UML 2 Activity Modeling for Domain Experts

Conrad Bock
NIST
conrad.bock@nist.gov

Overview

- UML for knowledge capture.
- Input to UML 2 Activities.
- UML 2 Activity elements.
- Systems engineering extensions.

UML For Domain Experts

- UML began as a language for domain experts to record their knowledge.
- Experts in electric motors design expressed concepts in diagrams, …
- … which automatically generated database table definitions.
- 3000% productivity improvement over informal textual descriptions.

- “Model Driven Design,” Cocks, D., Dickerson, M., Oliver, D., Skipper, J., INCOSE INSIGHT, 7:2, July 2004.

UML For CIM/Analysis

- **Computation-Independent Modeling = Analysis …**
- **… as in “Analysis and Design Task Force”.**
- **Early stages of software development capture end-user concepts (“analysis”) ...**
- **… before later stages choose how these are represented in software (“design”).**
- **Many diagrams shared between analysis and design (class, composition, behavior).**

UML For System Engineering

- SE specifications are agnostic about how they are implemented, in organizations, hardware, or software.
- Capture domain expert requirements, rather than how they will be satisfied.
- Executable models for over 15 years.
- UML is increasingly the major modeling language used in SE and military architecture communities.
- UML Profile for SE submitted.

- "Systems Modeling Language (SysML) Specification," SysML Submission Team, http://doc.omg.org/ad/05-11-01, November 2005.
- "Systems Engineering in the Product Lifecycle," Bock, C., International Journal of Product Development, 2:1-2, http://www.nist.gov/msidlibrary/doc/sysmlplm.pdf, 2005.
- "UML Profile for DoDAF/MODAF (UPDM)," OMG, http://doc.omg.org/dtc/05-09-12, September 2005.

UML For Ontology

- **Ontology languages becoming popular for expressing domain expert concepts.**
- **Enable automated consistency checking.**
- **UML has significant overlap with OL's …**
- **… eg, classes, properties, subclasses, subproperties, disjointness, and others.**
- **Many aspects of OL's are specialized kinds of constraints.**
- **UML Profile for RDFS and OWL submitted.**

- **"Ontology Definition Metamodel," IBM, Sandpiper, http://doc.omg.org/ad/2005-09-08, November 2005.**

UML for Business Modeling

- Foundation concepts for BM
  - Things (“classes”, “objects”, “entities”)
    - For documents, people, resources, etc.
  - Structured/assembled things
    - For organizations, structured entities.
  - Dynamics
    - For processes, collaboration, event monitoring.
- Continuity with other UML-based knowledge capture, and with IT implementation.

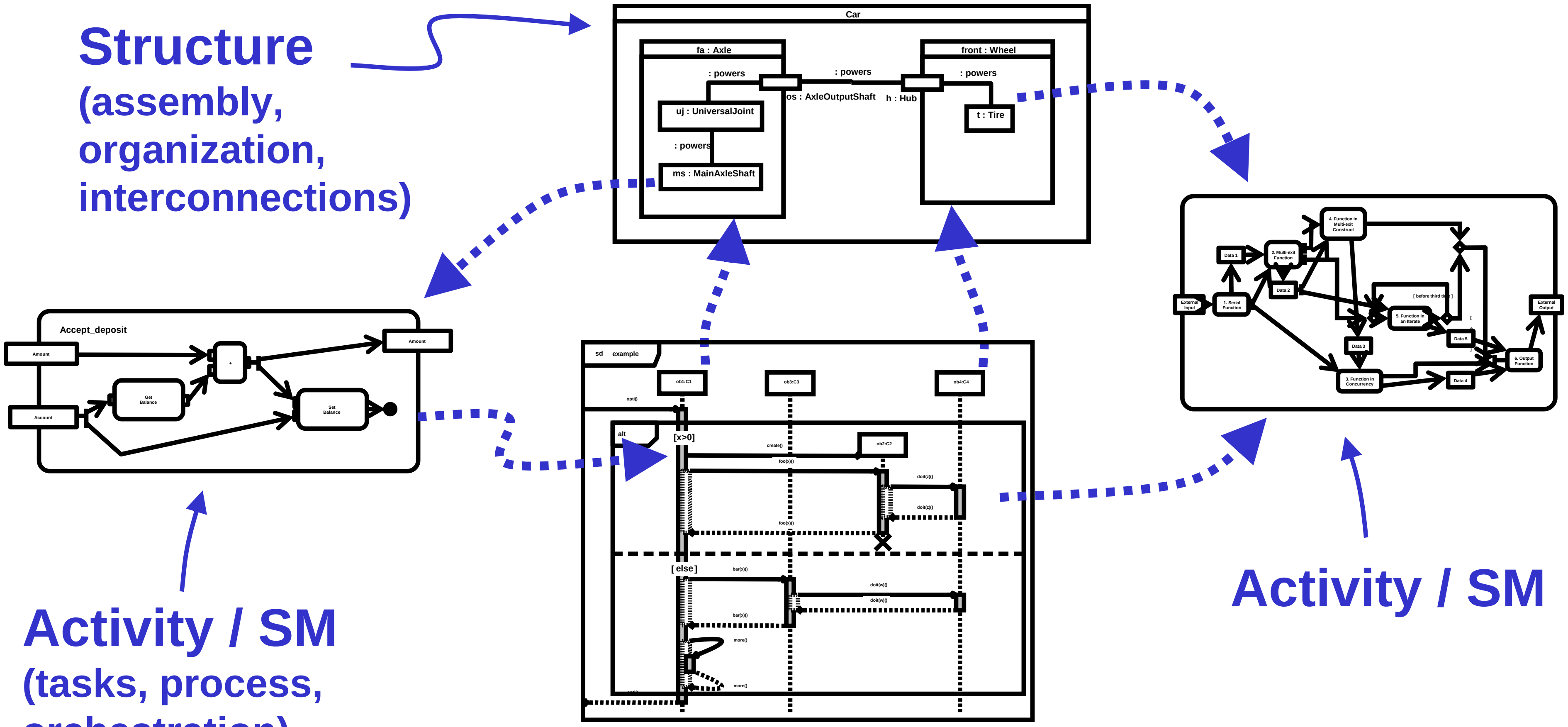
- Business Modeling with UML : Business Patterns at Work, Eriksson, H., Penkerbook, M., Wiley, January 2000.

UML for Process Knowledge

- UML includes three ways to express knowledge about dynamics …
- … each addressing different aspects:
  - Output to input dependencies (Activities)
  - Reaction to events (State Machines)
  - Message-passing (Interactions)
- … but also overlapping:
  - Sequencing, conditionals, concurrency.
- Virtual machines defined for execution.

Integrated Models

Structure
(assembly, organization, interconnections)



Activity / SM
(tasks, process, orchestration)

Activity / SM

Interaction
(messages, choreography, collaboration)

Integration with UML 1.5 Action / Procedure Model

- "Action Semantics".
- Activities fully executable.
- Covers the full range of flow models from flow charts to code.
- More about this later.

- OMG Unified Modeling Language Specification, OMG, Version 1.5, http://doc.omg.org/formal/03-03-01, September 2002.

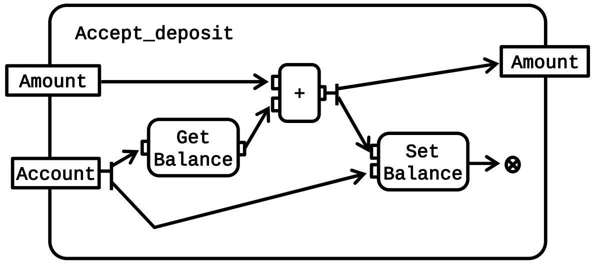
More Than Pictures

- Repository provides
  - API’s
  - XML interchange
  - Support for multiple notations

- UML notation stores to repository
  … and alternate notations can, too.

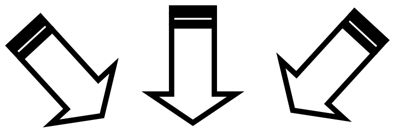
- Generate systems from repository:
  Notation → Repository → System

Repository-Centered

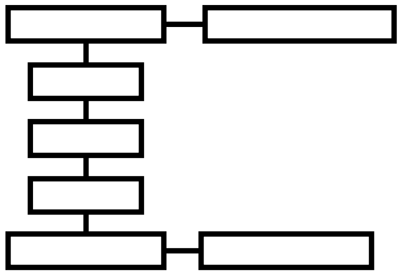


```
Amount function Accept_deposit
          (a : Account, d : Amount)
{
  Amount nb = a.balance + d;
  a.balance = nb;
  return nb;
}
```

