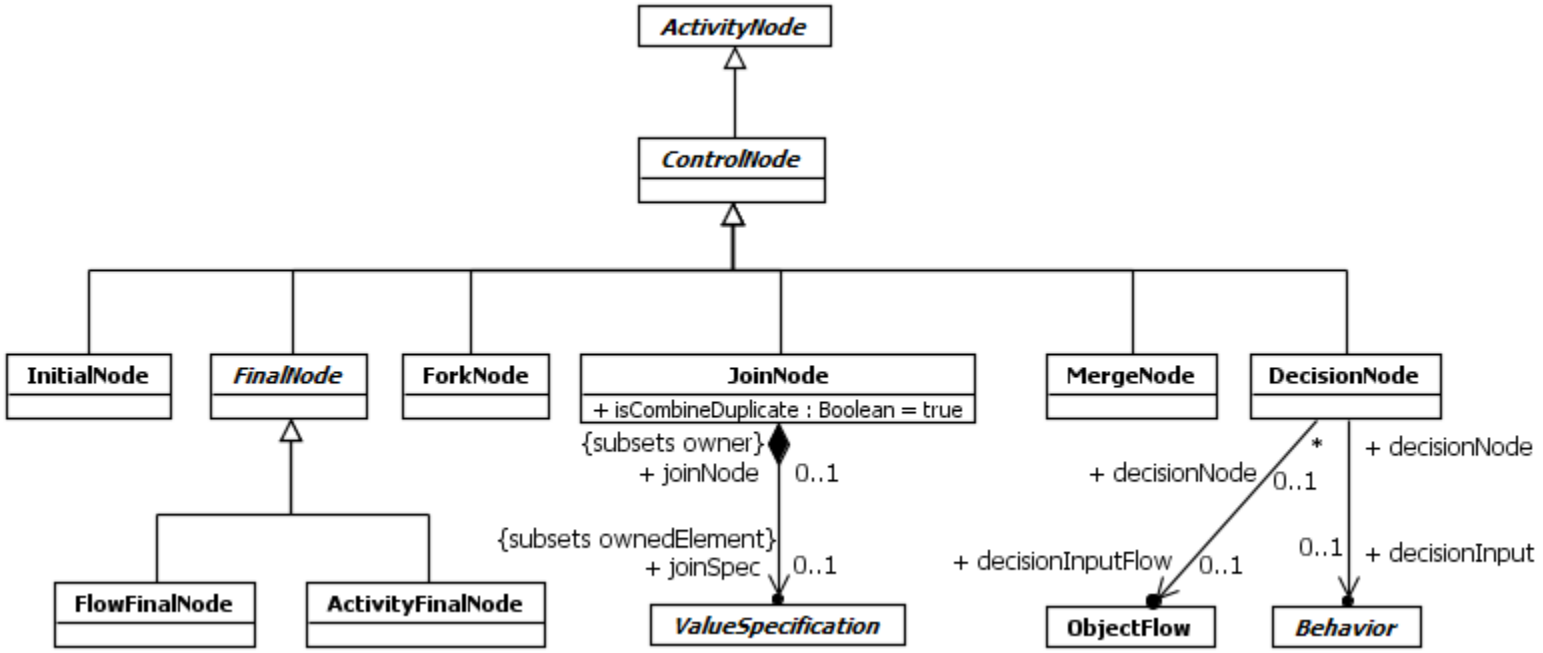
Executable Nodes

- Semantics:
  - If several incoming control flows, there must be tokens offered on each of them before beginning;
  - During execution the node hold one token;
  - After execution it offers one token on each control flow;
  - May consume/produce object tokens from object flows or input/output pins.

Université Nice Sophia Antipolis

Control Nodes

- Act as “traffic switches”
  - Tokens cannot rest (except Initial and Fork)



Université Nice Sophia Antipolis

Control nodes: initial and final

- Initial nodes [0..*] 
  - One token is offered at the beginning of the activity on each initial node (concurrent flows)
- Final nodes [0..*]
  - Flow final
    - The Token reaching this node is destroyed
  - Activity final
    - The first token reaching one activity final stops the activity;
    - All the tokens hold in object nodes are destroyed (except OutputParameterNodes);
    - All the behaviors called synchronously terminates;
    - Asynchronous behaviors called are not affected.

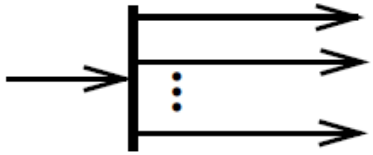
Activity final

Flow final

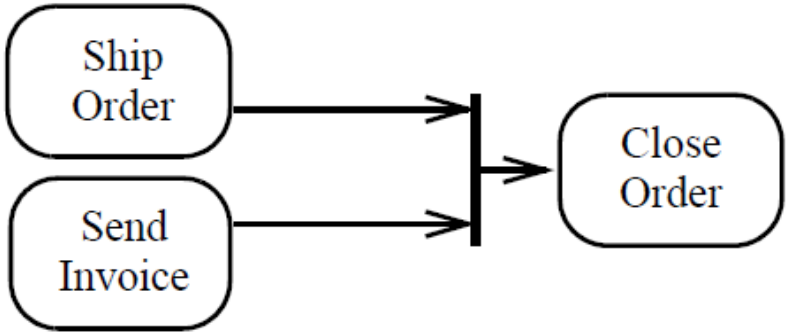
Control nodes: Fork and join

- Duplicate or merge tokens

Fork node (without flows)

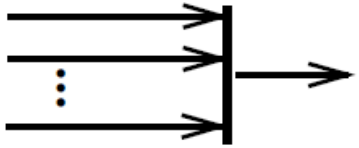


Fork node (with flows)

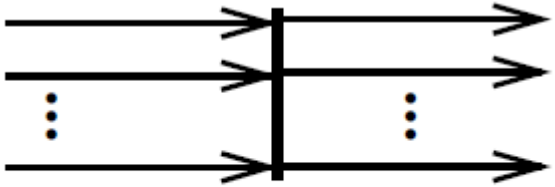


Join node (with flows and a join specification)

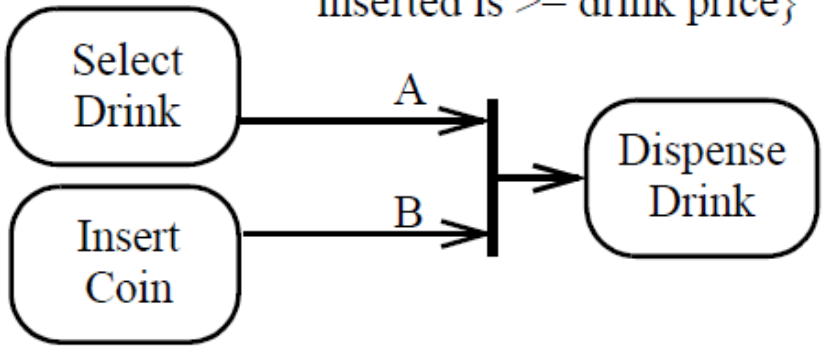
{joinSpec = ...}



{joinSpec =
A and B
and the total coin value inserted is >= drink price}



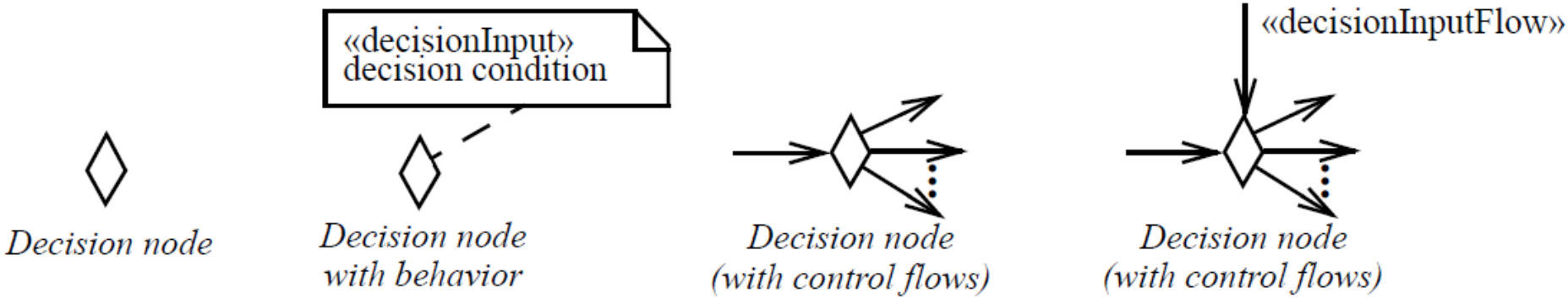
Join node and fork node used together, sharing the same symbol



