Contents(1)

- **Goal and Scope**
- **Basic Concepts on OOT**
  - Basic Concepts to represent the world
  - Basic Concepts for Reuse
  - Information Hiding Principle and Java Program
  - Superiority of OOT
- **Modeling Techniques**
  - Static Model: Class and Association
  - Dynamic Model: State Machine
  - Dynamic Model: Interaction Diagram
  - Concurrency Description: Active Object and Multi-thread Programming
  - Outline of UML2.0

Dynamic Model

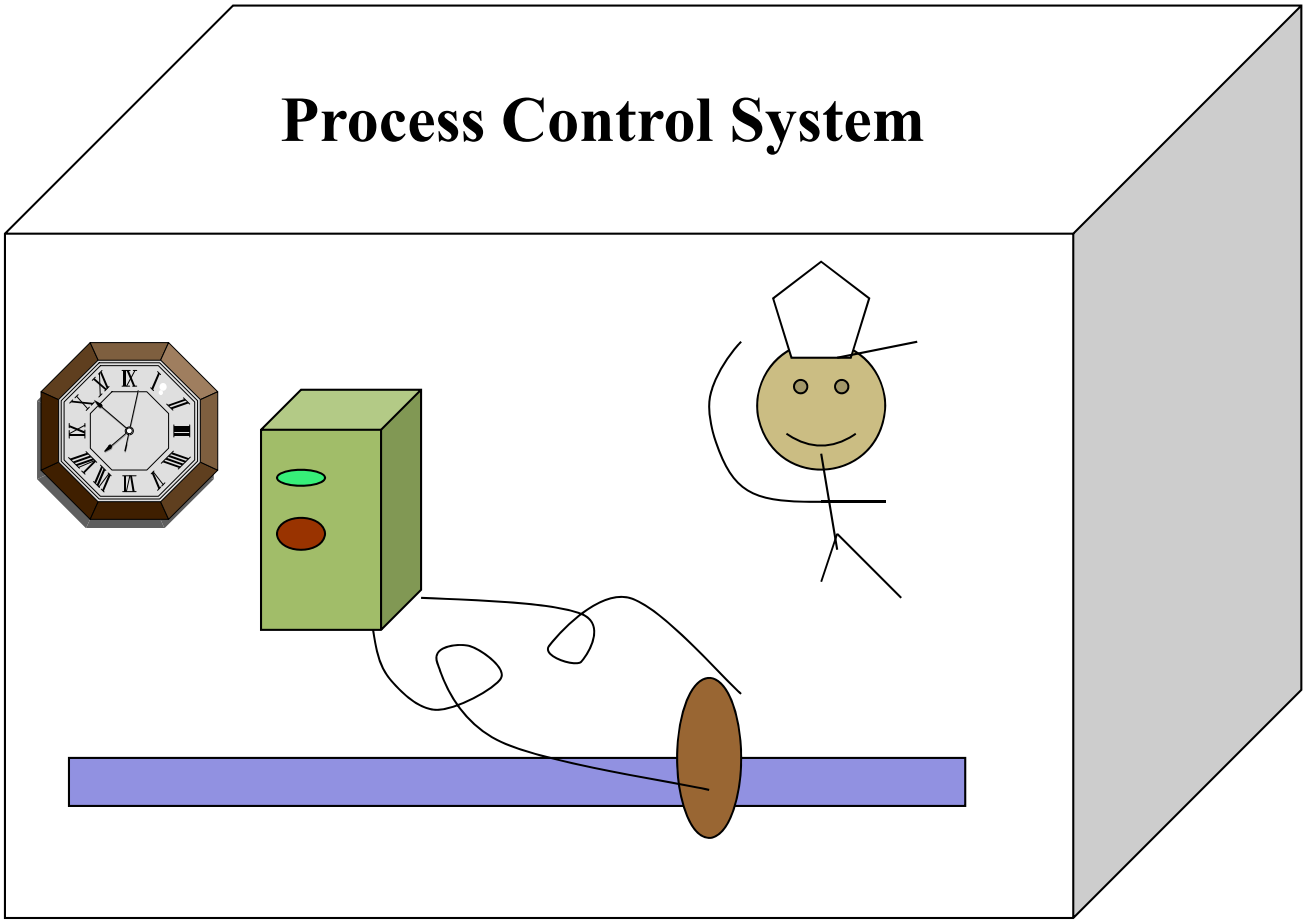
- **State Machine**
  - Describe which states an object can have during its life cycle
- **Sequence Diagram**
  - Describe how objects interact and communicate with each other
  - The primary focus in sequence diagrams is time
- **Communication Diagram**
  - Describe how objects interact
  - But the focus in a collaboration diagram is space

Object Interaction (Message)

- **Synchronous** 
  - procedure call or other nested flow of control
- **Return** 
  - return from a procedure call
- **Simple** 
  - Flat flow of control
- **Asynchronous** 
  - Asynchronous flow of control.

H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

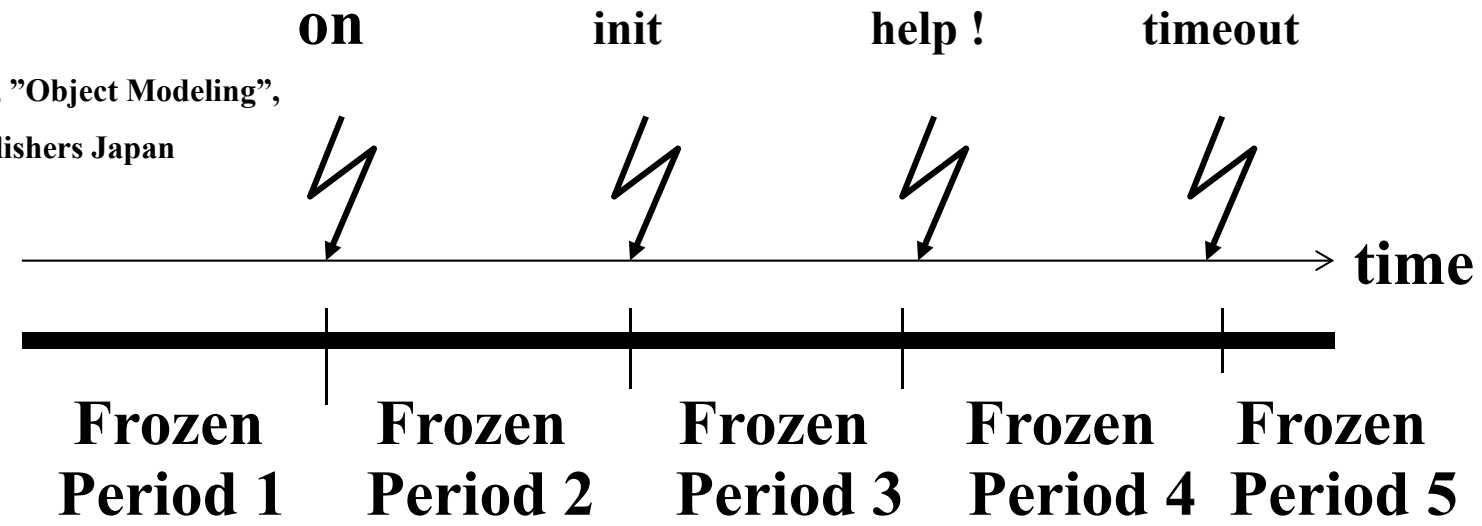
Defining behavior of the system by a state diagram

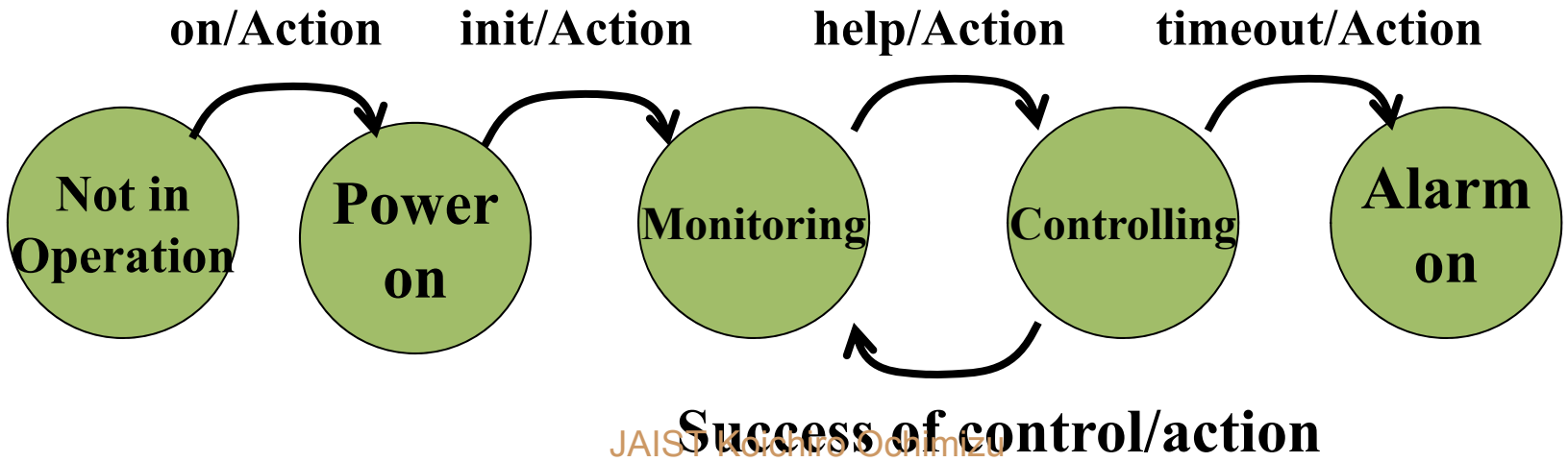


Ochimizu, Higashida, ”Object Modeling”, Addison-Wesley Publishers Japan

What is a State ?

Ochimizu, Higashida, ”Object Modeling”,
Addison-Wesley Publishers Japan

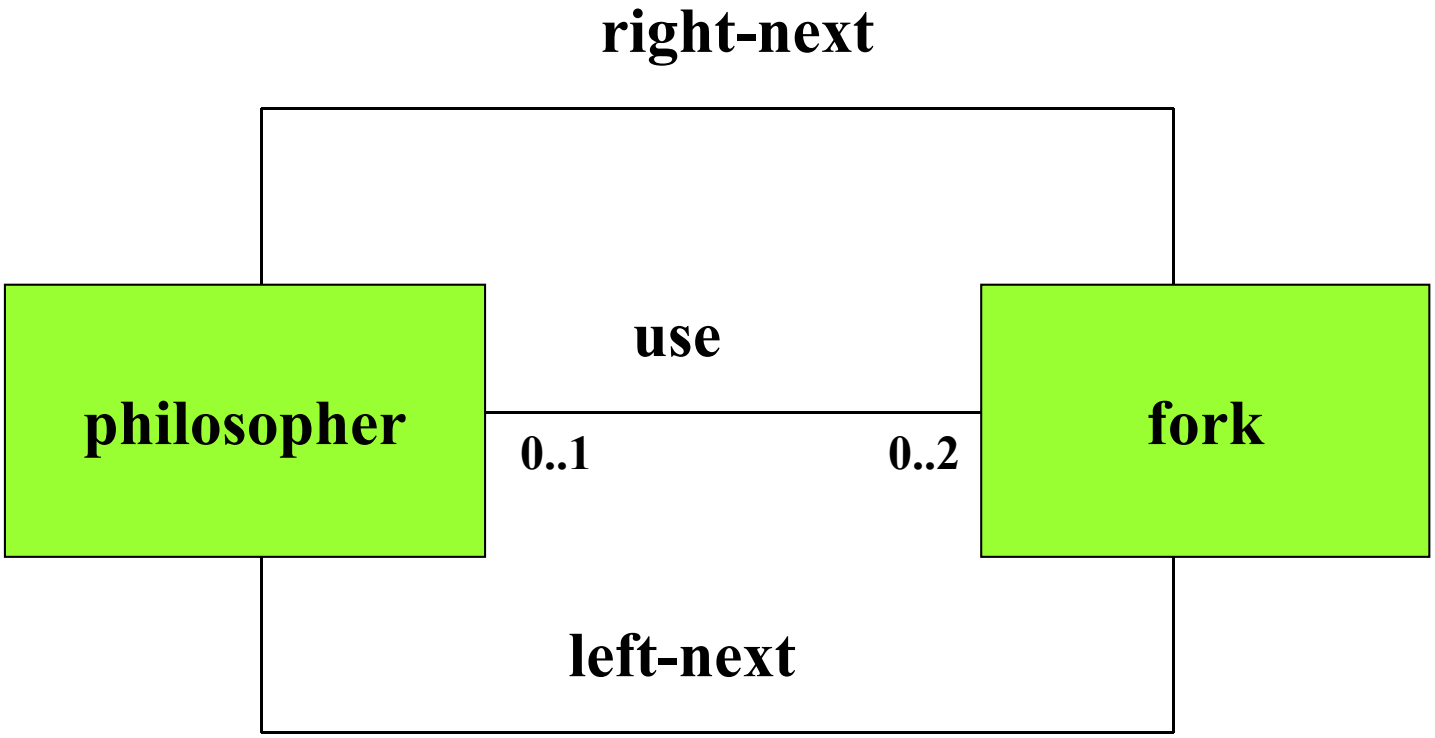




State Model

- Represent the behavior of an object by a state transition diagram.

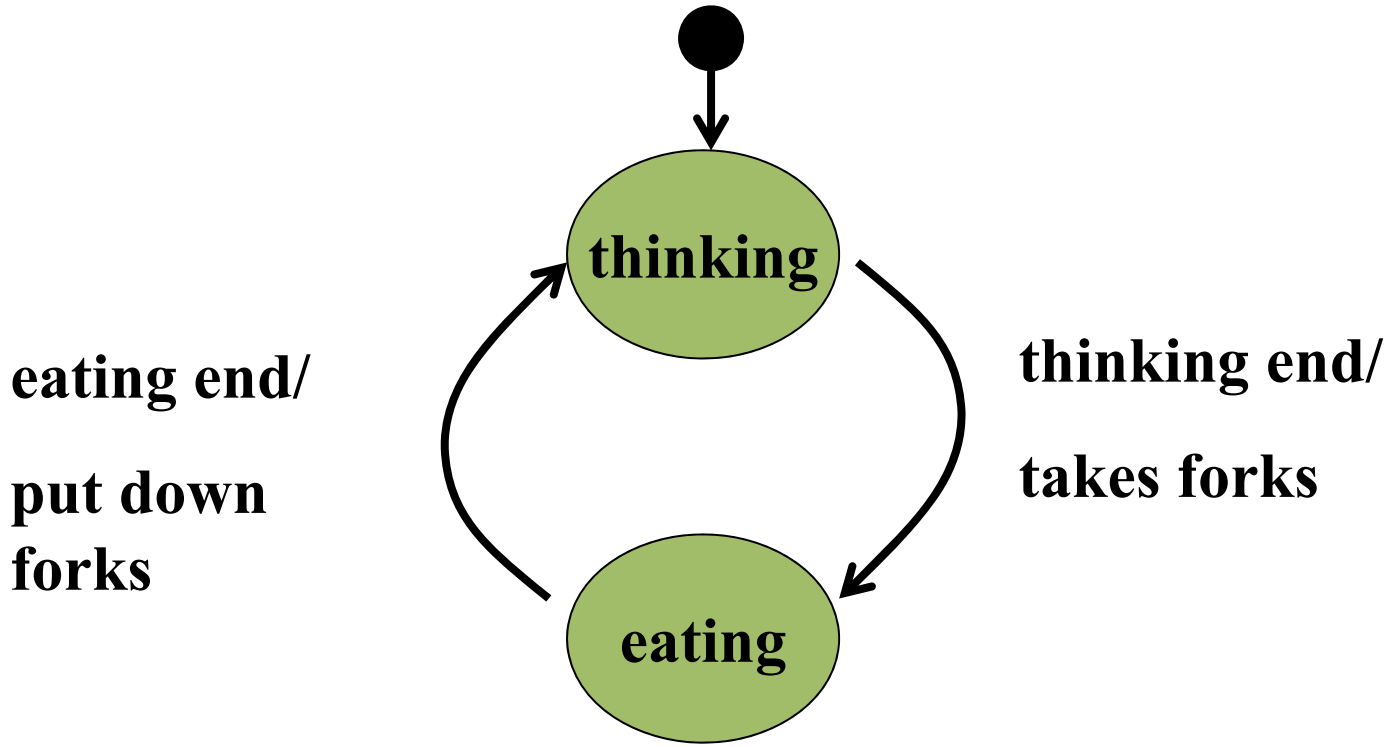
Defining the behavior of philosopher and fork