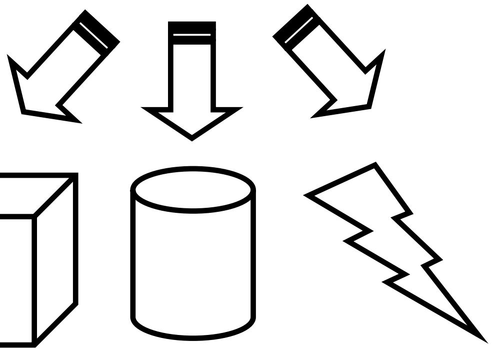
Presentation



Parsing



Repository



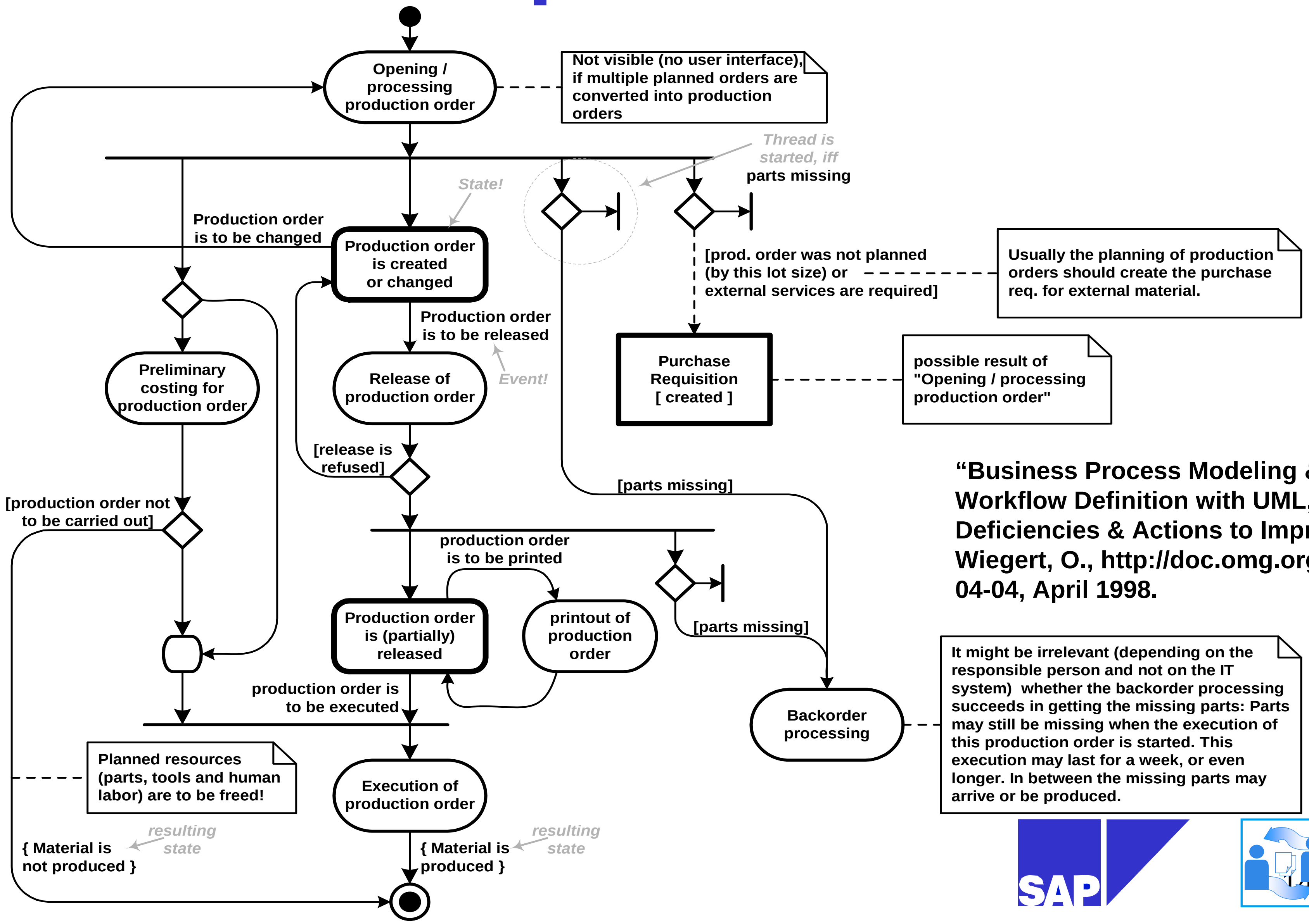
Model compilation

Actual system

Input to UML 2 Activities

- First workflow RFP discussion (HP, FileNet, NIST)
- SAP, Oracle, IBM
- EDOC, BPML, WPDL, BPEL (WSFL, XLANG), ebXML
- CaseWise, Odell and Associates, IntelliCorp.
- And others.

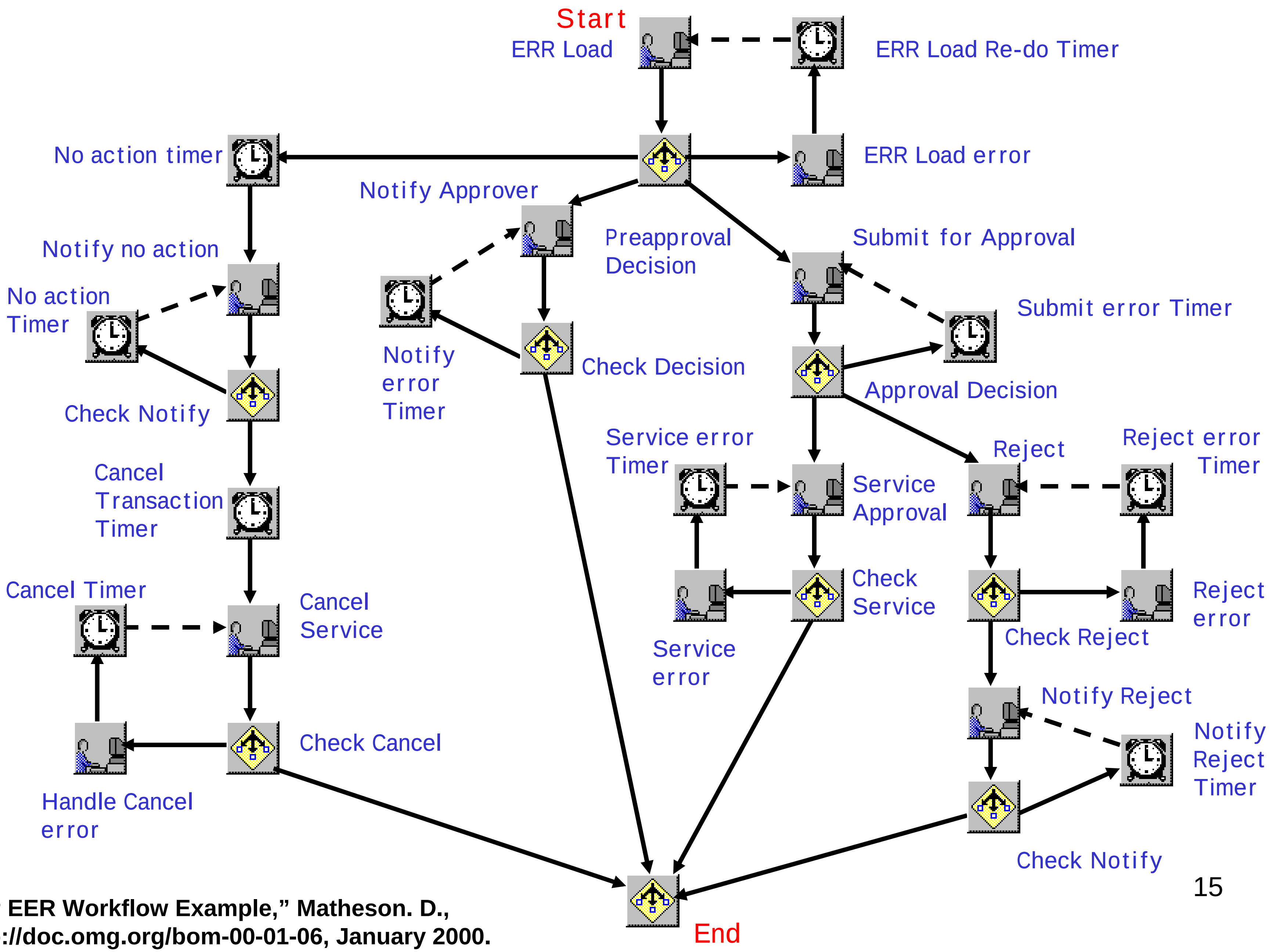
Example from SAP



Opening / processing production order
Not visible (no user interface), if multiple planned orders are converted into production orders
Thread is started, iff parts missing
State!
Production order is to be changed
Production order is created or changed
[prod. order was not planned (by this lot size) or external services are required]
Usually the planning of production orders should create the purchase req. for external material.
Production order is to be released
Event!
Preliminary costing for production order
Release of production order
Purchase Requisition [created]
possible result of "Opening / processing production order"
[release is refused]
[parts missing]
[production order not to be carried out]
production order is to be printed
Production order is (partially) released
printout of production order
[parts missing]
production order is to be executed
Backorder processing
It might be irrelevant (depending on the responsible person and not on the IT system) whether the backorder processing succeeds in getting the missing parts: Parts may still be missing when the execution of this production order is started. This execution may last for a week, or even longer. In between the missing parts may arrive or be produced.
Planned resources (parts, tools and human labor) are to be freed!
Execution of production order
{ Material is not produced }
resulting state
{ Material is produced }
resulting state

"Business Process Modeling & Workflow Definition with UML, Deficiencies & Actions to Improve," Wiegert, O., http://doc.omg.org/ad/98-04-04, April 1998.

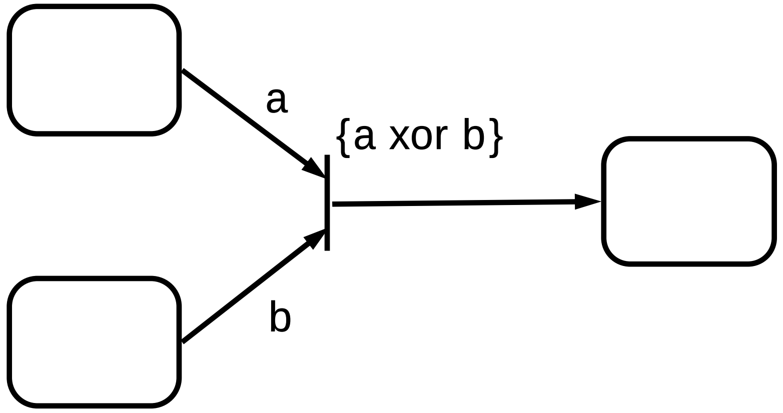
Example from HP

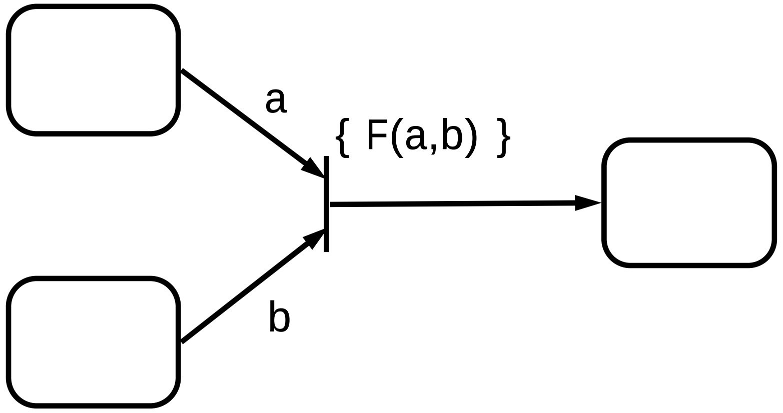


"HP EER Workflow Example," Matheson. D., http://doc.omg.org/bom-00-01-06, January 2000.

Feature	U2P AM	CaseWise	BPML	EDOC	WPDL-XML	WSFL	XLANG	ebXML/ebTWG
General								
Graphical notation	Y	Y	Not yet	Nonnormative	N	N (only for explanation in spec)	N	Y (UML 1.x activity graphs)
Metamodel	Y	N (not exportable anyway)	Not yet	Y	Y (not in UML)	N (but spec is written so that it could)	N	Y(Stereotypes of UML 1.x)
XML - any	Y	N	Y	Y	Y	Y	Y	Y (XMI, presumably)
Human usable textual notation.	N	N	Y (XML)	Not yet	Y (tags not too complex)	Y (tags not too complex)	Y (XML)	N
General Process Features								
Message or control/data flow model?	Control/Data	Control/Data	Message	Data	Control (with parameters)	Control/Data (message data only)	Message	Control/Data (UML 1.x activities)
Data and control on same diagram/model	Y (shown by usage in notation)	N	NA	NA	NA	NA (shown by solid/dotted in spec)	NA	Y
Business-specific features	N	Y (eg, location)	N	N	Y (eg, responsible party, manual activity, cost, etc)	N (but relates to WSDL)	N (but extends WSDL)	Y (stereotypes)
Activity - actor link	N (but has hook)	Y	Y	Y	Y	Y	Y	Y
Activity - artifact link	N (but has hook)	N	N	Y	Y	N	N	Y
Simplified subsets of functionality defined	Y (well-nested, flowchart)	N	N	N	Y (well-nested, acyclic)	N	N	N
Simulation-specific information	N	Y (a little for branching)	N	N	Y (timing attributes)	N	N	N
Transaction model	N	N	Y	N (very little)	N (activities are atomic)	N	Y	Y
Detailed Process Features								
Pin model	Y	N	N	Y	N	N (uses a single message input/output)	N	N
Objectflow "in the middle" model	Y	Y	N	N	N	N	N	Y (UML 1.x semantics)
Explicit control constructs	Y	N (guards for conditionals)	Y (uses message consumption for conditionals)	N (uses alternate output sets)	Y/N (part of invocation, guards)	N	Y (switch only, uses "Qname" for conditionals)	Y (but often uses guards on transitions)
"else" functionality for conditionals	Y	N	N	NA	Y	N	N	Y
Explicit merge construct	Y	N	N?	N (could use alternate input sets)	Y? (XOR enforced?)	N	N	Y
Optional inputs	N	N	N	Y (alternate input sets)	N	N	N	N
Optional outputs	Y (part of asych outputs)	Y (in control only)	N	Y (alternate output sets)	N	N	N	N
Alternative input/output sets	N	Y (in control only)	N	Y	N	N	N	N
Asynchronous inputs/outputs	Y	N	N	Y	N	N	N	N
Fork/join functionality	Y	Y	Y (flexible join using named spawn)	Y	Y (part of invocation, guards)	Y (part of invocation, guards)	Y (well-nested only)	Y (but has semantic problems)