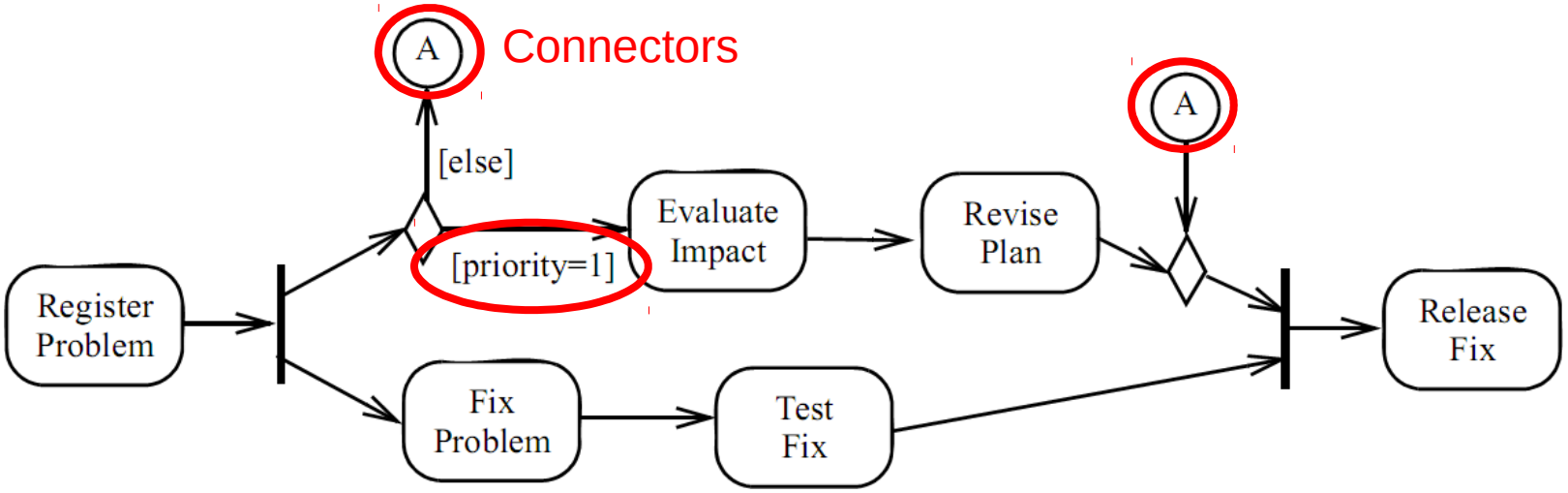
Université Nice Sophia Antipolis

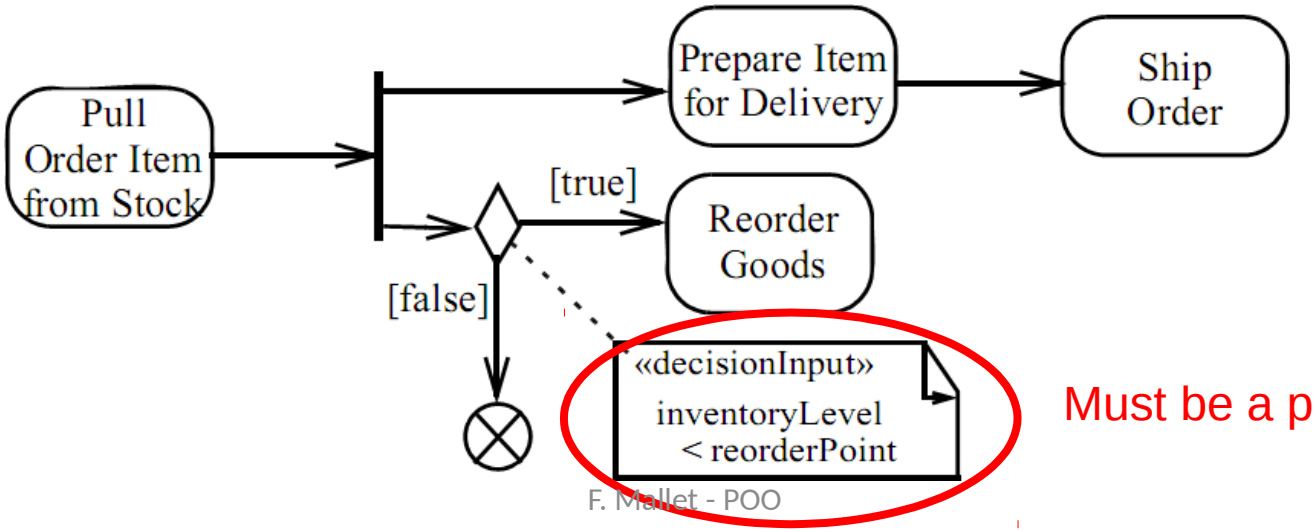
Decision Nodes

- Act as “Traffic switches”
  - One or two input flow edges
    - At least one primary input edge
    - If two, one is the decision input flow (must have its own object token)
  - Either only control edge or object flow edge
  - No duplication:
    - from the primary input edge to one of the output edge (non-determinism)
  - DecisionInput: a behavior with
    - Zero or One parameter (depending on decision input flow)
    - One return parameter (given to the guards on the output edges)



Priorities, Connectors & Decision inputs





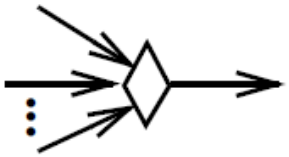
Must be a predicate

Université Nice Sophia Antipolis

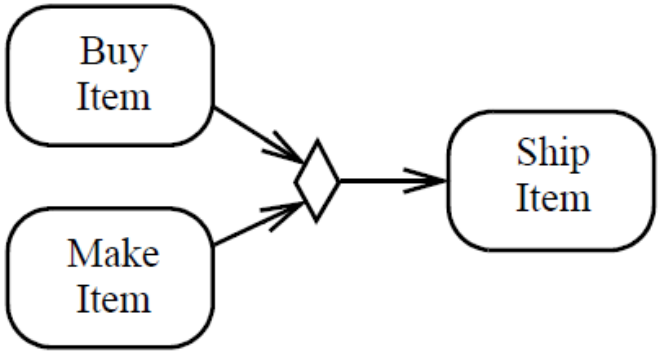
Merge nodes

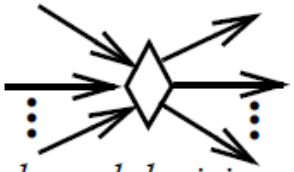
- Merge tokens
  - Only one token on one of its inputs
  - Either control or object flows but no mix

Merge node

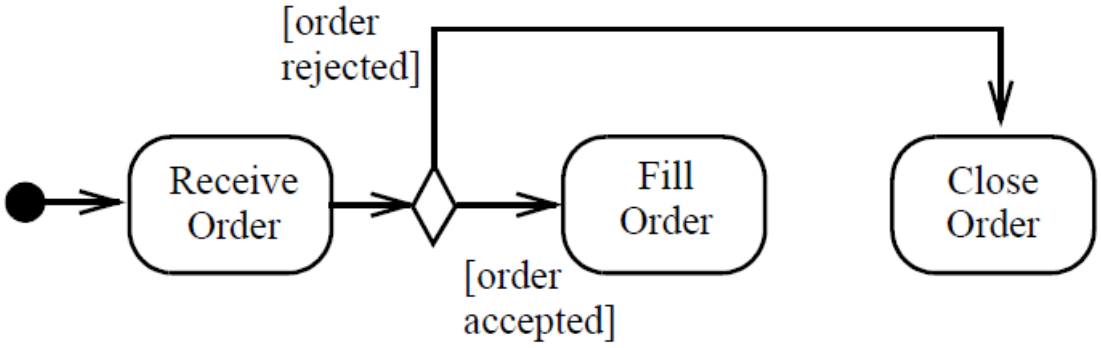


Merge node (with flows)