Each philosopher must use right next fork and left next one not used

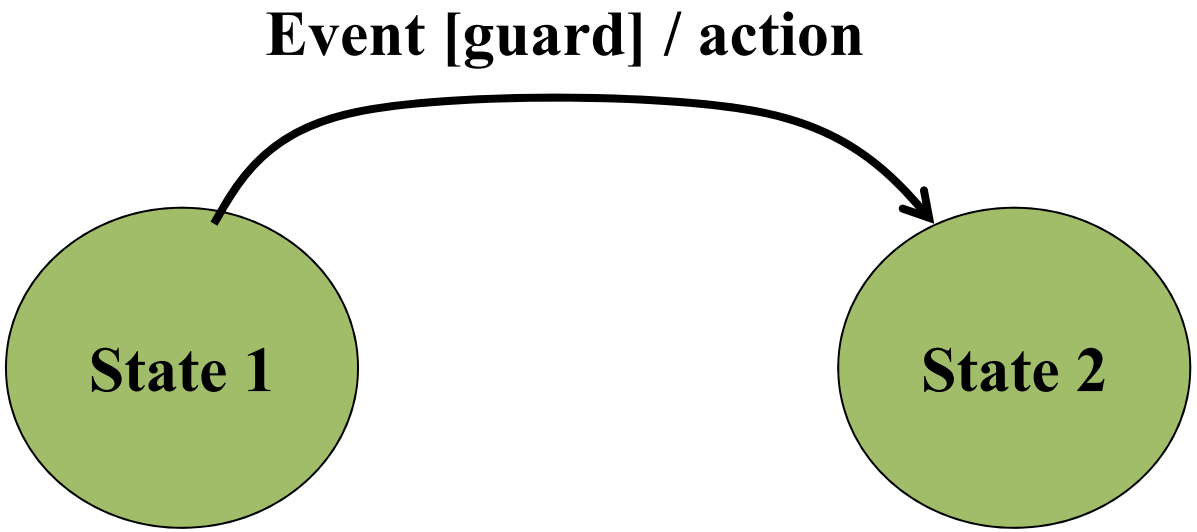
Ochimizu, Higashida,"Object Modeling", Addison-Wesley Publishers Japan

Behavior of a Philosopher



Ochimizu, Higashida, ”Object Modeling”, Addison-Wesley Publishers Japan

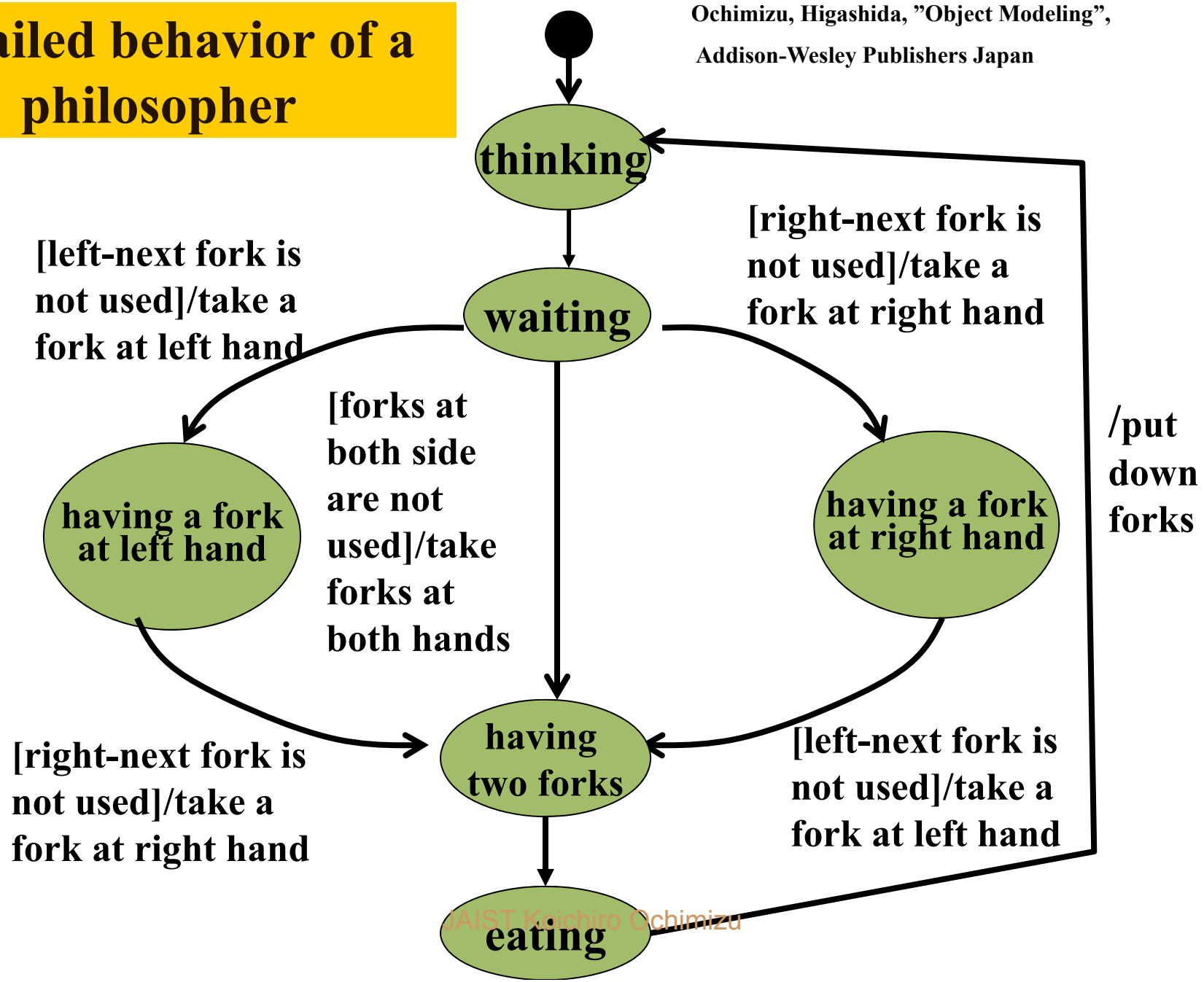
State transition depends on the state of forks both side



Ochimizu, Higashida, ”Object Modeling”, Addison-Wesley Publishers Japan

Detailed behavior of a philosopher

Ochimizu, Higashida, "Object Modeling",
Addison-Wesley Publishers Japan

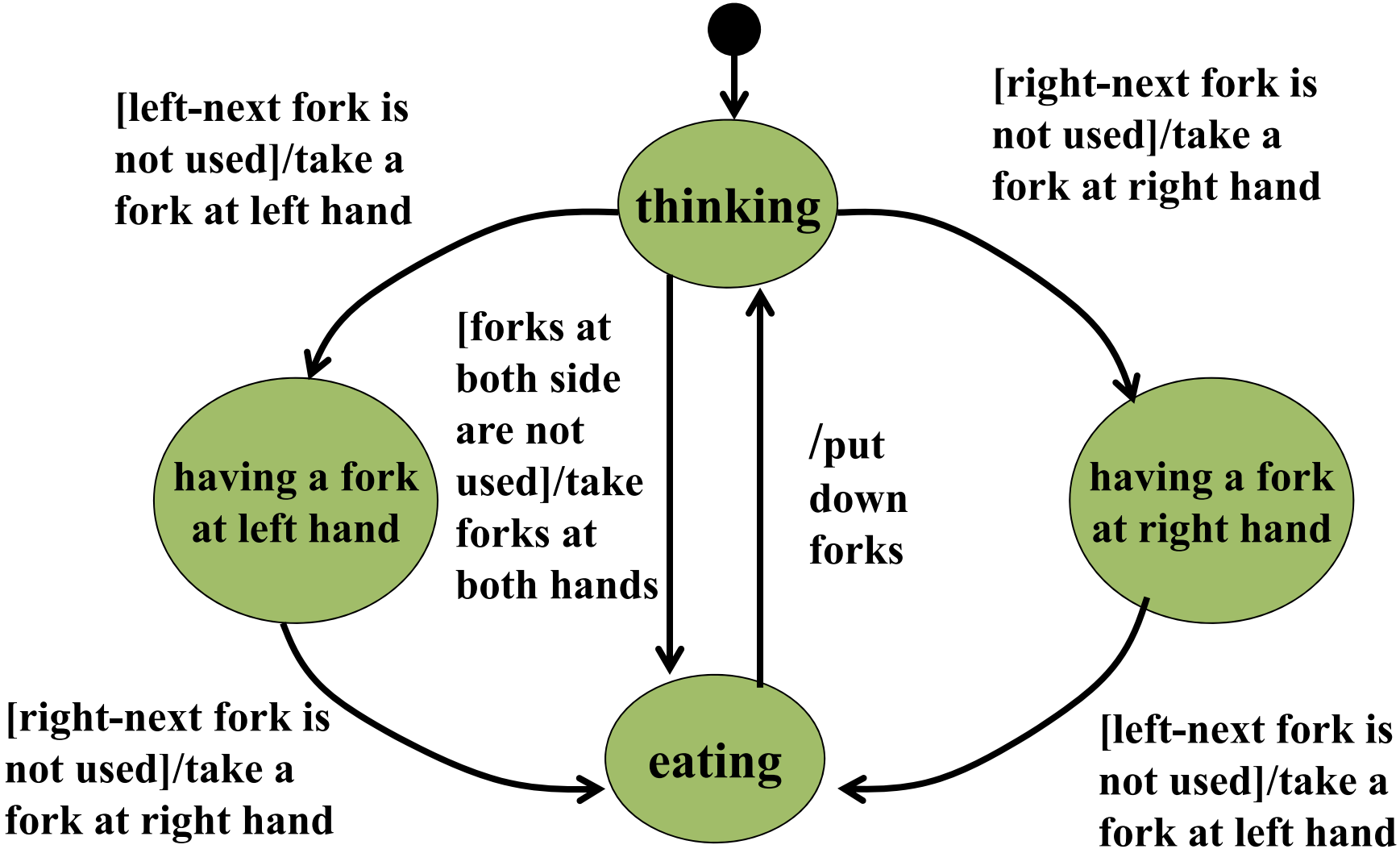


Passive Object and Active Object

- **Passive Object**
  - It is activated when it receives a message from other object (work when being hit)
- **Active Object**
  - It can change its state by itself and send a message to other object if necessary
- **In the case of philosopher**
  - **Philosopher's state of hunger changes autonomously during thinking and eating.**

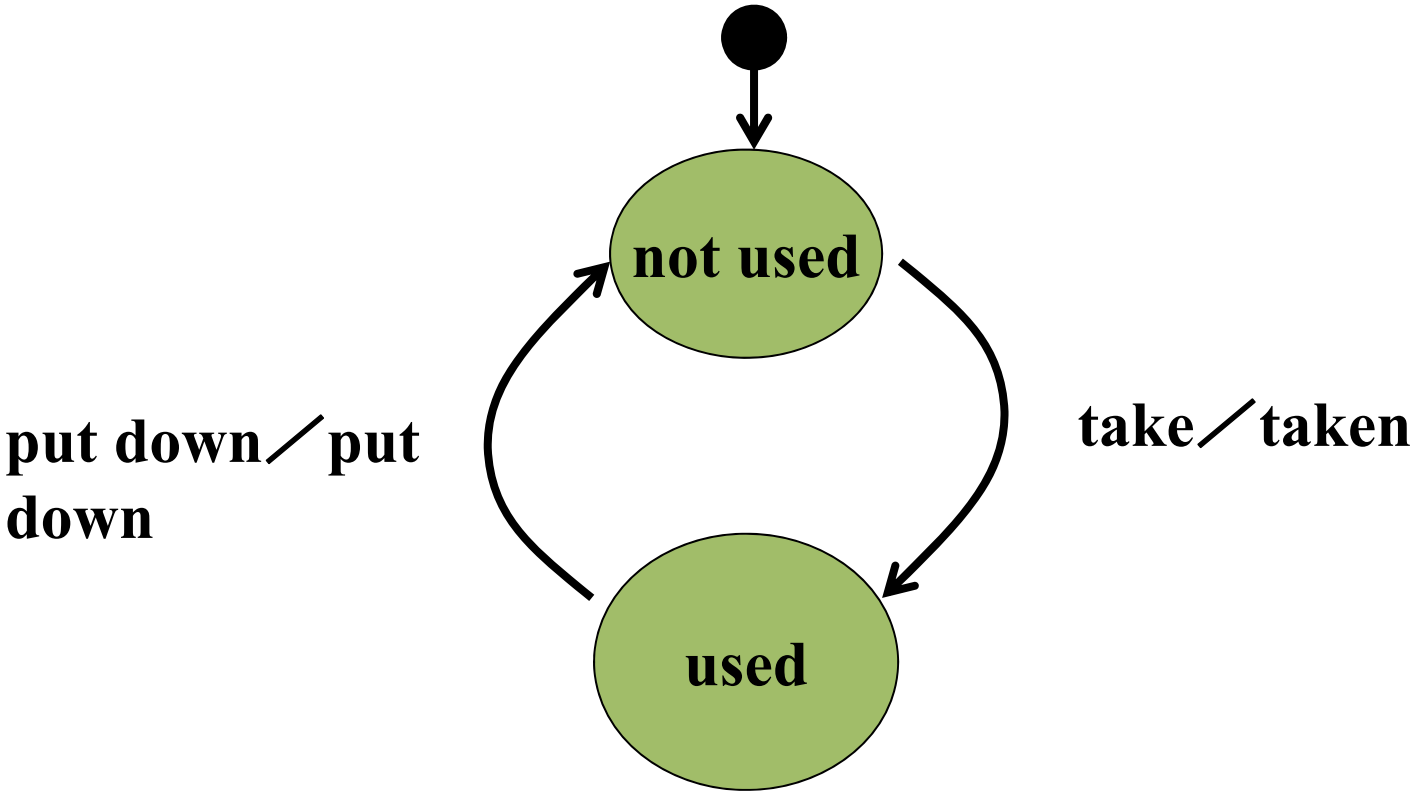
Ochimizu, Higashida, "Object Modeling", Addison-Wesley Publishers Japan