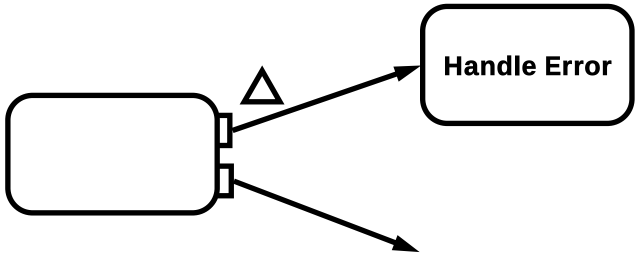
Exclusive/complex join

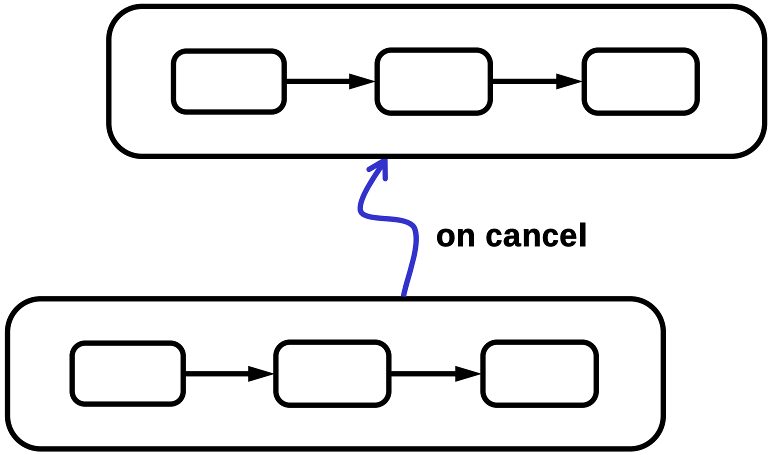




Feature	U2P AM	CaseWise	BPML	EDOC	WPDL-XML	WSFL	XLANG	ebXML/ebTWG
Exclusive join functionality	Y, (late flows ignored)	N	N	N	Y? (spec is ambiguous)	Y (late flows ignored)	N	N
Complex joins	Y	N	N	N	N	Y	N	N

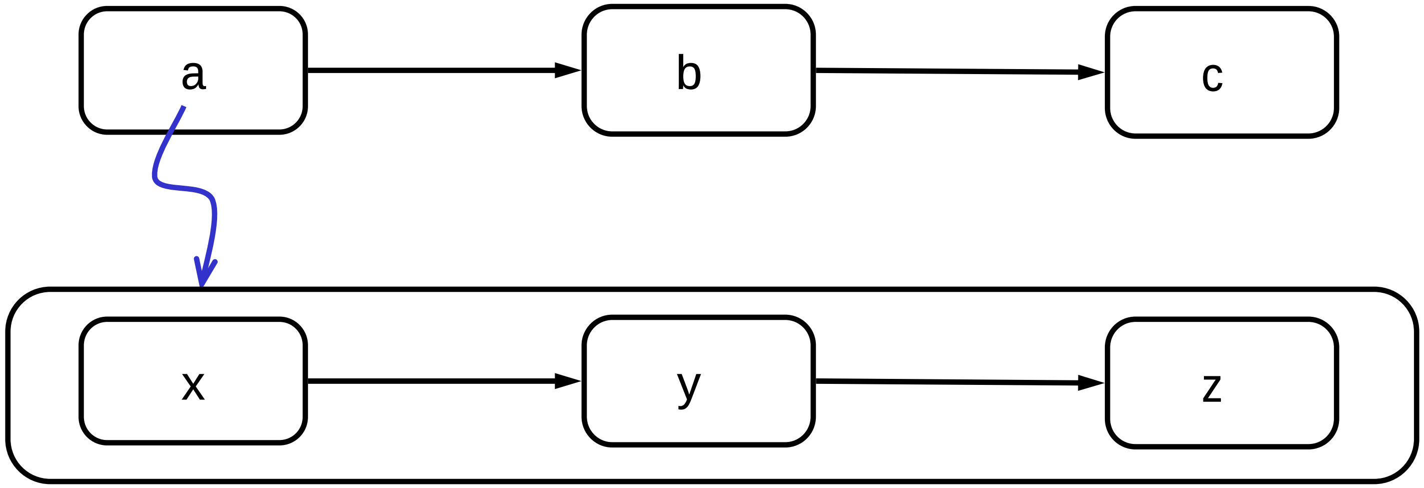
Error handling





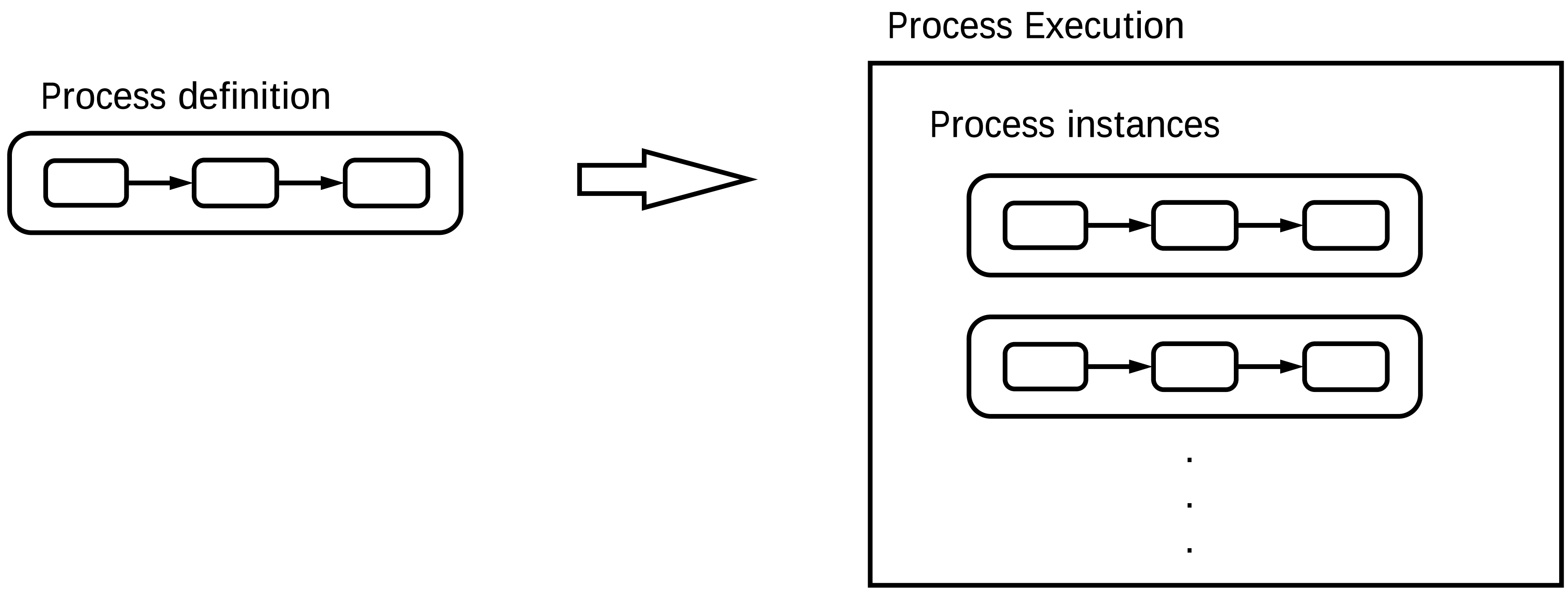
U2P AM	CaseWise	BPML	EDOC	WPDL-XML	WSFL	XLANG	ebXML/ebTWG
Y (exception outputs)	N	Y (compensation for completed activity called by an aborted activity)	Y (exception outputs)	N	Y (as part of data)	Y (compensation for completed activity called by an aborted activity)	Y

Asynchronous invocation



U2P AM	CaseWise	BPML	EDOC	WPDL-XML	WSFL	XLANG	ebXML/ebTWG
Y (for operations, signals, not subactivities)	N	Y (implements synch as two asych)	N	Y (for subactivities only)	N	Y (WSDL operations)	Y

Process instances



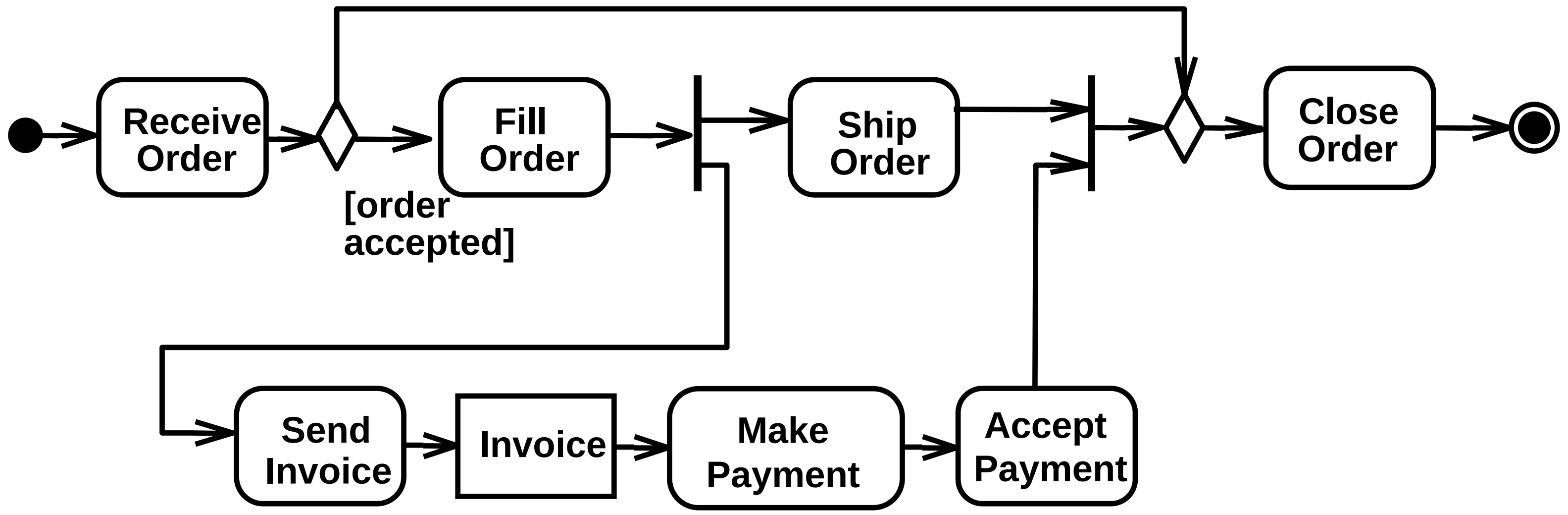
U2P AM	CaseWise	BPML	EDOC	WPDL-XML	WSFL	XLANG	ebXML/ebTWG
Y (structual features only, but extensible)	N	N	Y (scenario only)	N (but may through interop standard)	N	Y (identifier only, with mapping to messages)	N

Activity Modeling

- **Activity modeling emphasizes the output/input 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/what performs each task.**

