Subject:

Object-oriented analysis and design

Chapter 5: Communication, statechart diagram

Lê Văn Vinh, PhD
Department of Software Engineering
Faculty of Information Technology
University of Technical Education HCMC

Contents

I. Communication Diagrams

II Statechart Diagrams

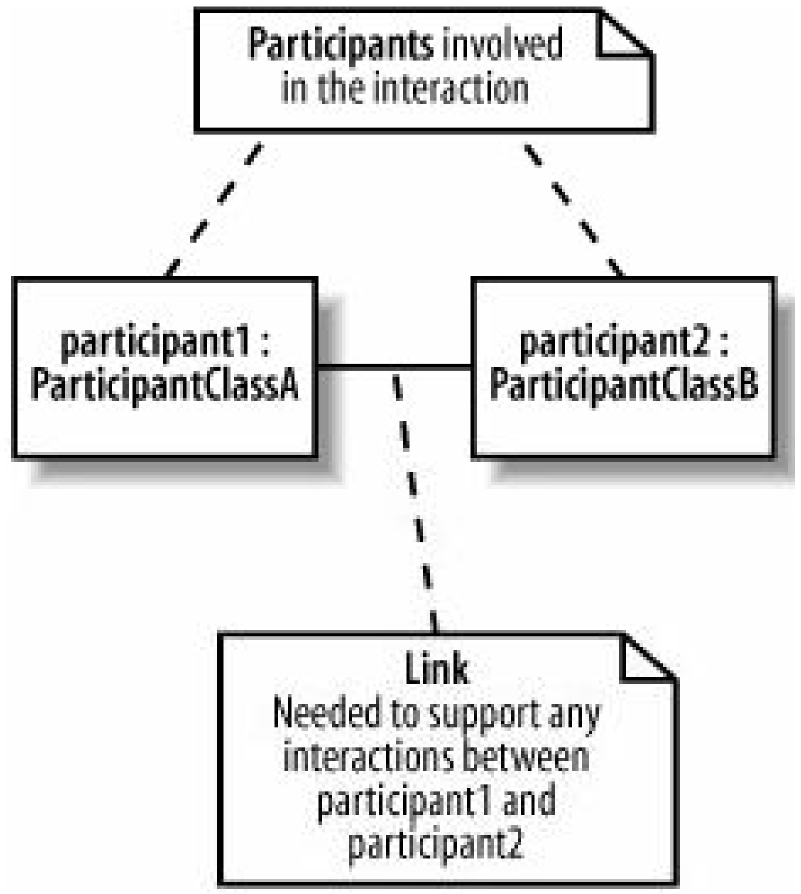
Communication Diagrams

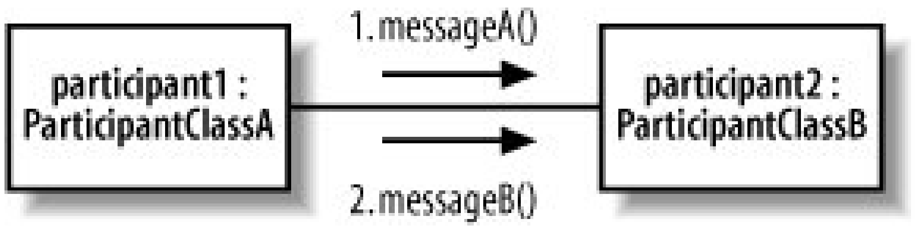
- **A communication diagram is an alternate way to represent the messages exchanged by a set of objects**
- **The diagram shows object interactions organized around the objects and their links to each other**

Communication Diagrams

- **A collaboration diagram contains:**
  - Objects
  - Links between objects
  - Messages exchanged between objects
  - Data flowing between objects, if any

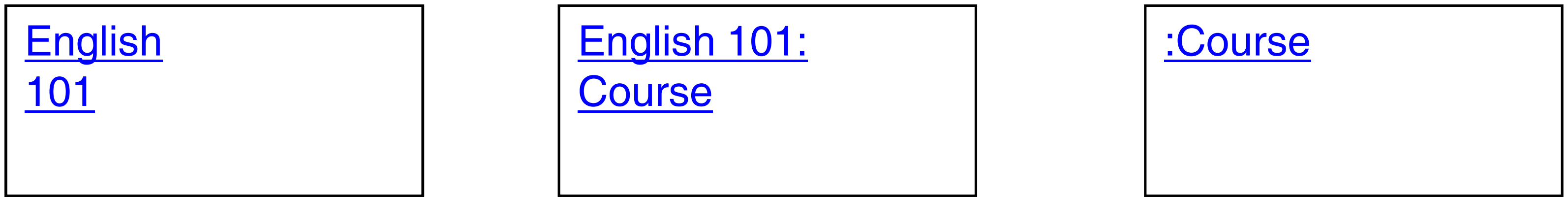
Participants, Messages and Links


Participants involved in the interaction
participant1 : ParticipantClassA
participant2 : ParticipantClassB
Link
Needed to support any interactions between participant1 and participant2


participant1 : ParticipantClassA
1.messageA()
2.messageB()
participant2 : ParticipantClassB

