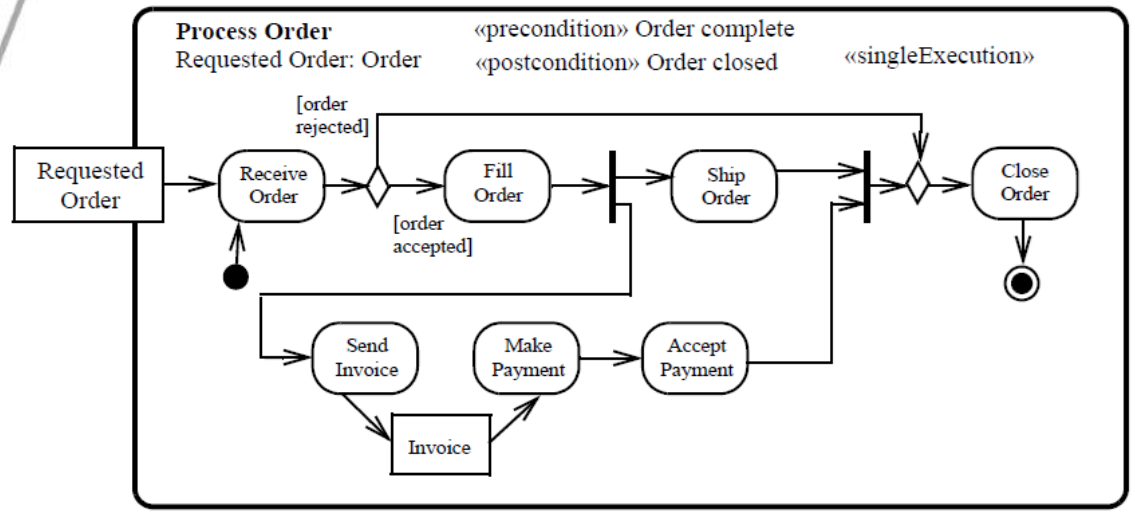
UML Behavioural Modelling

HUSZERL Gábor
huszerl@mit.bme.hu

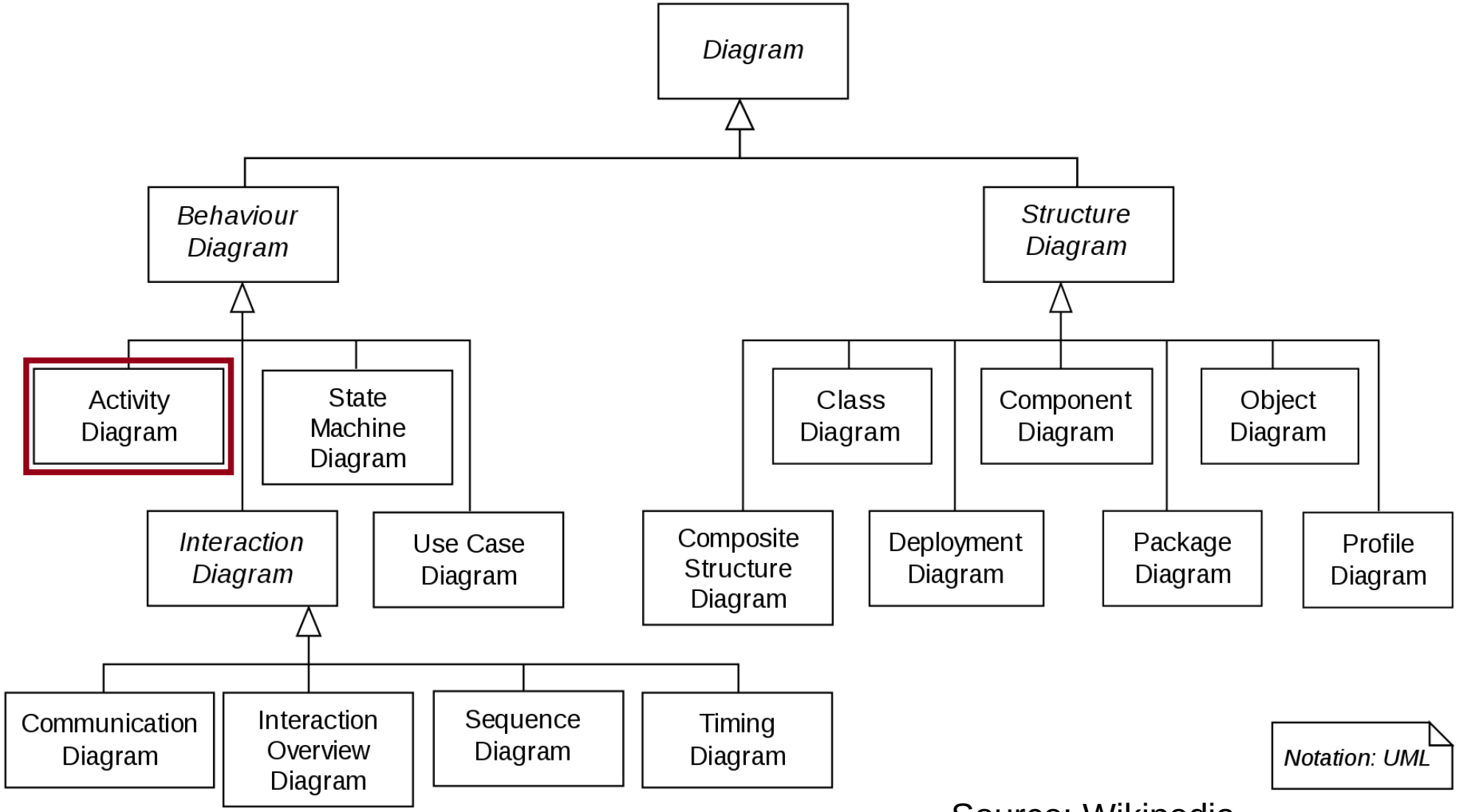
MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

UML Activity Models



Control and Data Flow

UML Diagram Types

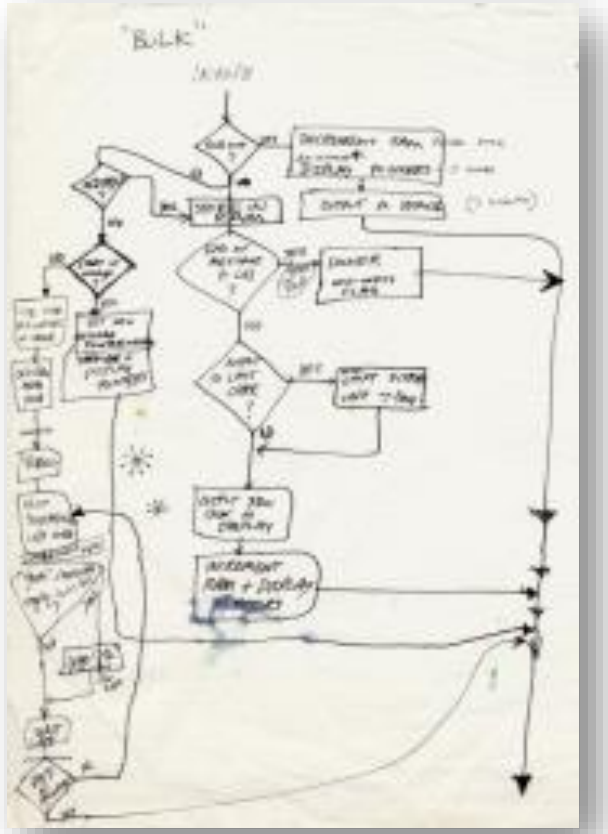


Source: Wikipedia

The Background of the Process Models

„**Process**: set of interrelated or interacting activities which transforms inputs into outputs.”

- Descriptions, old-established in different versions:
  - Control structures of programs
  - Brainstorming (Flow chart)
  - Scheduling project tasks (GANTT)
  - Descriptions of production and business processes



- **Common**: elementary steps and the dependencies among them

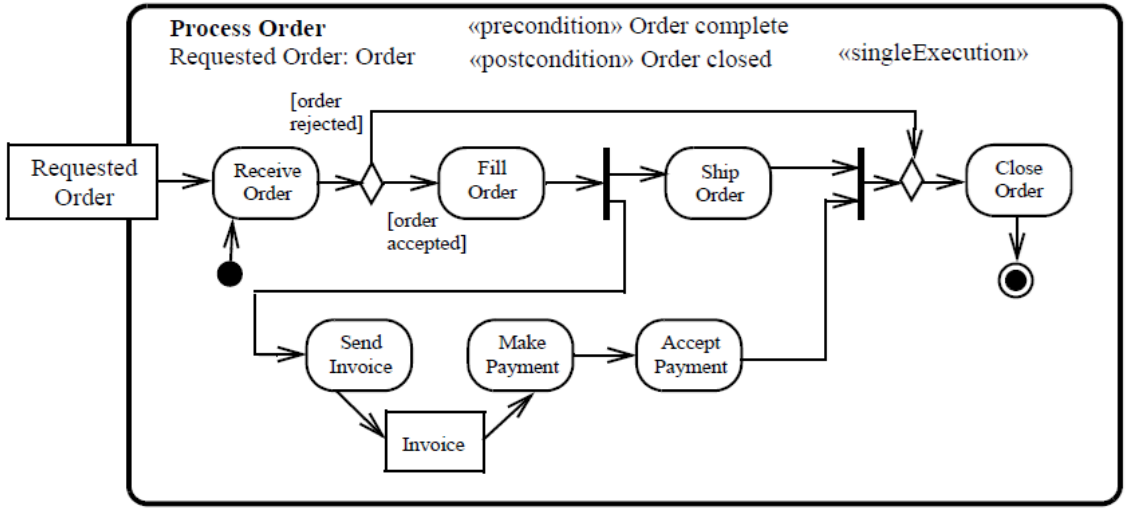
How They Can Be Used in Software Systems?

- High-level descriptions of business/system processes
  - E.g. explaining use cases
  - What are the input/output and main steps of the system
  - Which subtasks are performed by which actors

- Low-level descriptions of behaviour
  - Specification of a complex method/algorithm
  - Detailed behaviour in an executable model (e.g. effects of an event)
  - Actions: elementary operations on UML elements

UML Activities

- **Basic elements**:
  - Steps (nodes) and
  - Flow (edges) between them



- Dependencies (control and data flow):
  - Control: target step cannot be started before the source step is completed
  - Data: what (type of) data is required to execute the step

Describing potentially parallel execution!