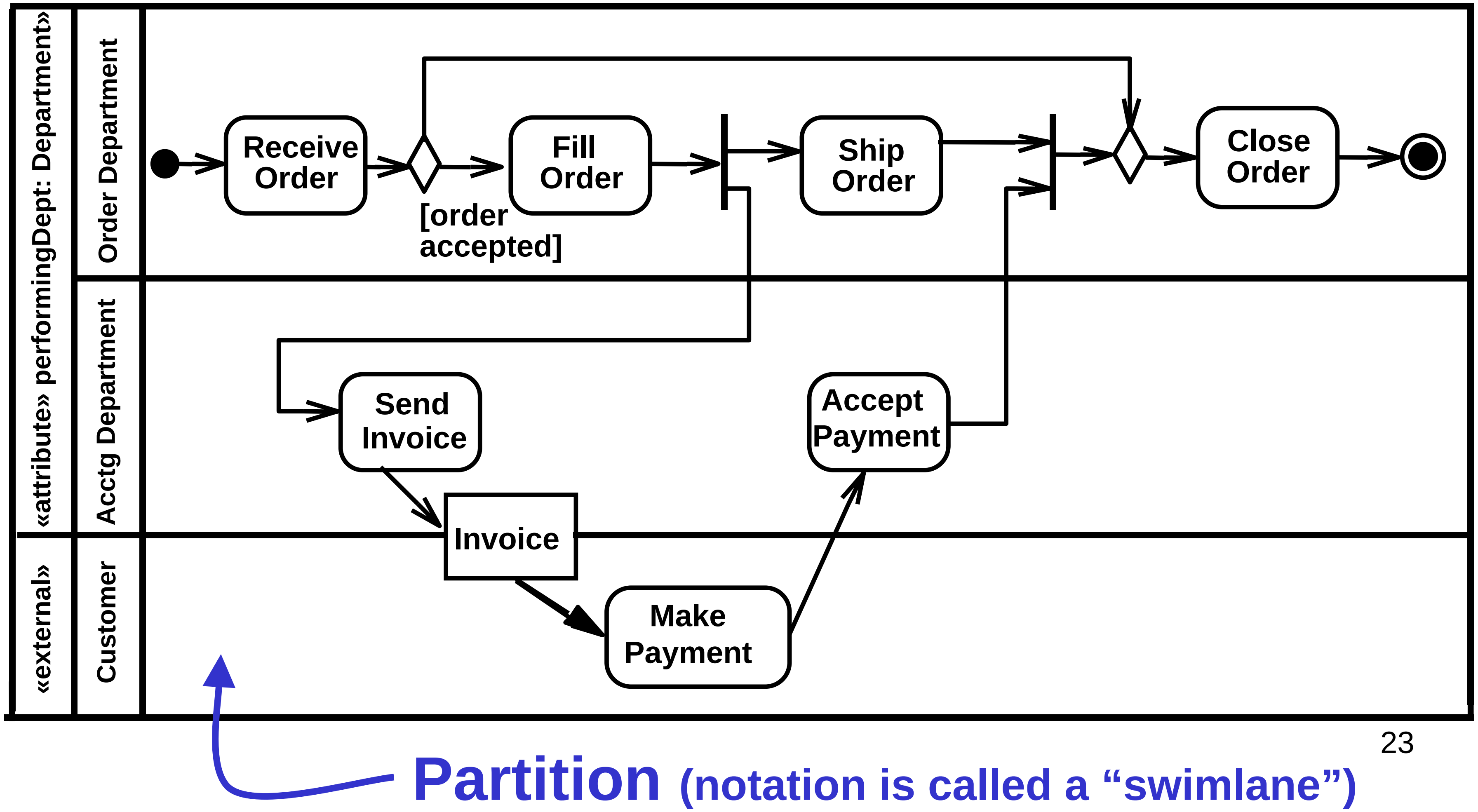
Activity Modeling

- **Tasks and ordering …**

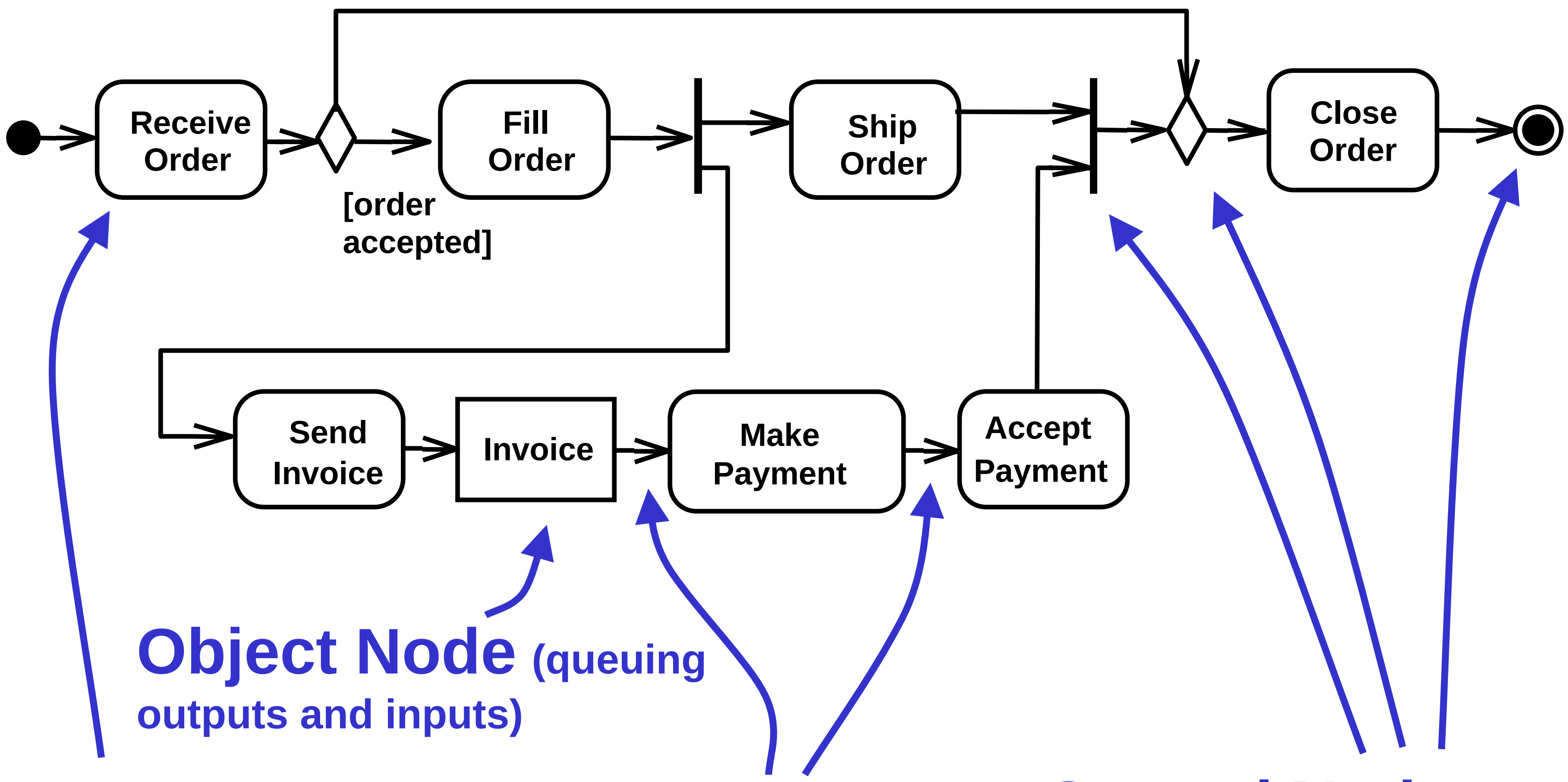


Activity Modeling

- … plus resource assignments.



Activity Elements



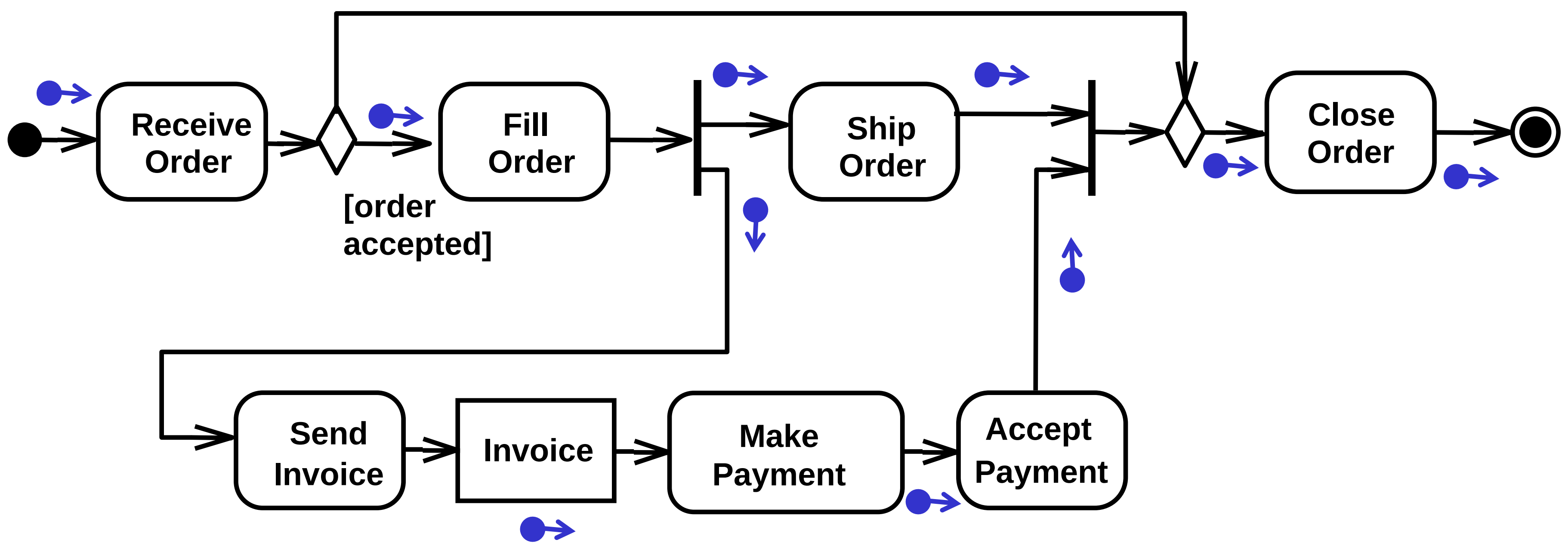
Object Node (queuing outputs and inputs)

Action (uses of other activities / tasks)

Edge/Flow (execution dependencies)

Control Node (routing control and objects)