Representing Objects

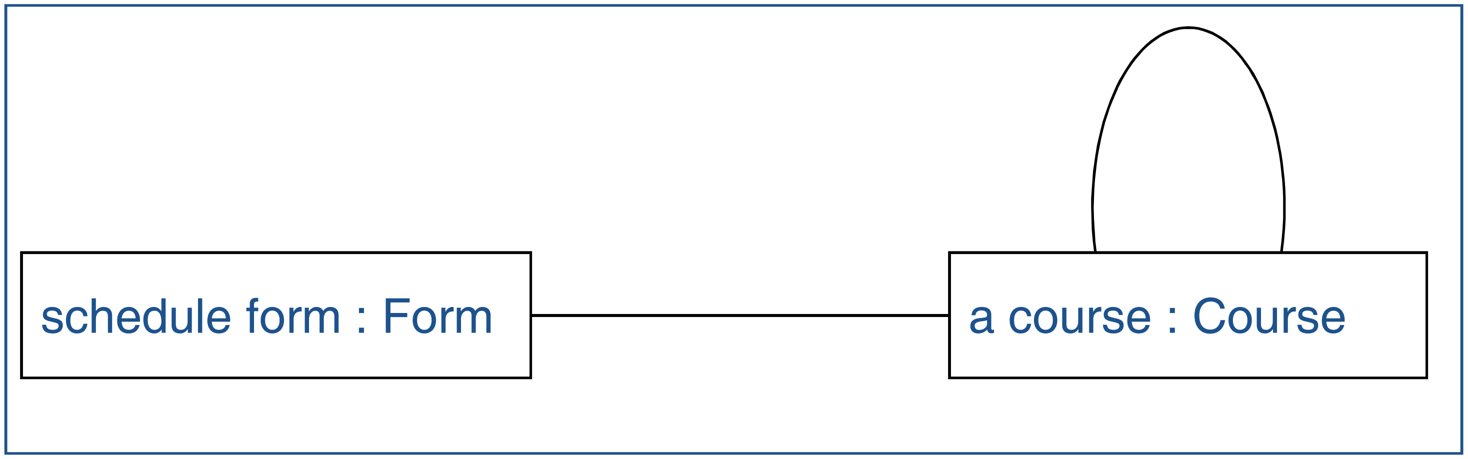
- Similar to that of the sequence diagram