Simplified behavior



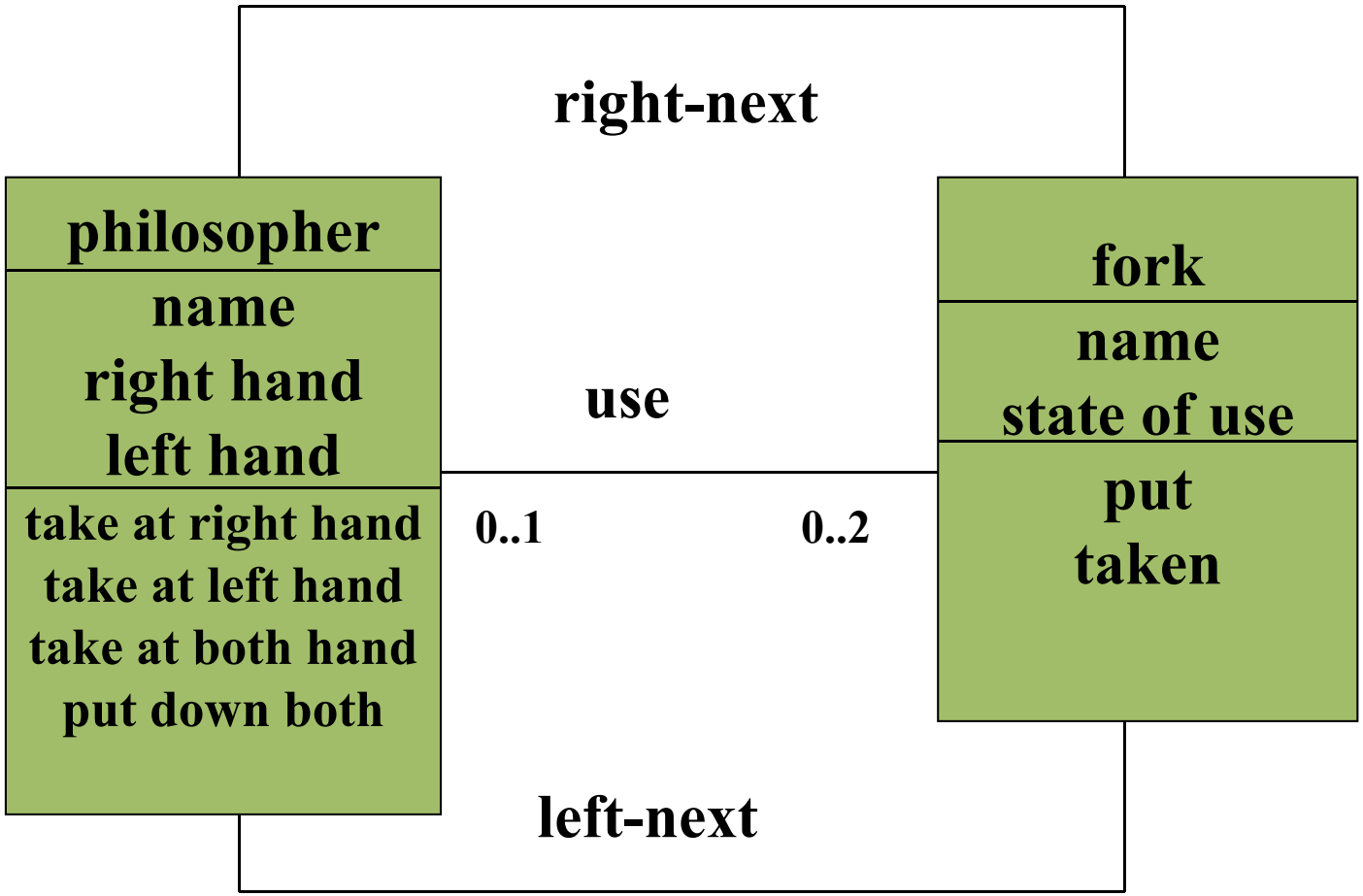
Ochimizu, Higashida, ”Object Modeling”, Addison-Wesley Publishers Japan

Defining the behavior of a fork



Ochimizu, Higashida, ”Object Modeling”, Addison-Wesley Publishers Japan

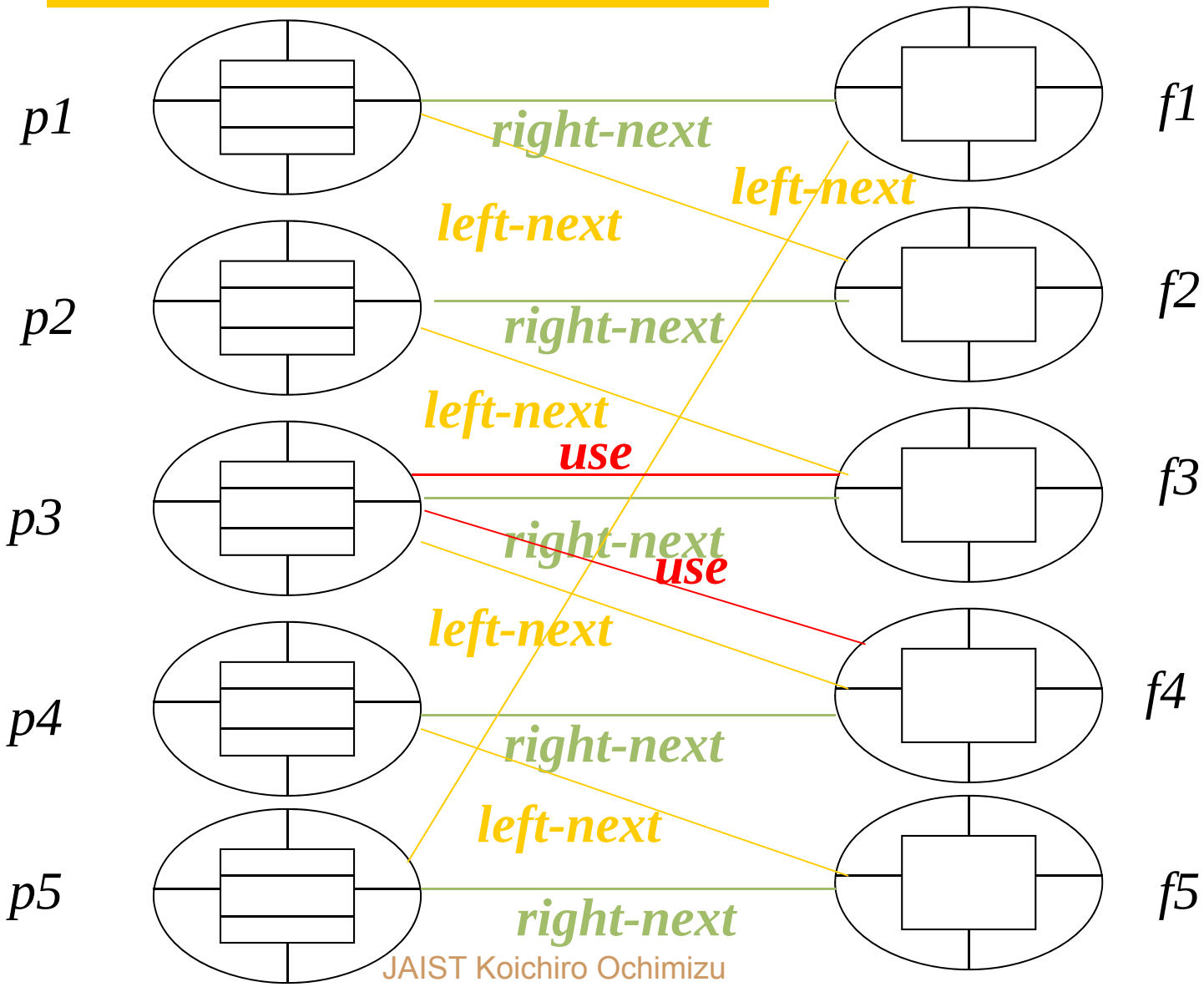
Class Diagram



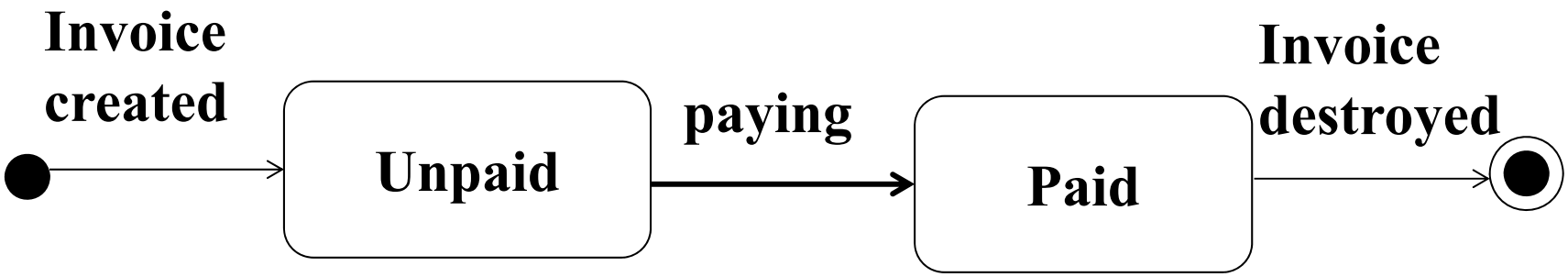
Ochimizu, Higashida, "Object Modeling", Addison-Wesley Publishers Japan

Execution

Ochimizu, Higashida, ”Object Modeling”,
Addison-Wesley Publishers Japan

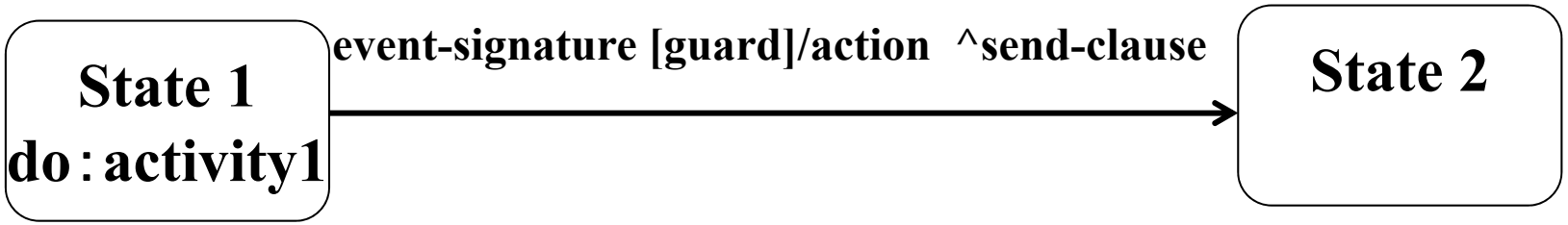


Initial State and Final State



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

State Transition



event-signature: event-name (parameter list)

draw (f: Figure, C: color)

draw()

send-clause: destination-expression . Destination-event-name (argument list)

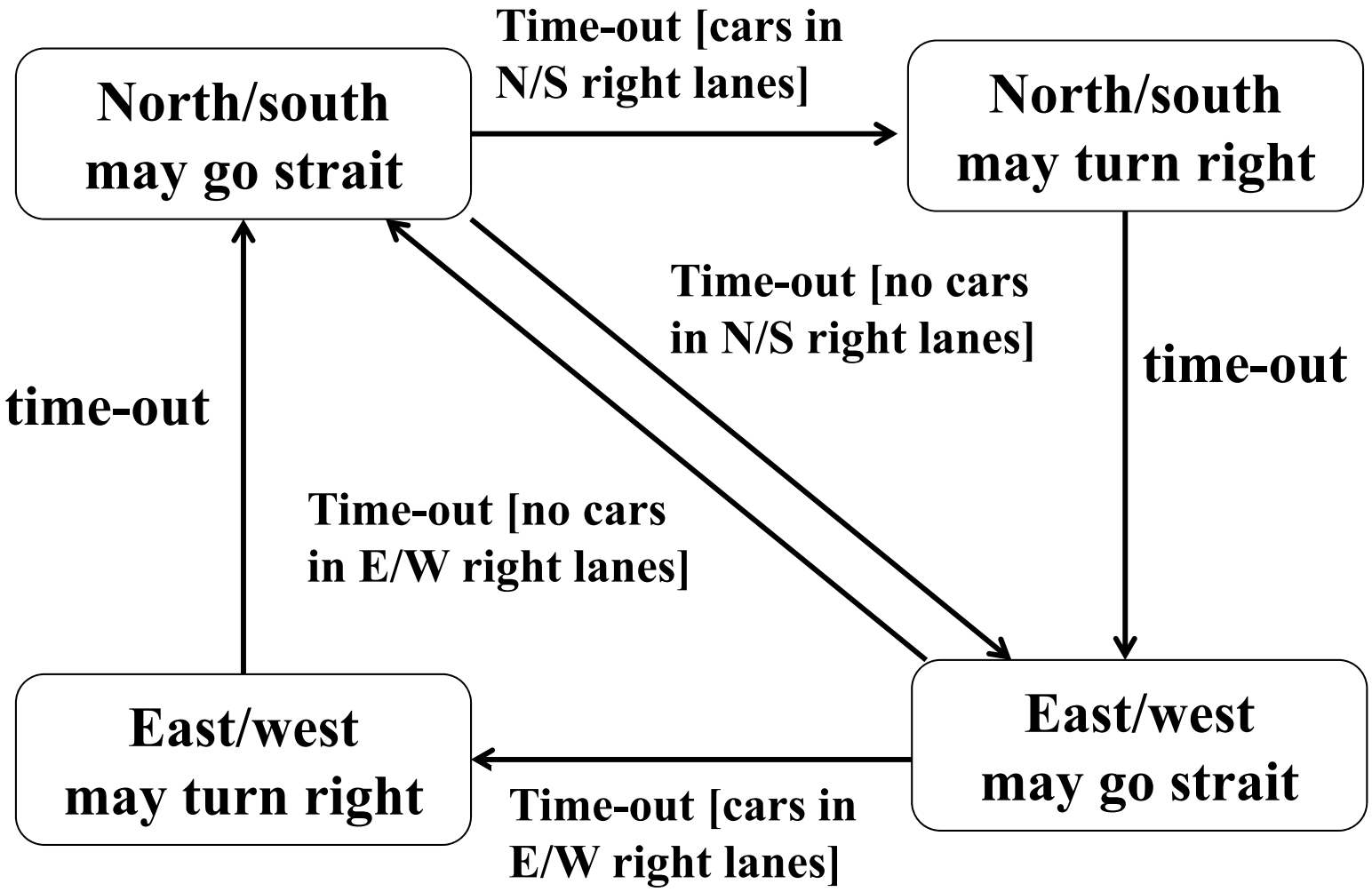
message sent during transition

[timer=time-out] ^ self.go down (first floor)

destination-expression: object or a series of objects

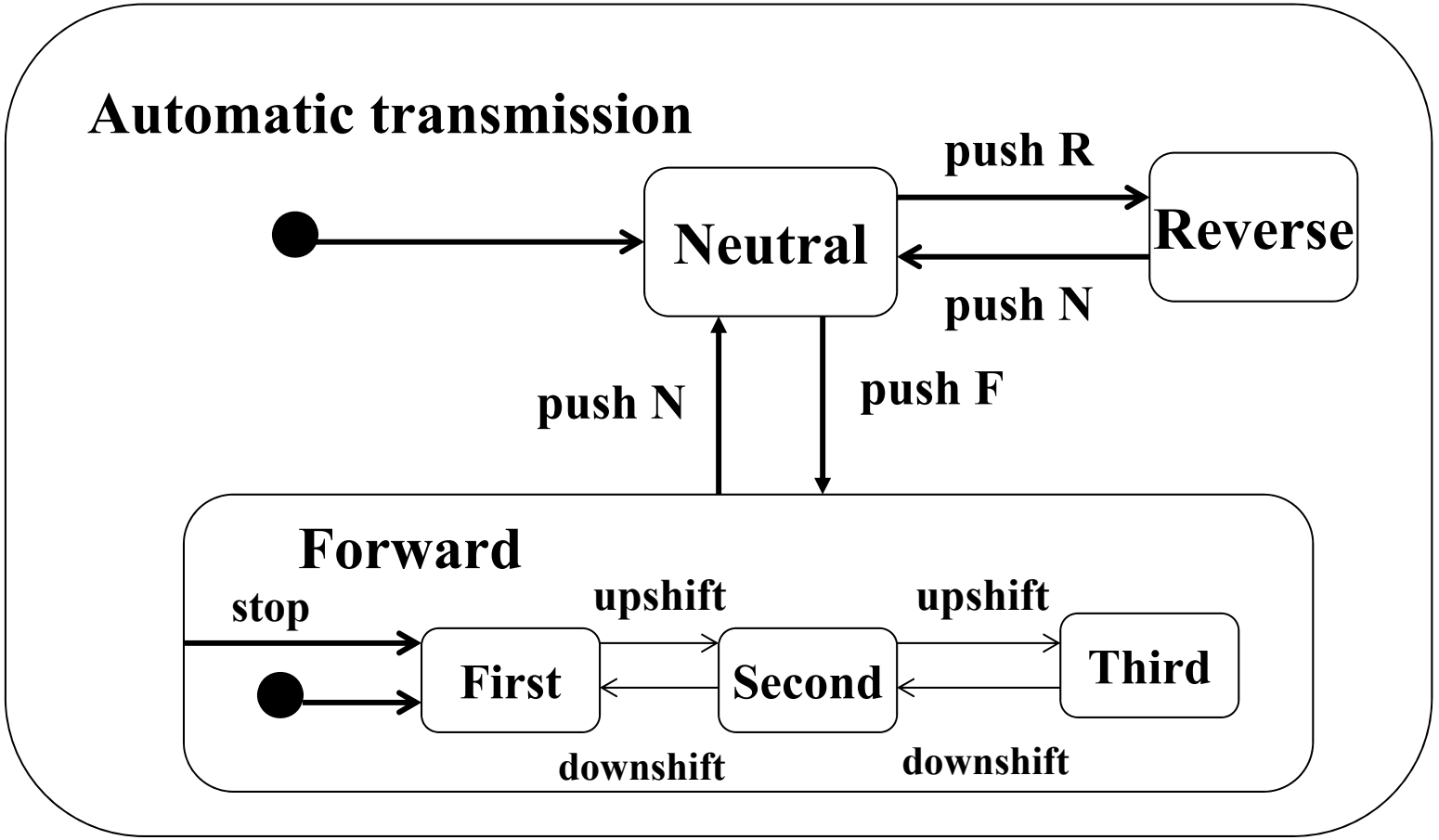
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Guard