“Flow” semantics

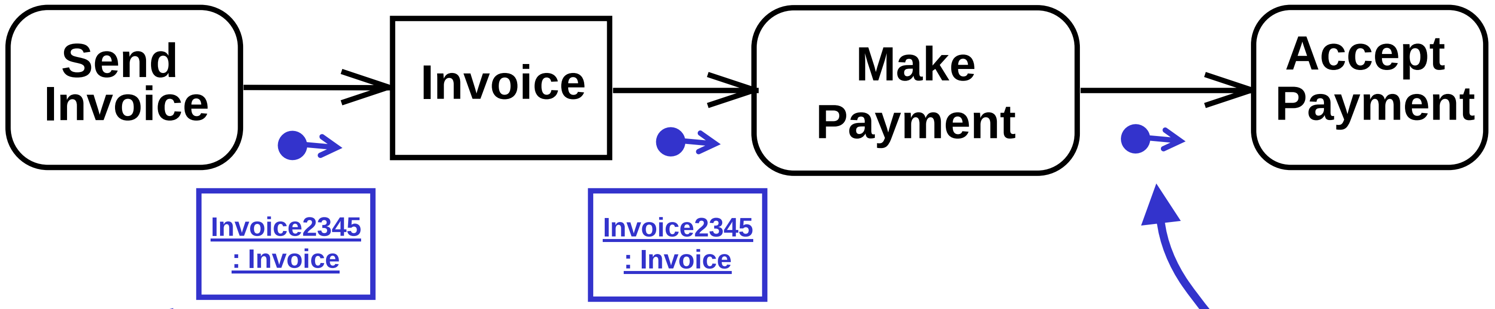


(●→ *not UML notation*)

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

Actions and Object Nodes

- **Accept inputs, start behaviors, provide outputs.**



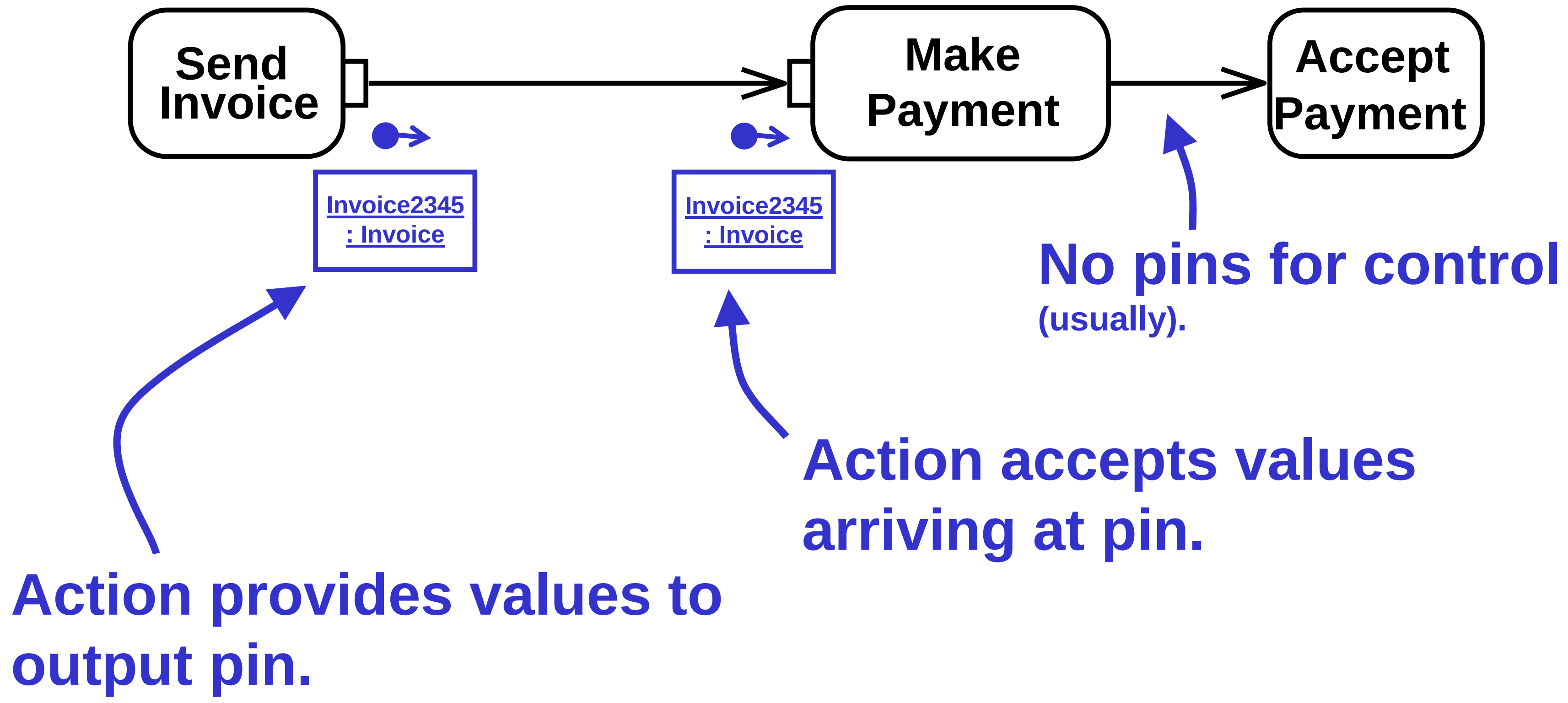
Action starts when input arrives.

Sequencing: control "flows" when action is complete.

Output provided when action is complete.

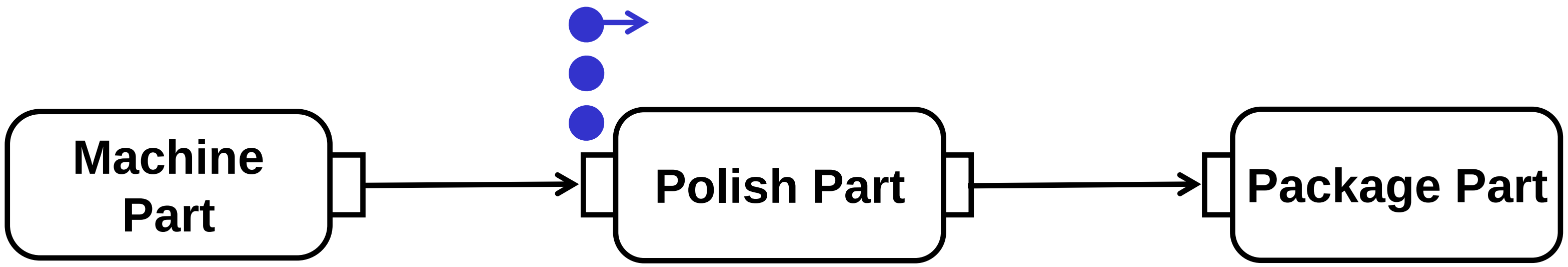
Actions and Object Nodes

- Alternate object node notation (pin).



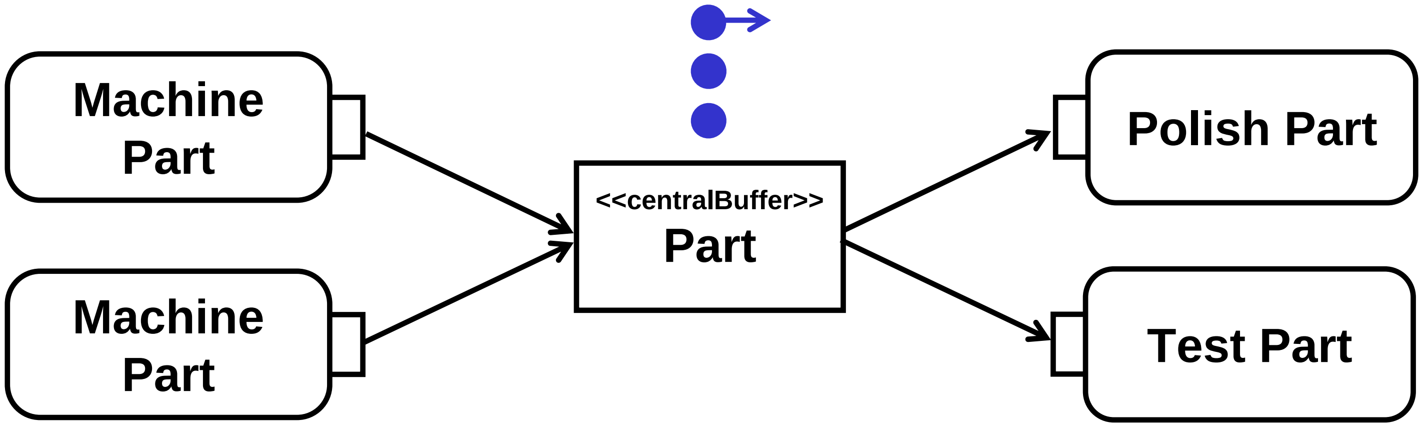
Must use this notation if the output type is different than the input type. The underlying repository stores pins.

Queuing



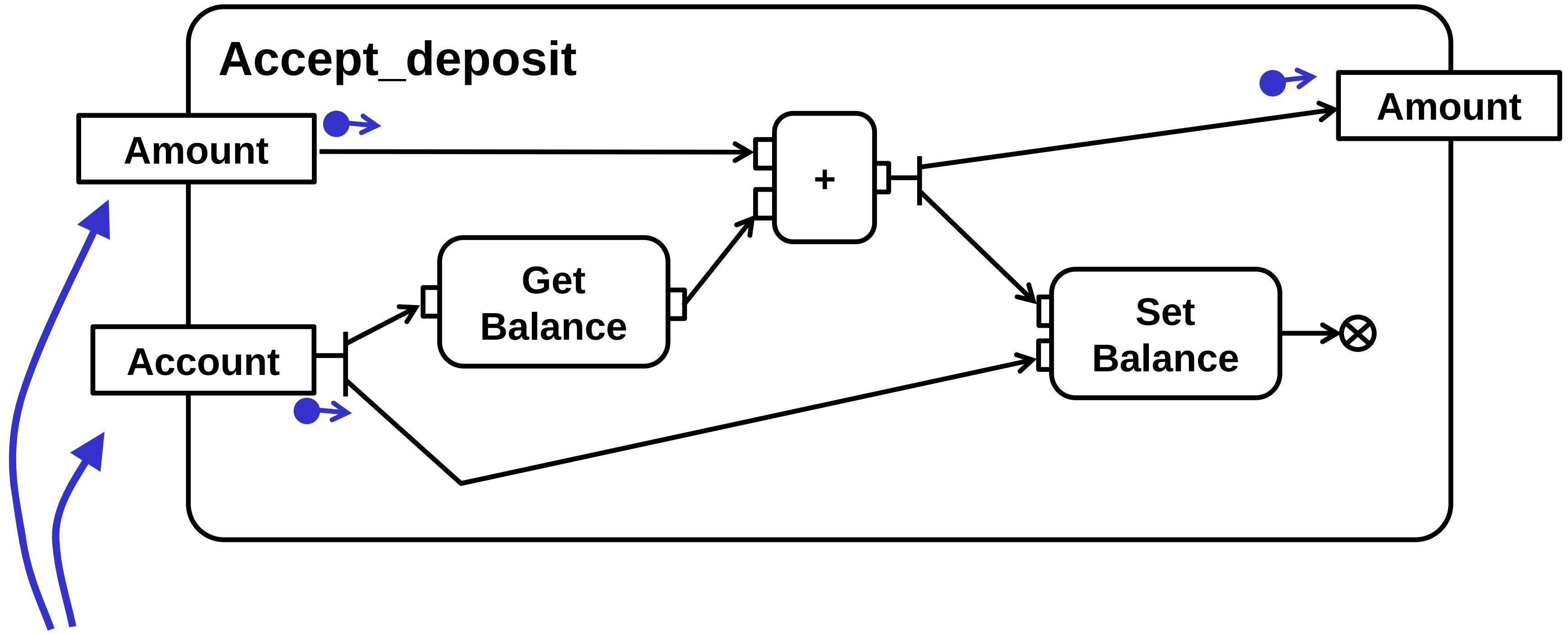
- Tokens can
  - stack up in “in/out” boxes
  - backup in network
  - prevent upstream behaviors from taking new inputs
- Applicable to systems with significant resource constraints, such as physical or manual processes.