Object only **Object and Class** **Class only**

Representing Links

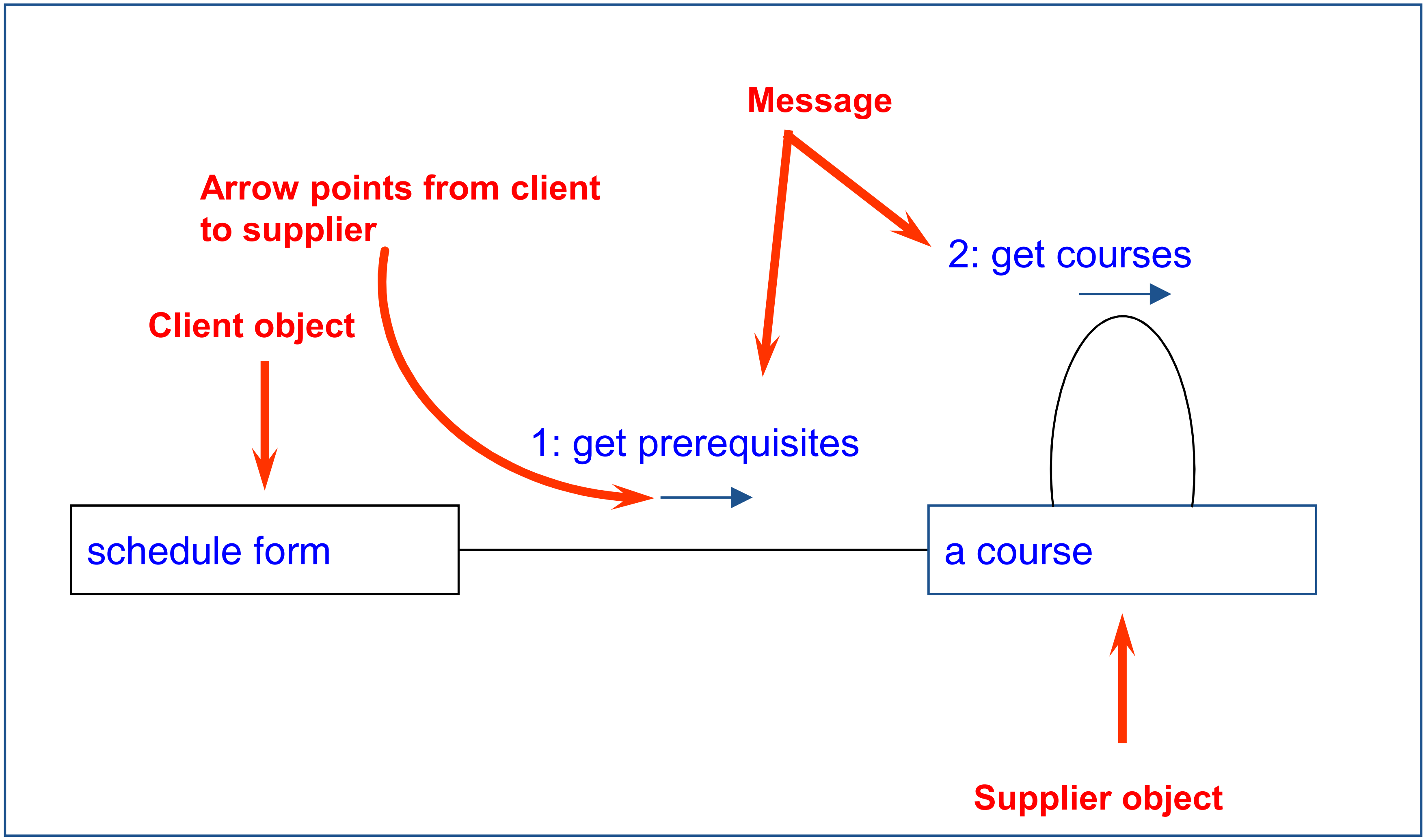
- A interaction link in a collaboration diagram is represented as a line connecting object icons
- A link indicates that there is a pathway for communication between the connected objects