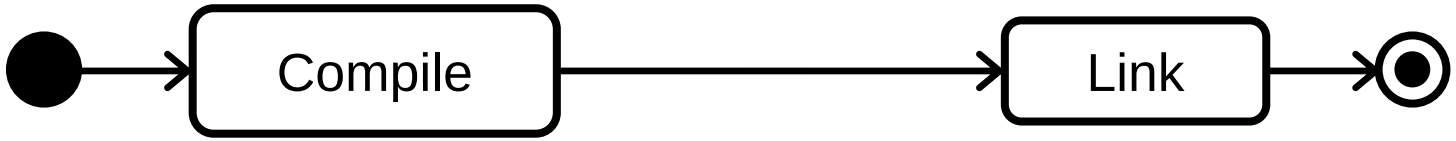
Elements of a Control Flow Model

Elementary Activity

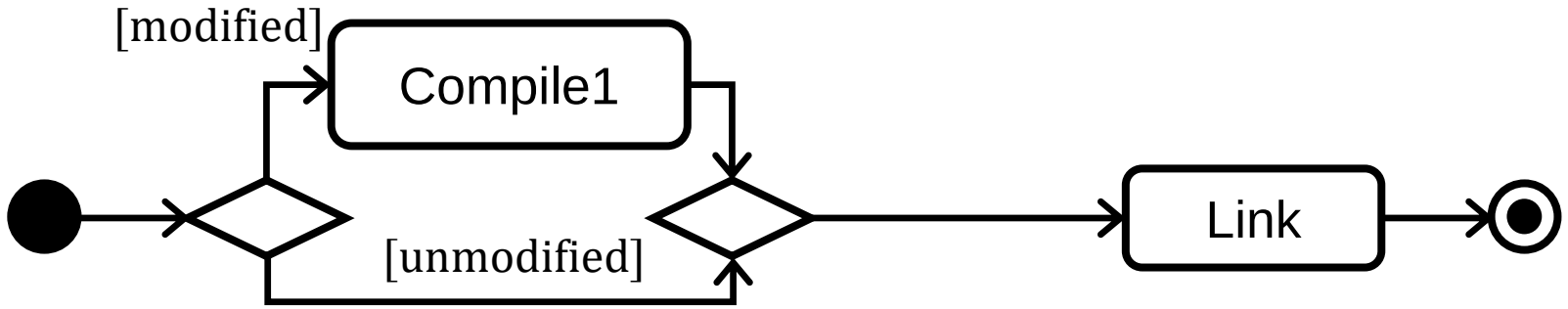
Compile

Initial and Final Nodes

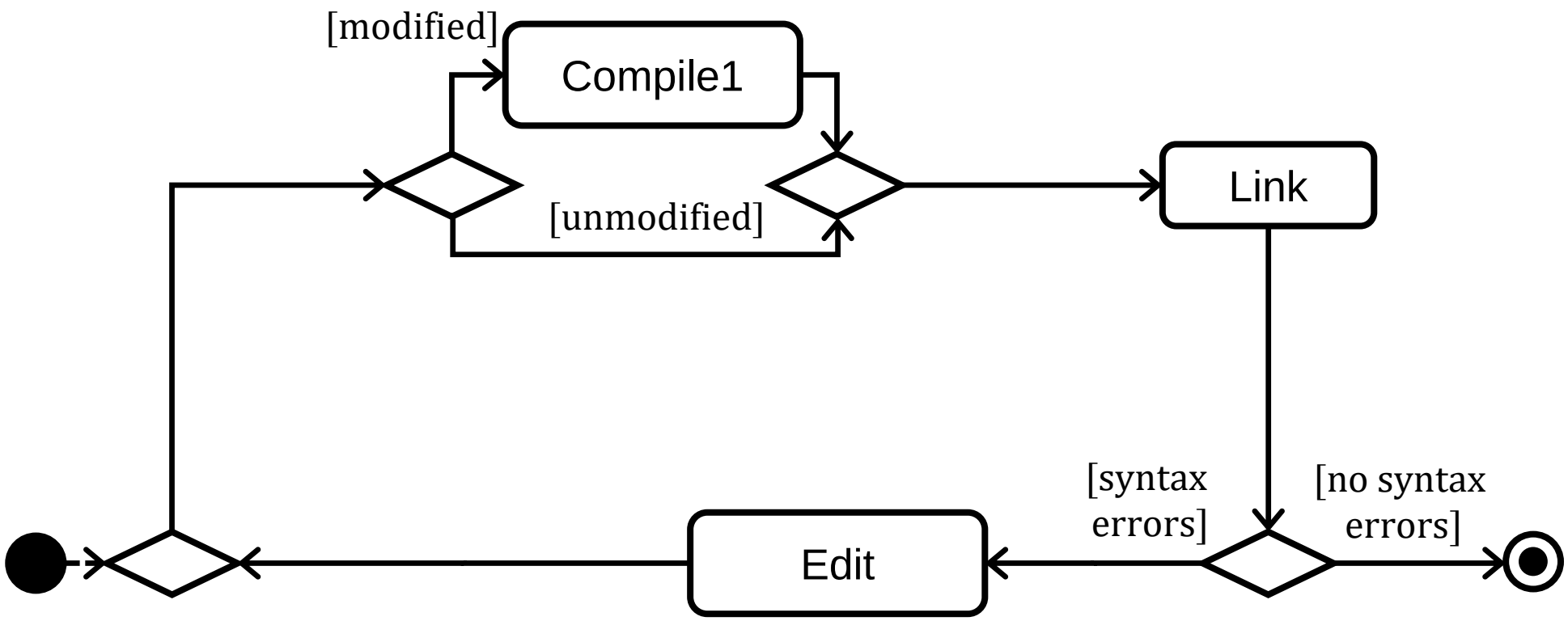
Control Flow



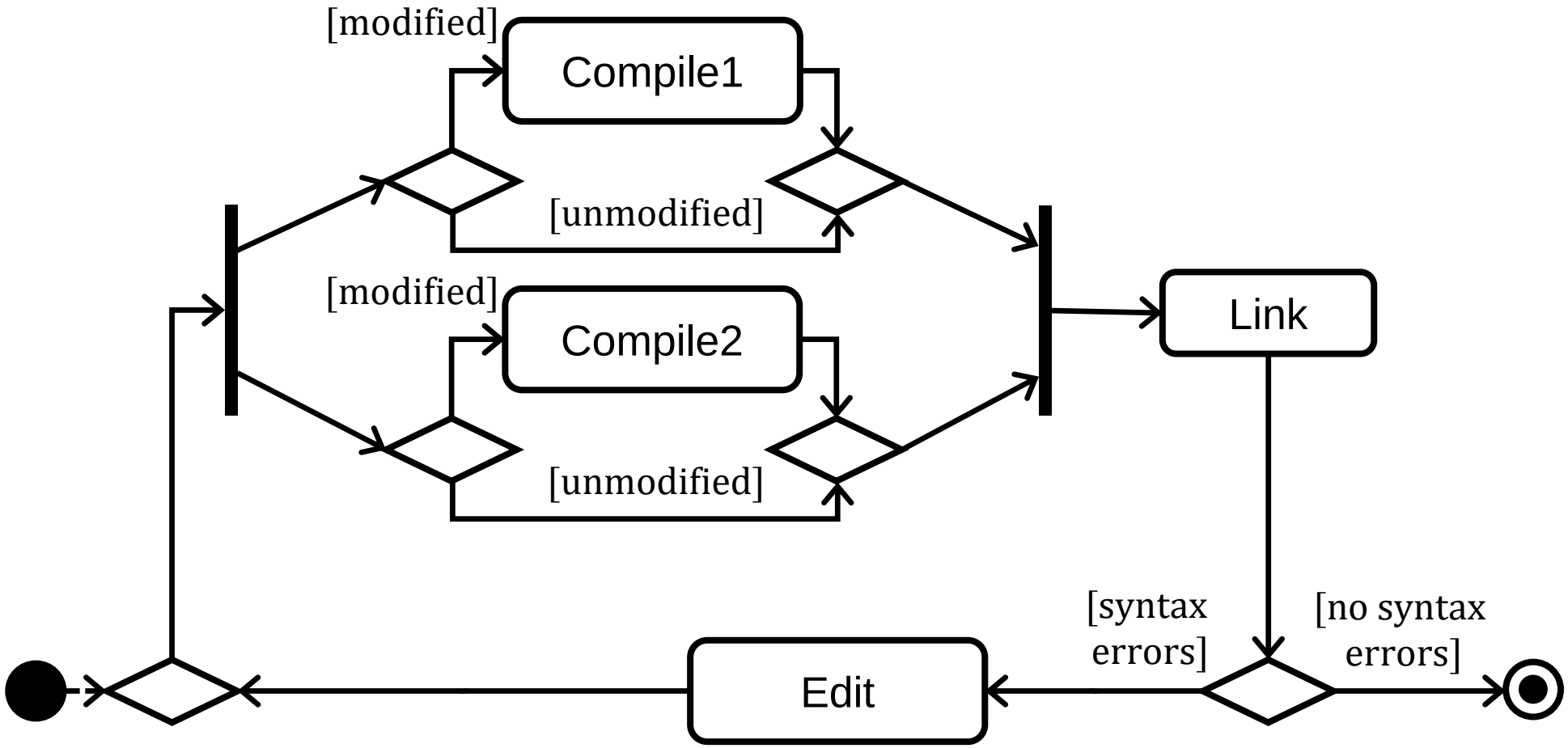
Decision and Merge



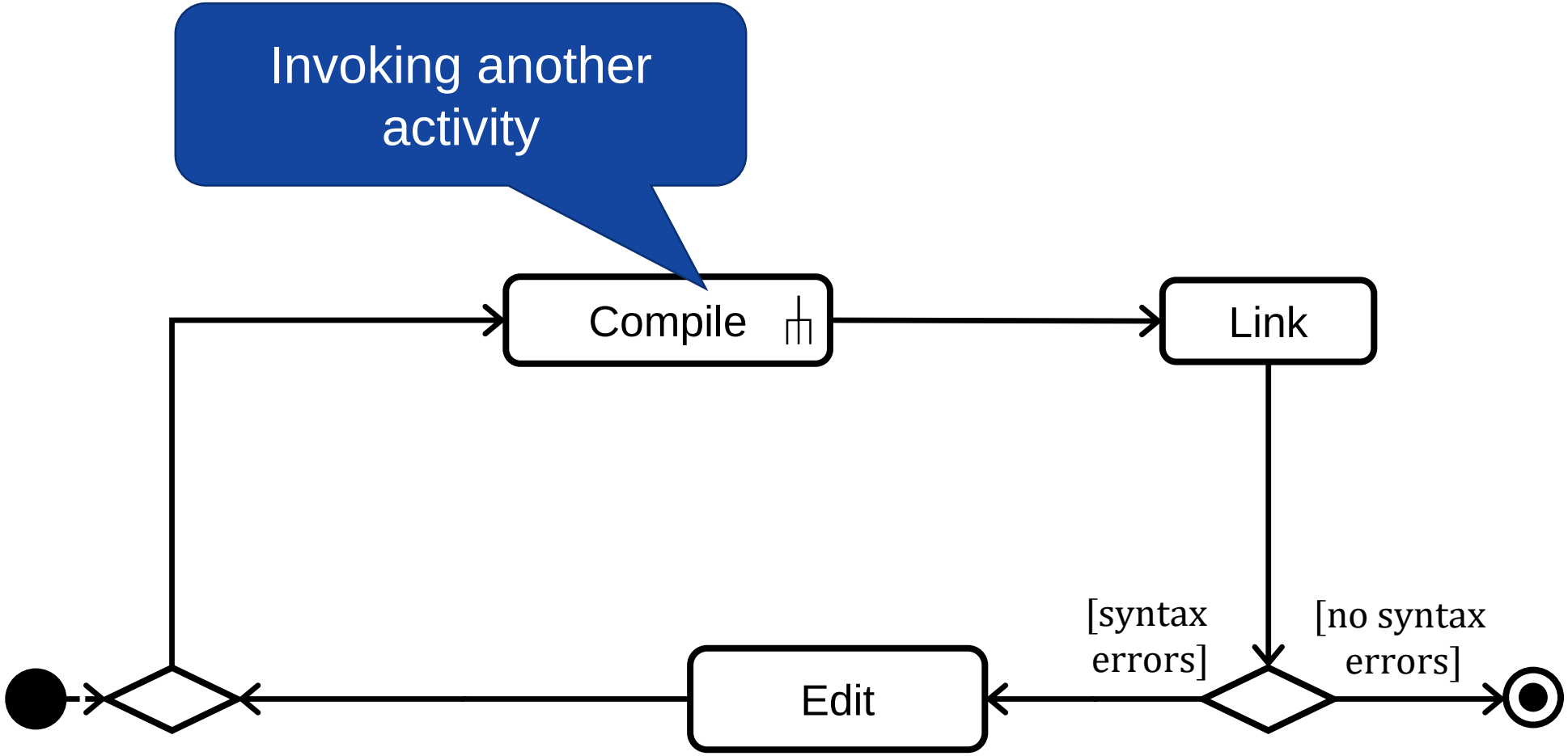
Modelling Loops by Decisions



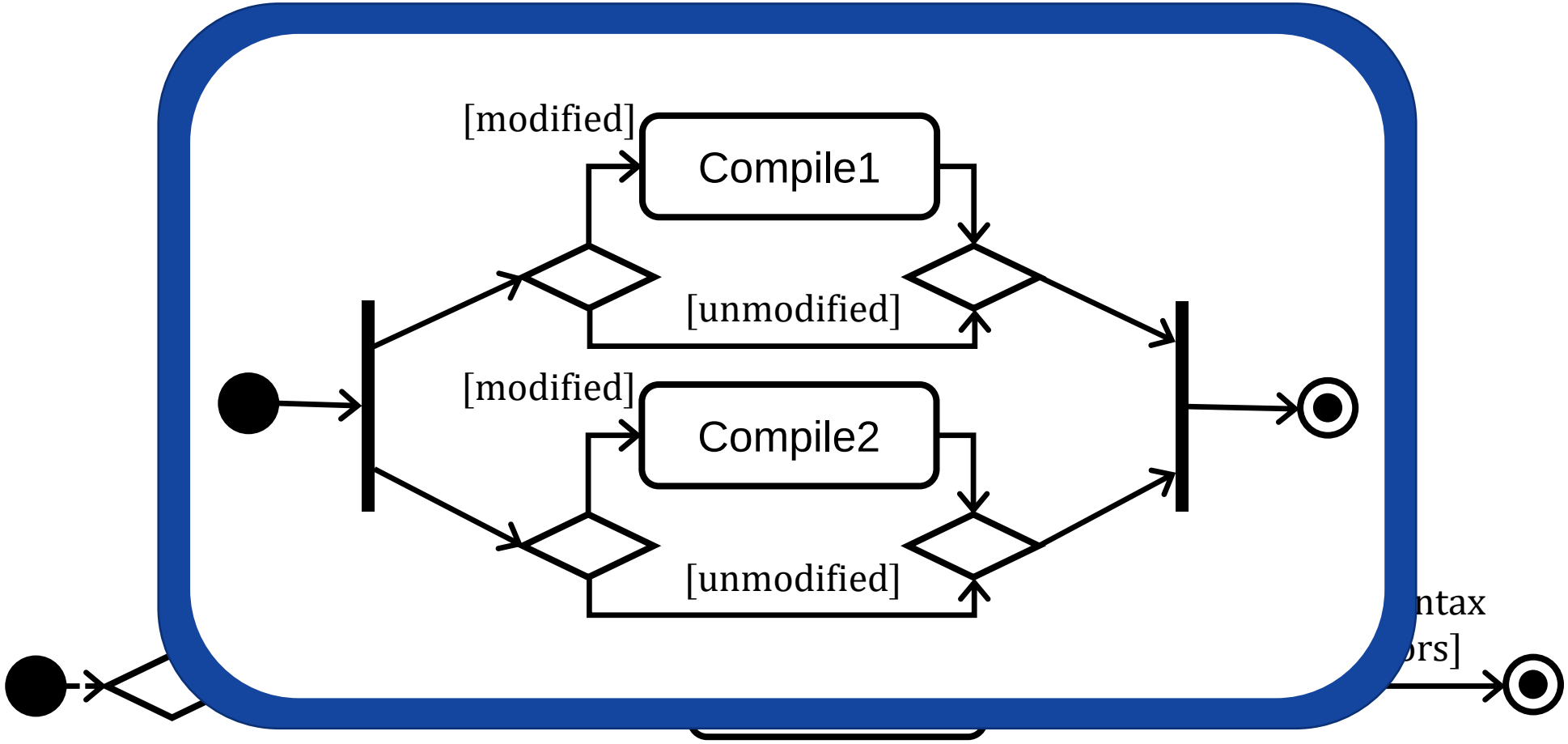
Parallel Execution (Fork and Join)



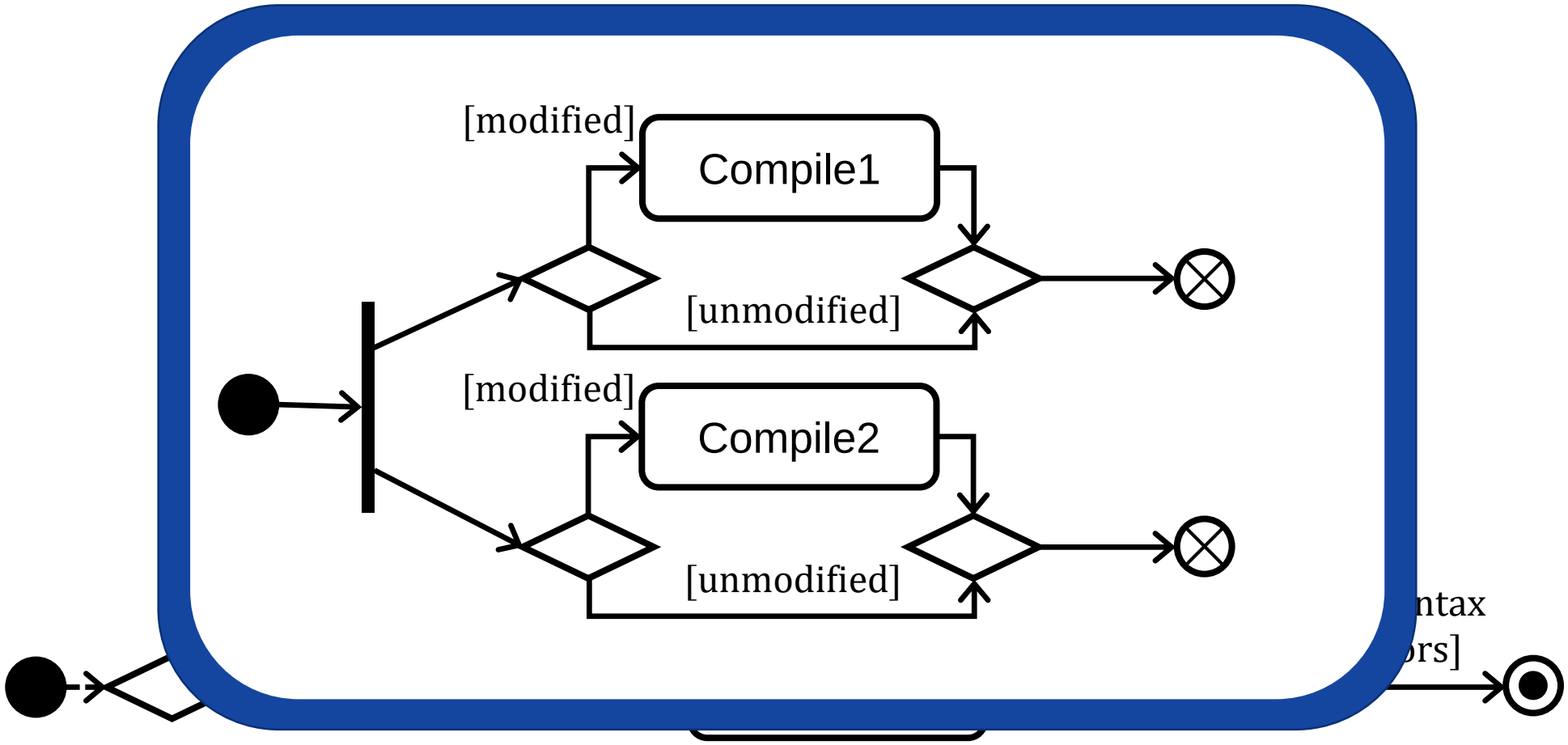
Refinement of Activities (Hierarchy)