Queuing



- Tokens can be
  - Stored temporarily
  - Divided between flows
- Tokens cannot
  - Flow in more than one direction, unless copied.

Activity Parameter Nodes

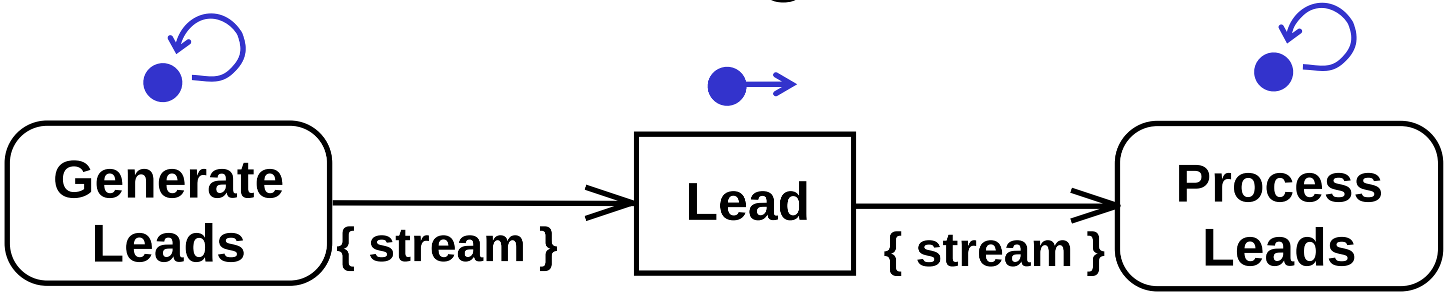


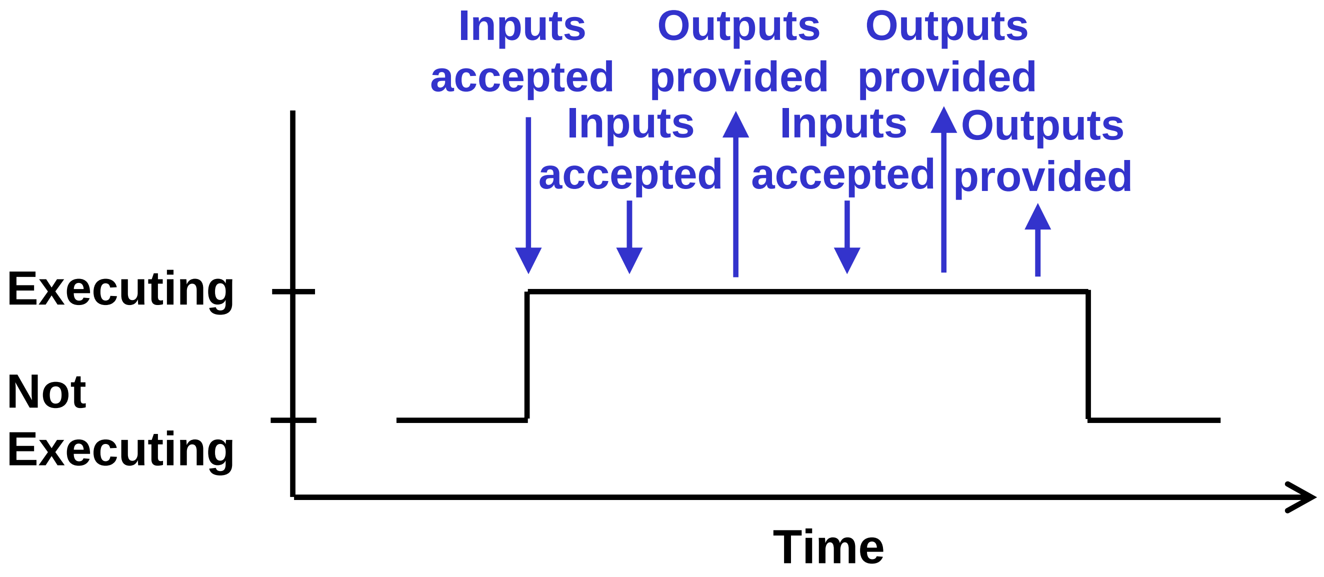
Activity Parameter Node
(uses of objects/data, a kind of object node)

- **Parameter nodes accept and provide values to/from whatever behavior uses this activity.**

Streaming Parameters

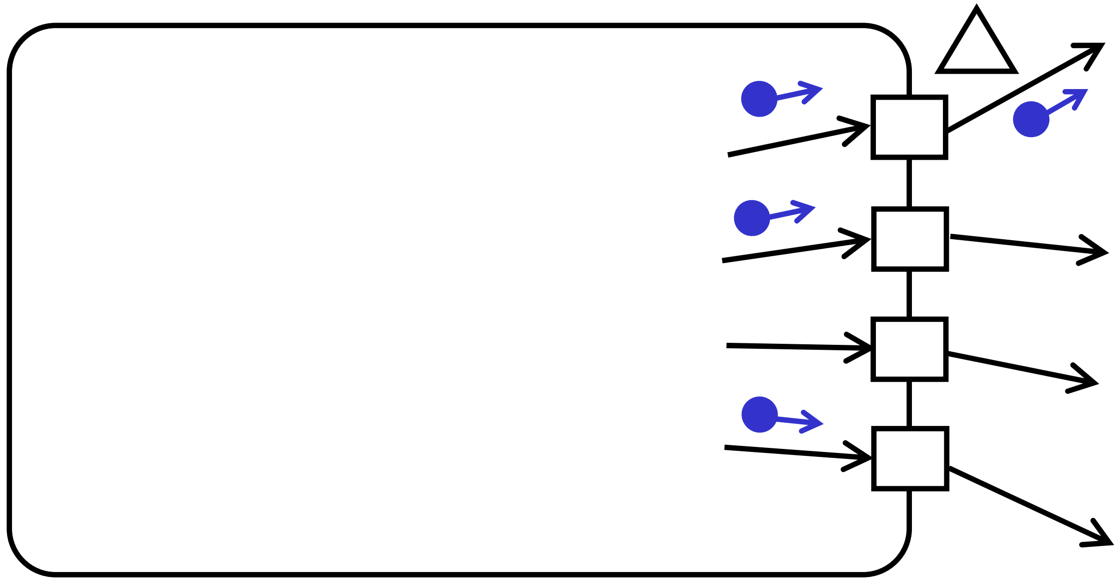
- Values accepted and provided while action is executing.





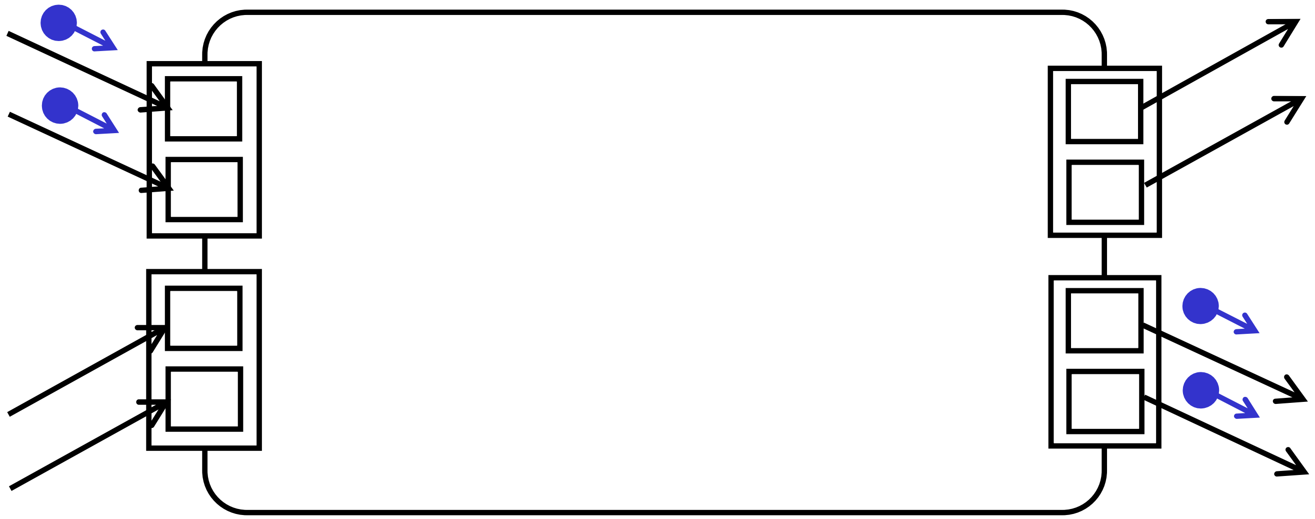
Exception Parameters

- Outputs that are exclusive of others, and aborts the activity.



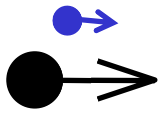
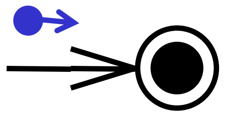
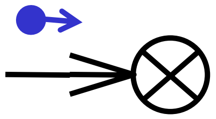
Parameter Sets

- **Parameters accepting input or providing output exclusive of each other (for each execution).**



Control Nodes

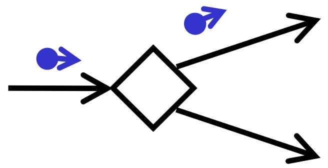
- Route objects/data
- At beginning and end of activity:

Initial Node		Gets control when containing activity starts. Flows out immediately.
Activity Final		Accepts input, aborts containing activity.
Flow Final		Accepts input, does nothing.

Control Nodes

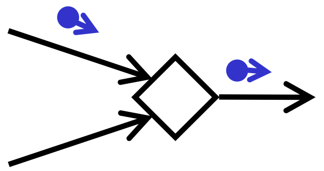
- Route objects/data
- In middle of activity:

Decision



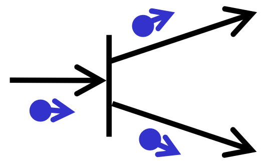
Flows out in exactly one direction.

Merge



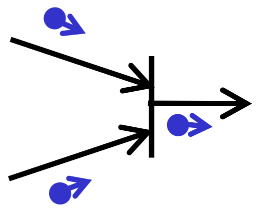
Flows through immediately. Does not combine the tokens.

Fork



Copies inflow to multiple multiple outflows.