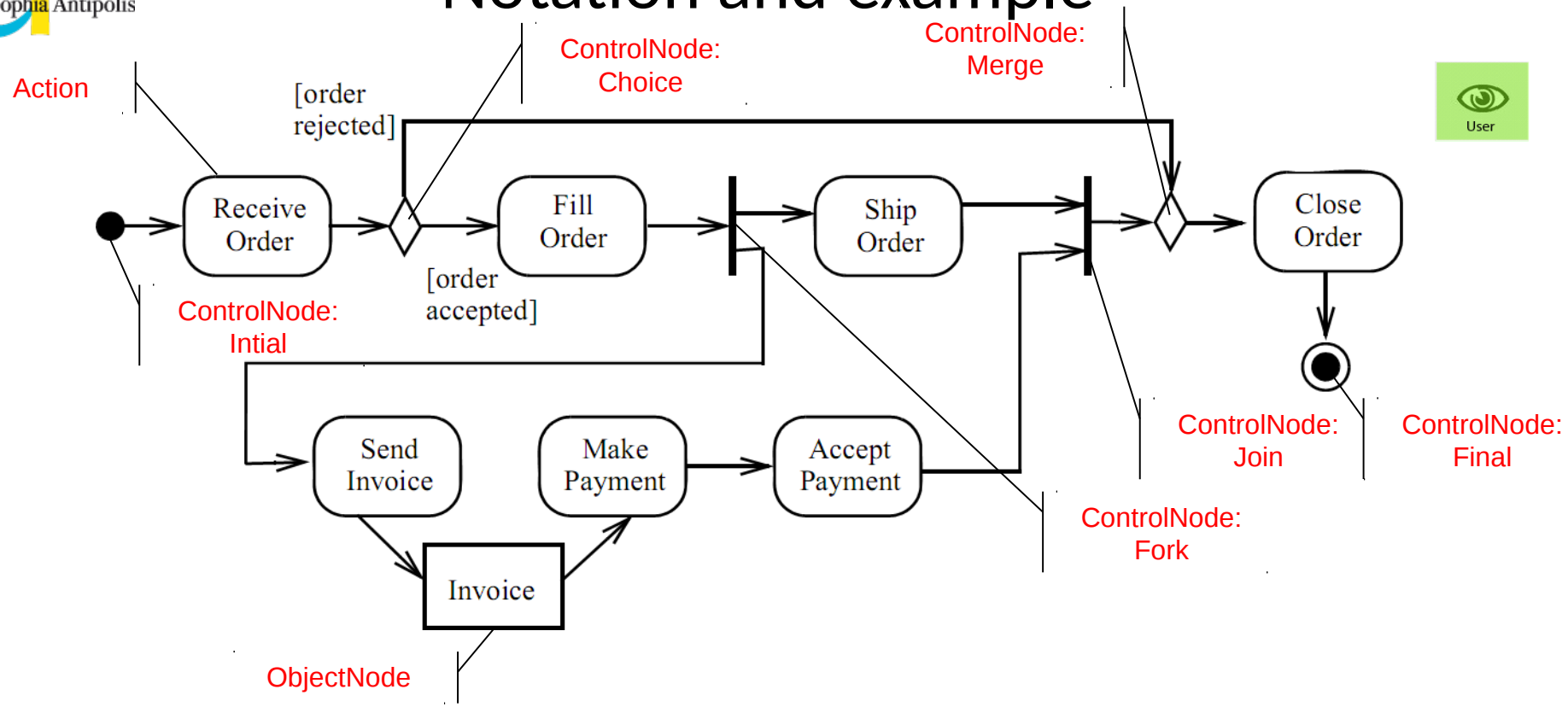




Merge node and decision node used together, sharing the same symbol



Notation and example

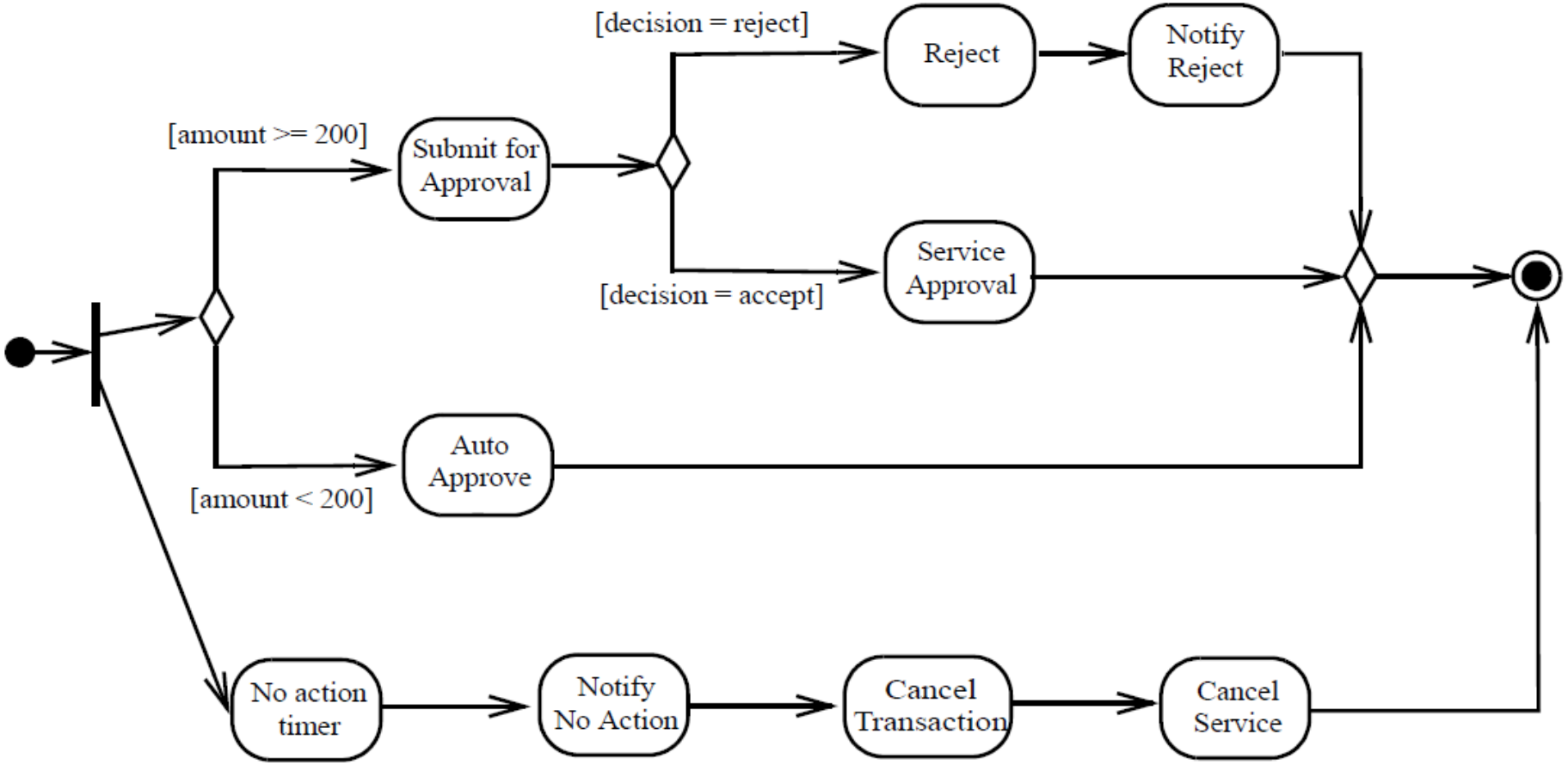


- Token flow semantics (Petri Net)
  - (control/object) Nodes are places (unbounded)
  - Edges are transitions
  - Tokens = data or control

Université Nice Sophia Antipolis

Activity Final Node

- Example of concurrent flows



Université Nice Sophia Antipolis

Exercice

- Faire un diagramme d'activité pour le problème suivant:

Quand un distributeur a un projet d'aménagement ou d'extension de ses équipements, il doit obtenir l'aval du siège, qui se traduit par sa participation au financement de l'opération.

Une fois établi, le dossier de projet est donc soumis simultanément à la banque et au siège, qui répond très rapidement.

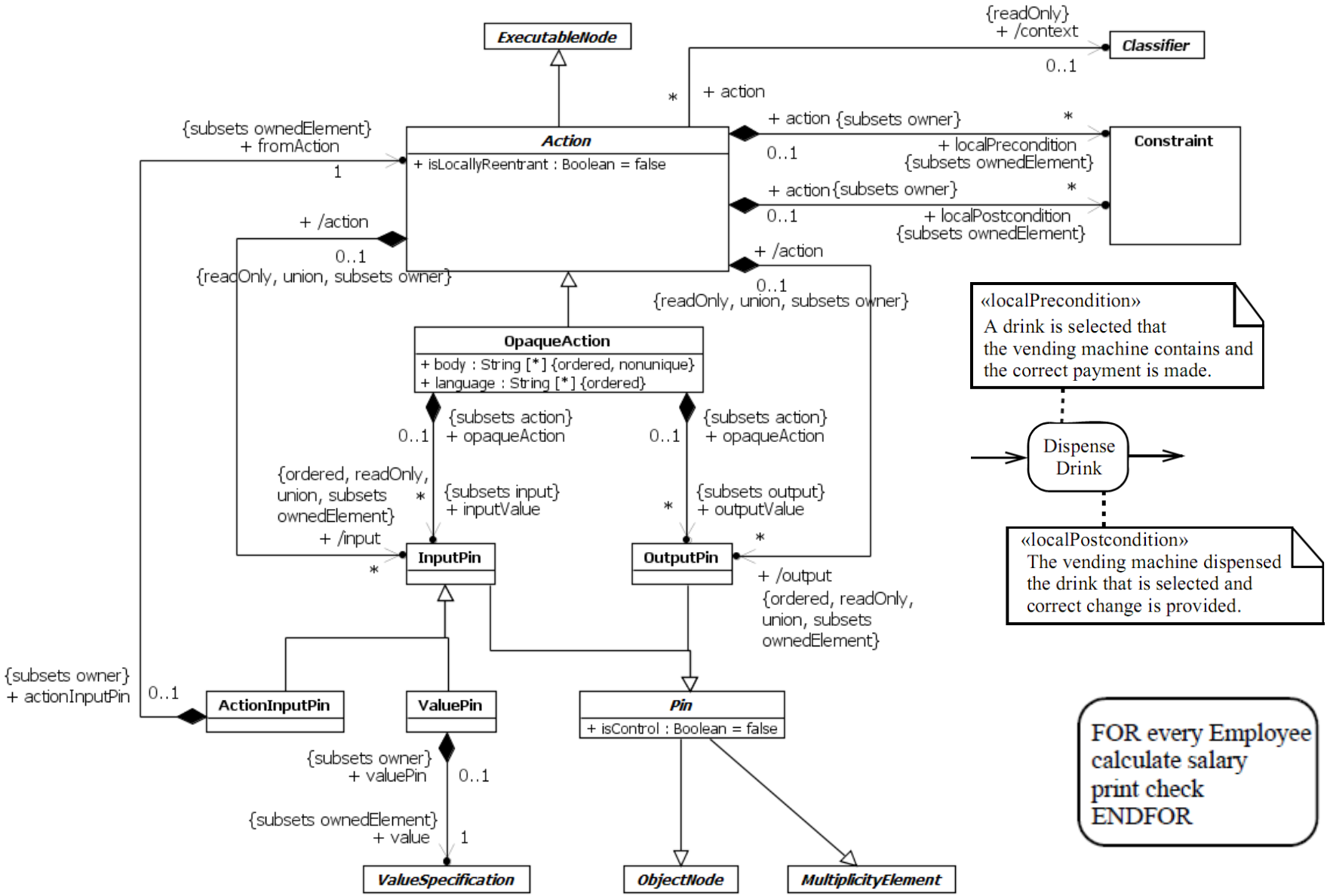
Si le siège est défavorable, le projet est abandonné et la banque est prévenue.

Si le siège accepte de co-financer le projet, on attend la réponse de la banque pour décider de poursuivre ou de réétudier le dossier.

Quand les deux réponses sont positives, un dossier de financement définitif est établi puis le projet est lancé.

Université Nice Sophia Antipolis

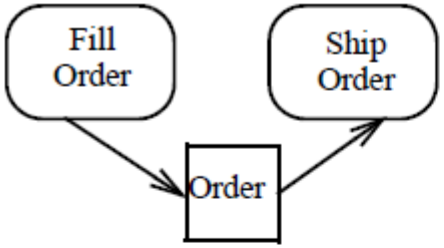
Actions

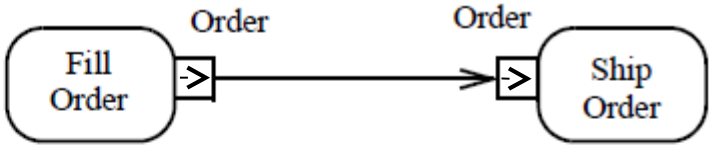


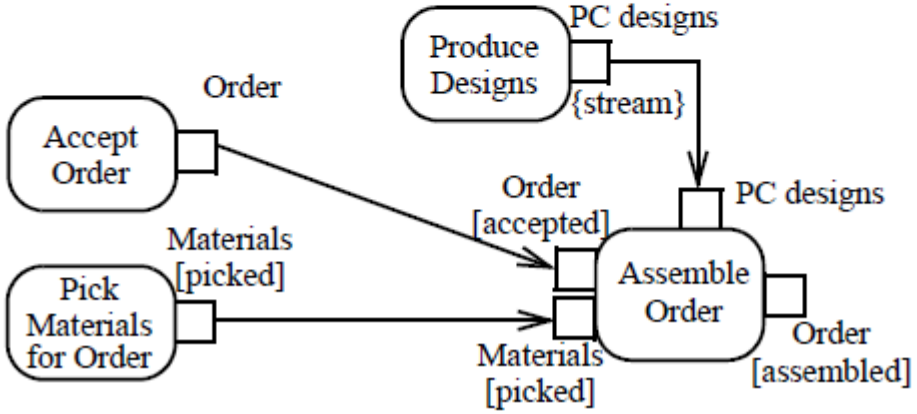
Université Nice Sophia Antipolis

Actions and pins

- (Input/Output) Pins are object nodes
  - They are connected to object flows (not control flows)