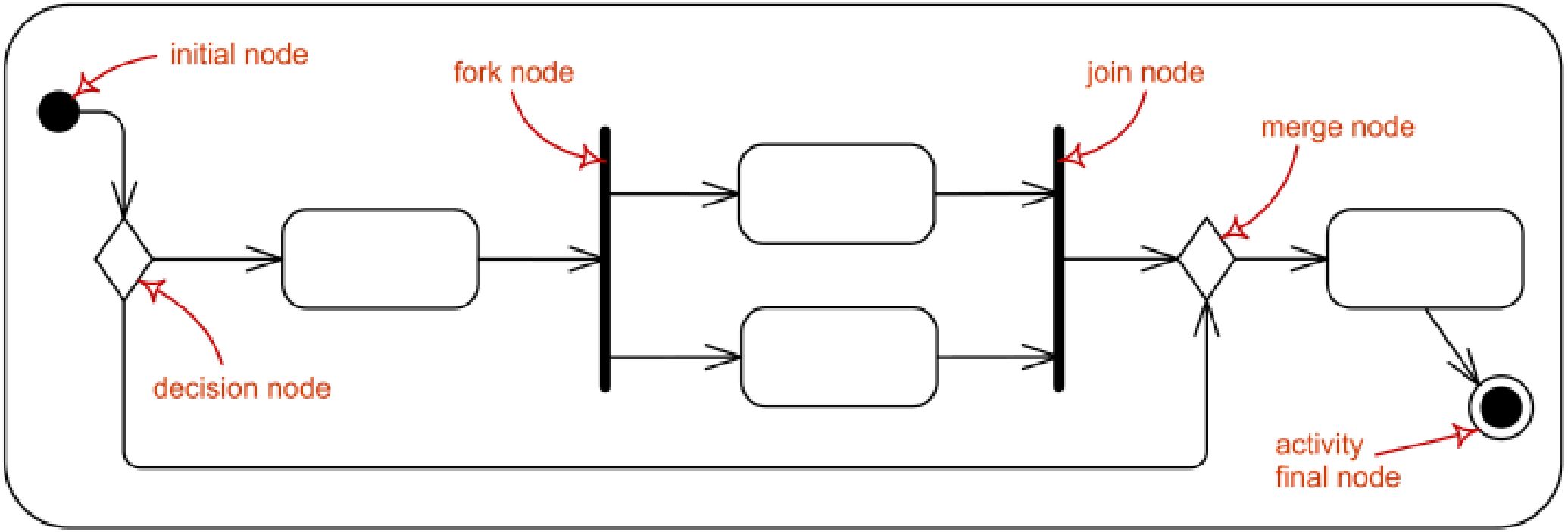
Refinement of Activities (Hierarchy)



Flow End

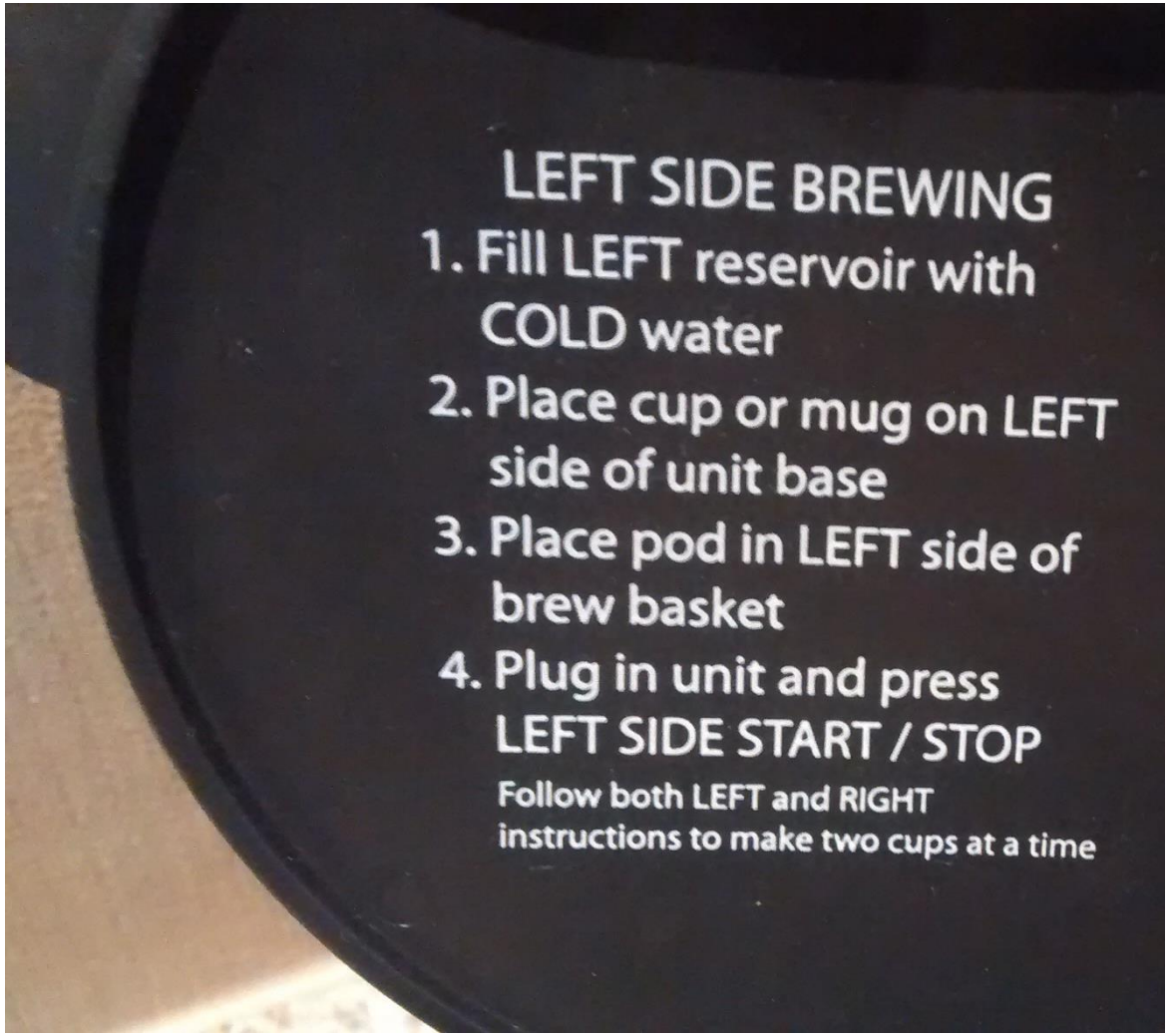

[modified]
Compile1
[unmodified]
[modified]
Compile2
[unmodified]
ntax
ors]

Summary of the Control Elements



Source: uml-diagrams.org

EXERCISE: Coffee Machine



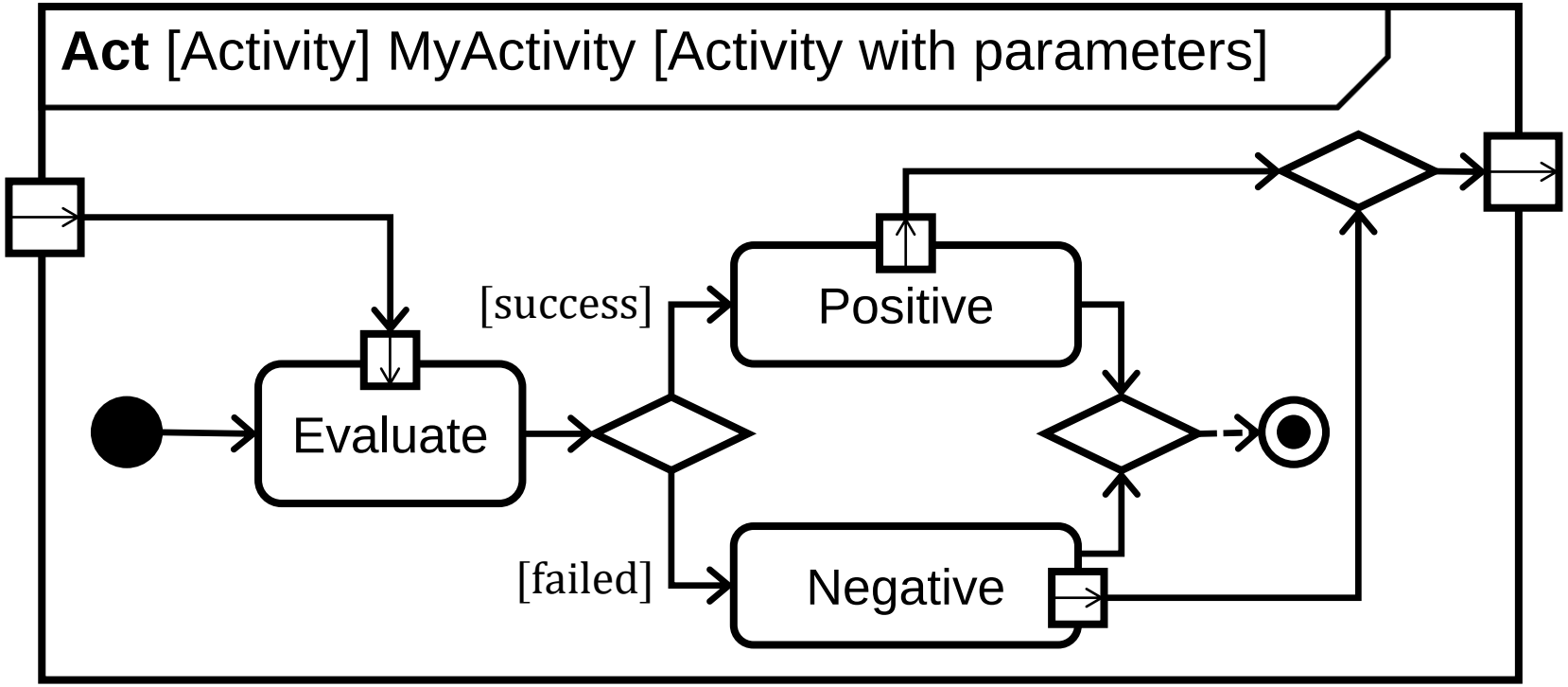
Elements of a Data Flow Model

Modelling a Data Flow

- Activities usually **produce and consume data**
  - Data in a broader sense: object instance, material item, …
  - The produced data may be the input of another activity
- Data generally: **ObjectNode**
- Data connected to actions: **Input/Output pin**
  - May have a name and type
  - Data flow edge must not go into an Action directly (may into a ControlNode)

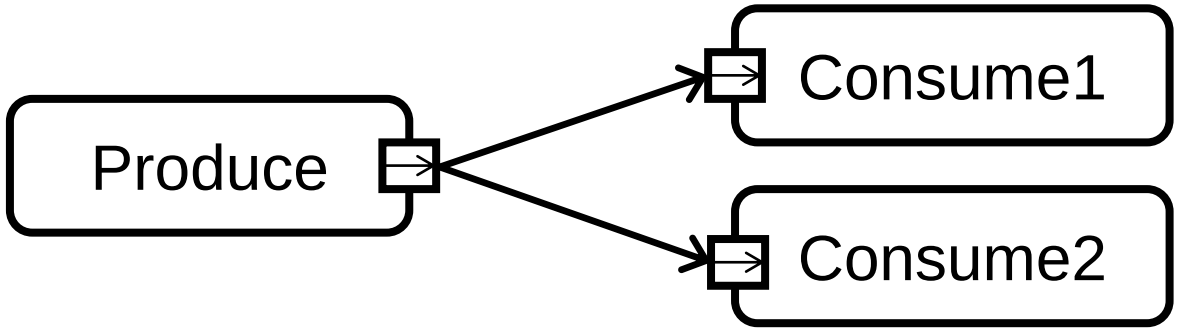
Parameters of an Activity

- Activity may have parameters
  - Parameter pin: represents an incoming or outgoing parameter
  - Notation: square on the border of the activity

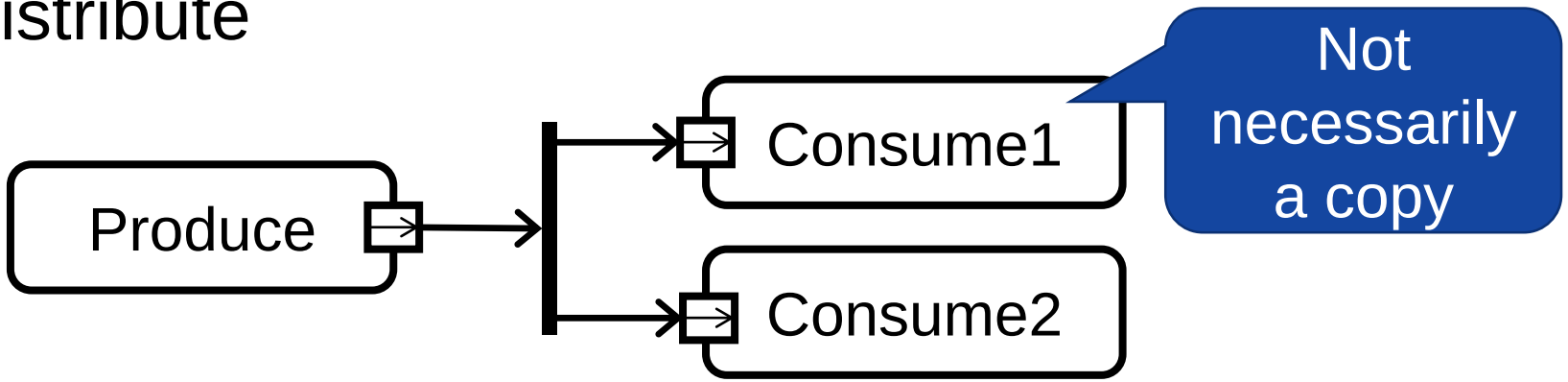


Distributing Data Objects

- By default, output pins produce a **single data object**
  - Input pins connected to the same output pin compete for it



- Use **fork** element to distribute



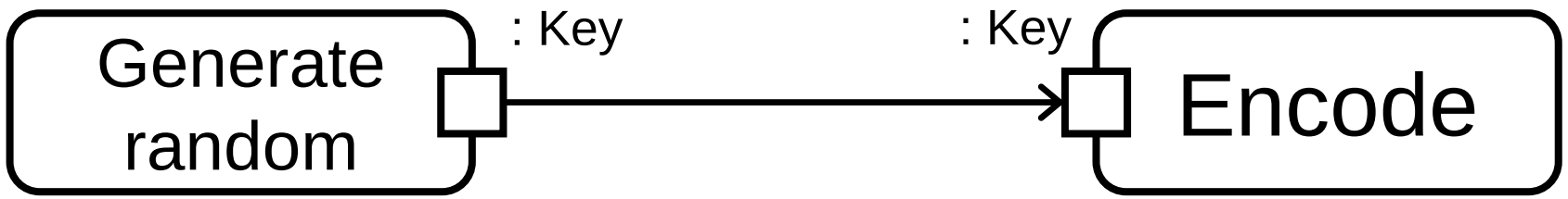
Control Flow vs. Data Flow

- **Data flow** represents data dependency
  - One step requires the data produced by another one
- **Control flow** represents control dependency
  - One step cannot start before another one finishes