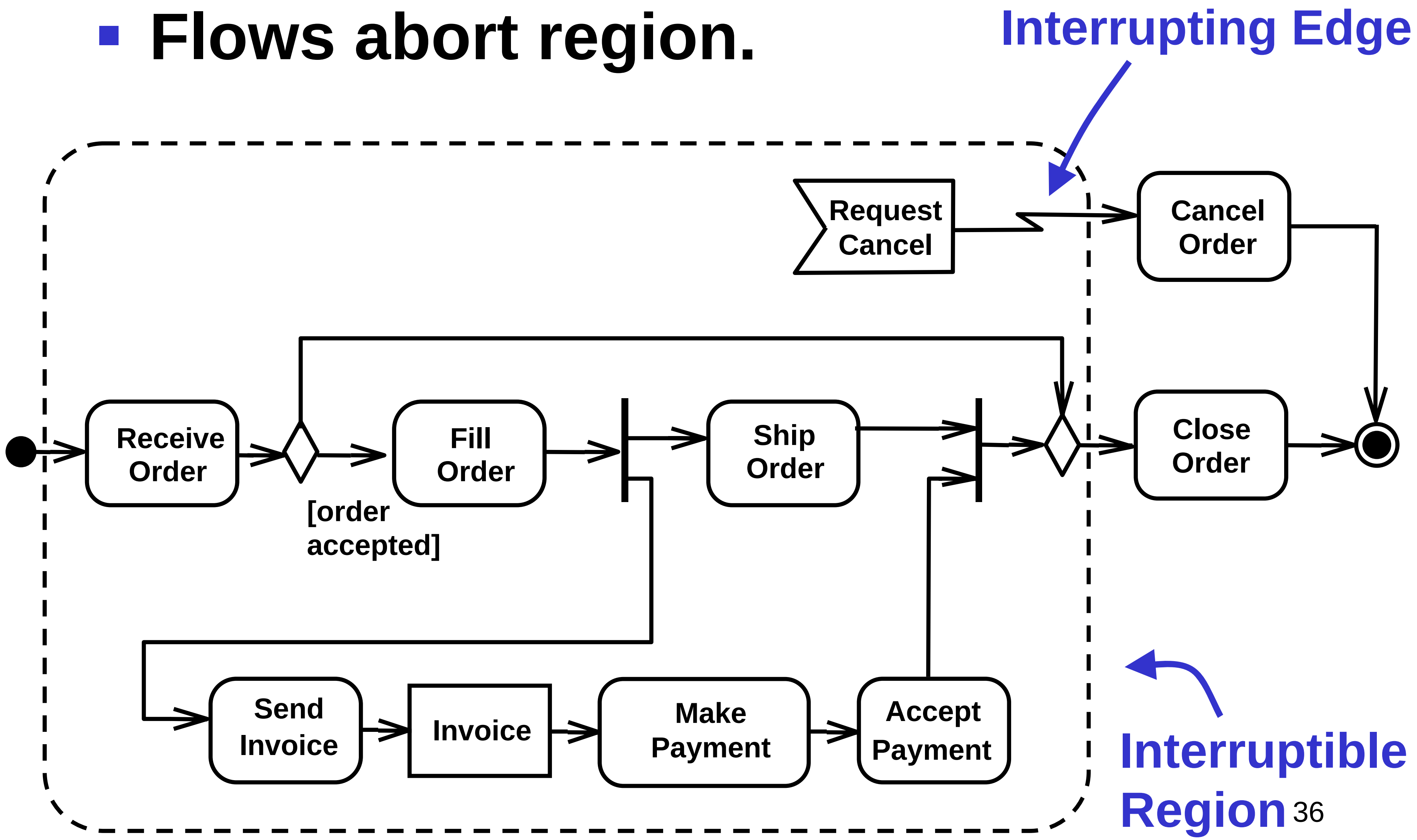
Join



Flows out when all inflows arrive. Combine tokens when possible.

Interruptible Region

- Flows abort region.

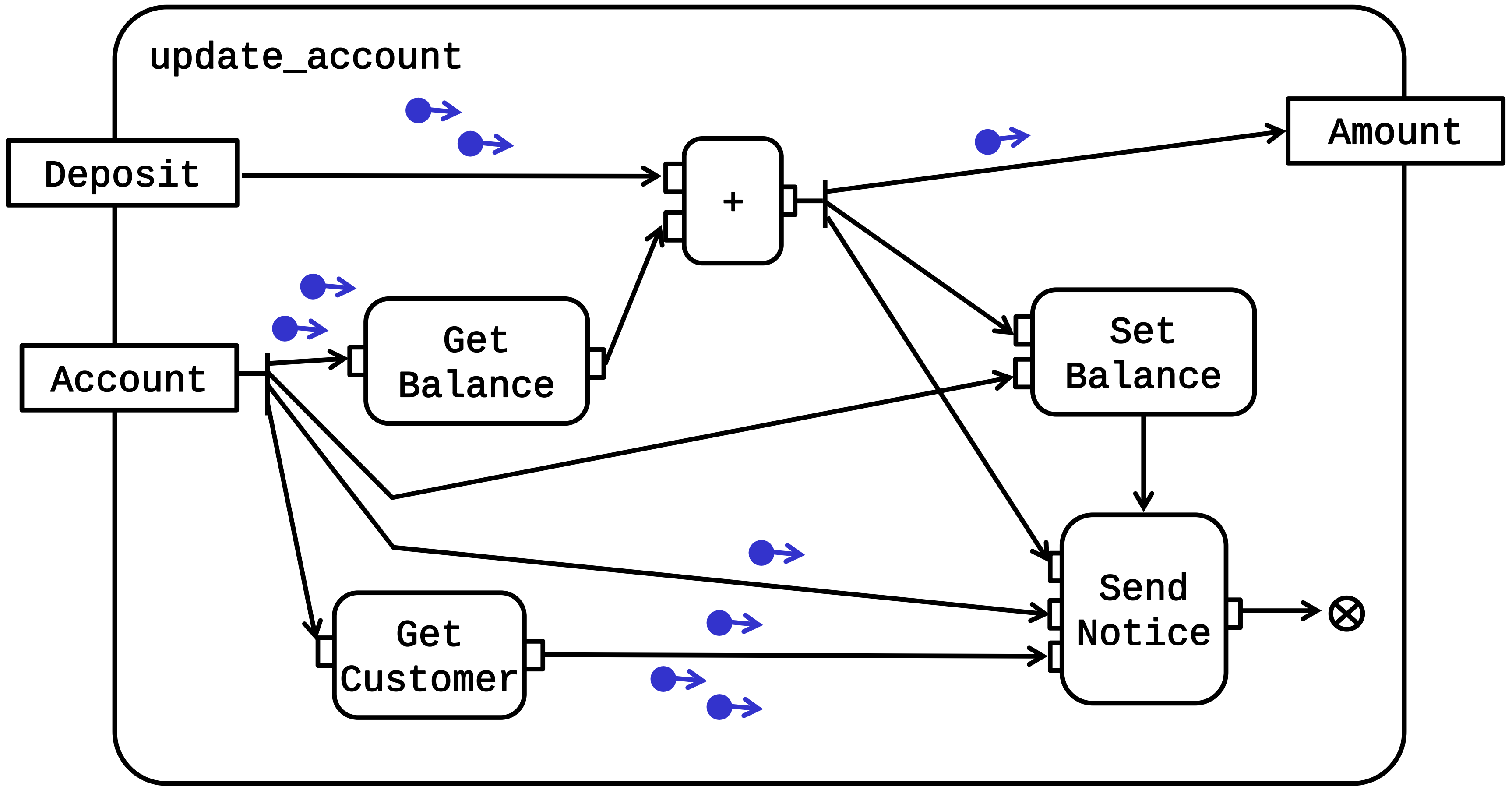


Reentrant Activities

- No token interaction.
- For domains without resource constraint, such as computation.

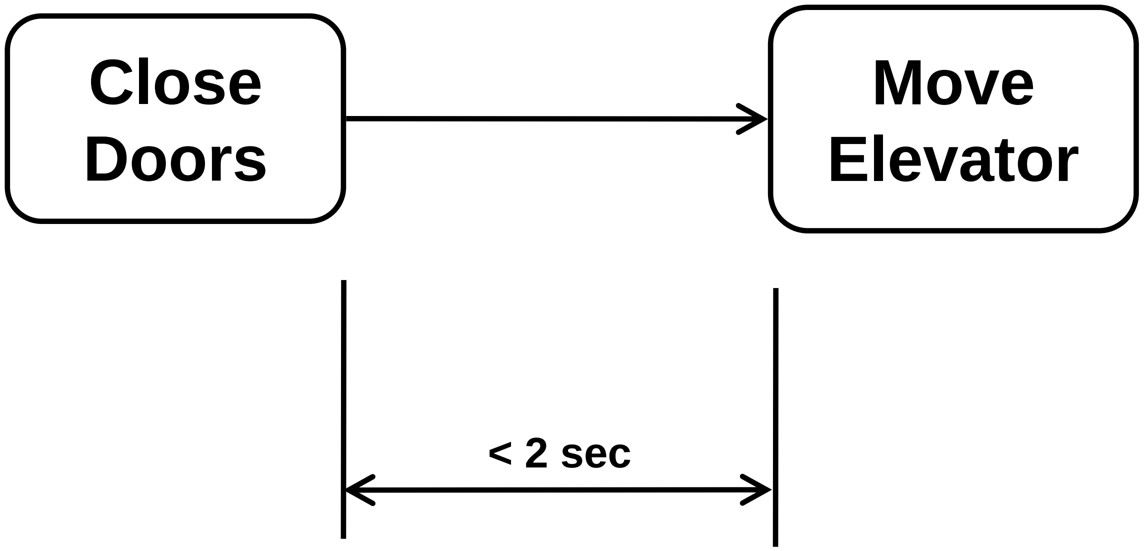
```
Amount function update_account
            (a : Account, d : Amount)
{
  Amount nb = a.balance + d;
  a.balance = nb;
  send_notice (a.customer, a, nb);
  return nb;
}
```

Reentrant Activities



Time Model

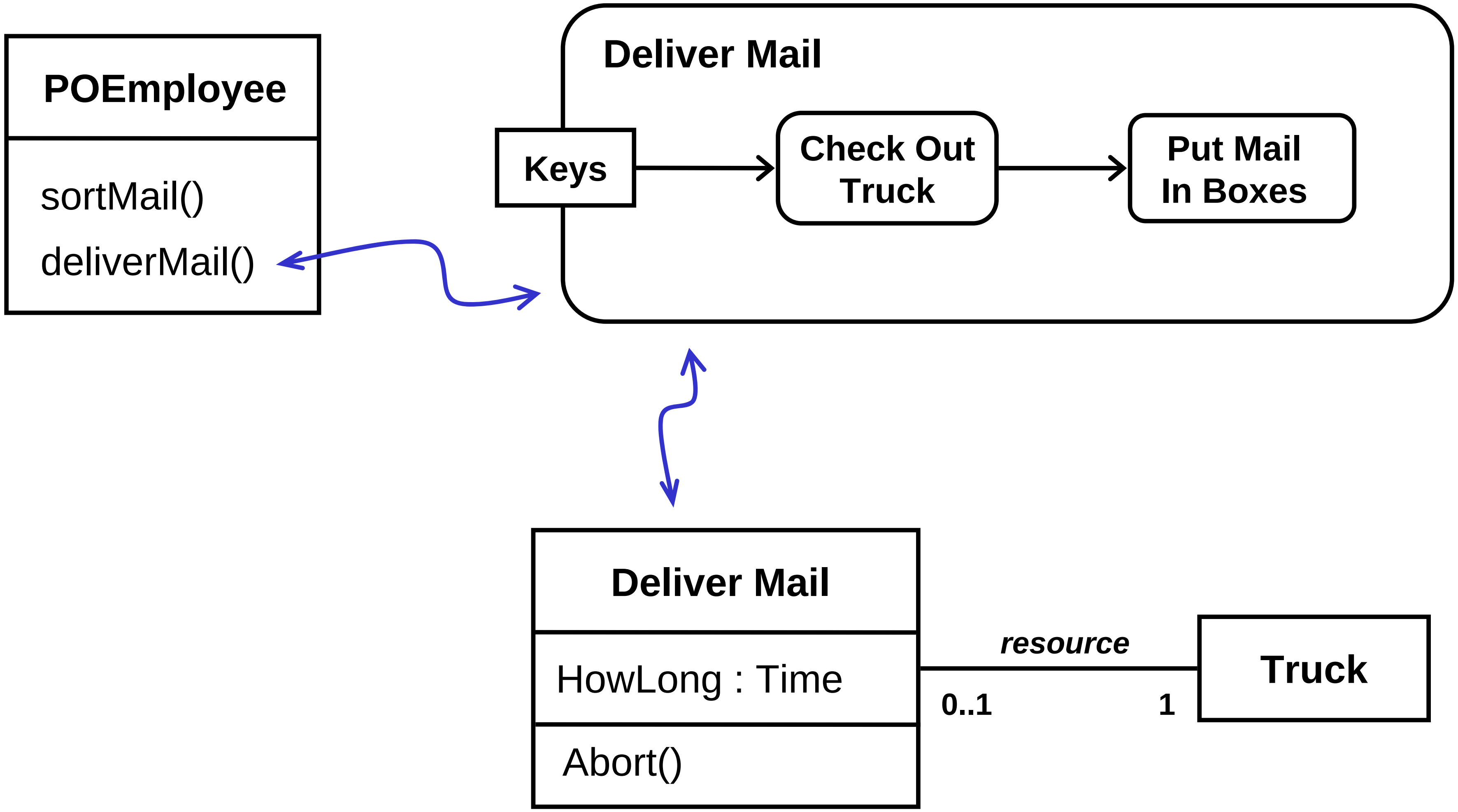
- **Can be used to state constraints on processes:**