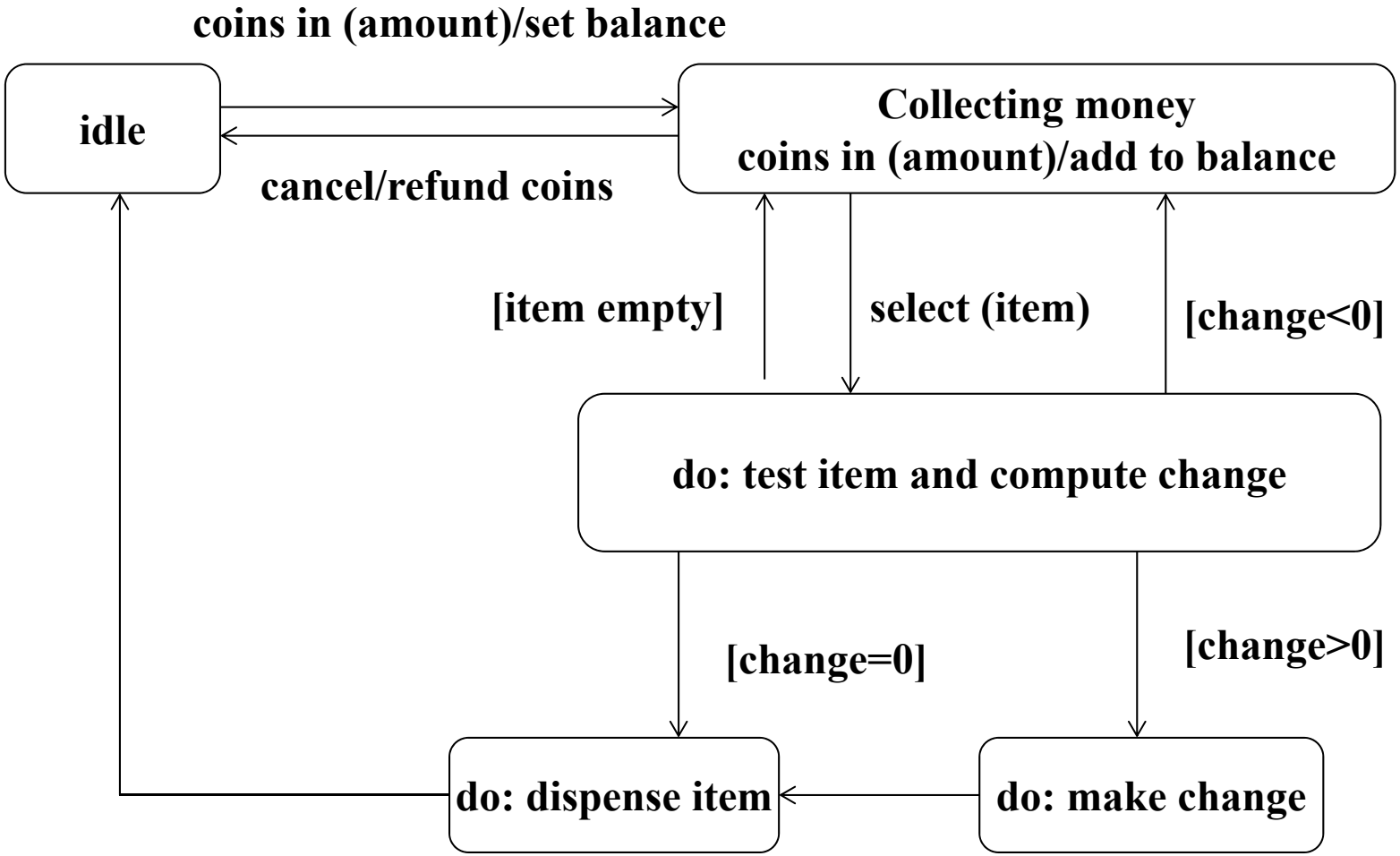
J.Rumbaugh,"Object-Oriented Modeling and Design", PrenticeHall,199.

State Generalization



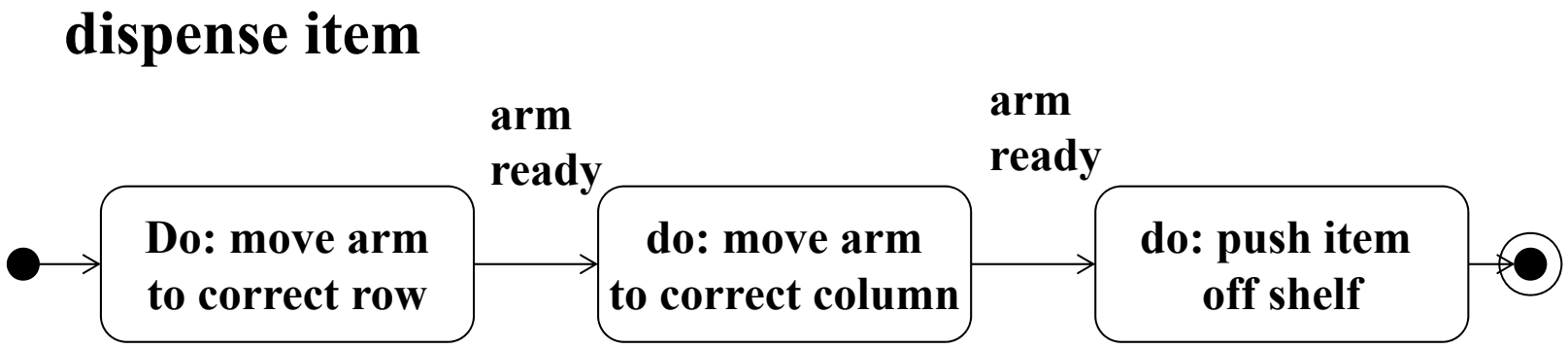
J.Rumbaugh,"Object-Oriented Modeling and Design", PrenticeHall,199.

Automatic Transition



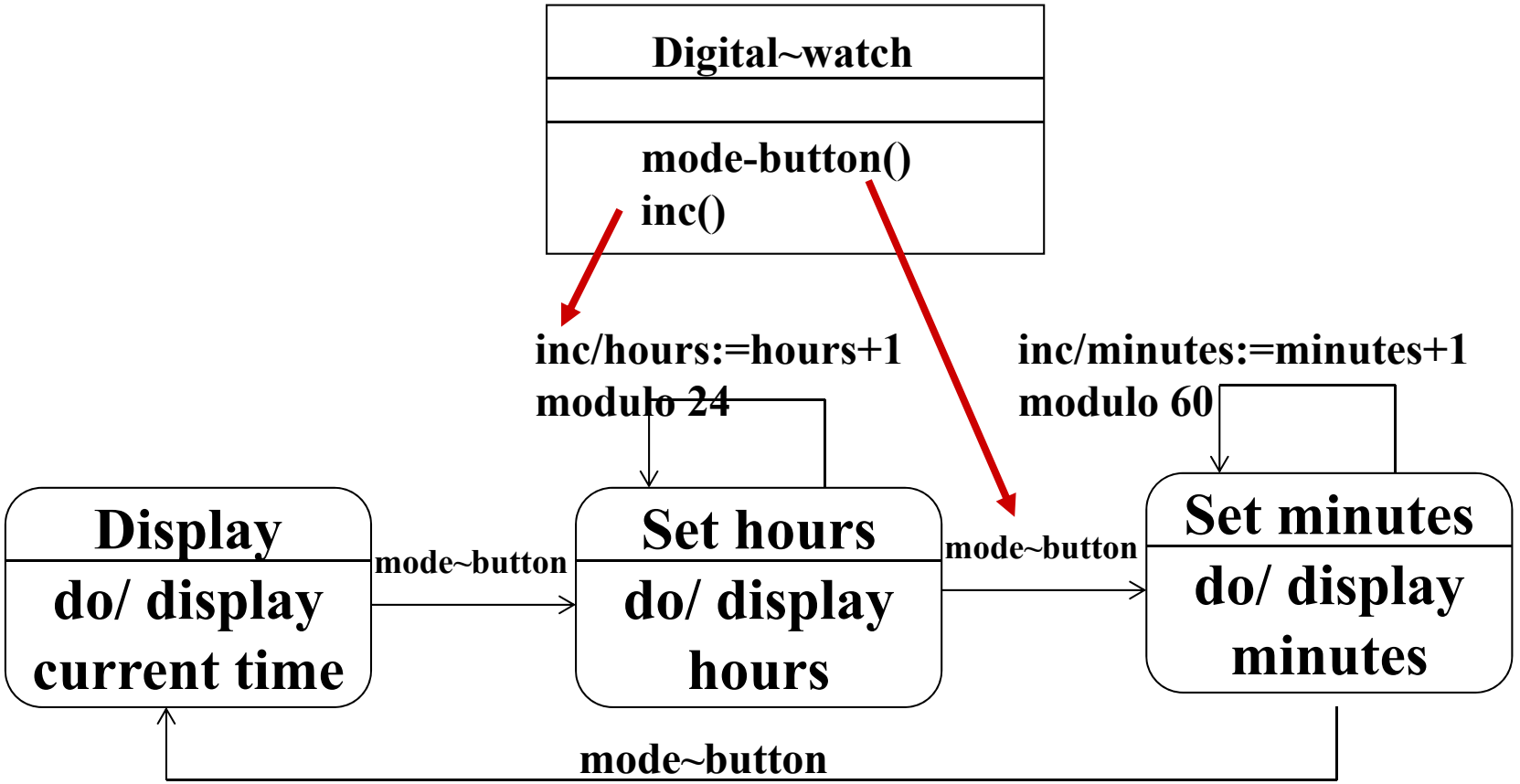
J.Rumbaugh,"Object-Oriented Modeling and Design", PrenticeHall,199.

Sub diagram



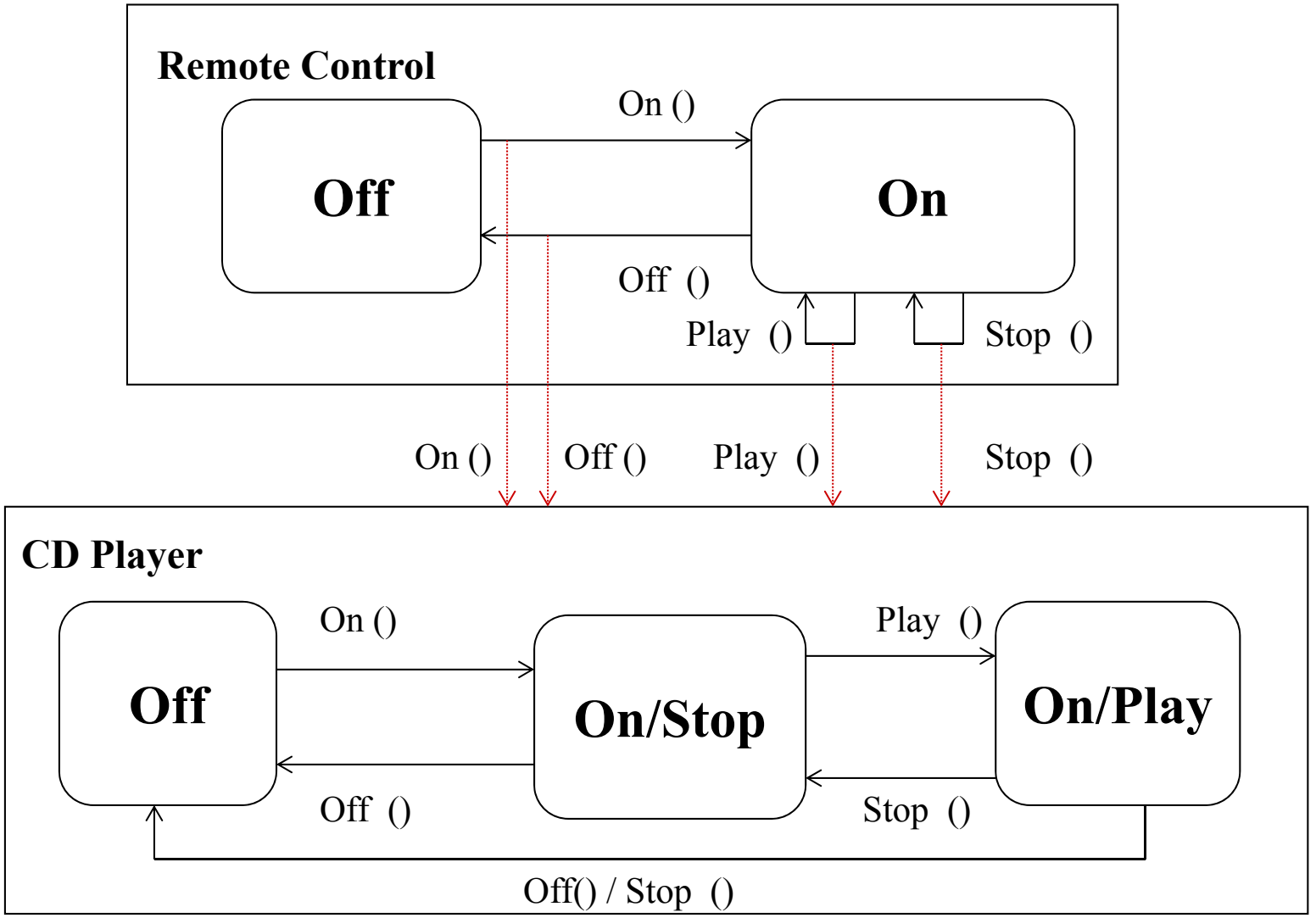
J.Rumbaugh,"Object-Oriented Modeling and Design", PrenticeHall,199.