- Data flow **may substitute** the control flow
  - Control dependency may be omitted if there is already a data dependency
    - It may be worth showing it if that helps understanding the model
  - Control flow may be considered as a data flow without type and value

Simplified Representation of Data flow

- Two pins of the same type/name are connected to each other:



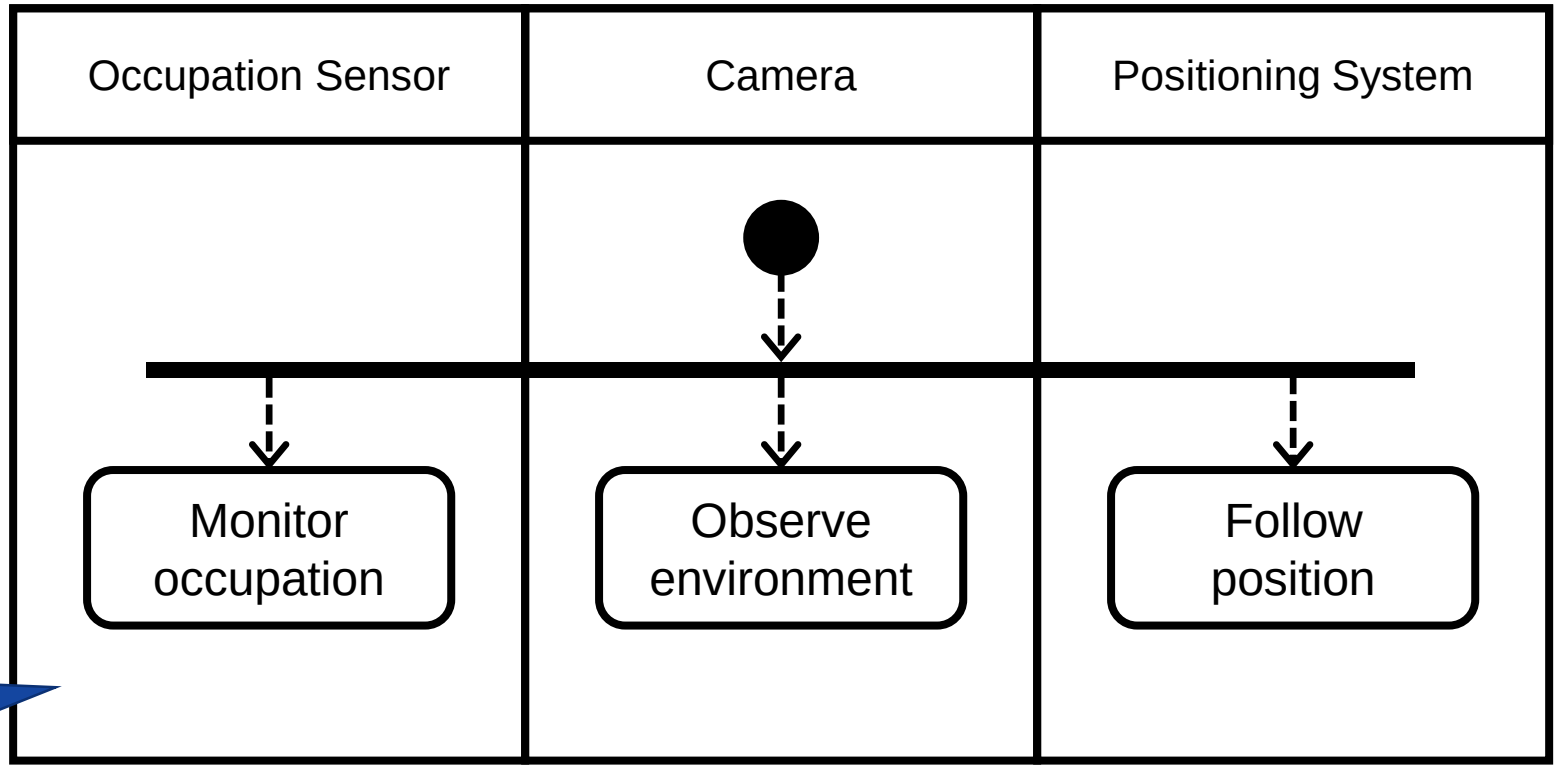
- Simplified representation:

Built-In Actions (Excerpt!)

Primitive action	Primitive action	E.g. reading or writing an attribute, reading a link
Send signal	signal, target, <Signal>	Sending a signal to the given target
Accept event	<Event1>, …, data	Waiting for an event, making the received data available
Accept time event	after(…)	Signalling after the timer expires
Call behavior	param, Call behavior, result	Executing behaviour (e.g. another Activity)

Activity Partitions

- Nodes may be assigned to roles
  - Role: Classifier, instance, Property, … (may also be external)
  - *Who is responsible for the behaviour? Which resource will execute it?*



Partition (swimlane)

Executing Activities

Semantics

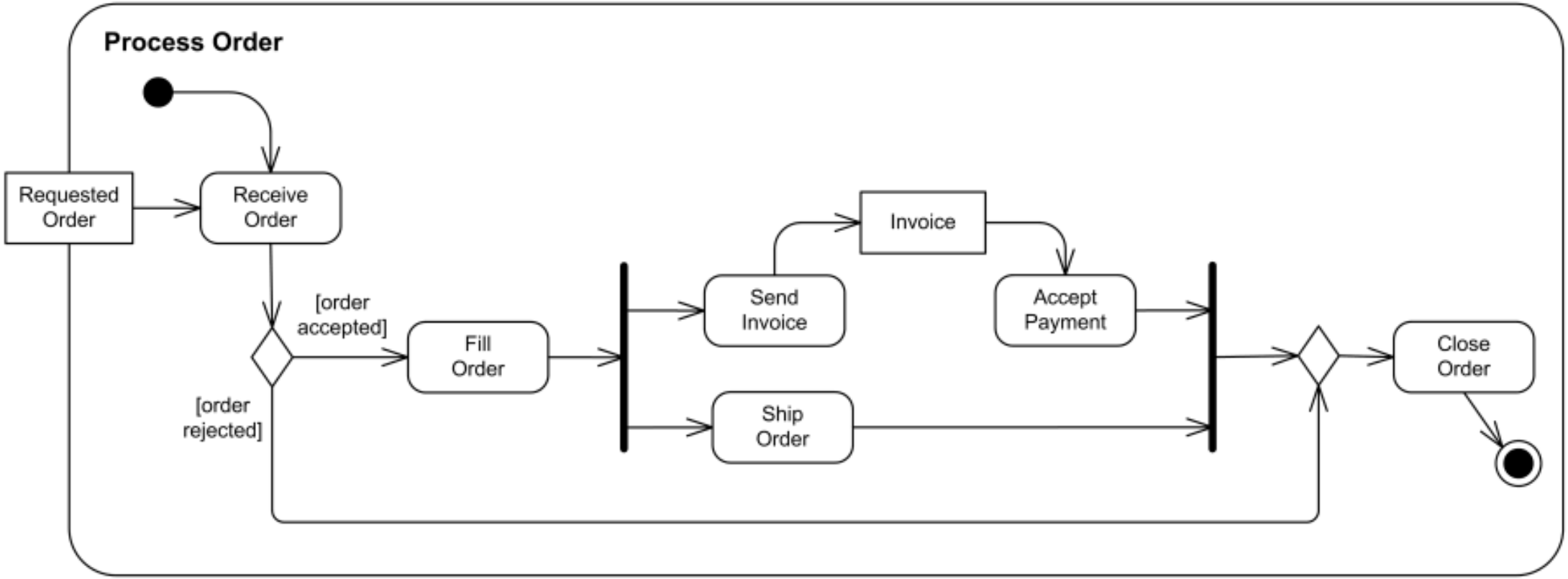
The Key Idea

Token game

- **Token**: representing the state of the execution
- Tokens “flow” along the edges
  - Nodes offer and receive tokens
- The meaning of the node defines what happens with the token

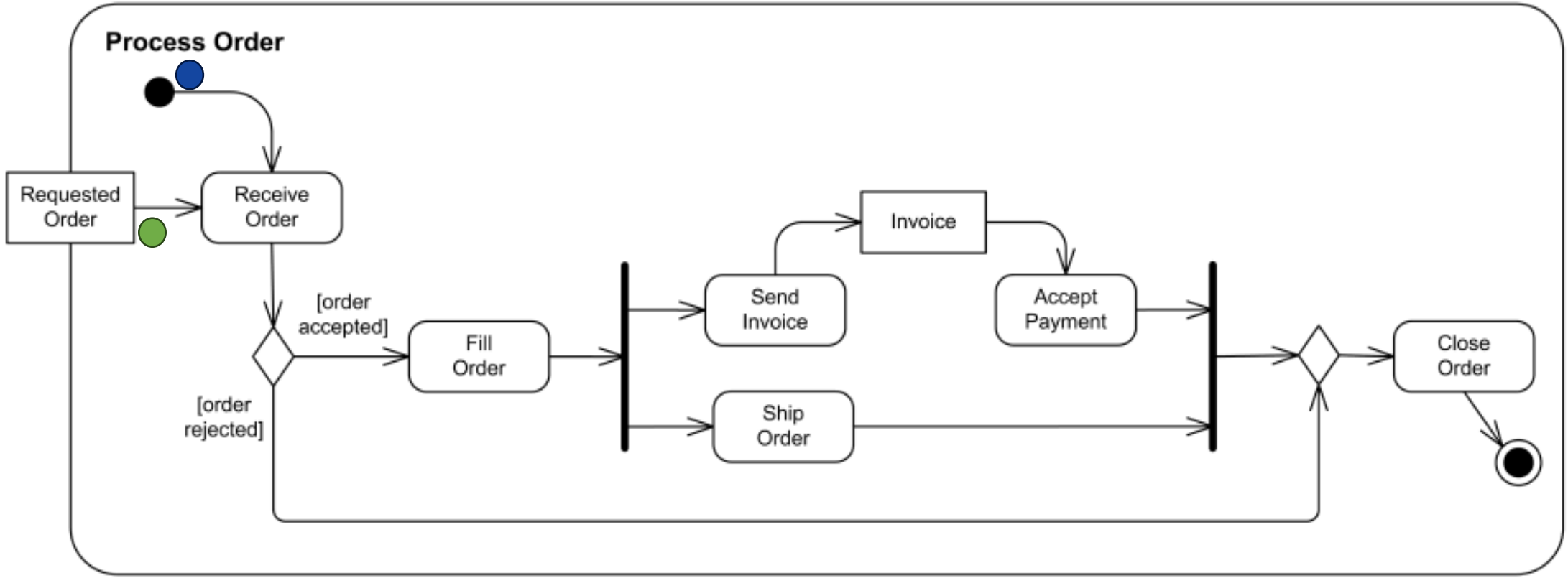
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



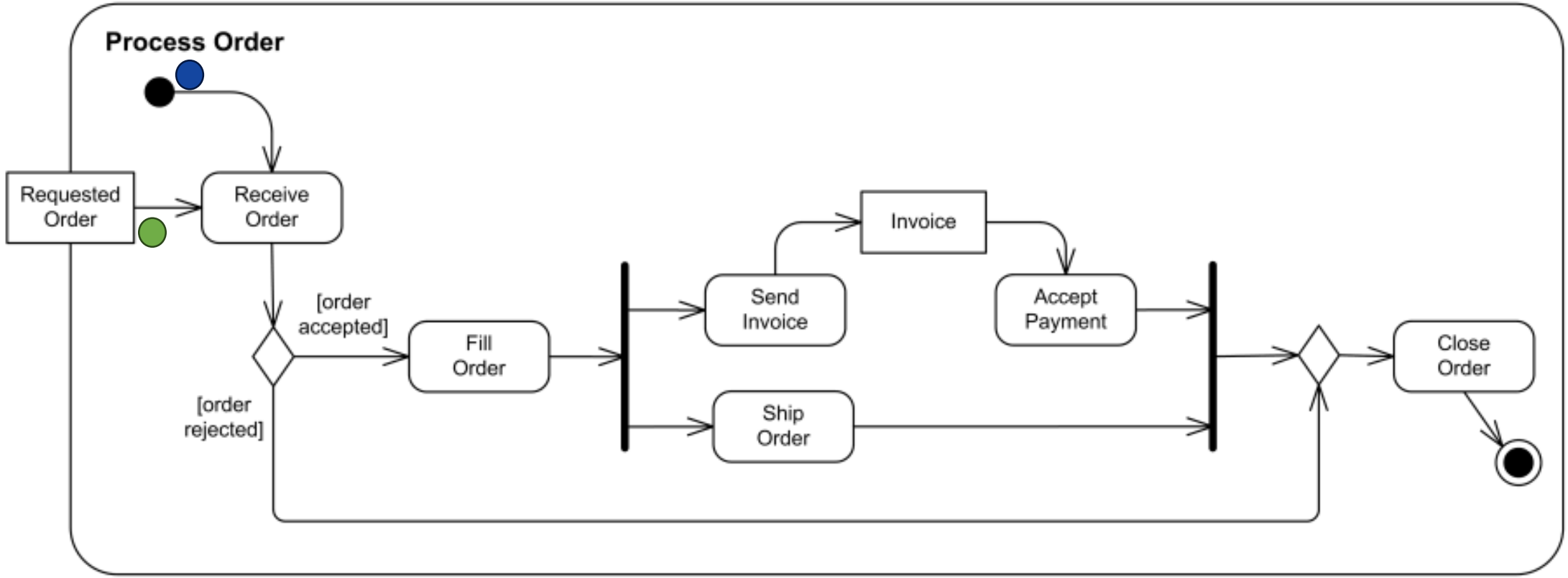
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