First-class Behavior Model

- Object-orientation not required to model dynamics …
- … but supported when needed.
- Flexibility in using/not using:
  - Behaviors owned by objects.
  - Messages and Polymorphism
- Integrate with OO for:
  - Relating internal execution to exchanges with between partners.
  - Transformation to implementation

First-class Behavior Model



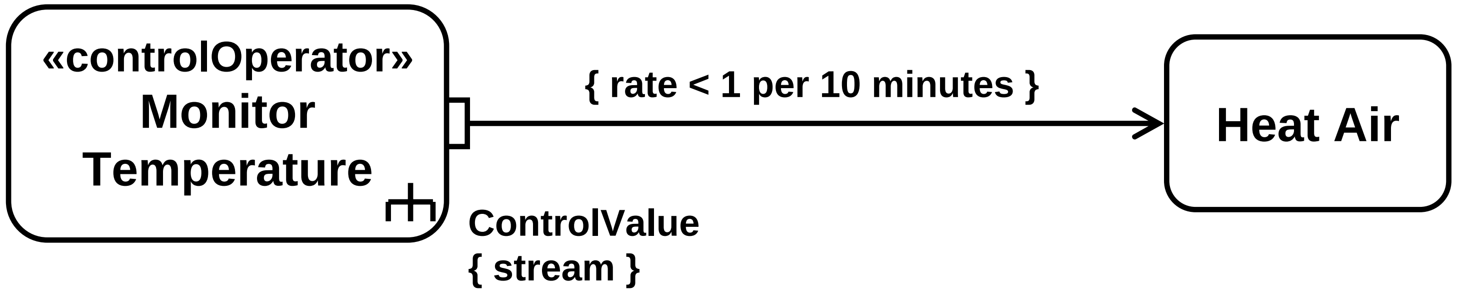
Full Action Model

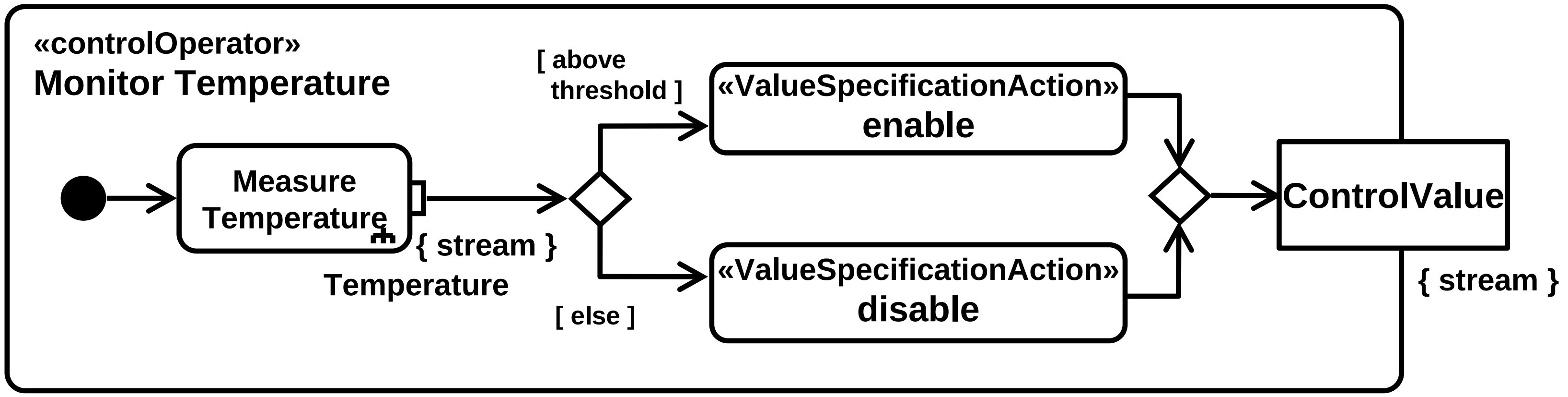
- Kinds of actions include:
  - Invoking behaviors/functions.
  - Creating/destroying objects.
  - Getting/setting property values.
  - Structured nodes (conditionals, etc).
  - Exception handling.
- For fully-executable models and simulations.

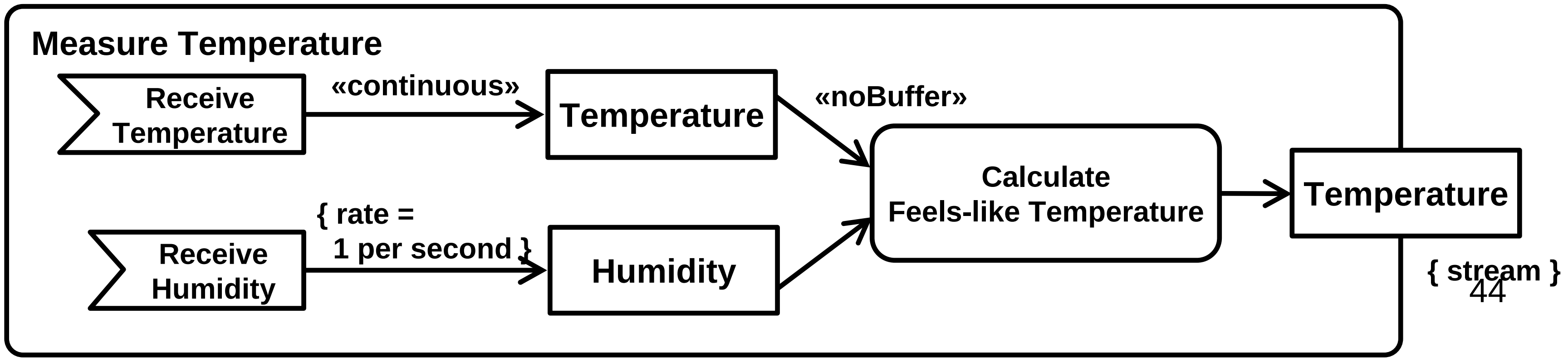
SE Extensions

- 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 node.
- Behavior decomposition.

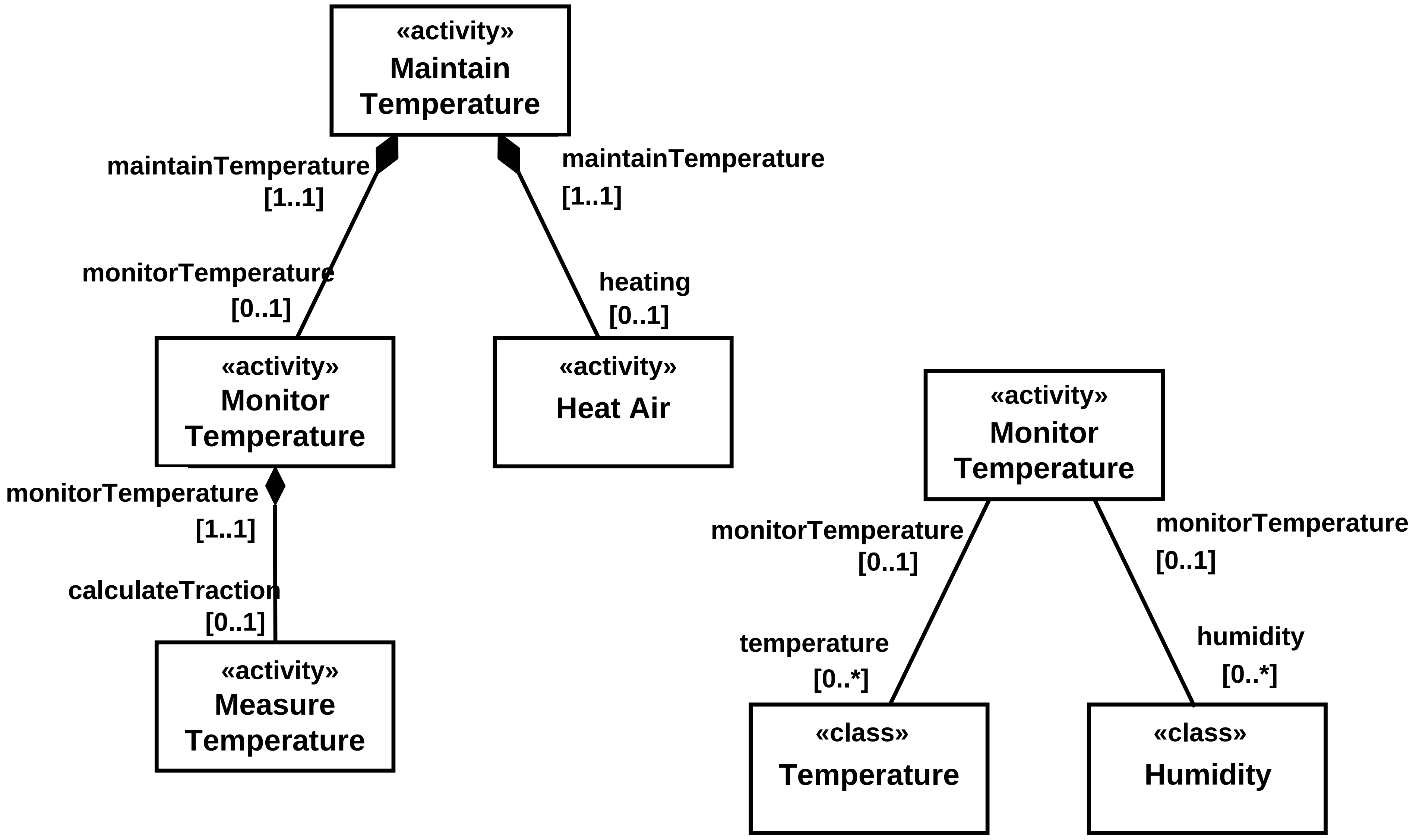
Rate and Buffer Reduction







Activity Decomposition



Validation

- Systems Engineering
  - UML 2 developed completely separately from SE …
  - … SE execution semantics matched UML 2 activities almost exactly.
- High-throughput data flow applications
  - Concurrent/pipeline hybrid.
  - Optimized concurrent flow rate.
  - Used for coordinating networks and business applications in telecom and financial applications.

More Information

- UML 2 specification: http://doc.omg.org/formal/05-07-04
- UML 2 Activity articles: http://www.conradbock.org/#UML2.0
- SysML submission: http://doc.omg.org/ad/05-11-01