Events in the state diagram correspond with operations within the class



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

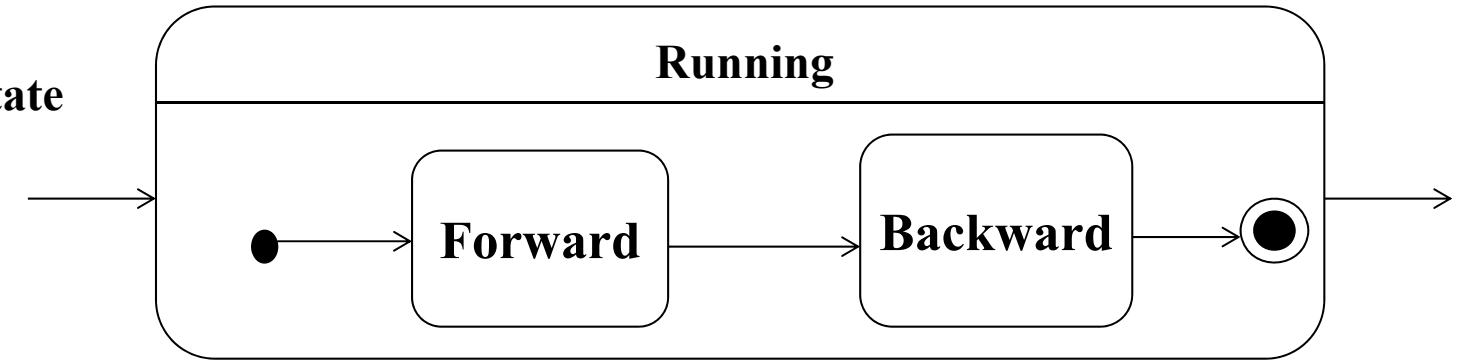
Message passing between two state diagram



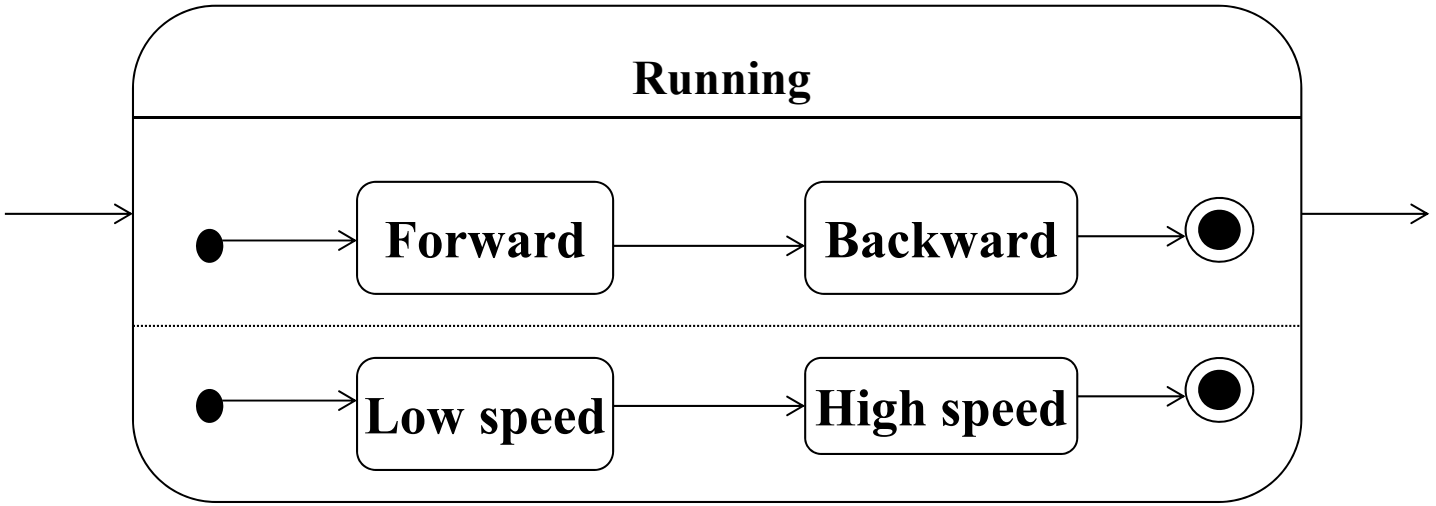
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Or-substate and And-substate

Or-substate



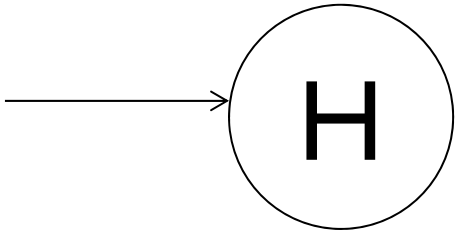
And-substate



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

History Indicator

- is used to memorize internal state
- Support roll-back
- If a transition to the indicator fires, the object resumes the state it had last within that region
- Shown as a circle with an H



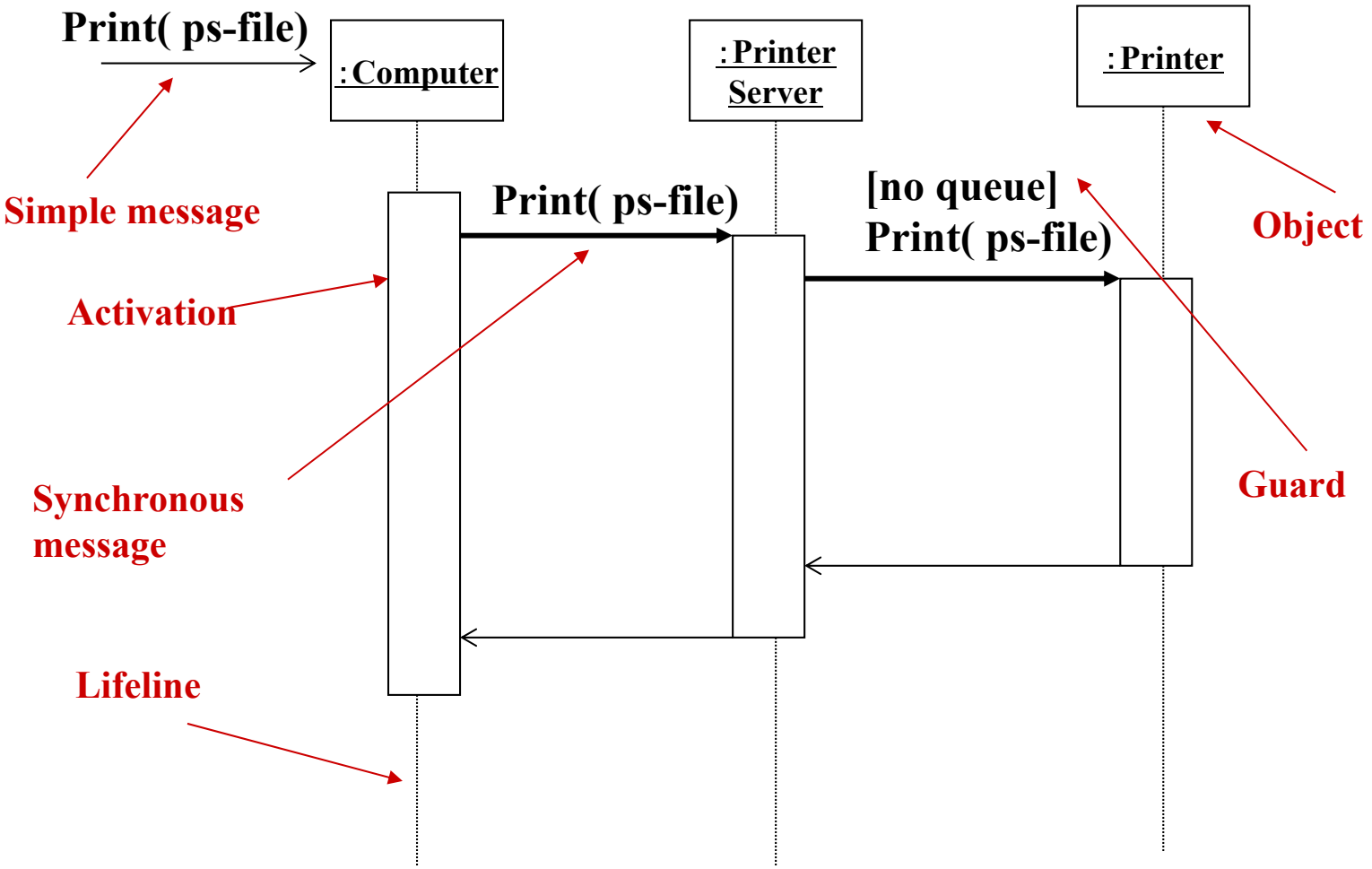
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Sequence Diagram

- Sequence diagrams illustrate how objects interact with each other.
- They focus on message sequences, that is, how messages are sent and received between a number of objects.
- Sequence diagrams have two axes: the vertical axis shows time and the horizontal axis shows a set of objects.
- The instance form describes a specific scenario in detail
- The generic form describes all possible alternatives in a scenario, therefore branches, conditions, and loops may be included.:

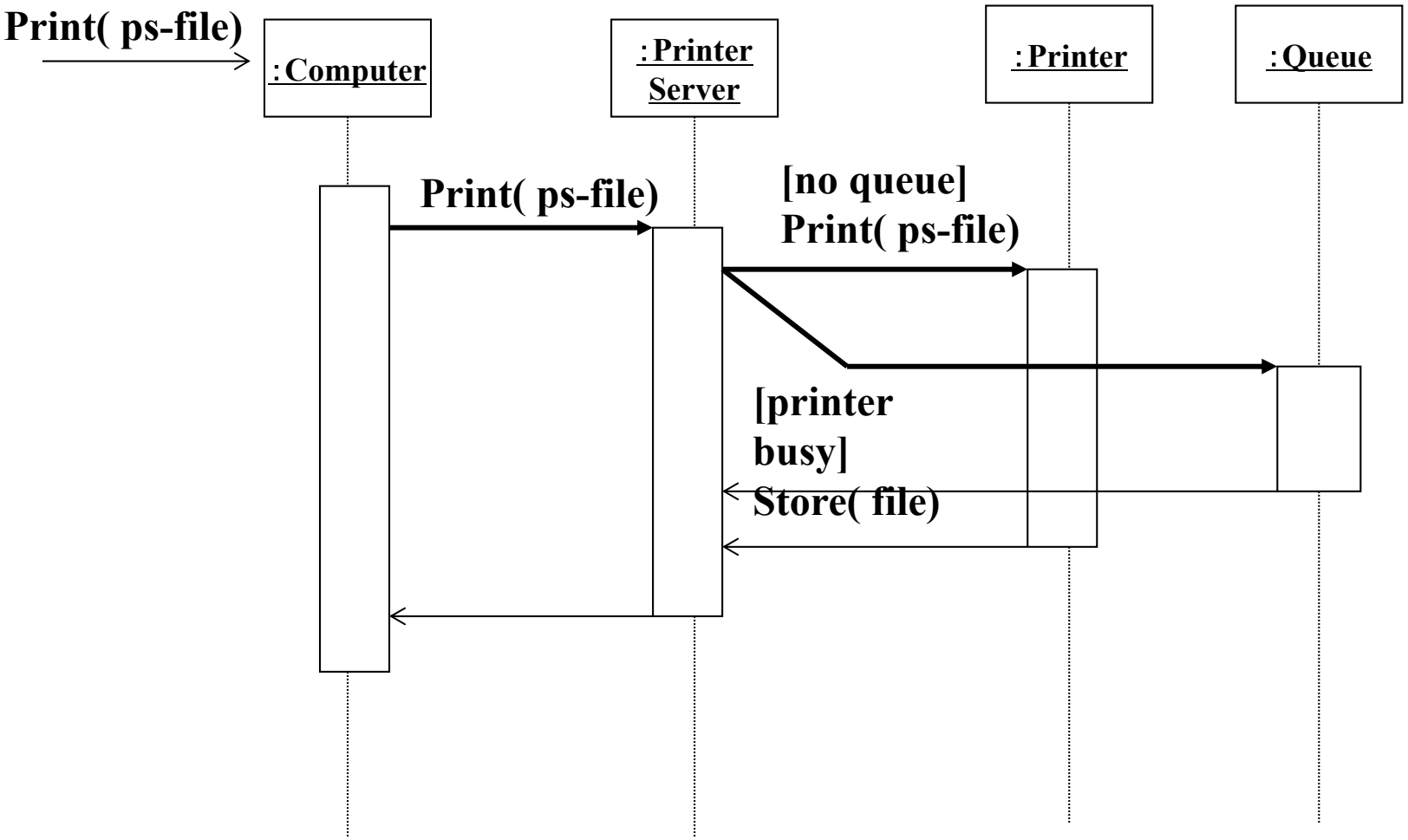
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

The Concepts used In a sequence diagram



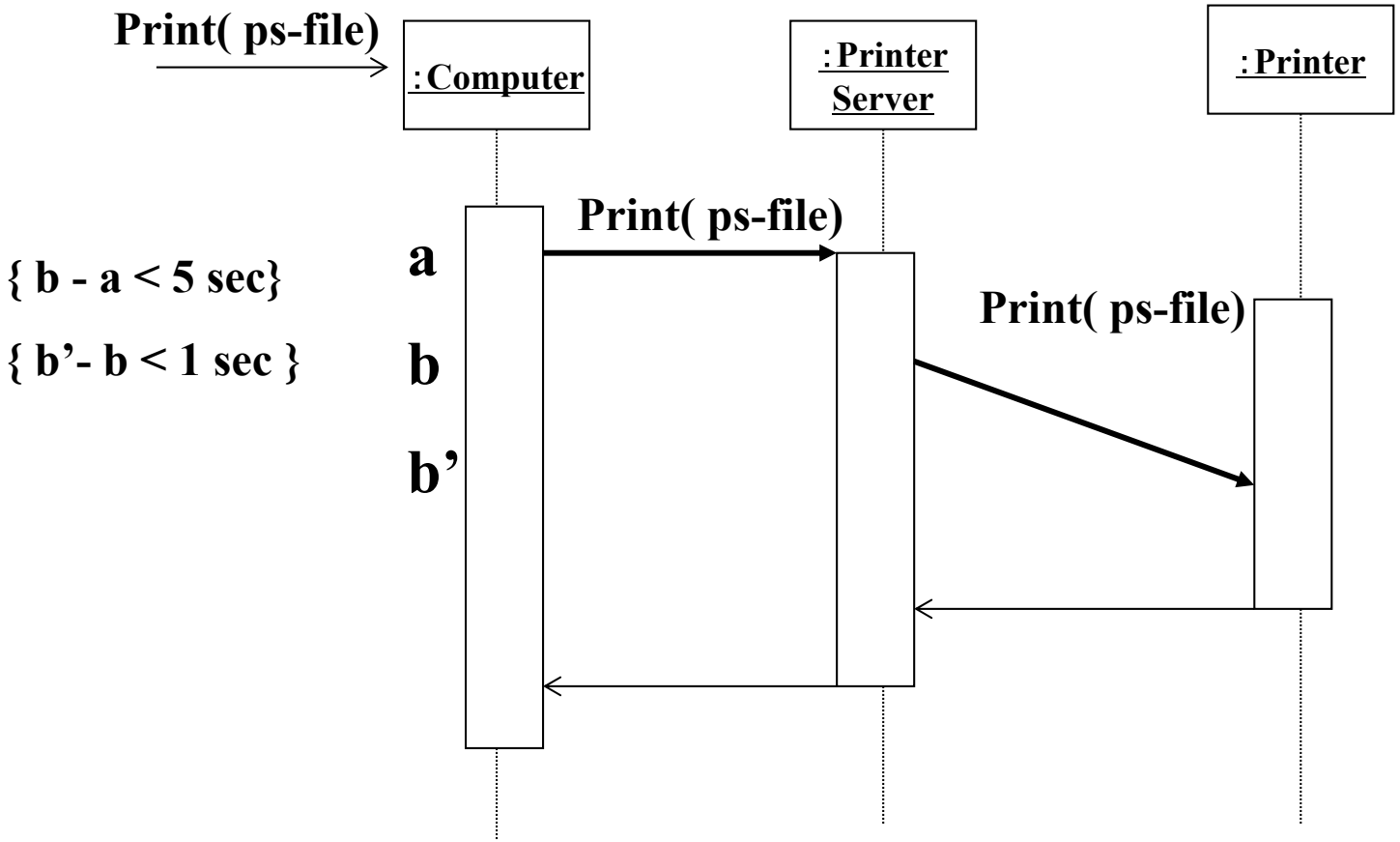
H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Sequence diagram with Branch