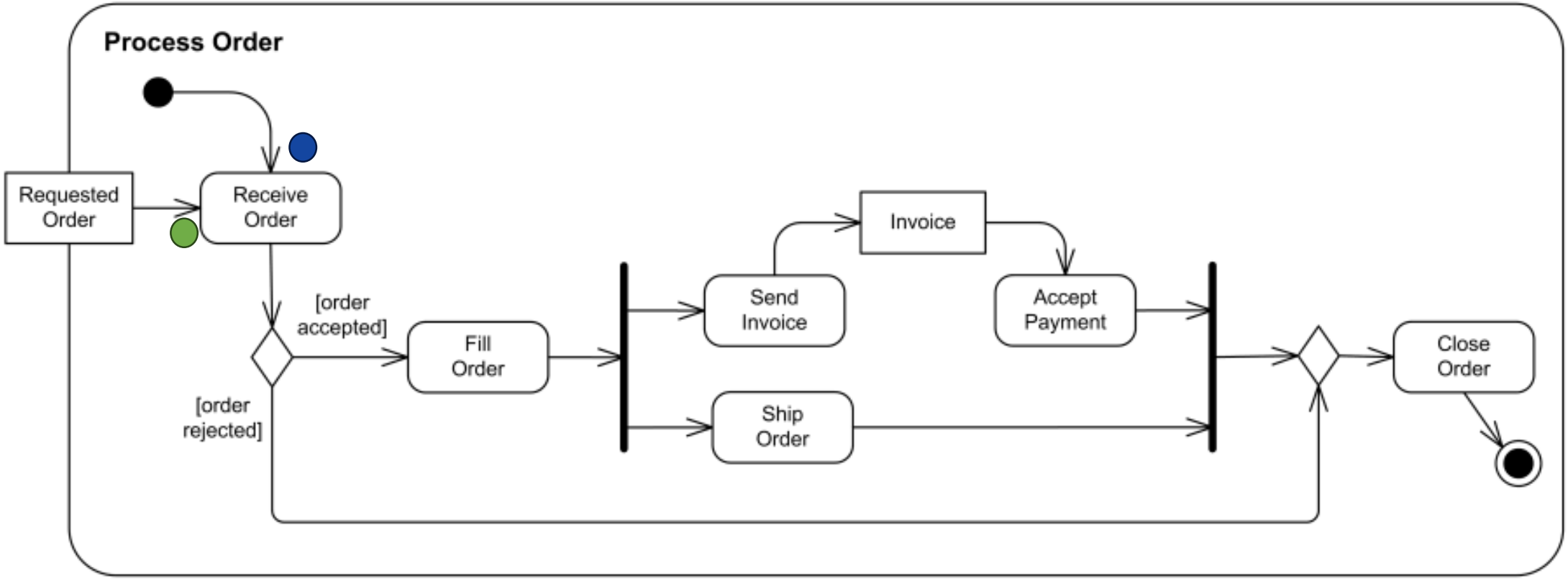
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



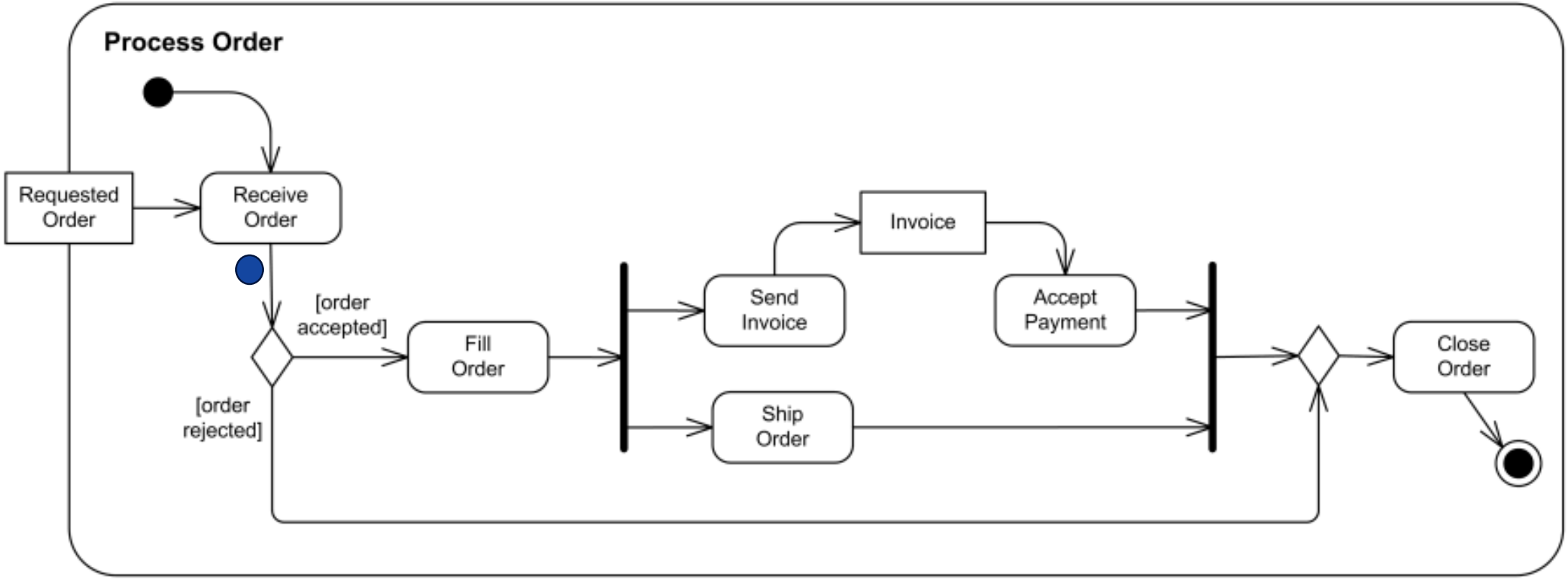
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



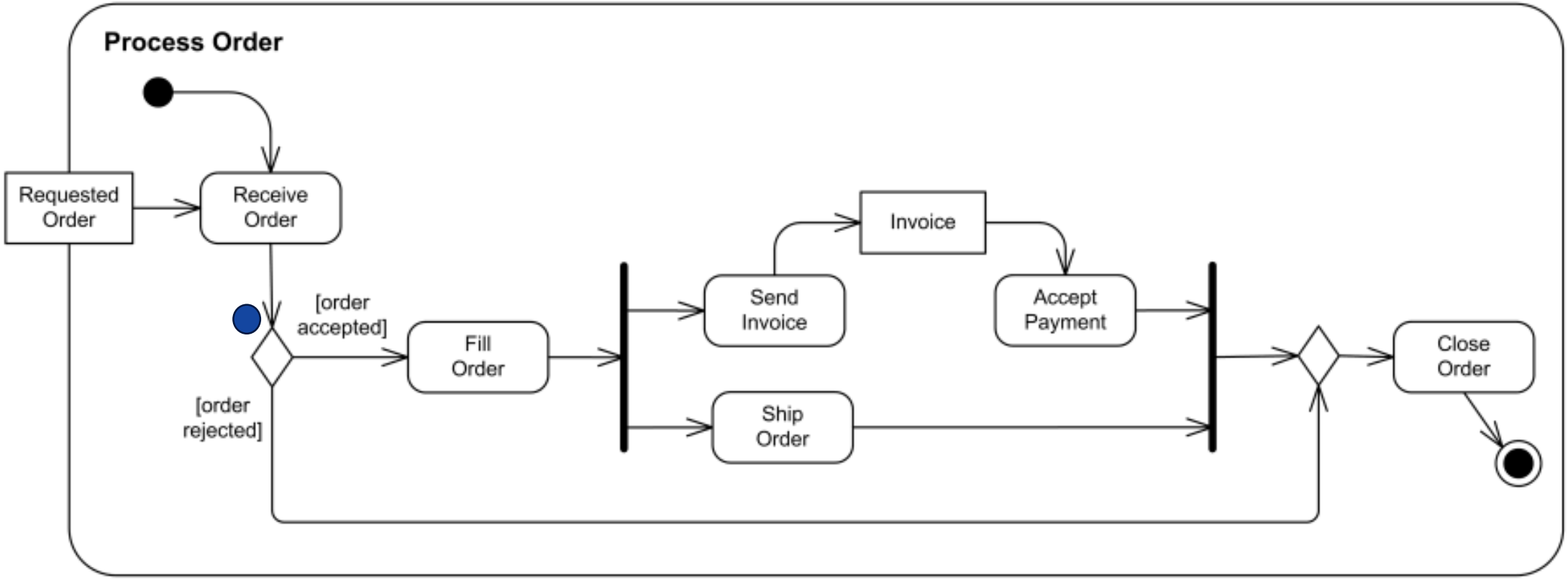
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



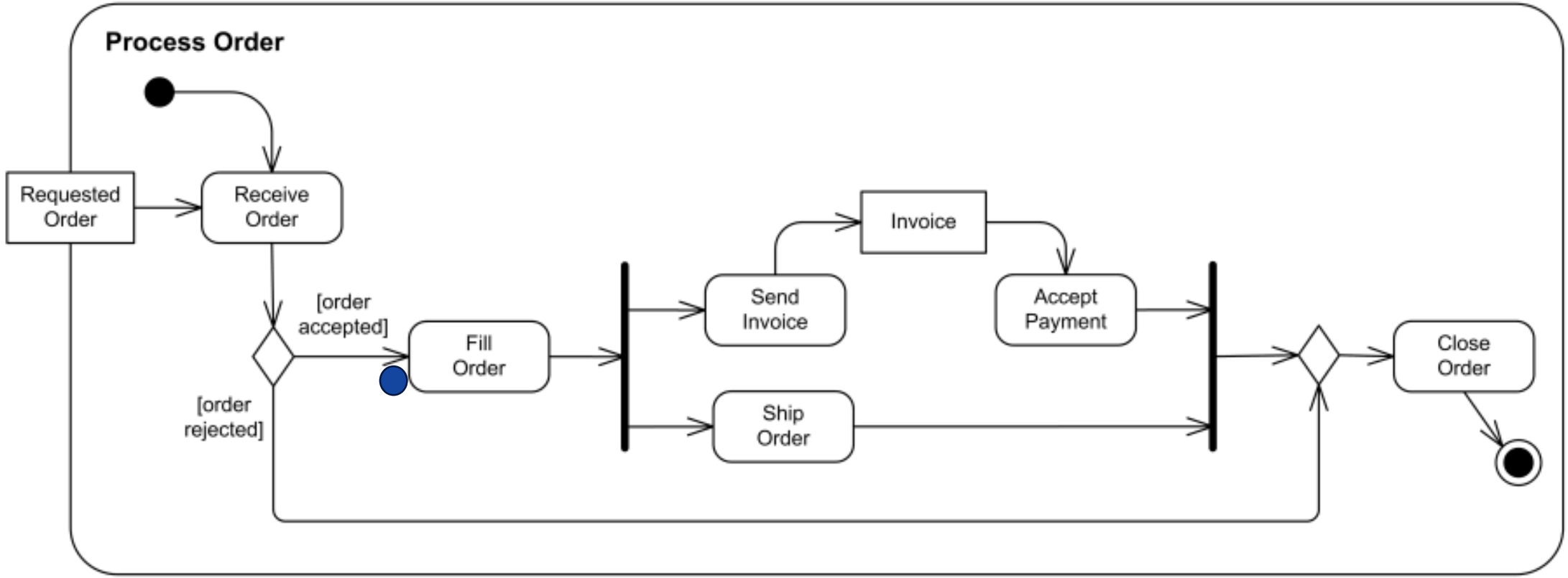
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



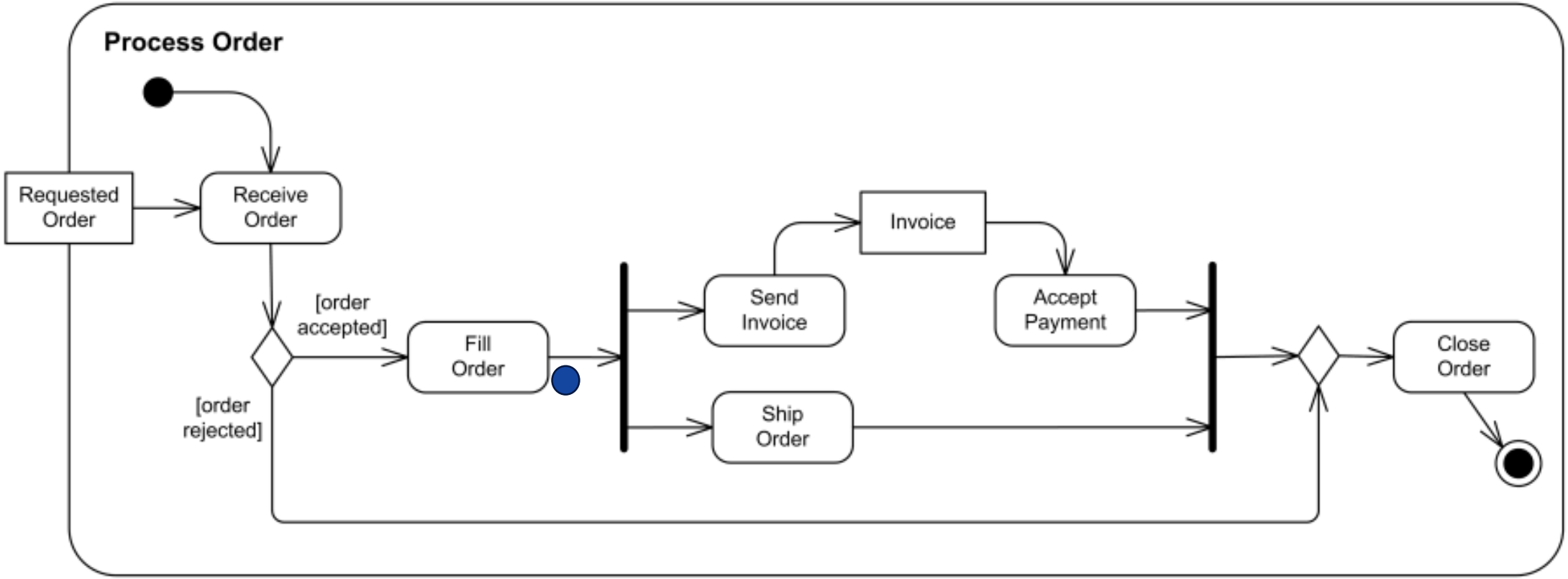
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



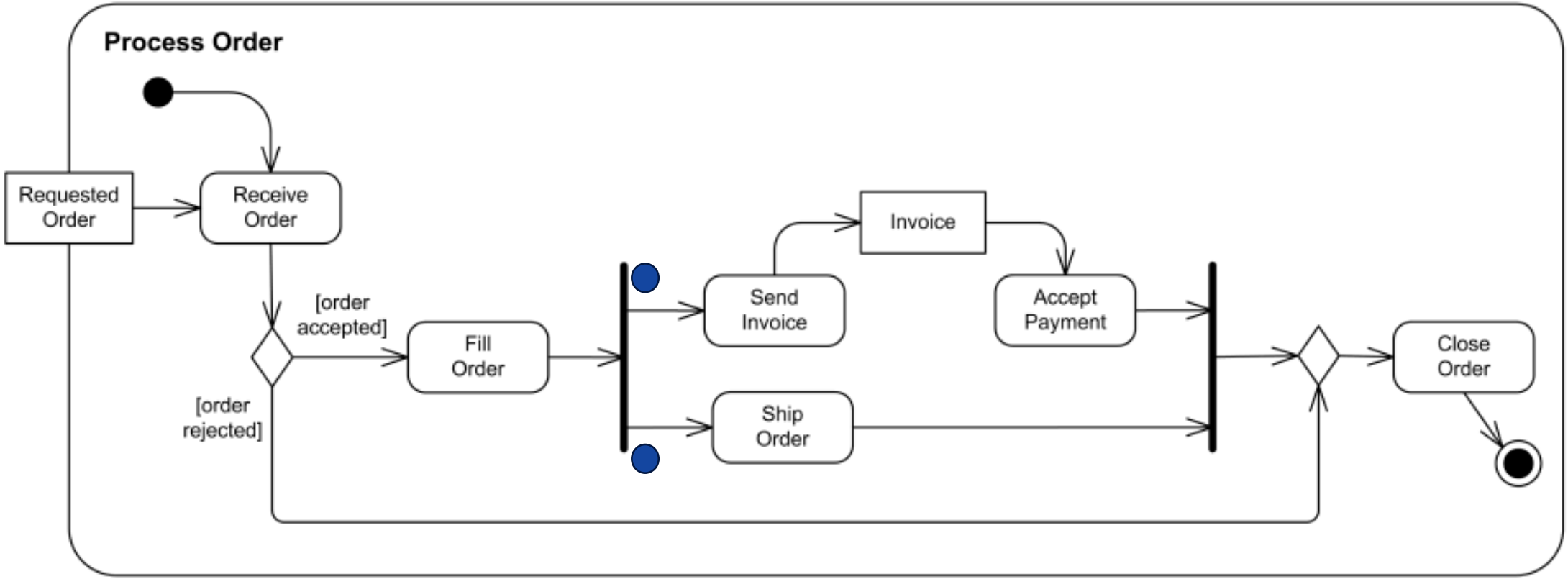
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