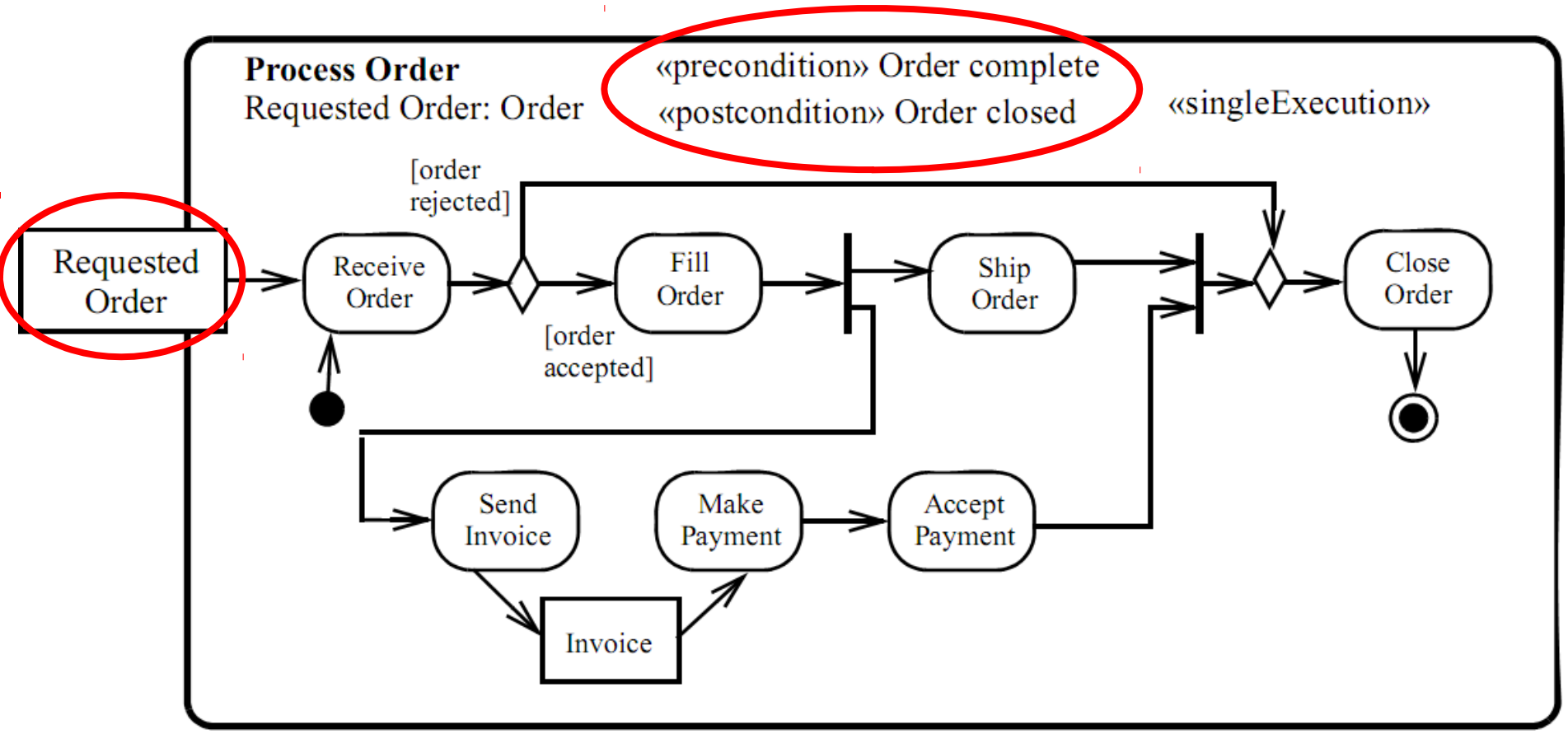






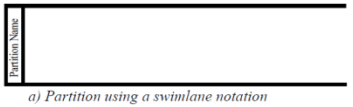
Université Nice Sophia Antipolis

Input parameters & pre/post conditions

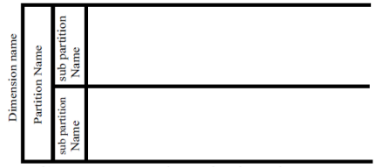


Université Nice Sophia Antipolis

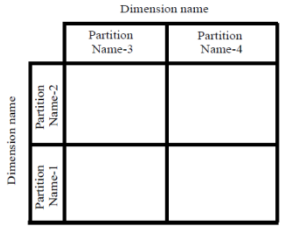
Activity Partitions



a) Partition using a swimlane notation



b) Partition using a hierarchical swimlane notation



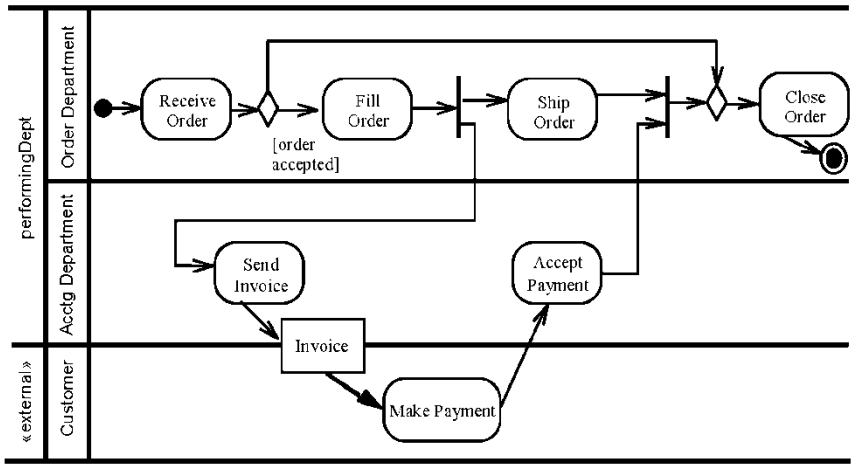
c) Partition using a multidimensional hierarchical swimlane notation

- Rationale
  - Group some nodes
  - Partitions can share nodes
  - Partitions do not affect the token flows
- A partition can represent
  - A Classifier
    - All the actions are in the context of one instance of its classifier
  - An Instance Specification
    - All the actions are applied to this particular instance
  - A Property
    - The context is the type of the Property
    - All the actions are applied to the Property