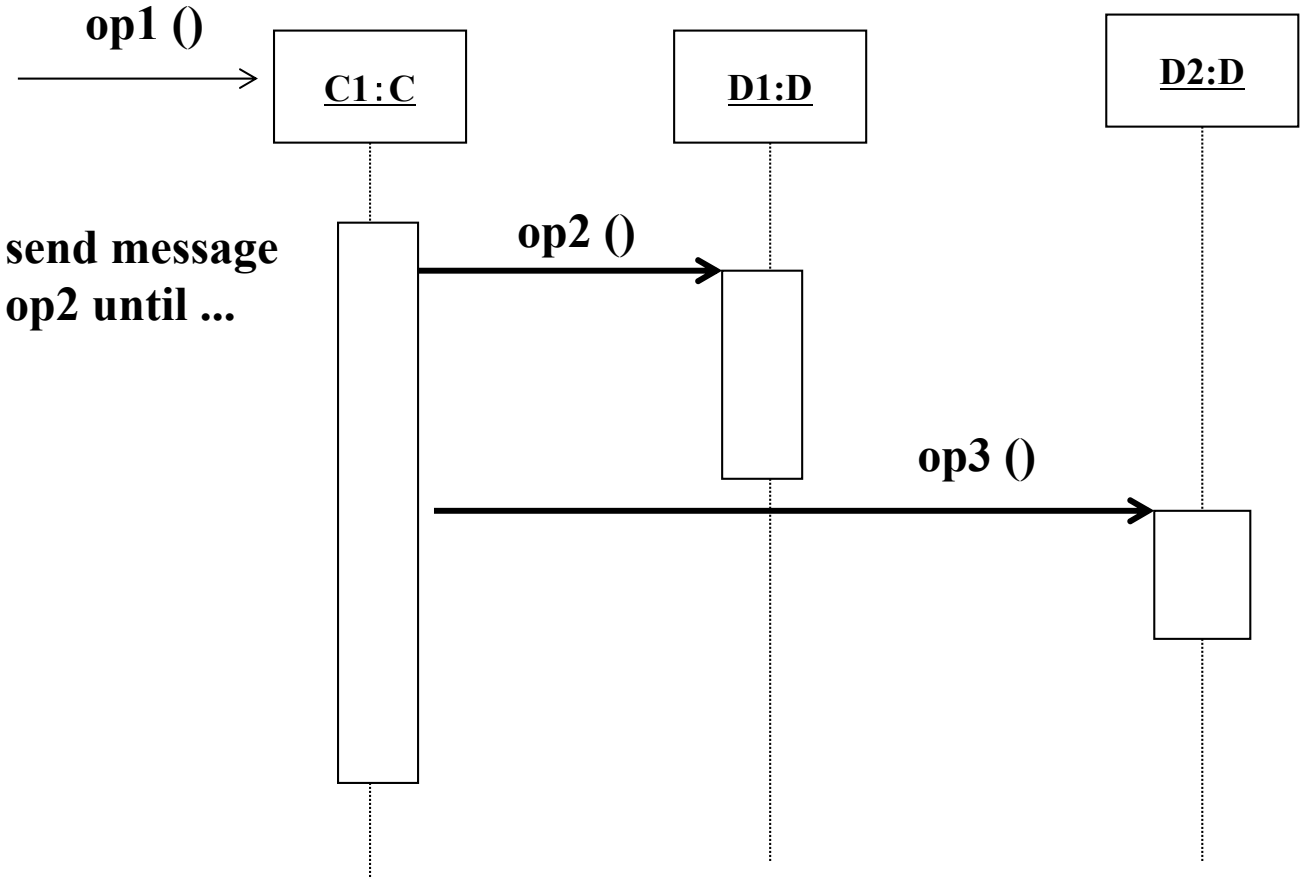
H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Timing Constraint



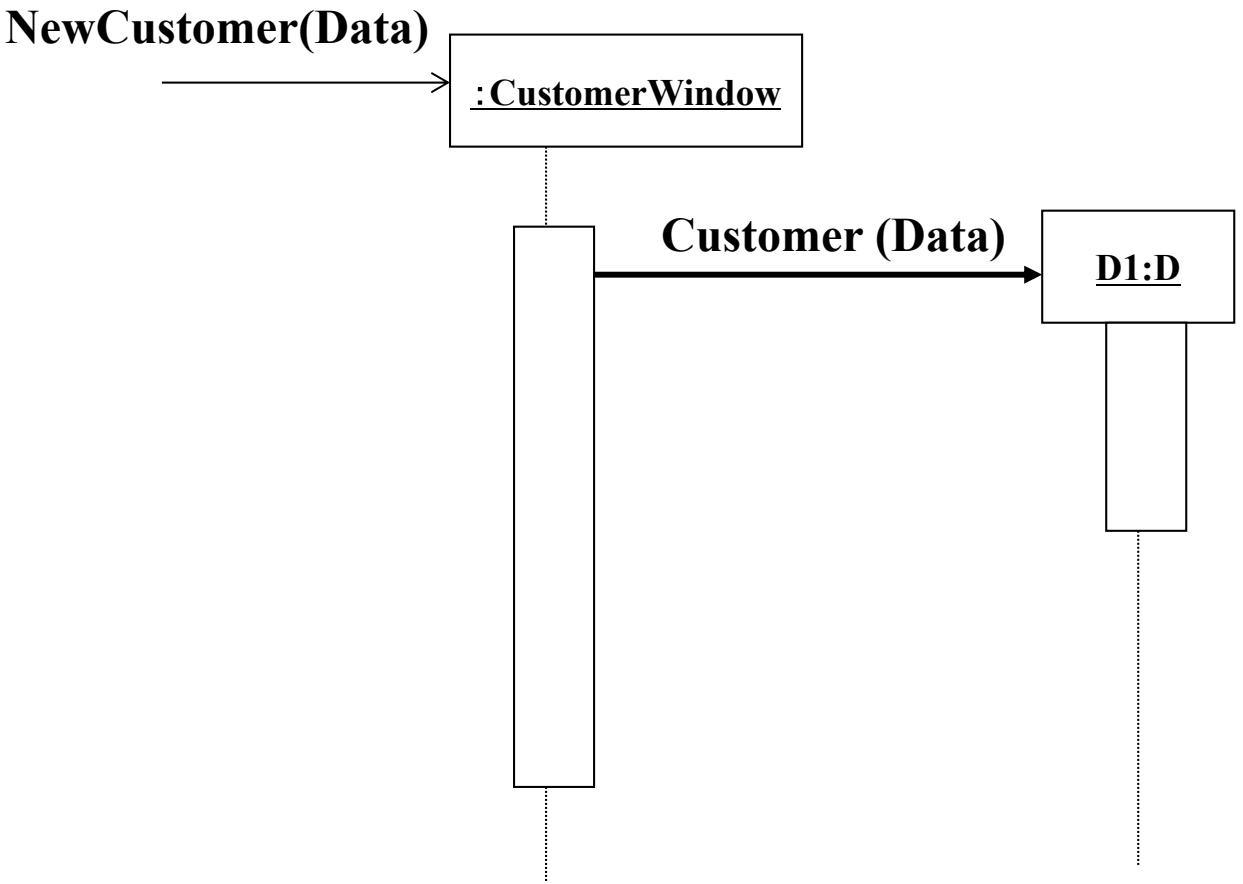
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Iteration



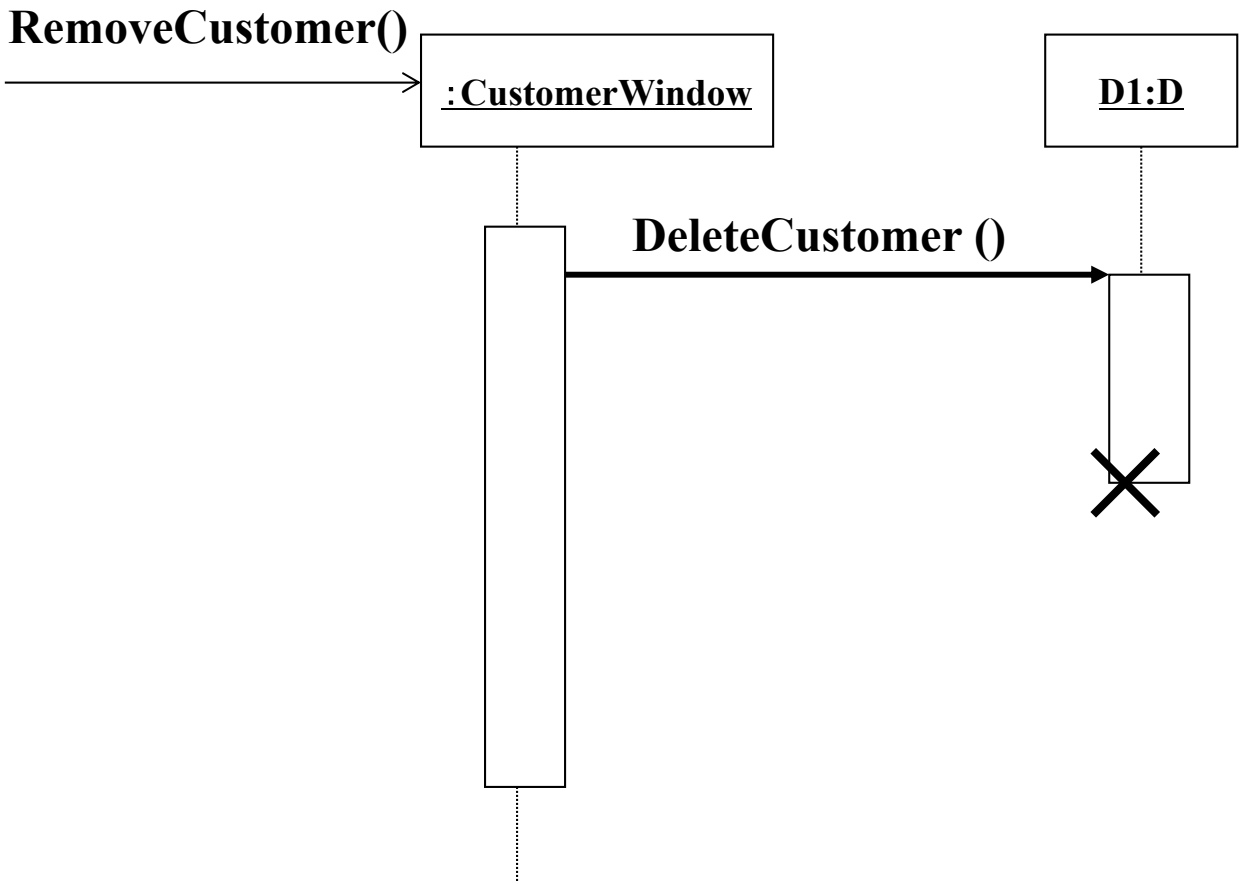
H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Creating Object



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Destroying Object

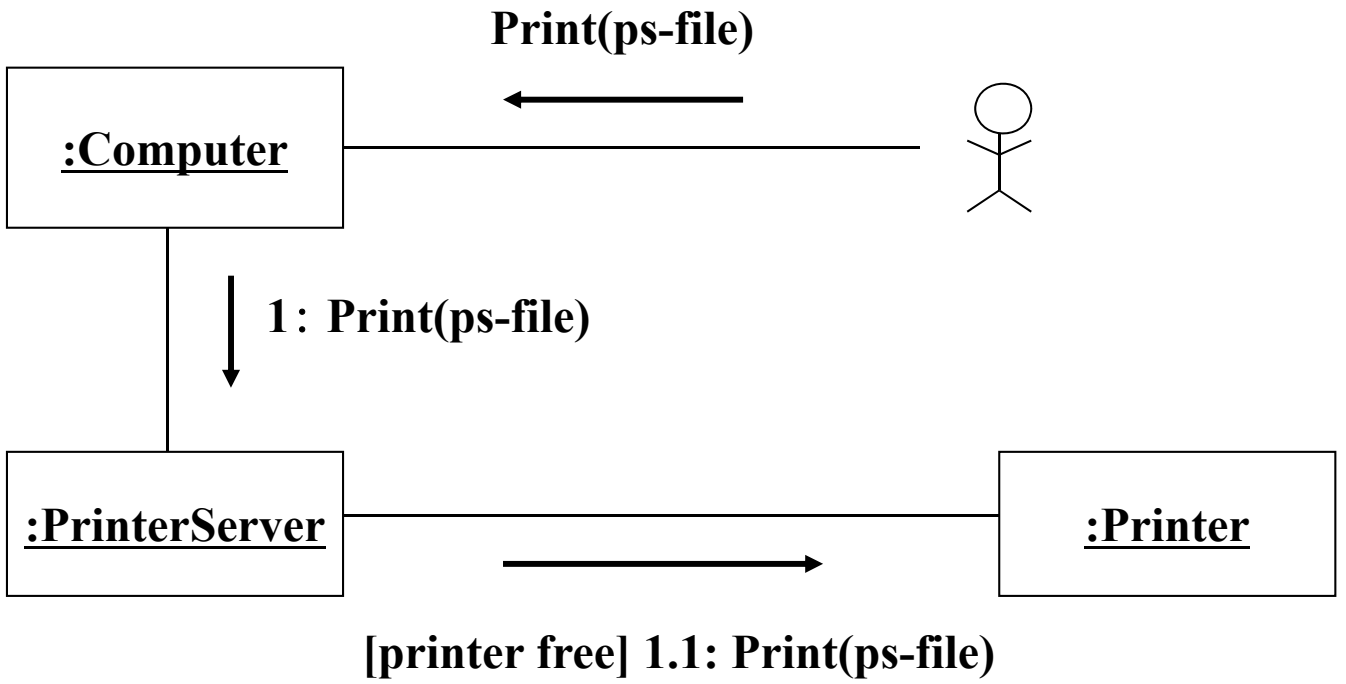


H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Communication Diagram

- Communication diagrams focus on the interaction and the links between a set of collaborating objects. The sequence diagram focuses on time but the communication diagram focuses on space.

An example of a communication diagram



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Message Label

predecessor guard-condition sequence-expression return-value := signature

sequence-number , … /

The predecessor is an expression for synchronization of threads or paths, meaning that the messages connected to specified sequence-number must be performed and handled before the current message is sent.

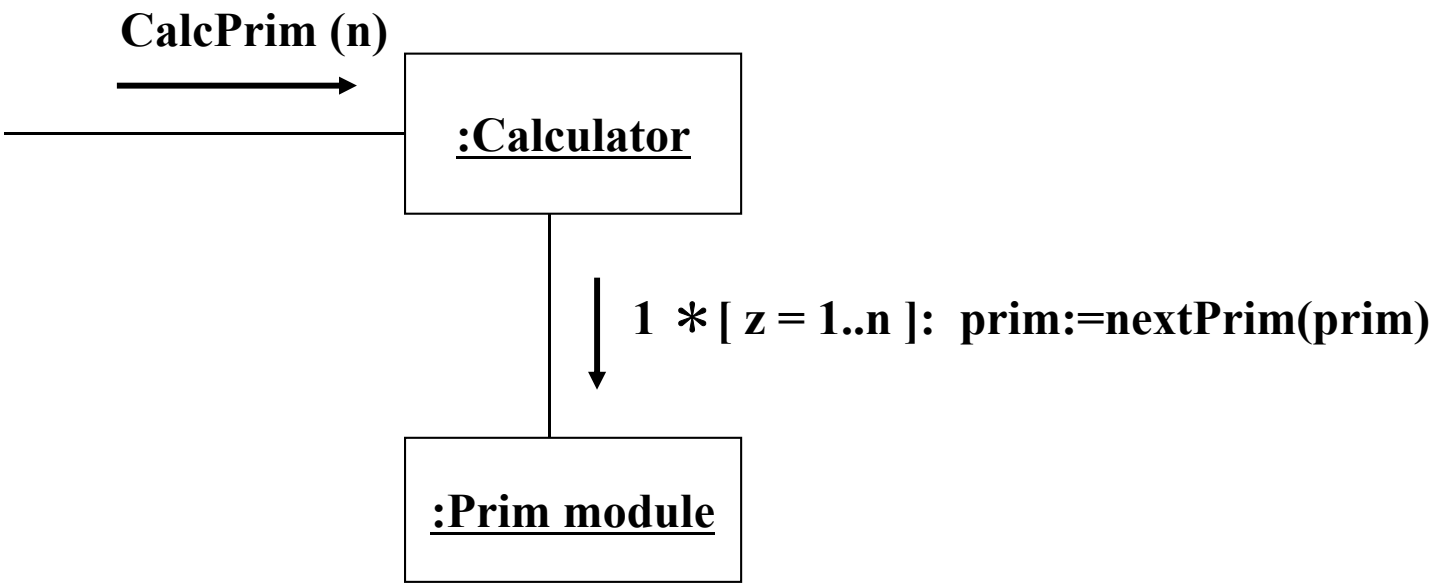
[integer | name] [recurrence]

- integer a sequence-number specifying the message order 1.2.1
- name concurrent thread of control 1.2a 1.2b
- recurrence * [iteration-clause] [condition-clause]
  a conditional or iterative exrcution

1: display()
[mode = display] 1.2.3.7: redraw()
2 * [n:=1..z]: prime:=nextPrim(prim)
3.1 [x < 0] : foo()
3.2 [x => 0] : bar()
1.1a, 1.1b/ 1.2: continue()