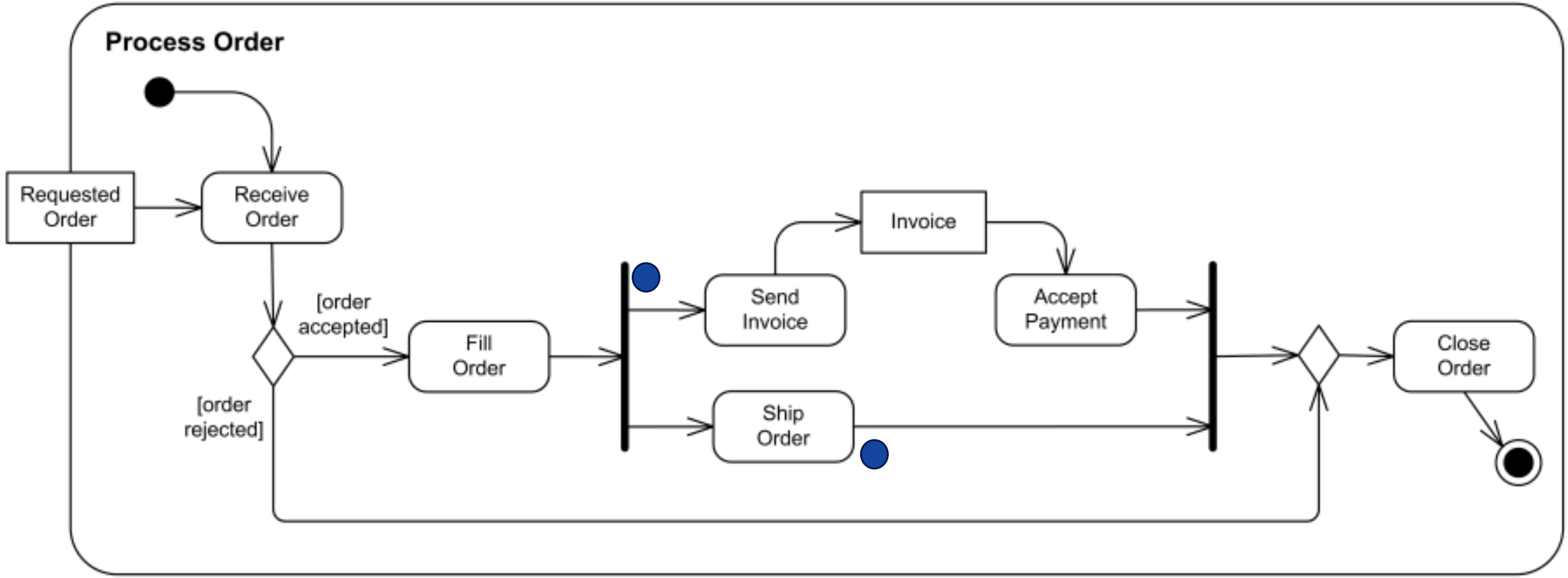
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



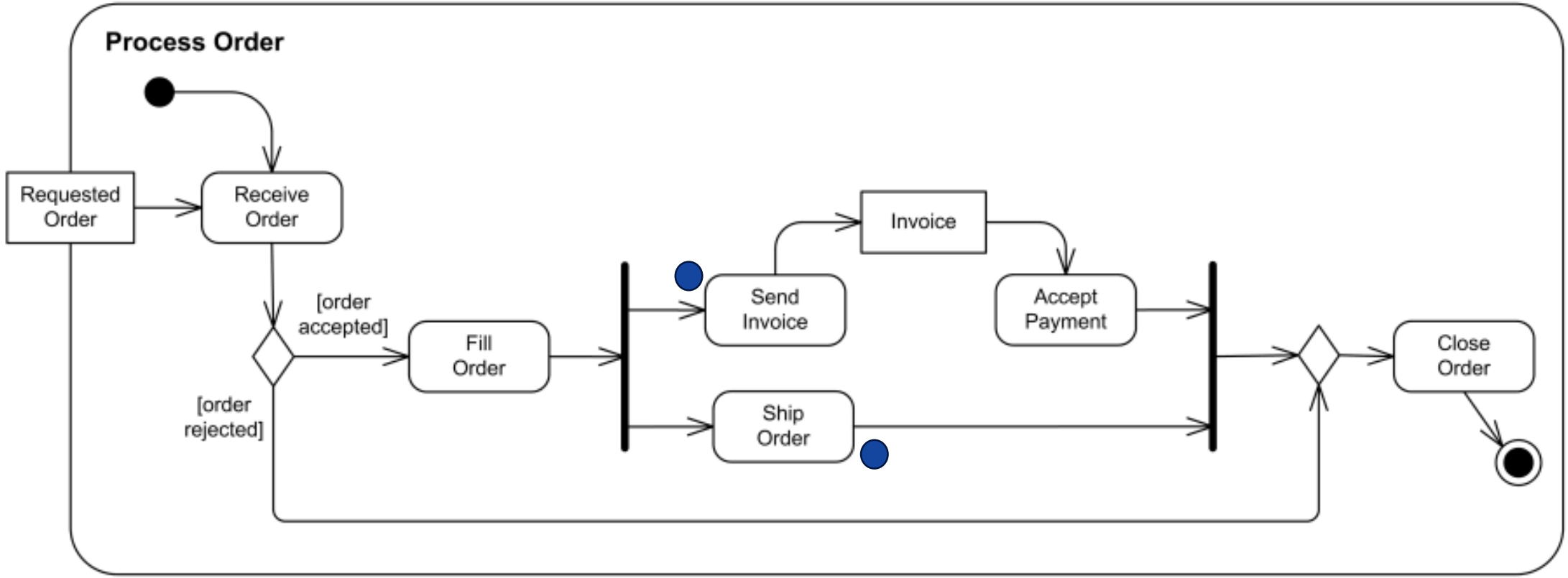
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



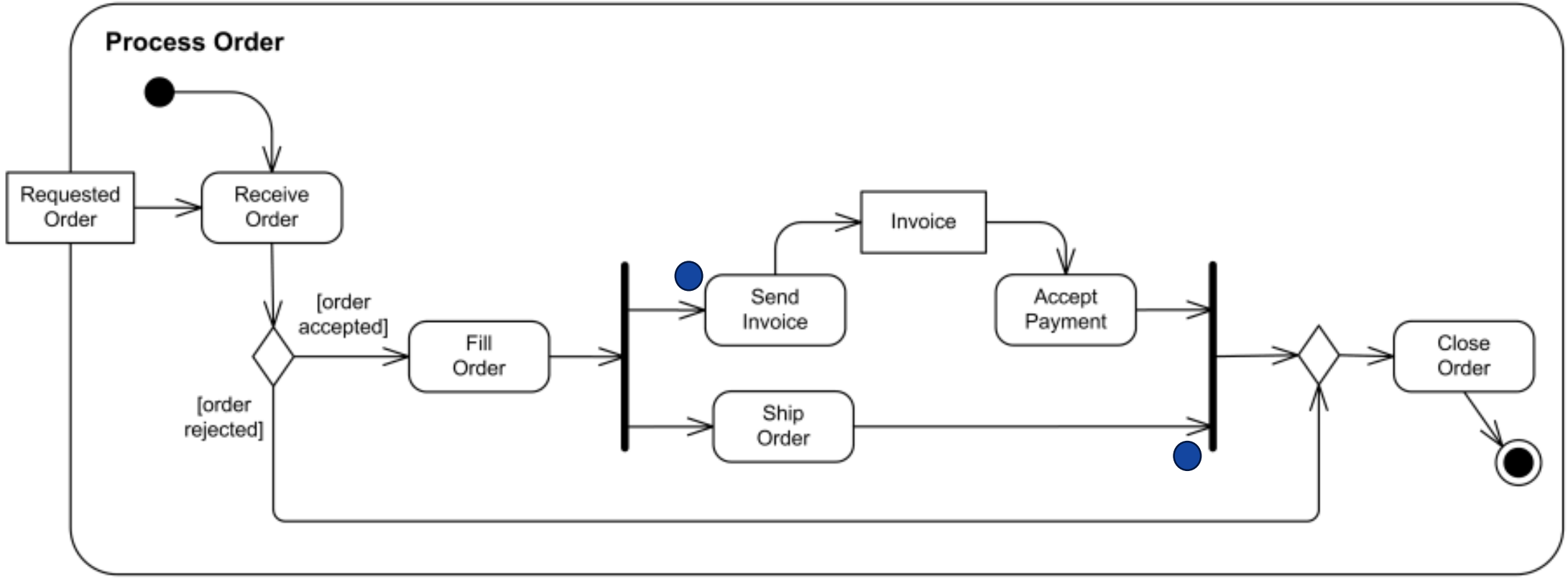
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



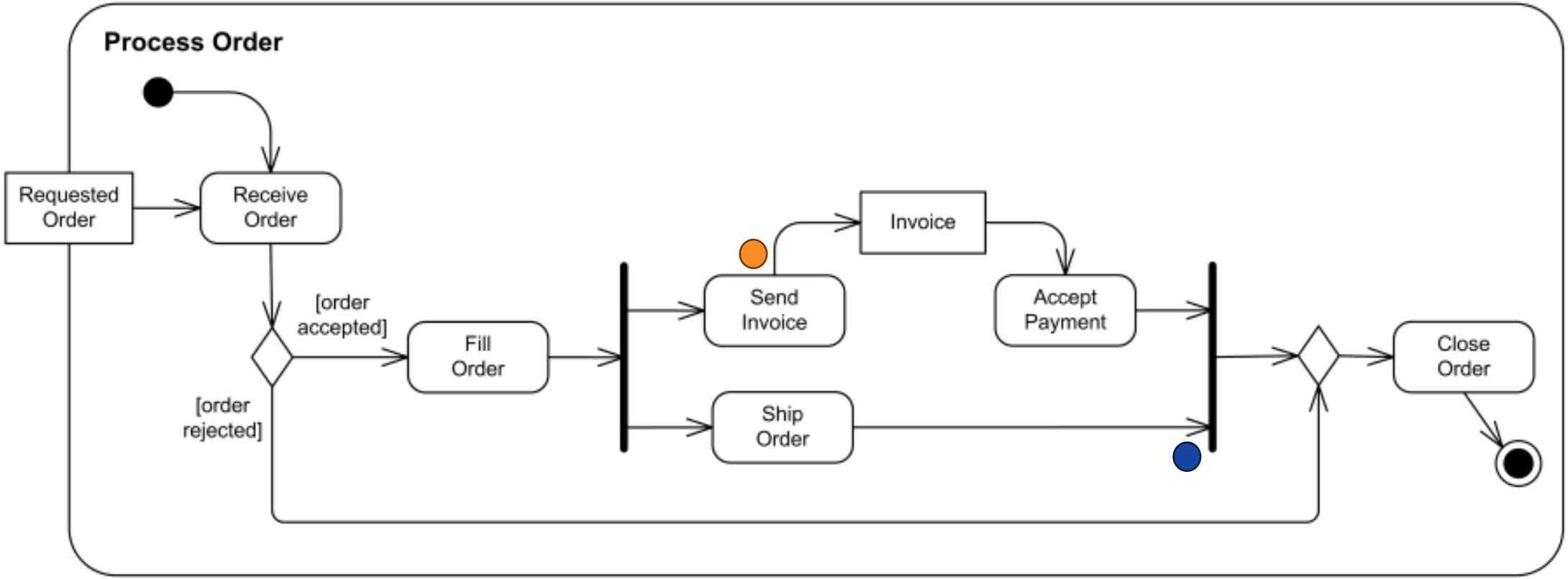
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



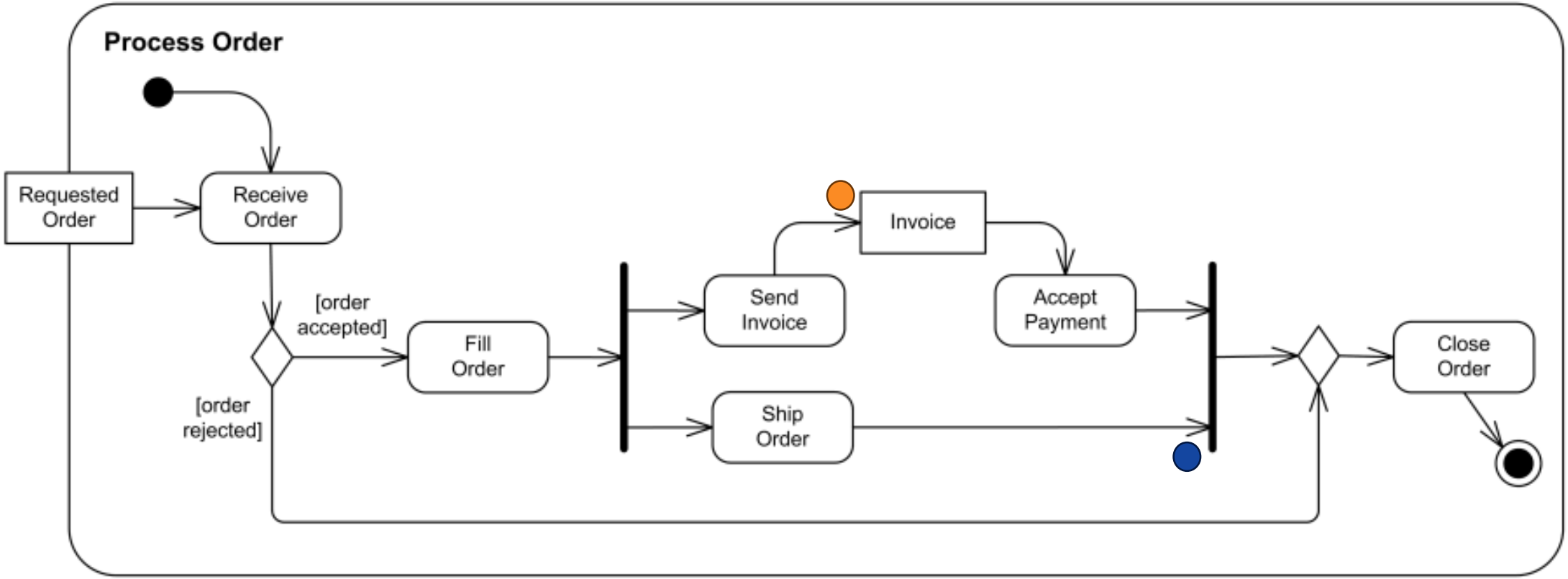
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