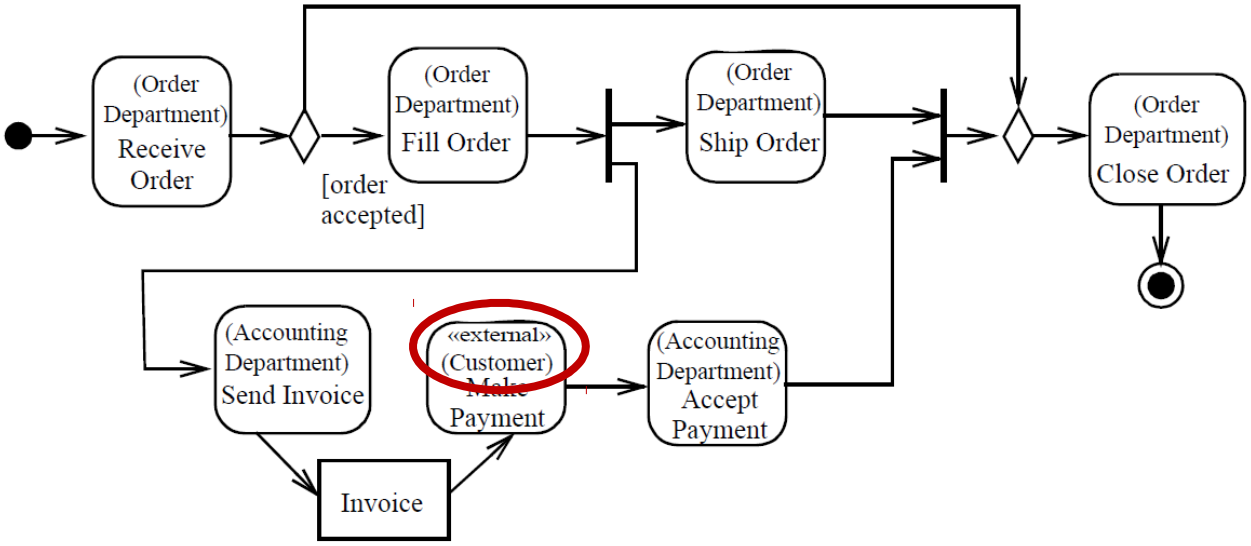
Université Nice Sophia Antipolis

Activity Partitions

- Examples
  - Swimlanes

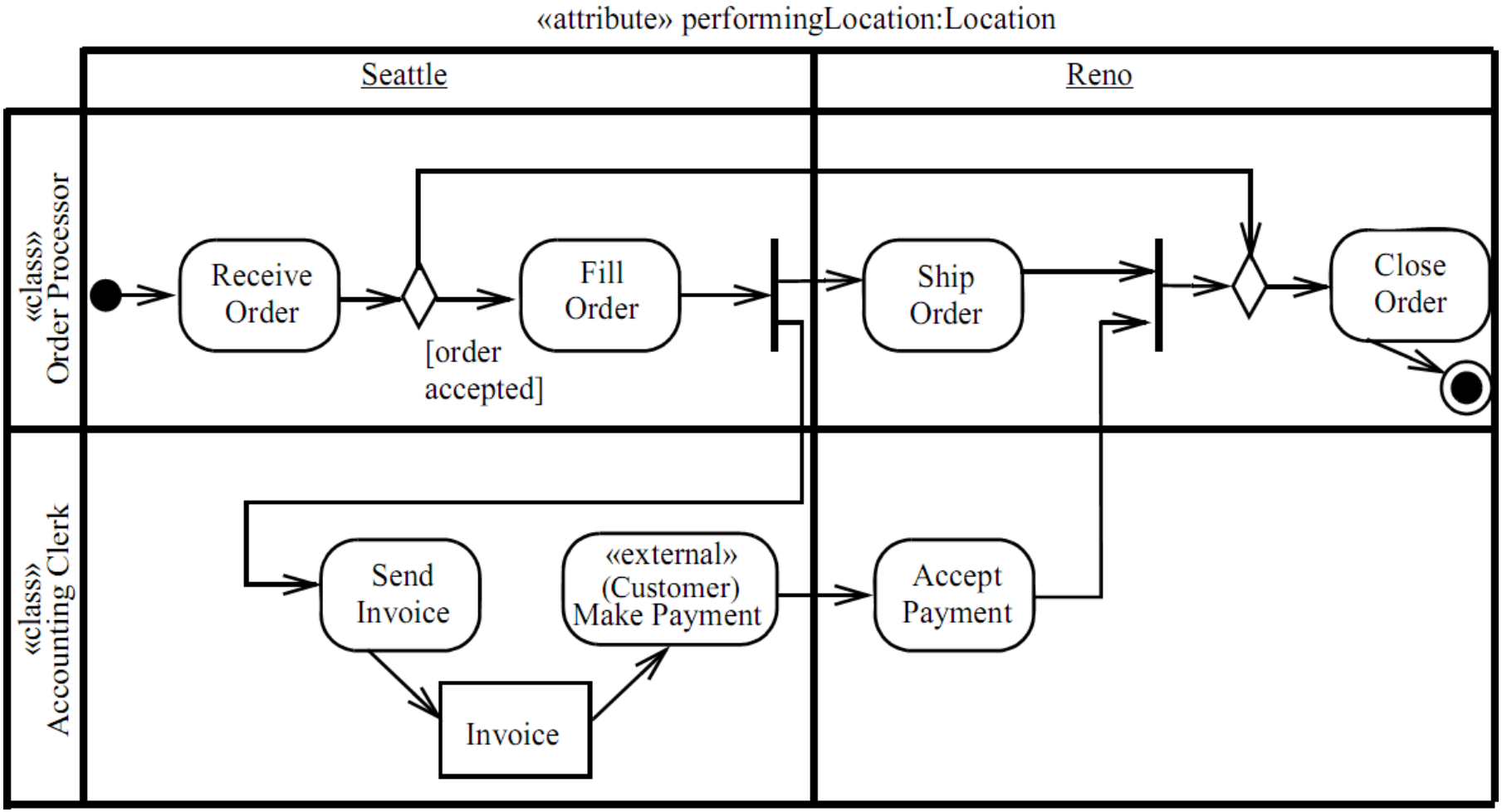


  - Annotations



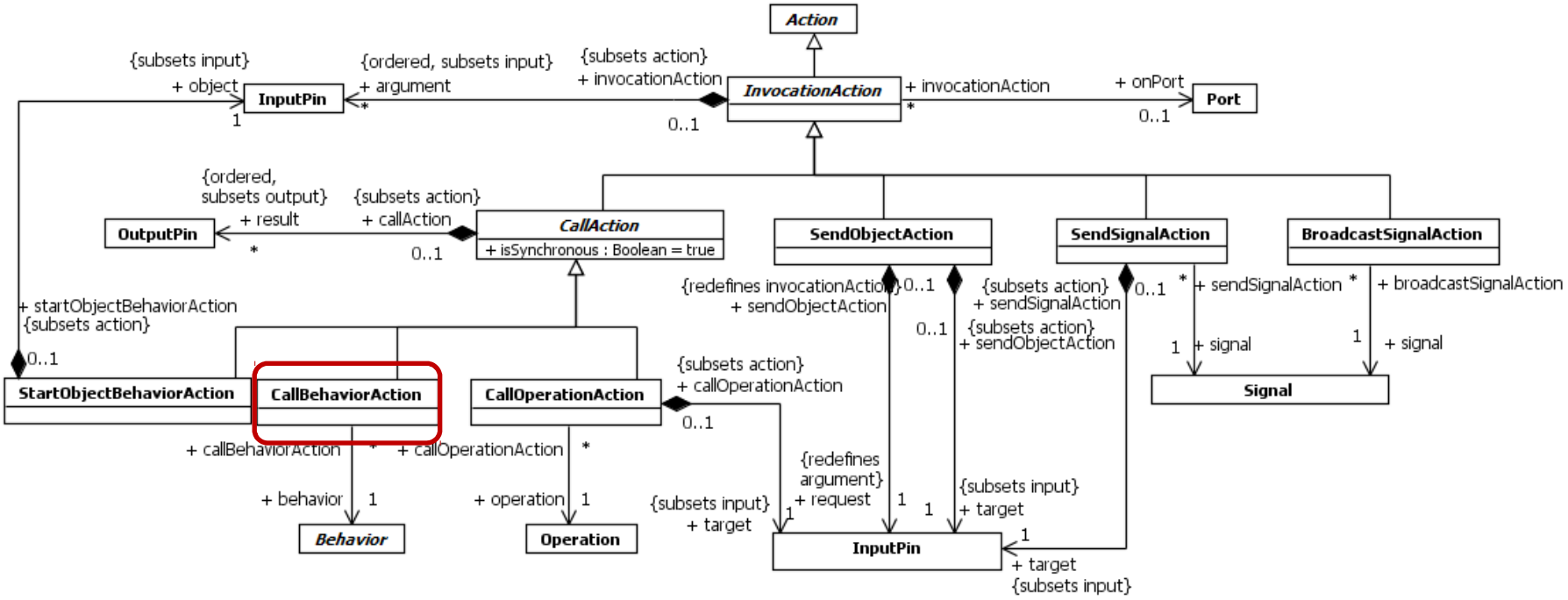
Université Nice Sophia Antipolis

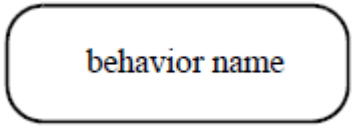
SwimLanes

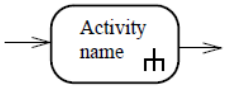


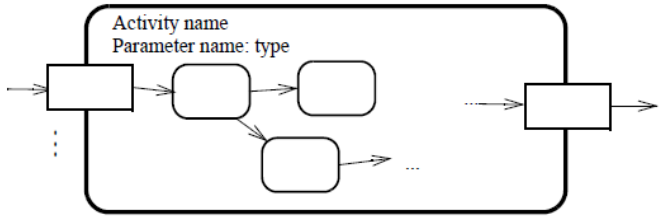
Invocation Actions: CallBehaviorAction

- Actions can be refined into different classes





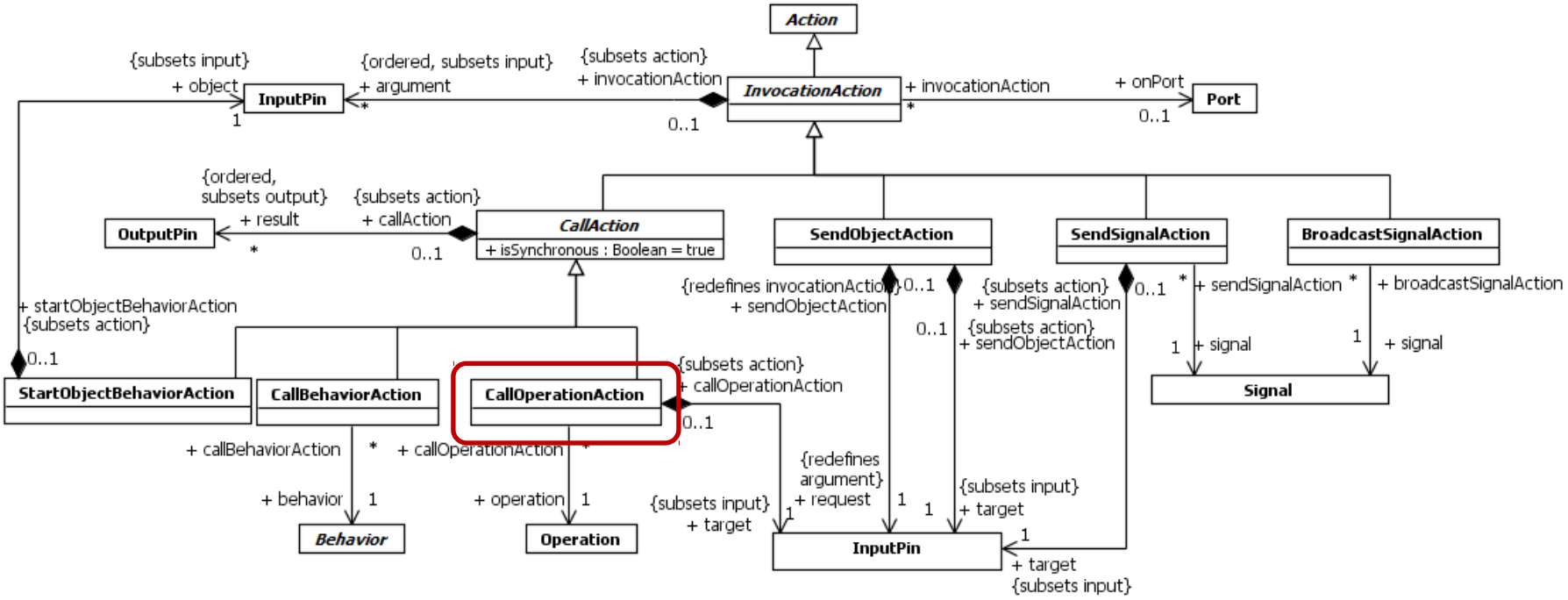




(Note: the border and name are the notation; the other symbols are present to provide clarity, only.)

Invocation Actions: CallOperationAction

- Actions can be refined into different classes



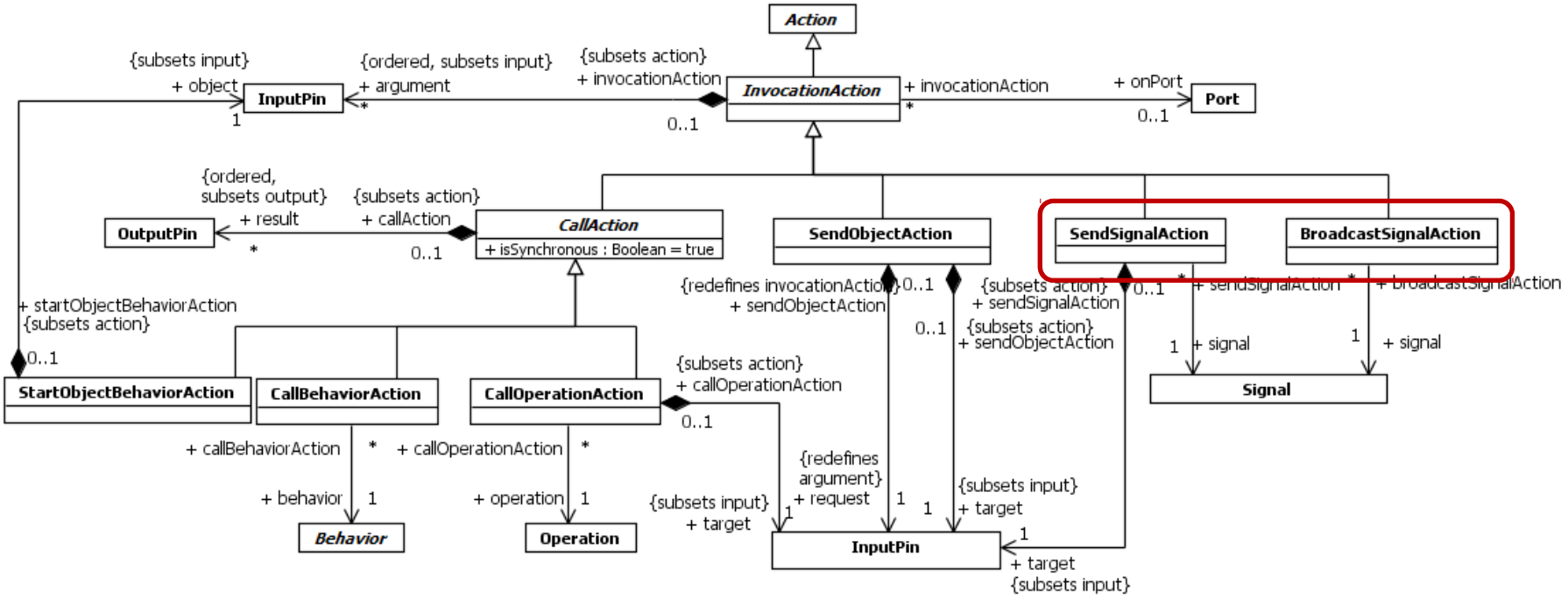
operation name

name
(ClassName::)

name
(ClassName::OperationName)