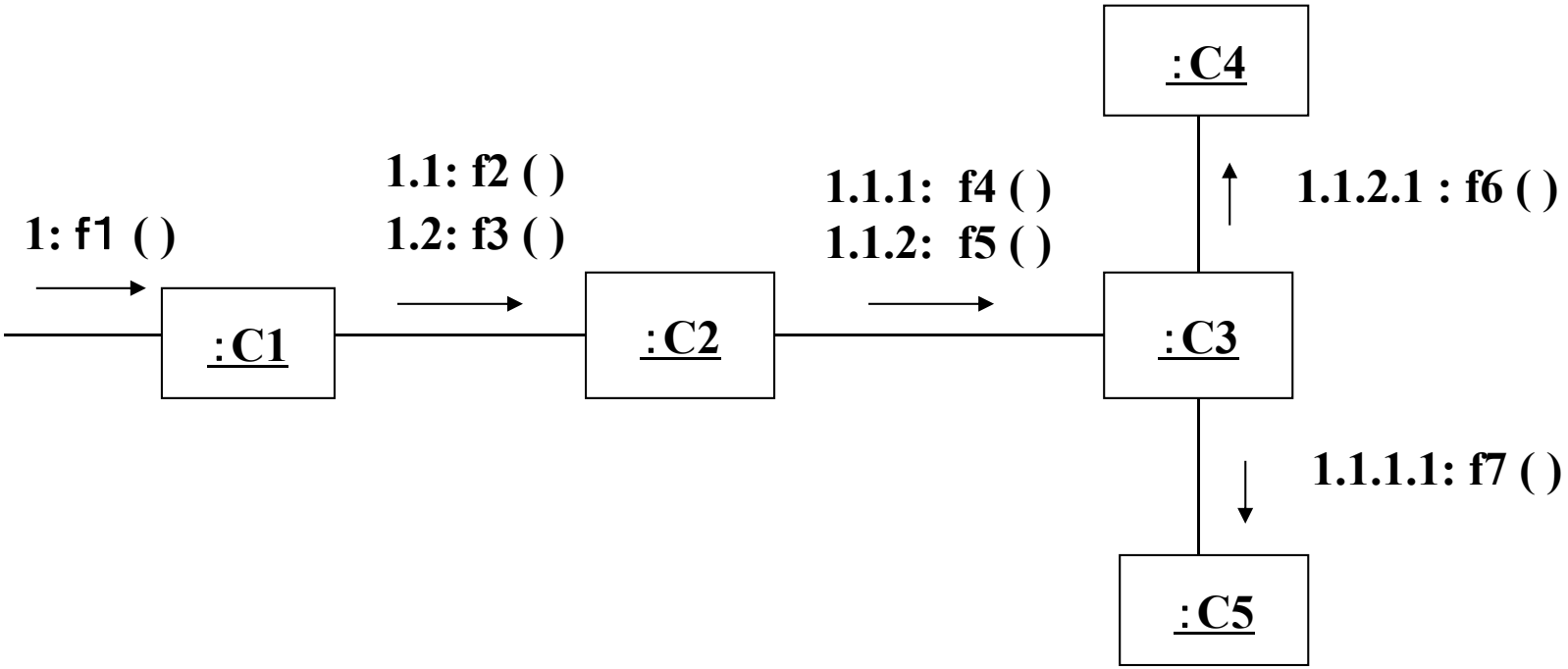
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Iteration in a communication diagram



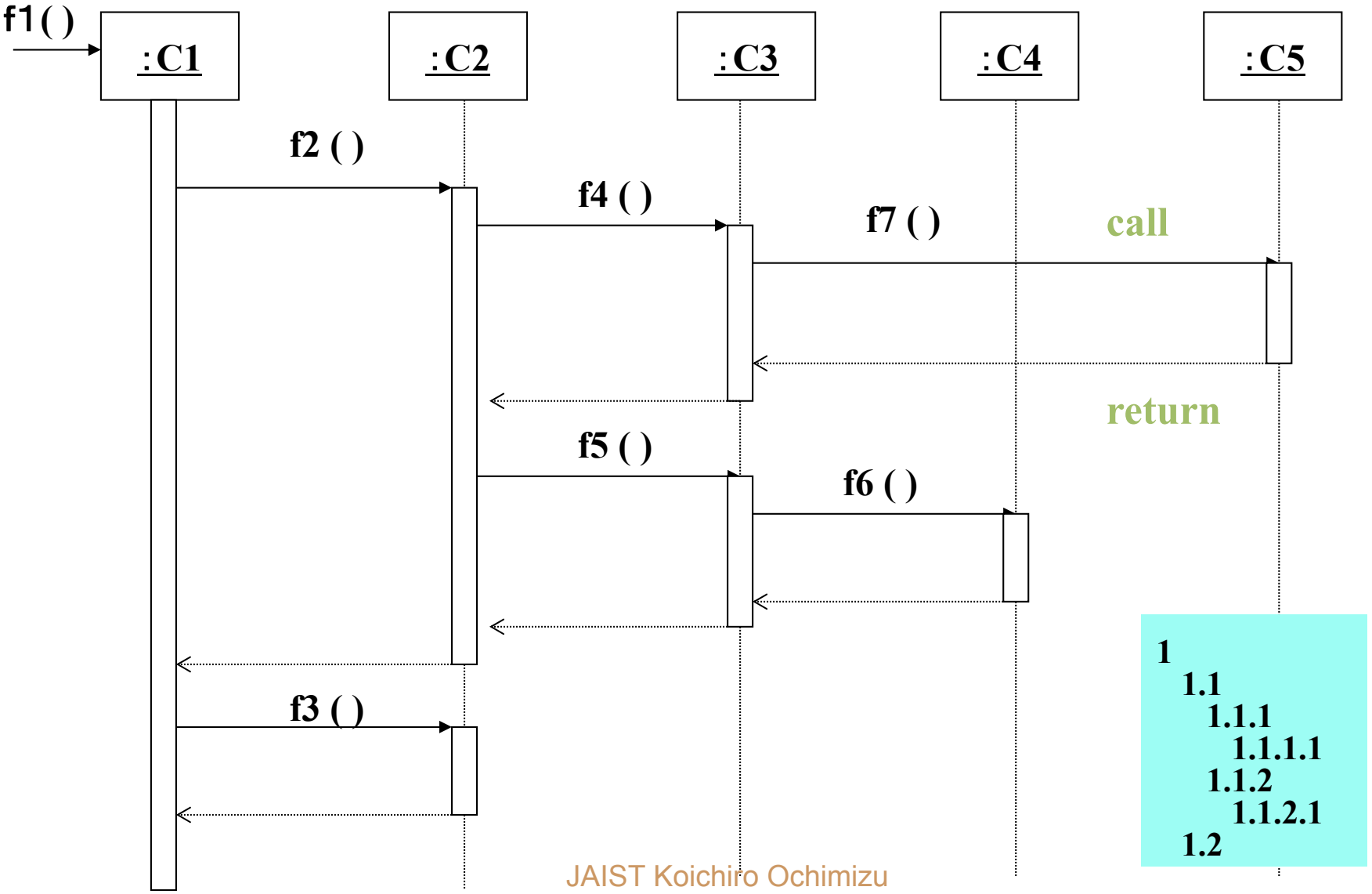
H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Exercise



Answer

UML Ver.1.3



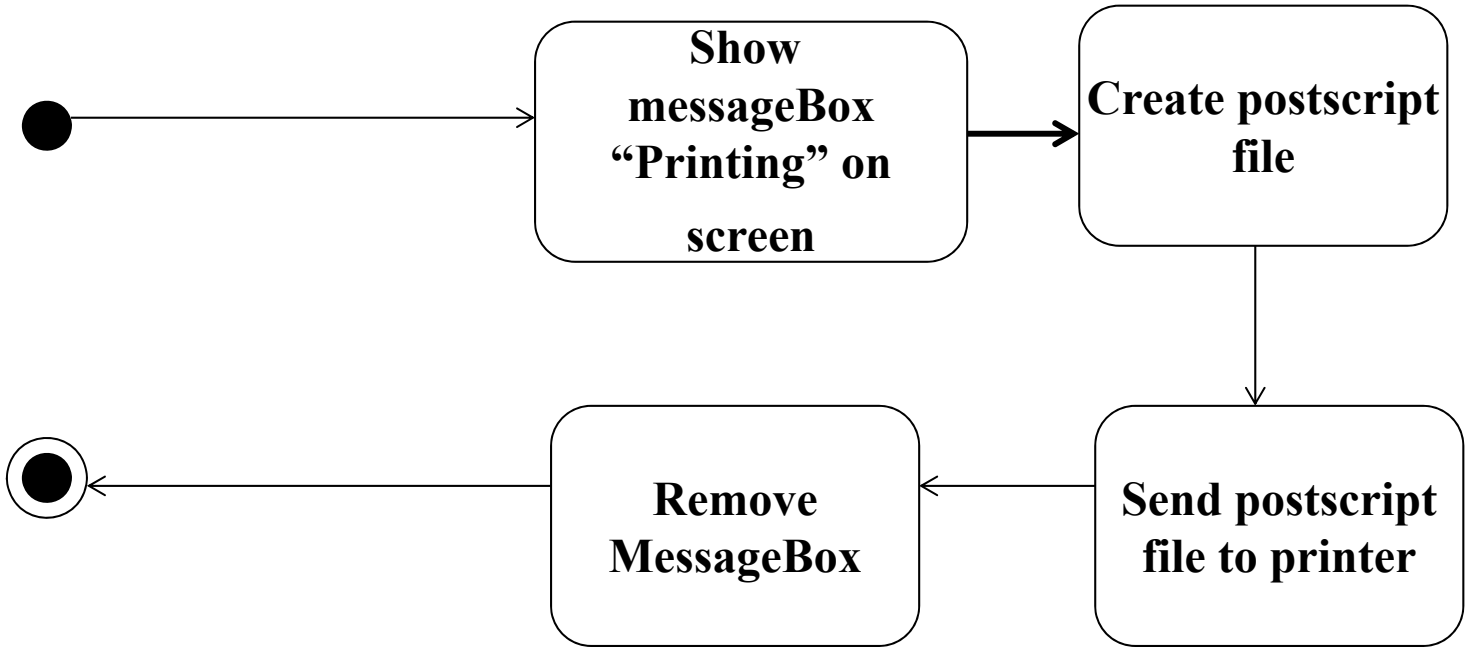
1
- 1.1
  - 1.1.1
    - 1.1.1.1
  - 1.1.2
    - 1.1.2.1
- 1.2

Activity Diagram

- An activity diagram is essentially a flowchart, showing flow of control from activity to activity

Actions and Transitions

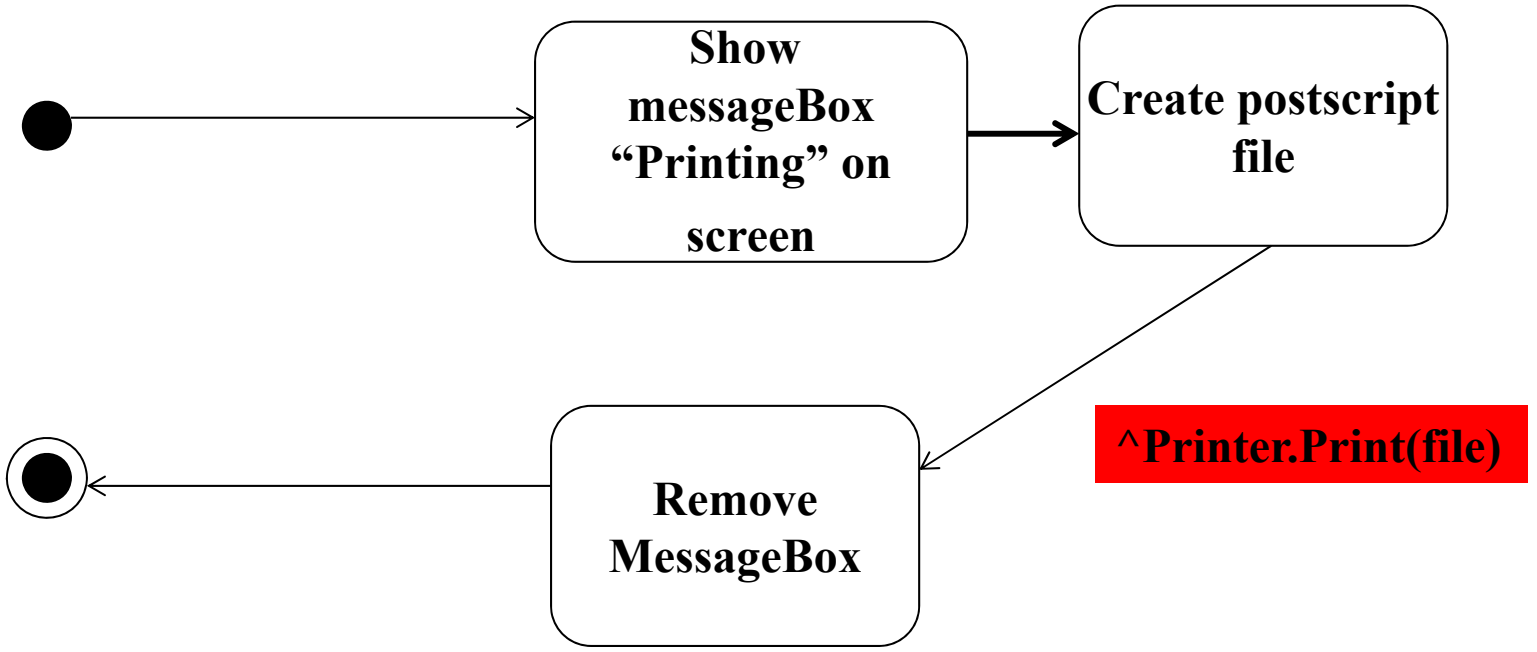
CustomerWindow.PrintAllCustomer()



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

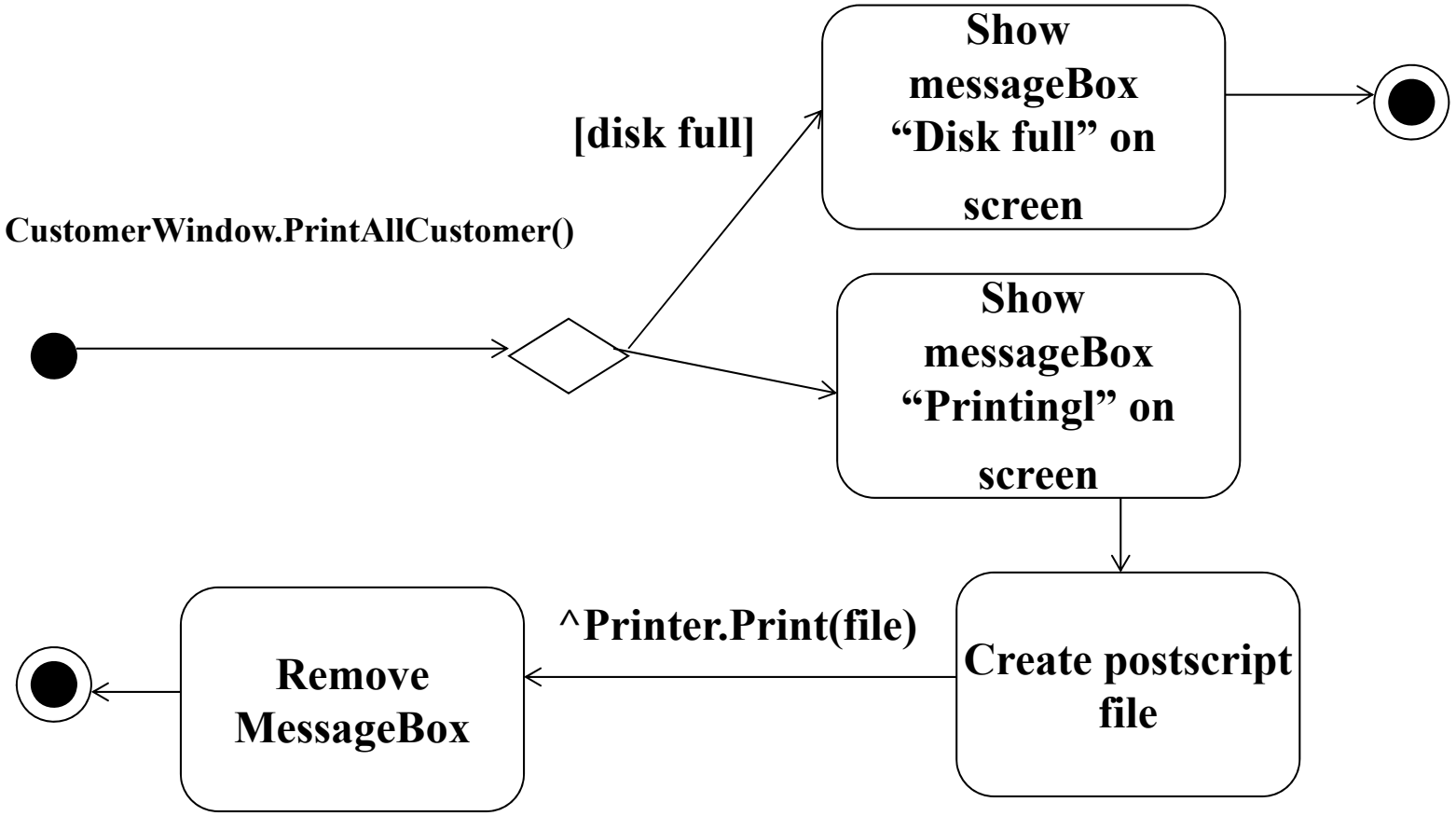
Transition with send-clause

CustomerWindow.PrintAllCustomer()