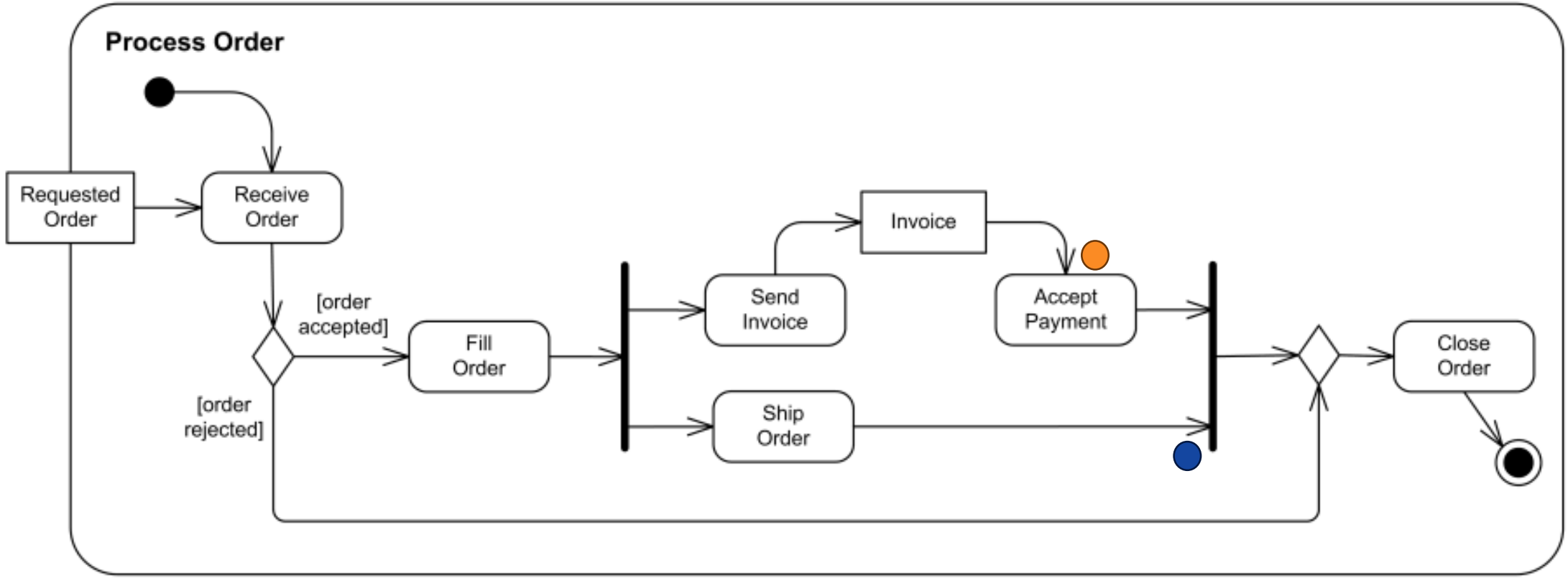
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



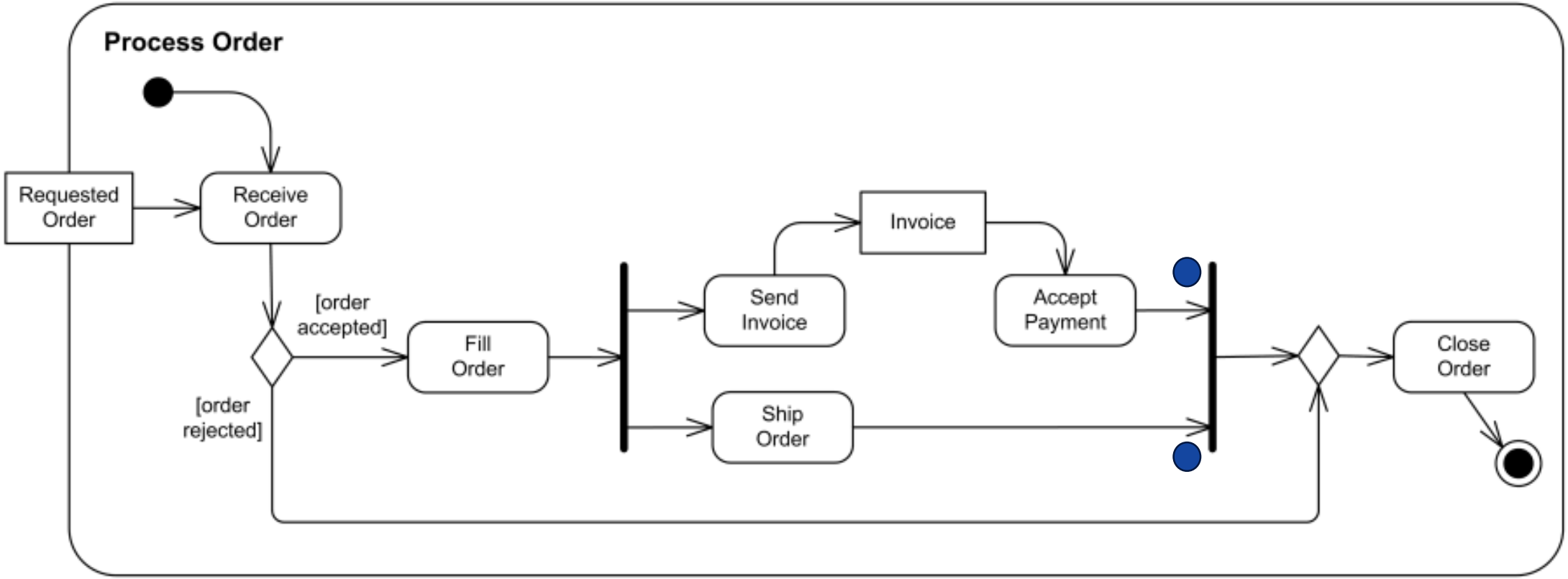
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



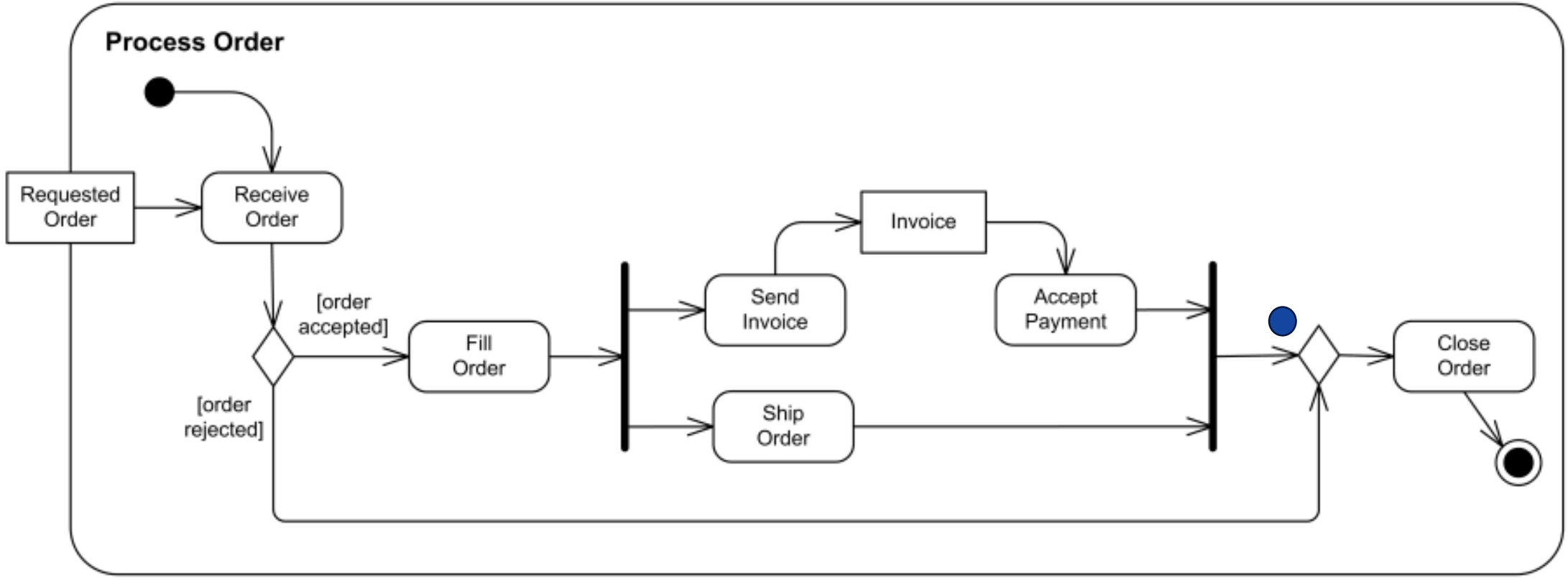
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



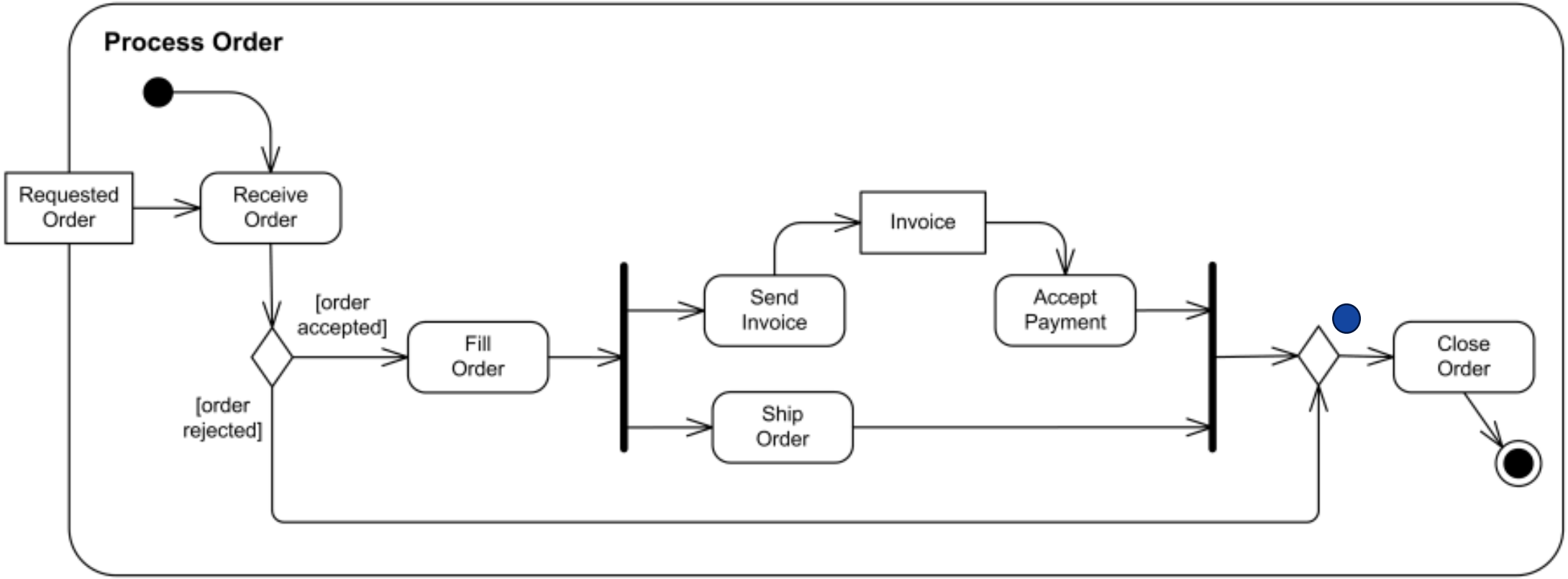
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



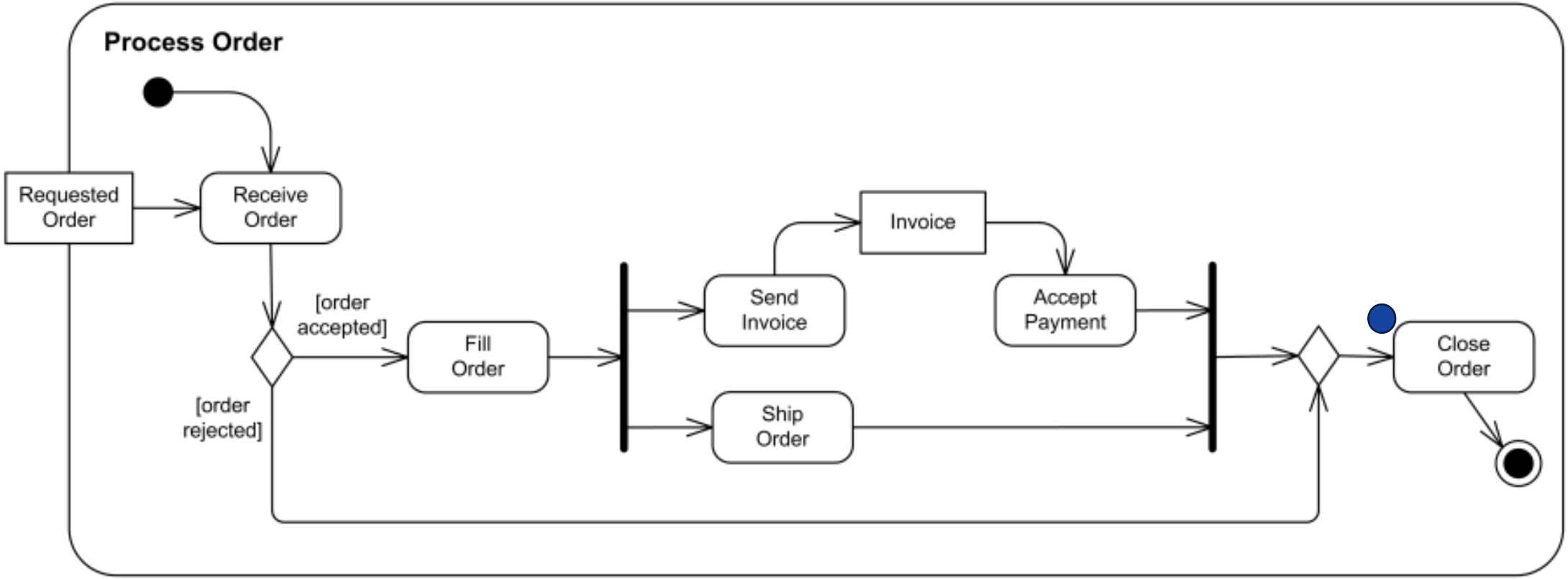
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