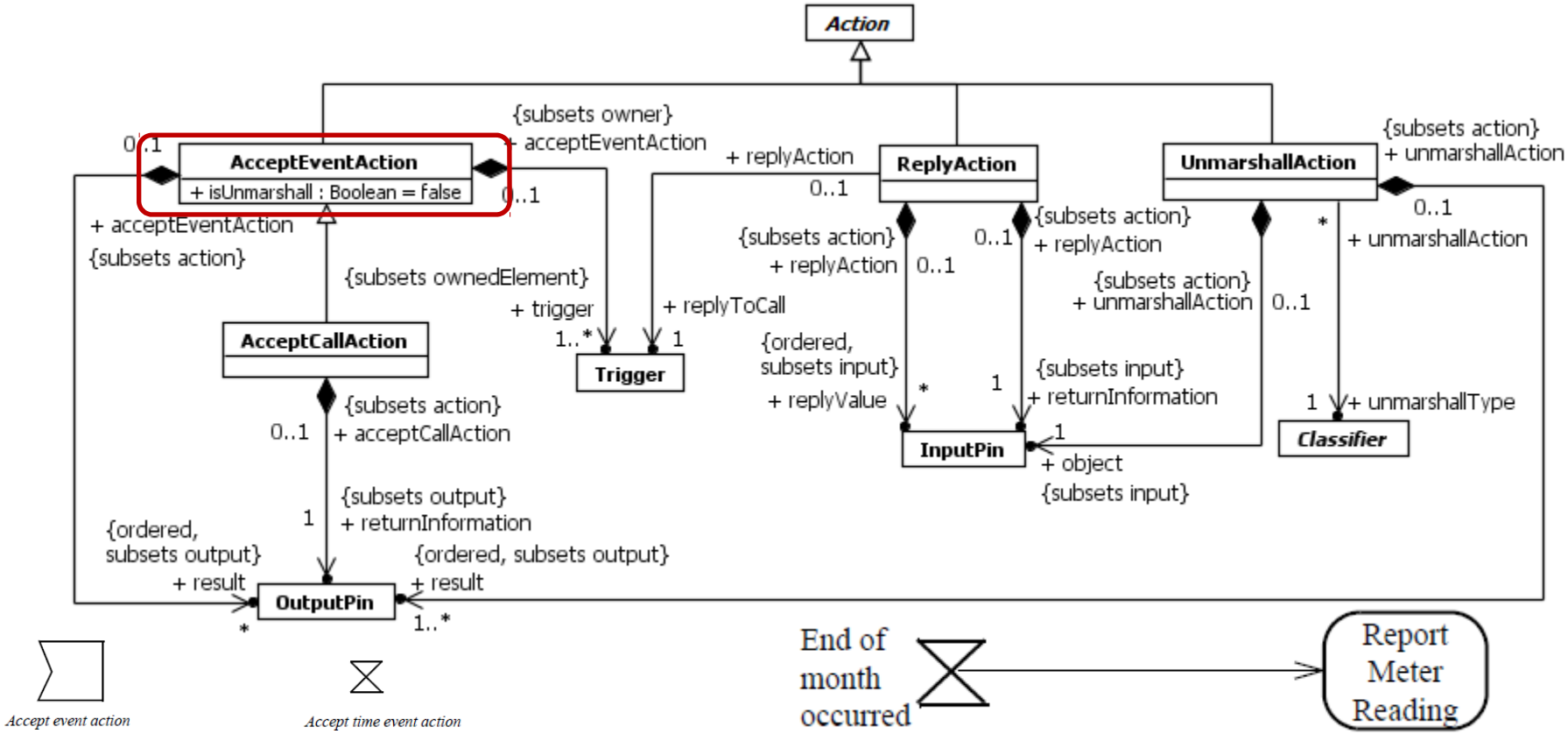
Invocation Actions: SendSignalAction

- Actions can be refined into different classes



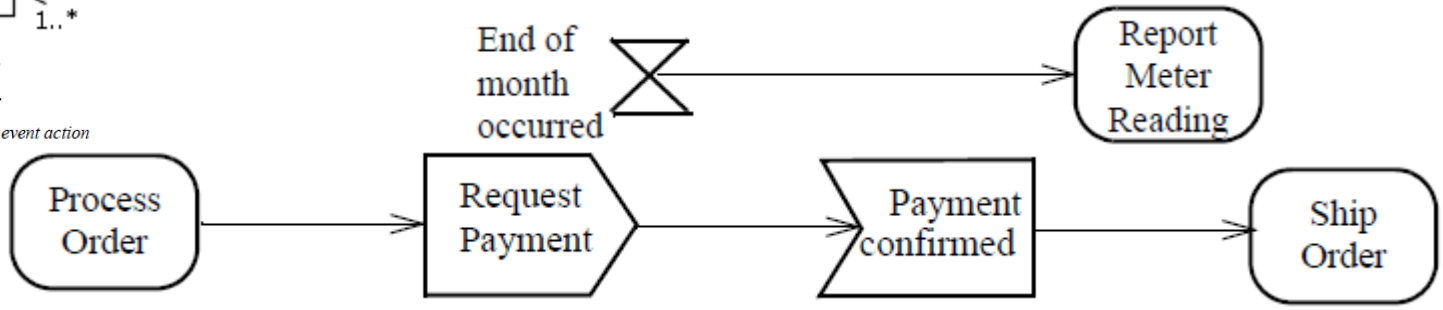
AcceptEventActions

- Receive Events (see SendSignalActions)



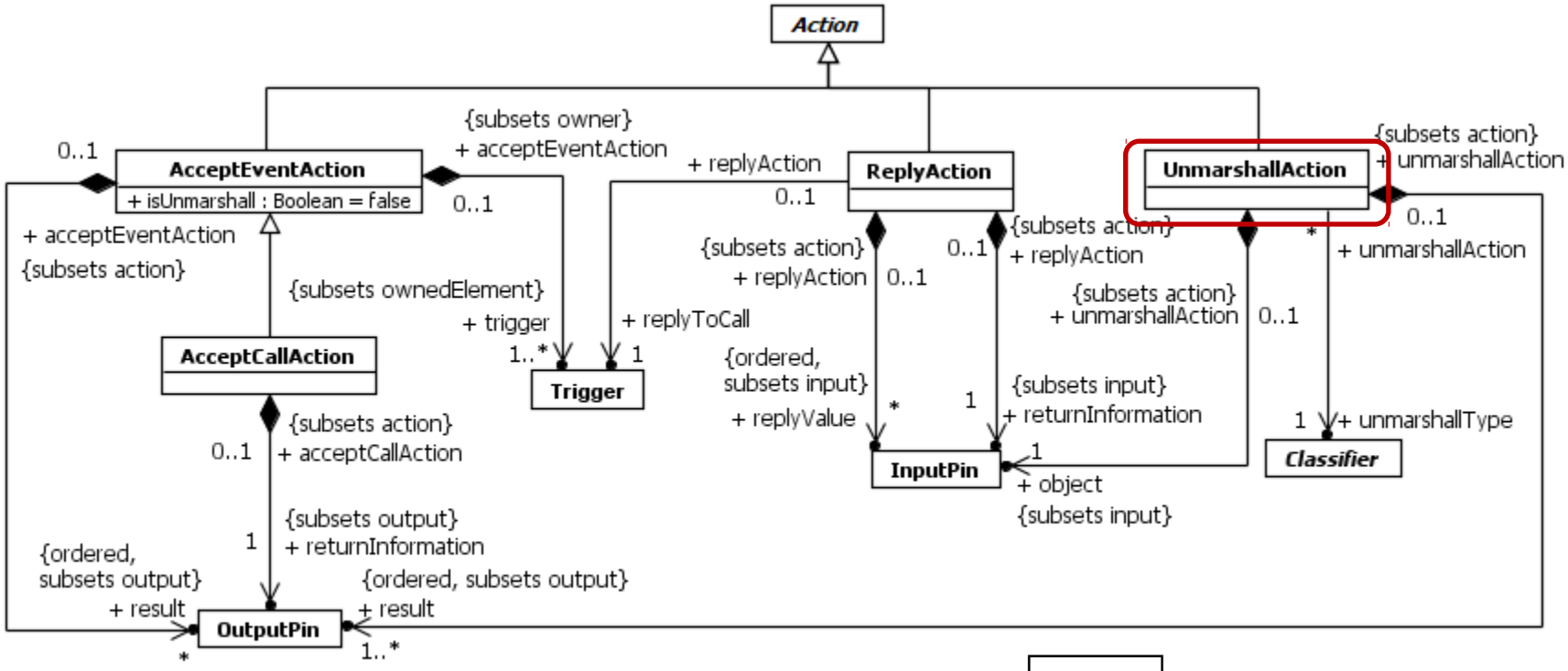
Accept event action

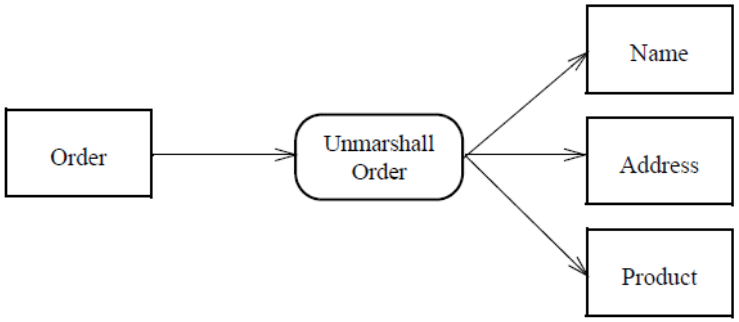
Accept time event action



UnmarshallAction

- Open an object as a set of properties





Université Nice Sophia Antipolis

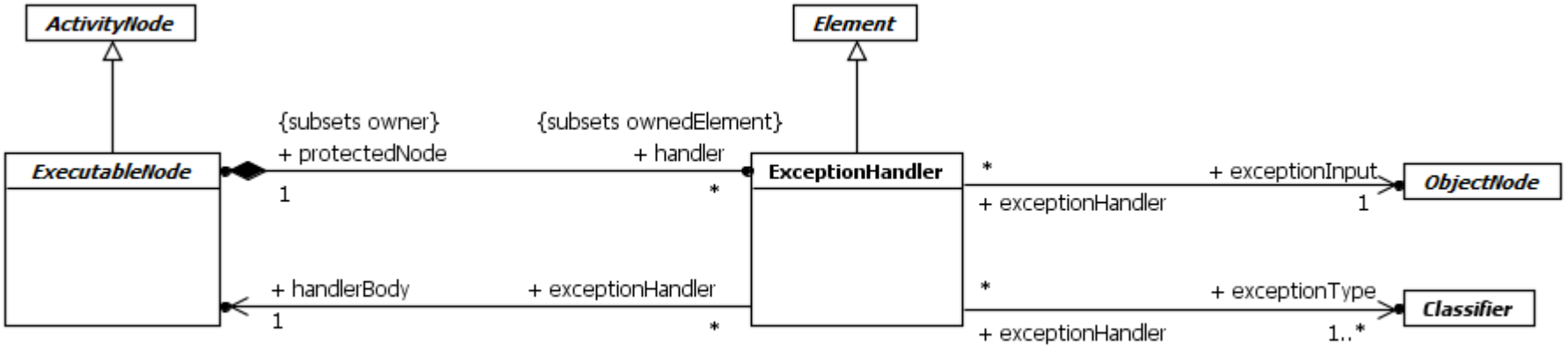
Exercice

- PRÉPARATION CRÈME ANGLAISE ET SES ÎLES FLOTTANTES

1. Rassembler les ingrédients.
2. Faire bouillir le lait avec le sucre vanillé dans une casserole. Fouetter vivement les jaunes d'oeuf et le sucre fin jusqu'a ce que le mélange blanchisse et devienne assez épais.
3. Verser tout doucement et progressivement le lait sur le mélange sucre-oeuf tout en remuant. Reverser le tout dans la casserole et faire épaissir la crème **7 minutes** à feu doux en tournant constamment avec une cuillère en bois.
4. Retirer la crème immédiatement du feu pour la mettre à refroidir dans un saladier. Réserver au réfrigérateur pendant **3 heures** au minimum.
5. Préparer les blancs en neige bien ferme tout en incorporant le sucre fin en 2 fois.
6. Dans une grande casserole faire frémir 1 l d'eau. Poser délicatement une grosse cuillère à soupe de blancs en neige sucrés. Les faire cuire **20 secondes** de chaque côté. Les retirer immédiatement sur du papier absorbant. Laisser refroidir.
7. **Au moment du repas** remplir des coupes à dessert de crème anglaise, déposer 1 ou 2 blancs en neige, mettre un léger coulis de caramel.