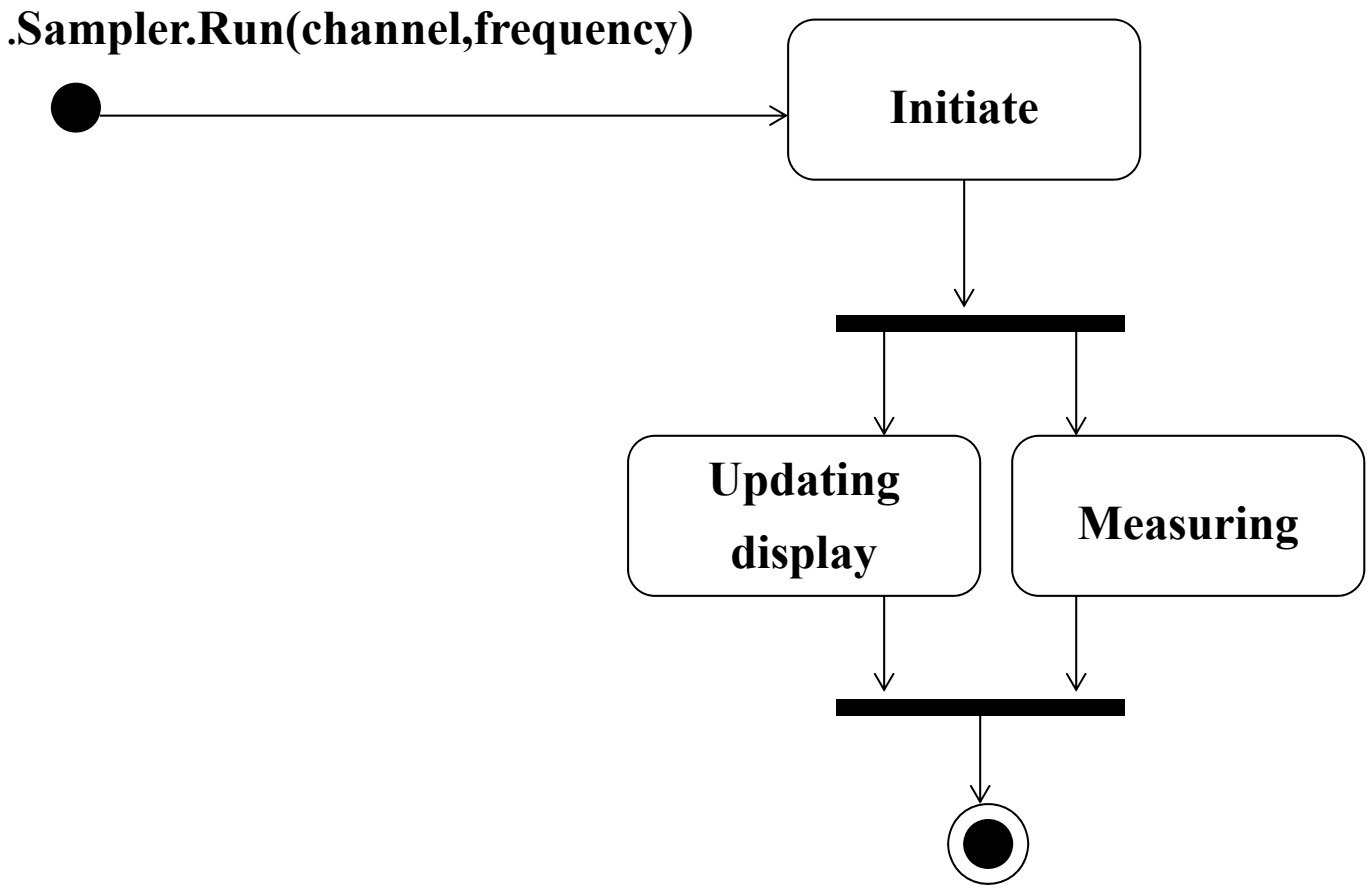
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Transitions are protected by guarded-conditions



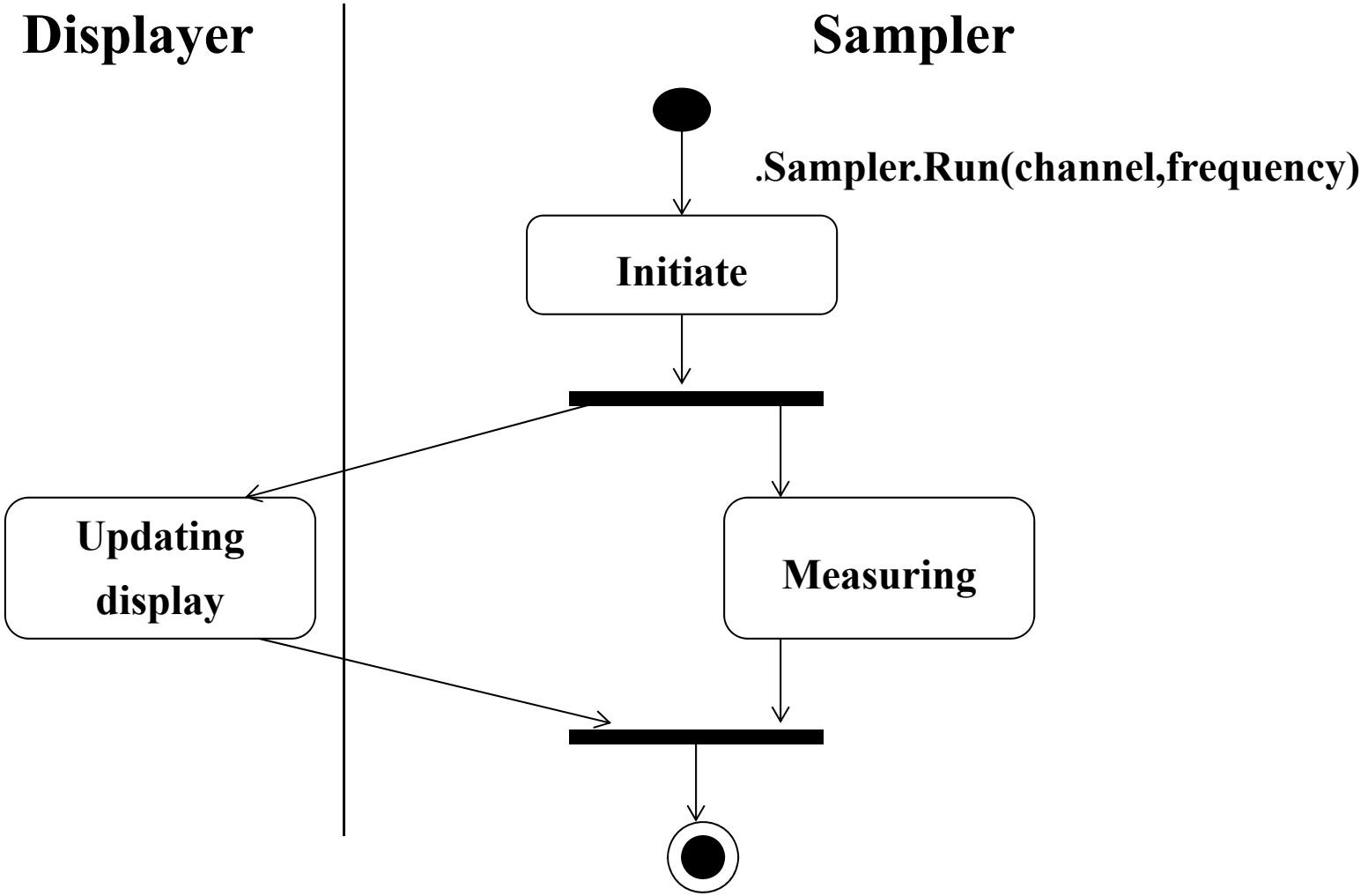
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Parallel Actions



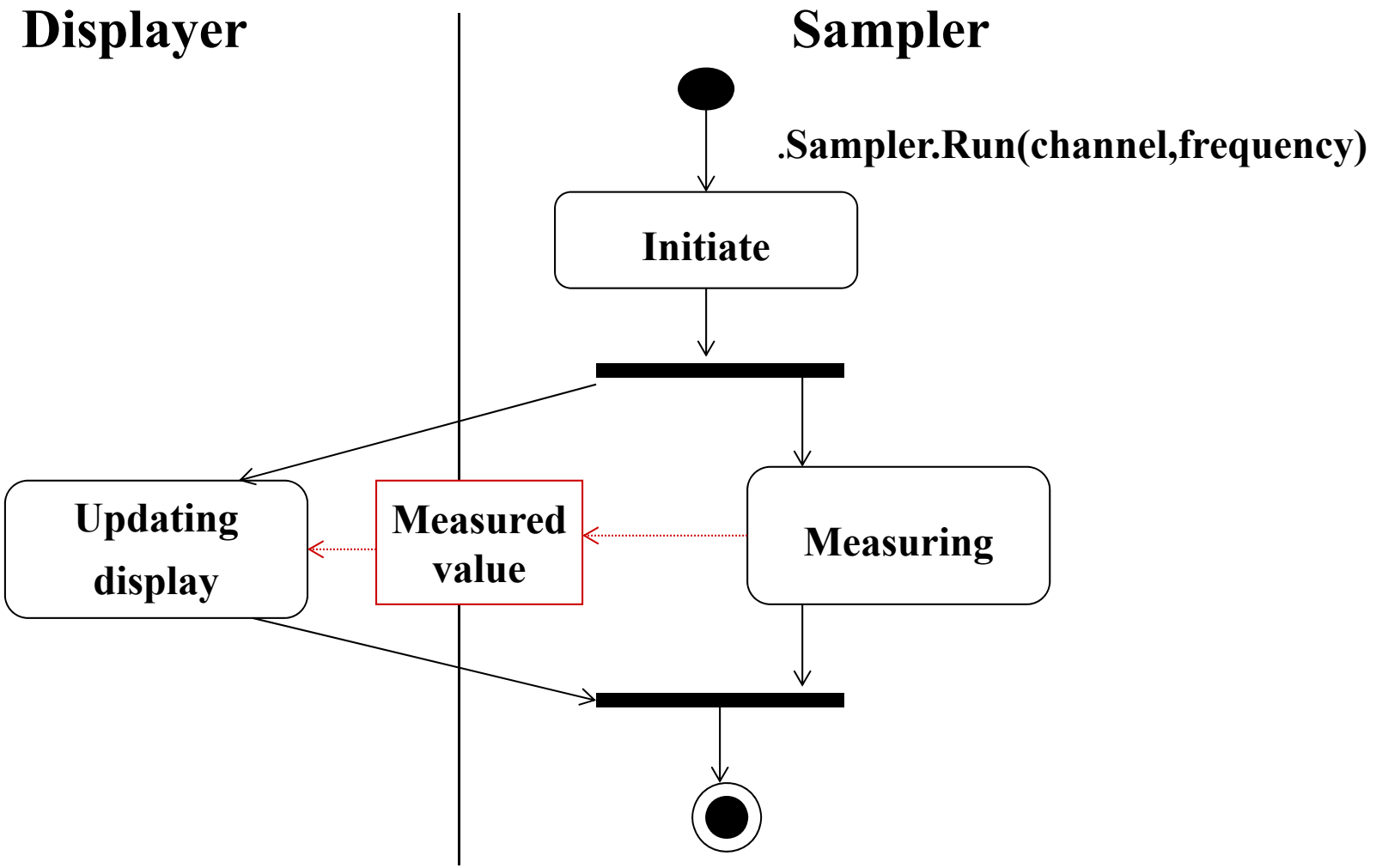
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Swim lanes



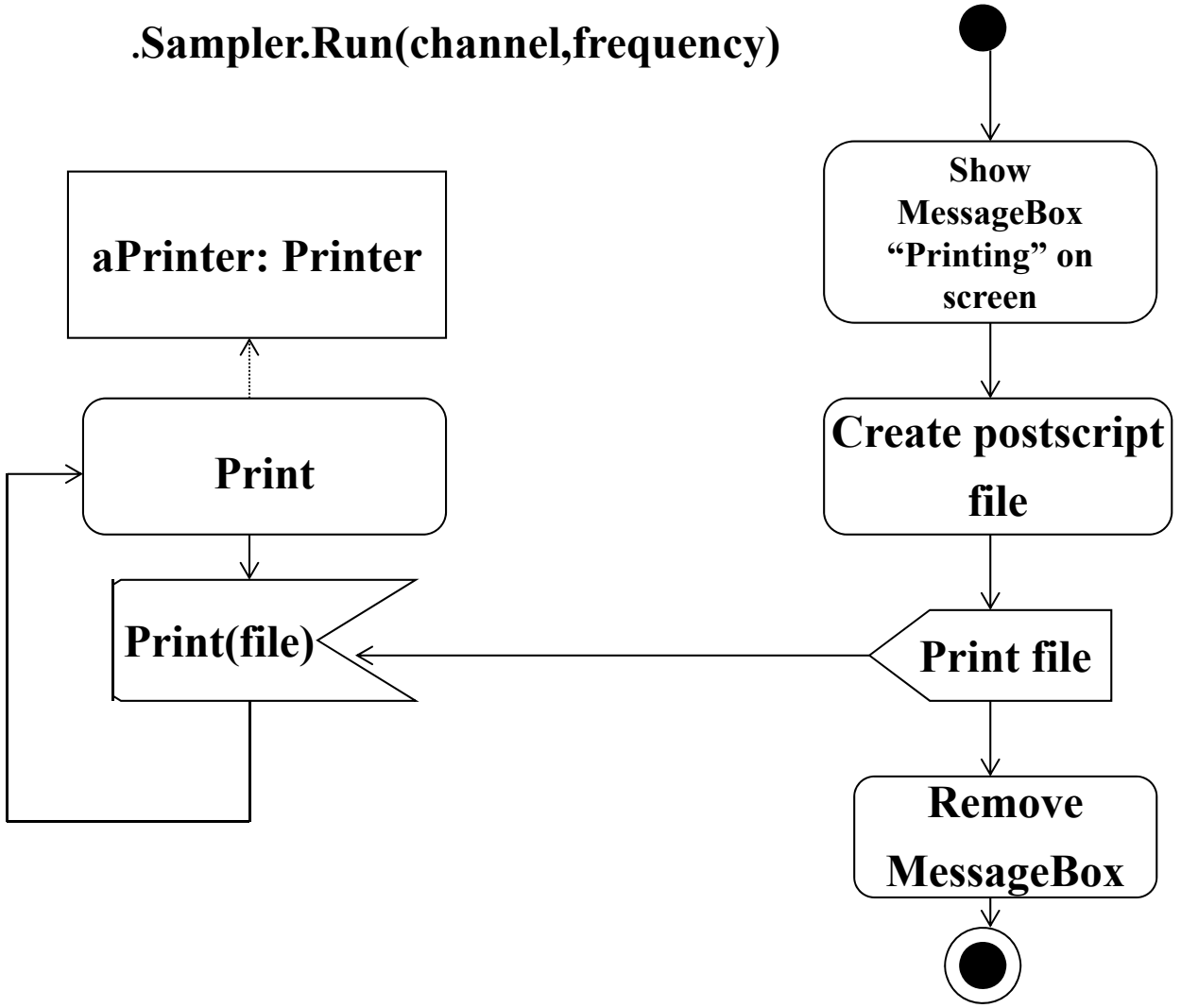
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Object Flow



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

Sending/receiving signals



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.