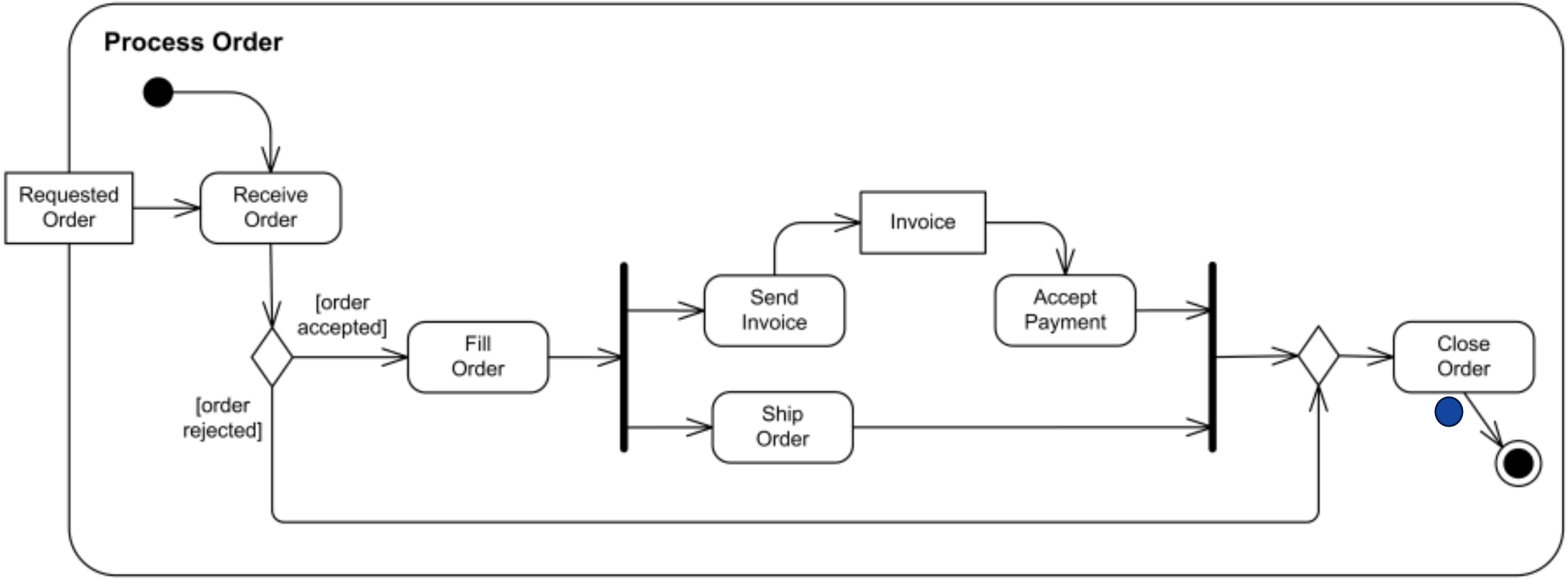
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



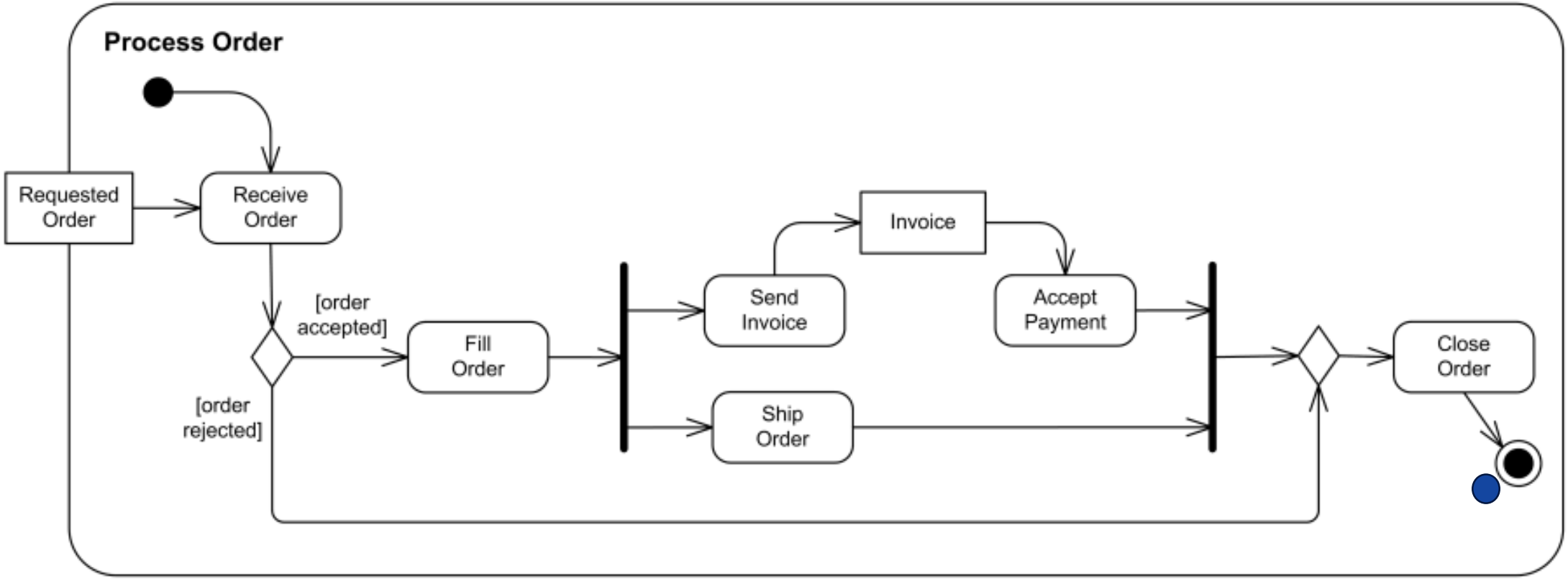
Illustrating the Execution by Tokens

- control token
- data token (Order)
- data token (Invoice)



Illustrating the Execution by Tokens

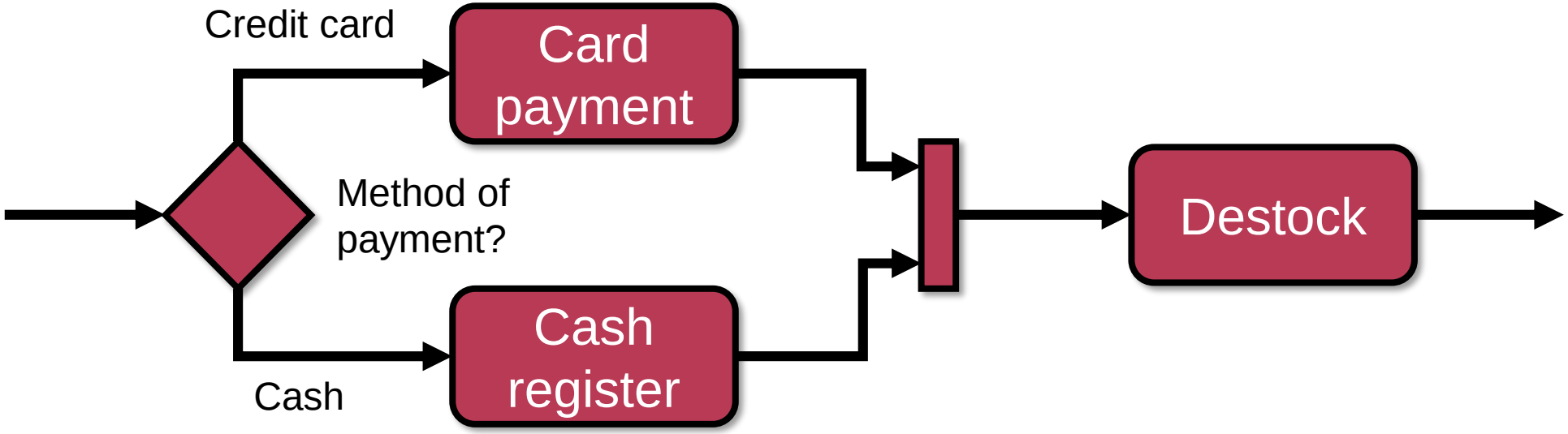
- control token
- data token (Order)
- data token (Invoice)



Advices for modelling activities

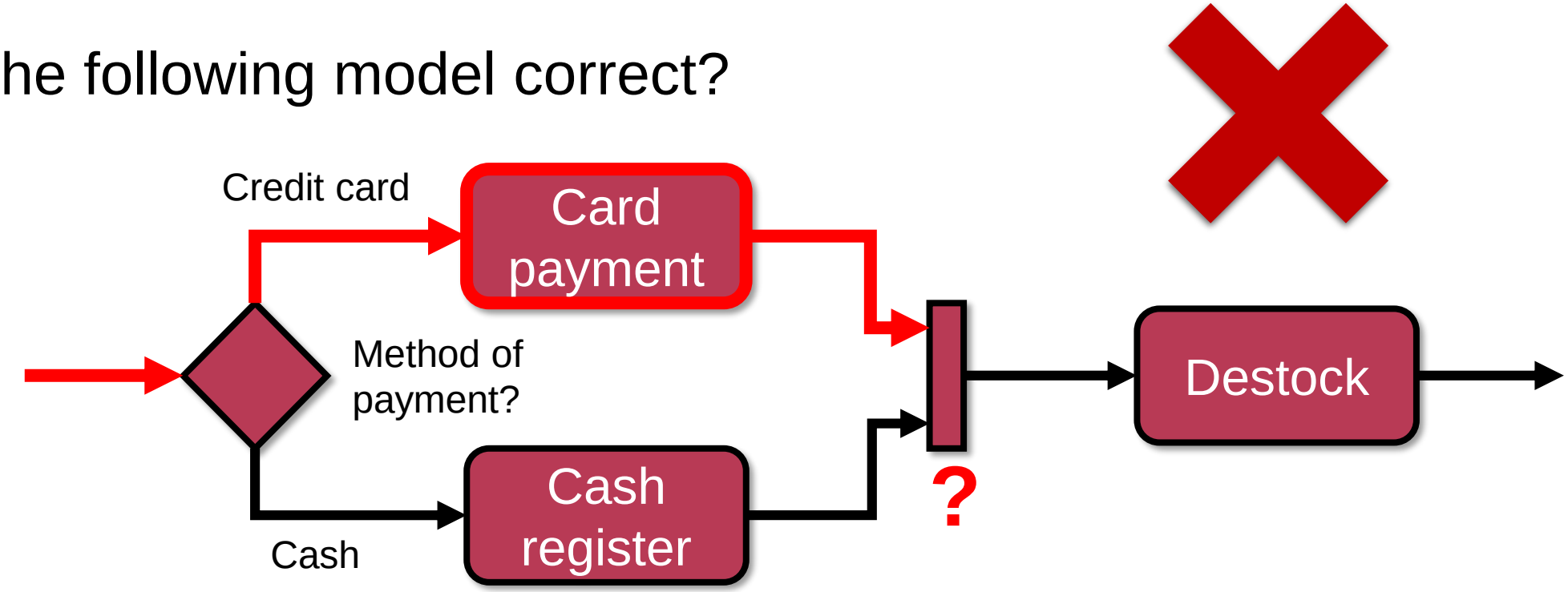
Decision and Join

- Is the following model correct?



Decision and Join

- Is the following model correct?



  - Join: only proceeds if tokens are arrived at each

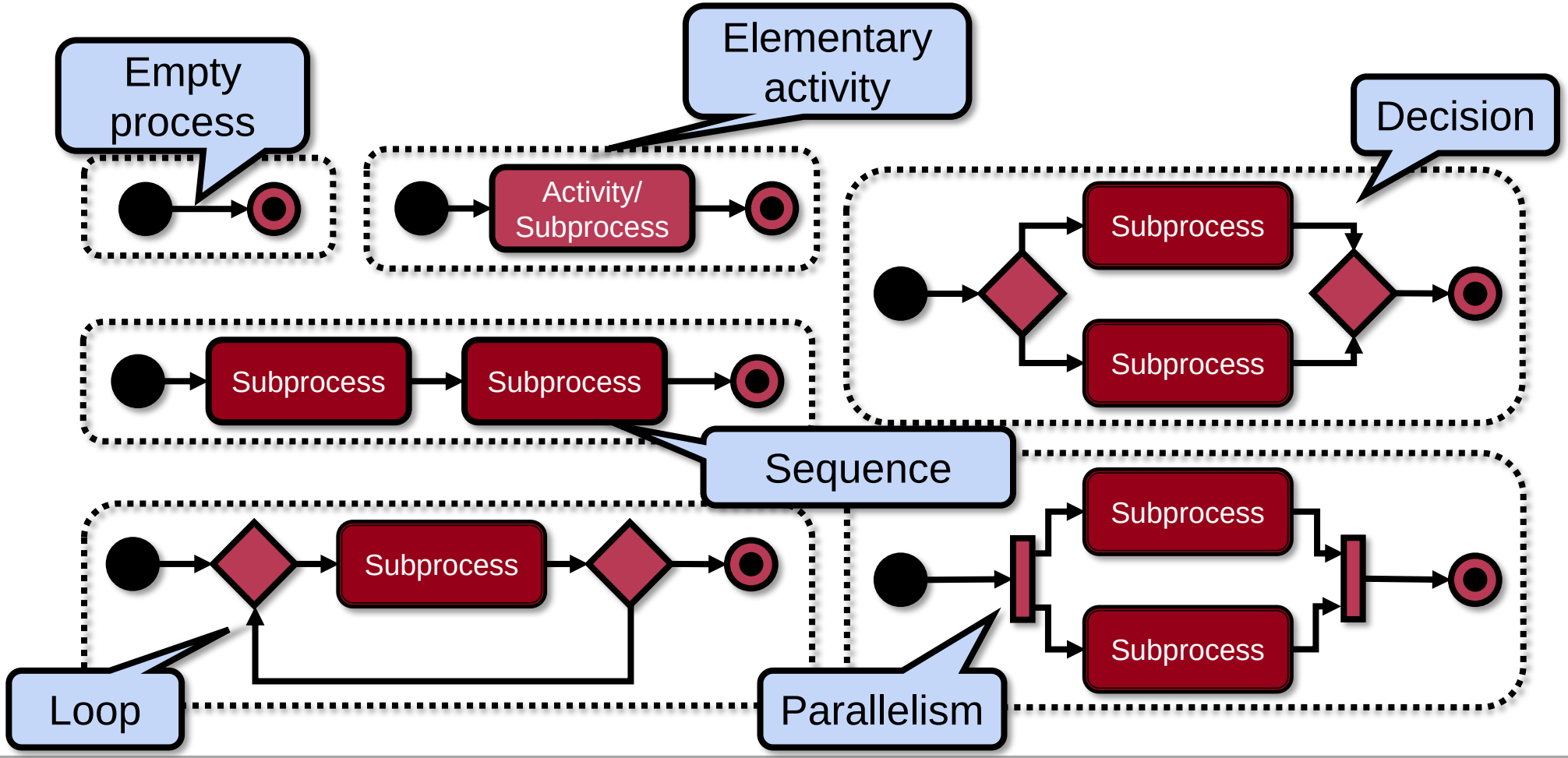
→ DEADLOCK

Typical Errors in Process Models

- Deadlock
- Decimerge, Joinfork
- Lack of synchronisation (e.g. not a join after a fork)
- Missing dependencies
- Early termination

Well Structured Process Models

- **Allowed patterns:**



Further Advices for Activities

- Start the process in the top left corner
  - Continue to the right, try to avoid right-to-left
- Guards should be complete and non-overlapping
- Add pins, if the type of the dependency is unclear (data vs. control)
- If several flow edges go out of an activity, then check:
  - The tokens will pick one of them? Add a Decision node.
  - Tokens will be put to each of them? Add a Fork node.