Exceptions and ExceptionHandlers

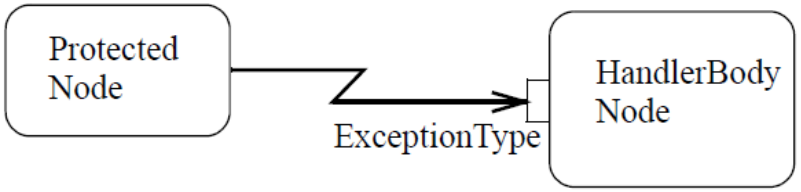
- Similar to programming mechanisms
  - Protected a node with one or several exception handlers
  - Exception Handler
    - One exception input
    - One or more exception types
    - One handler body (with no incoming/outgoing edges)



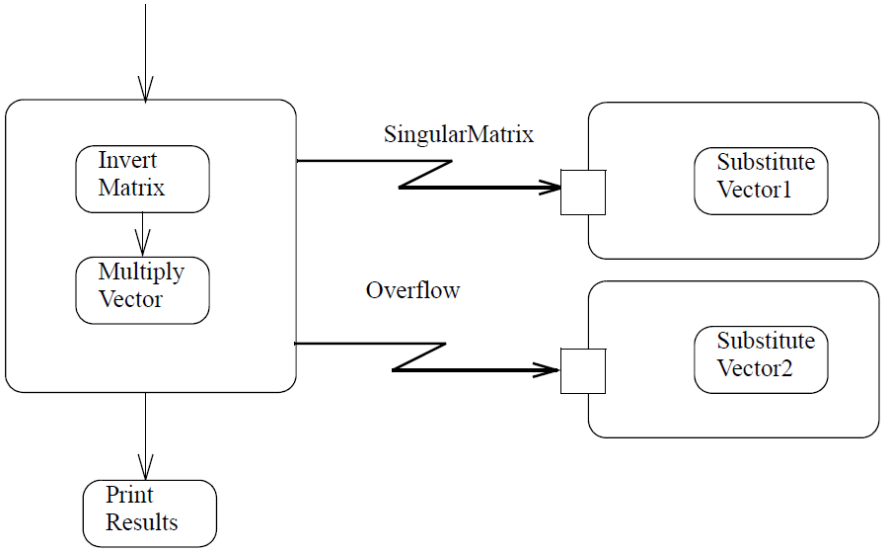
Université Nice Sophia Antipolis

Exceptions

- Notations



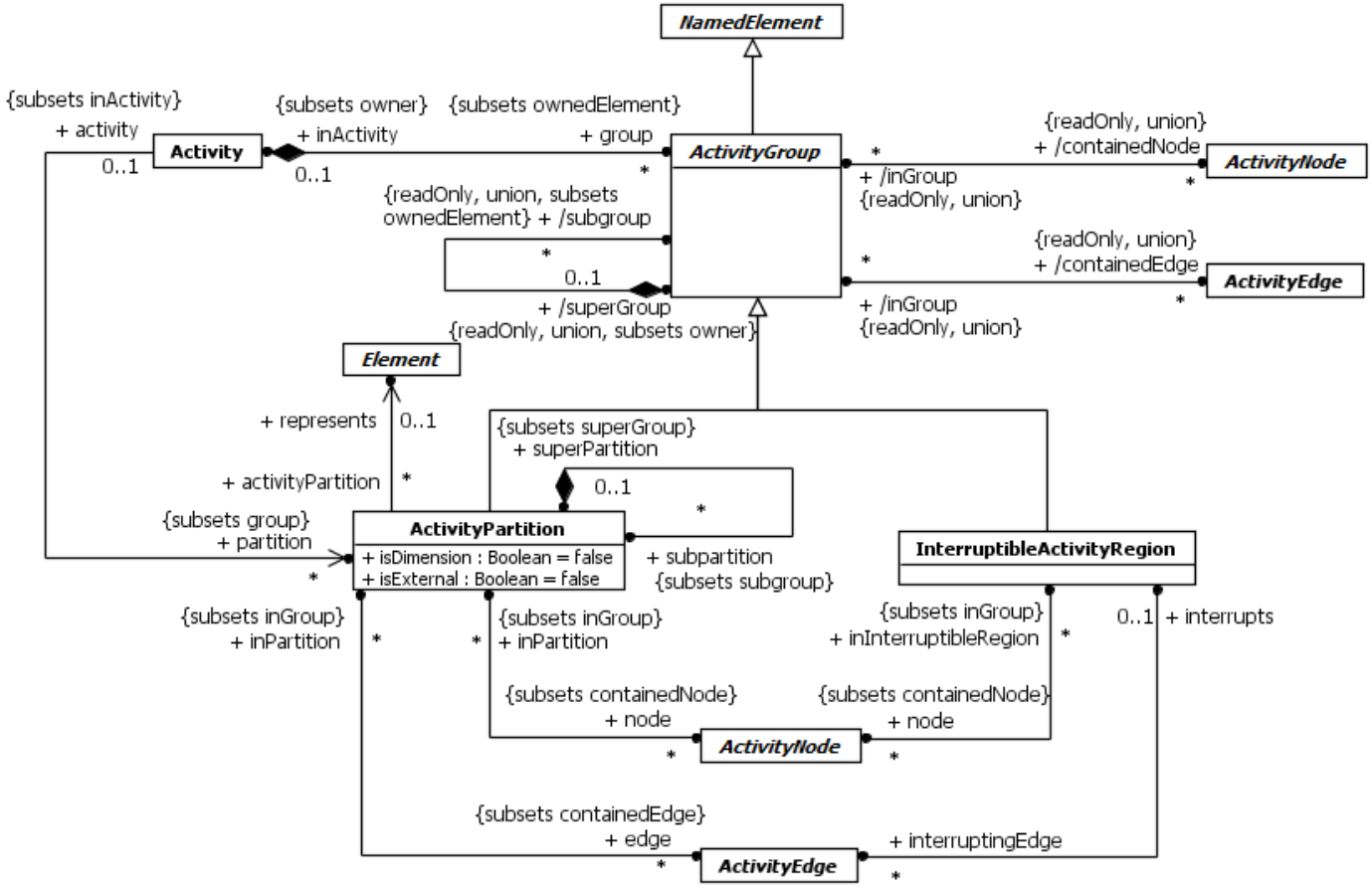
- Example



Université Nice Sophia Antipolis

InterruptibleRegion

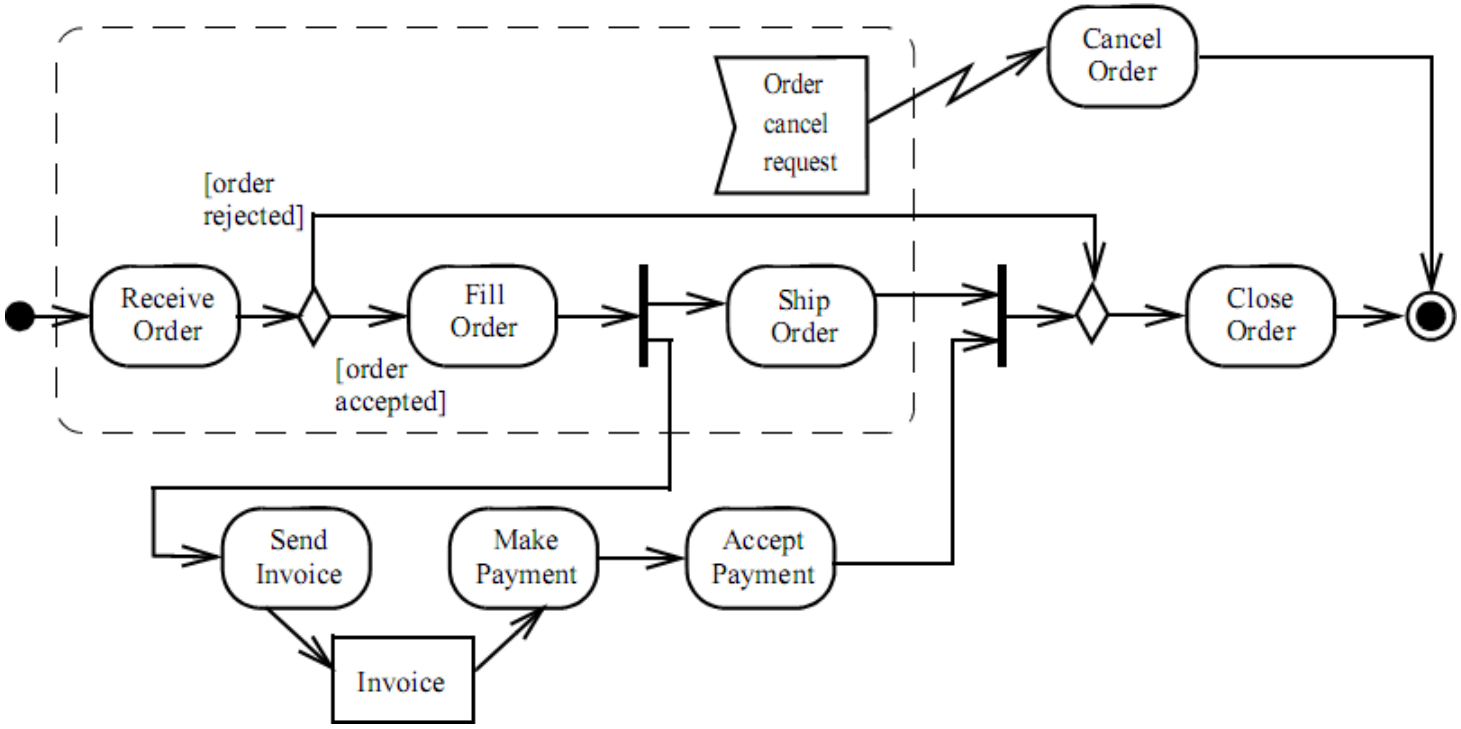
- Metamodel



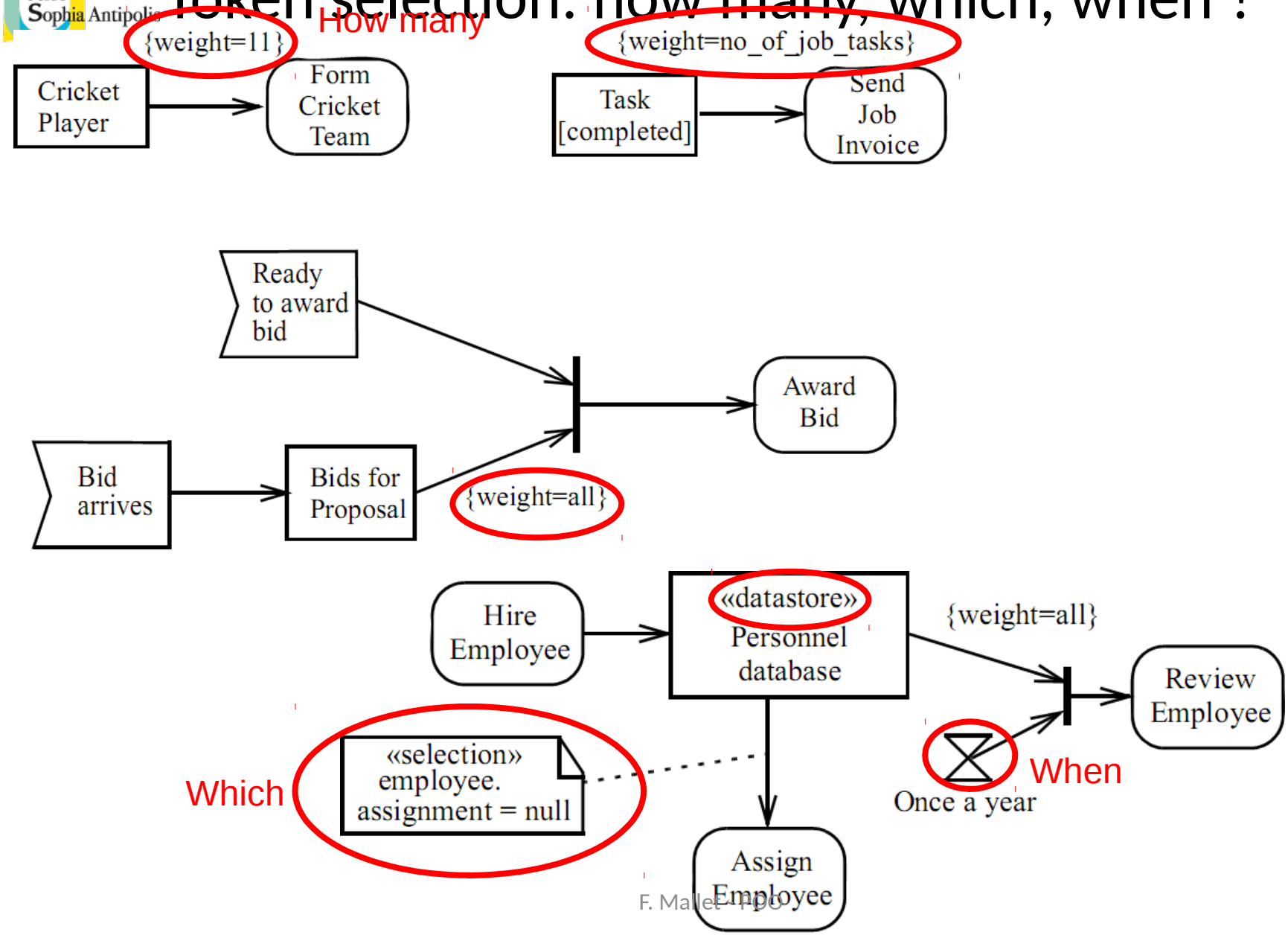
Université Nice Sophia Antipolis

InterruptibleRegion

- Semantics
  - Group of nodes interrupted when one token reaches a designated interruptingEdge
  - All the flows are aborted

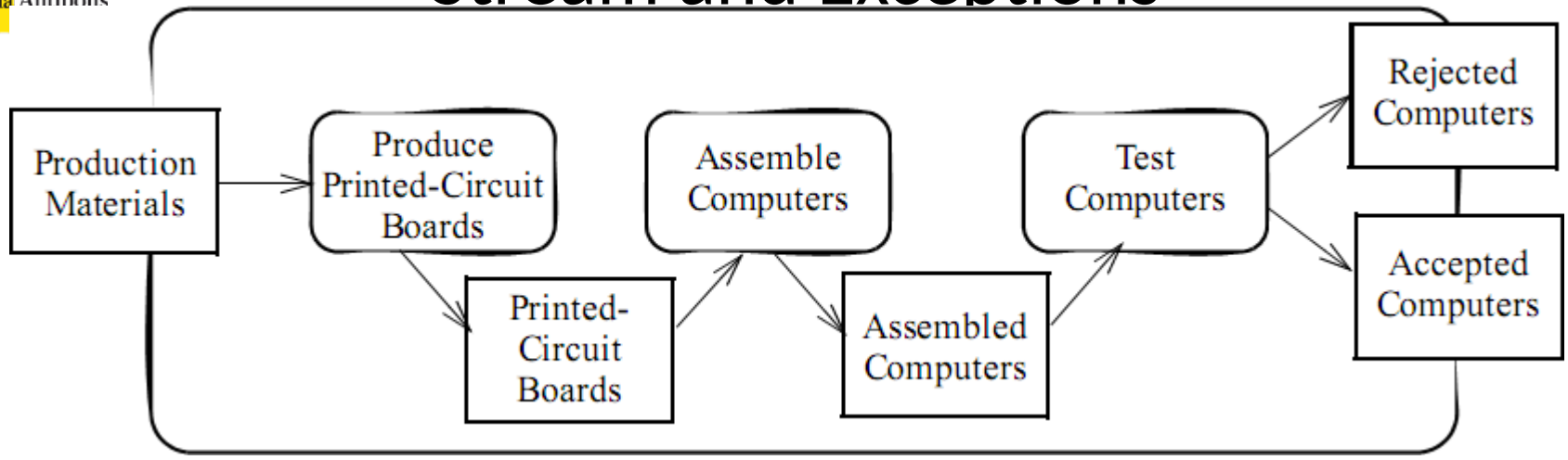


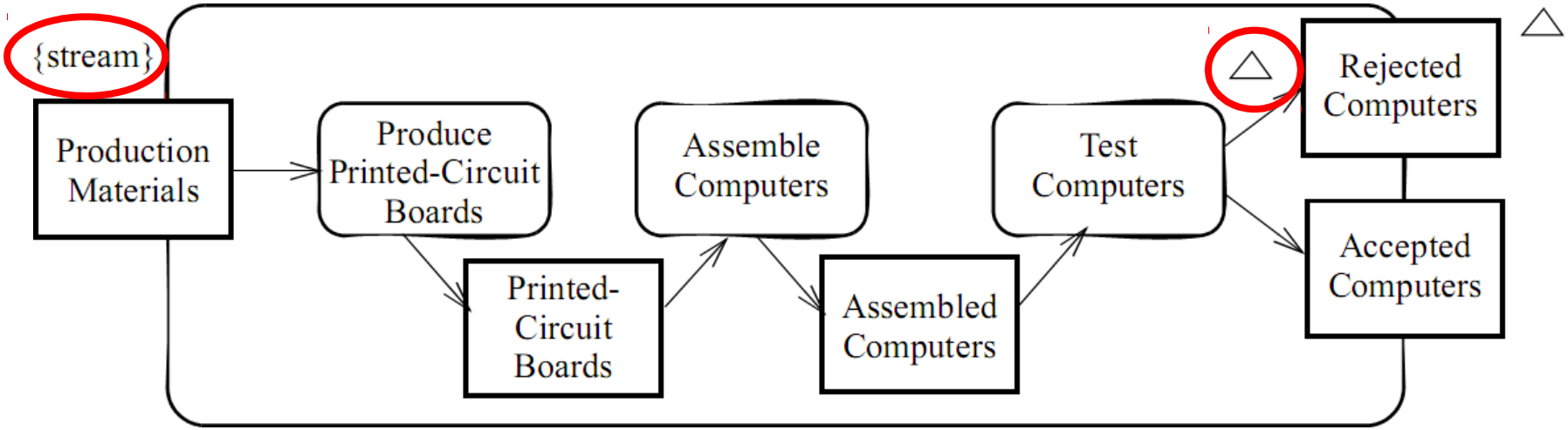
Token selection: how many, which, when ?



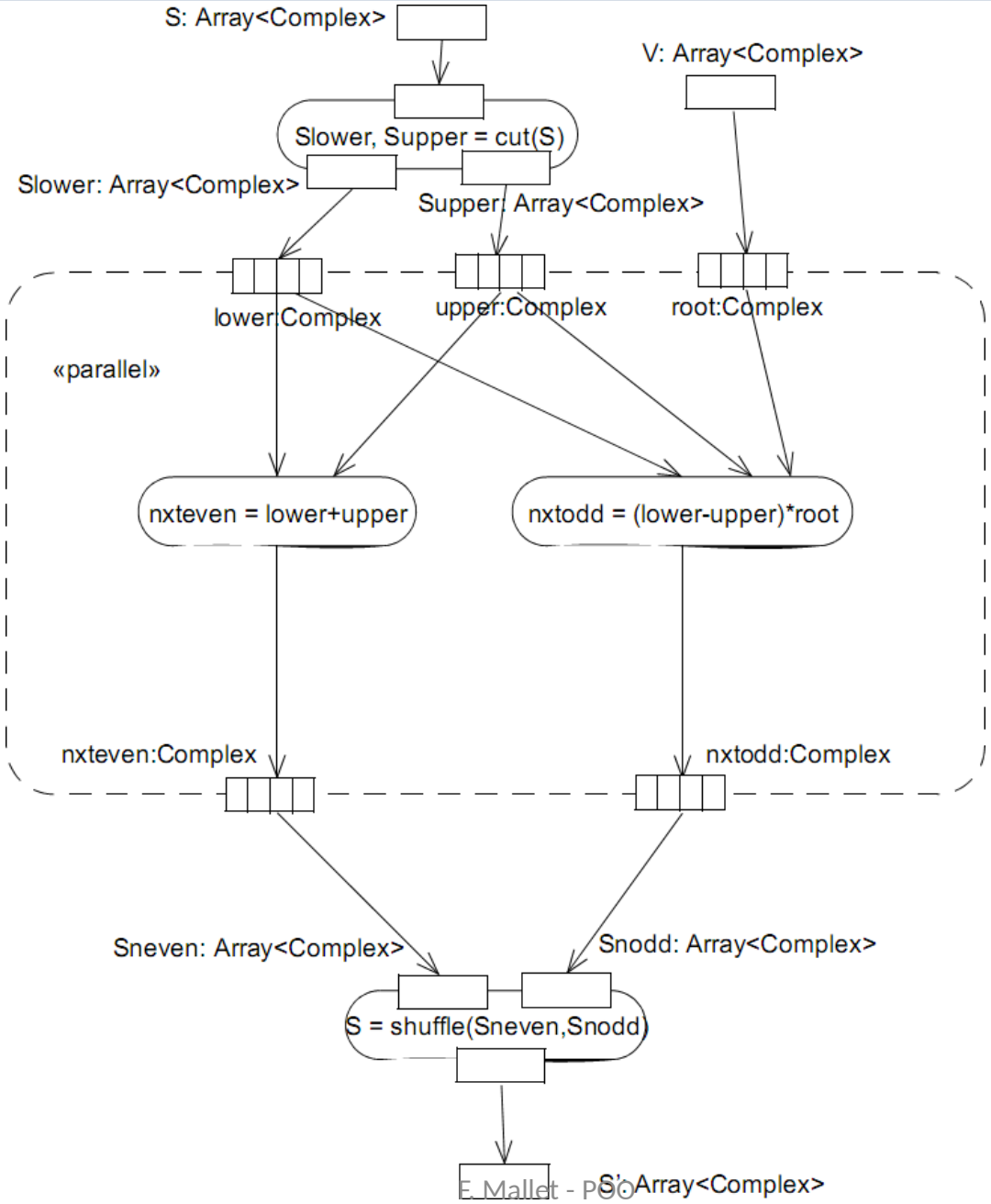
Université Nice Sophia Antipolis

Stream and Exceptions





Activités


S: Array<Complex>
V: Array<Complex>
Slower, Supper = cut(S)
Slower: Array<Complex>
Supper: Array<Complex>
lower:Complex
upper:Complex
root:Complex
«parallel»
nxteven = lower+upper
nxtodd = (lower-upper)*root
nxteven:Complex
nxtodd:Complex
Sneven: Array<Complex>
Snodd: Array<Complex>
S = shuffle(Sneven,Snodd)
S': Array<Complex>

Université Nice Sophia Antipolis

UML semantics of Activities

- The semantics of activities is based on token flow.
  - Edges represent dependencies between nodes;
  - A 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;
  - A node may begin execution when specified conditions on its input tokens are satisfied; the conditions depend on the kind of node;
  - When a node begins execution, tokens are accepted from some or all of its input edges and a token is placed on the node;
  - When a node completes execution, a token is removed from the node and tokens are offered to some or all of its output edges.