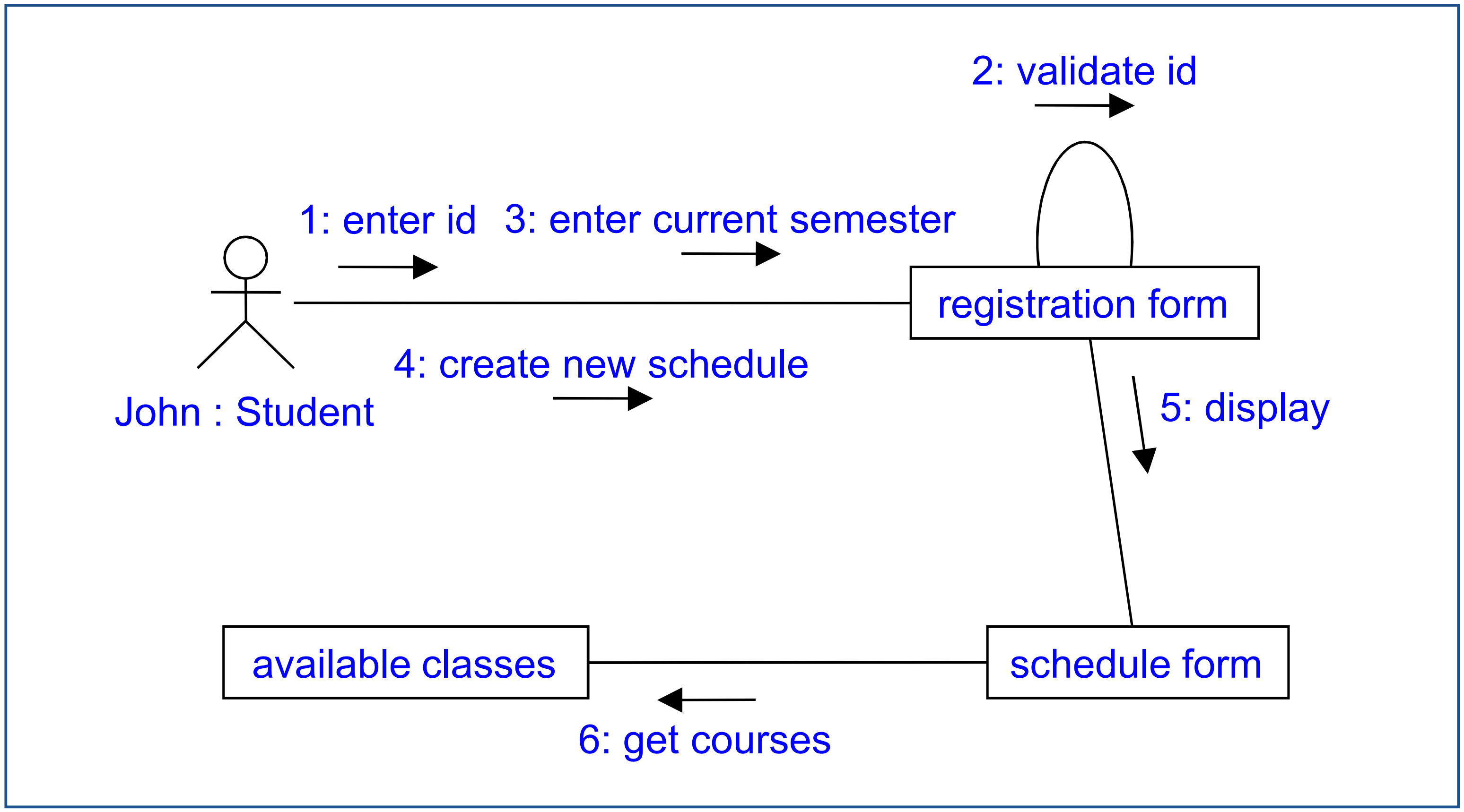
Link Annotations

- An interaction link in a collaboration diagram can be annotated with:
  - An arrow pointing from the client object to the supplier object
  - The name of the message with an optional list of parameters and/or a data return value
  - An optional sequence number showing the relative order with which the messages are sent

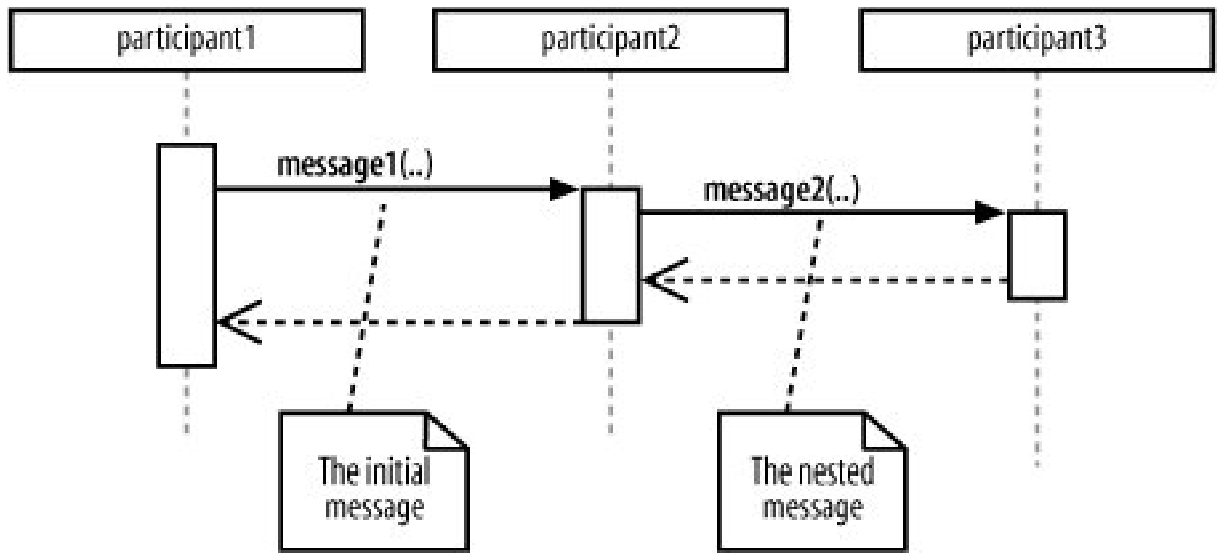
Link Notation

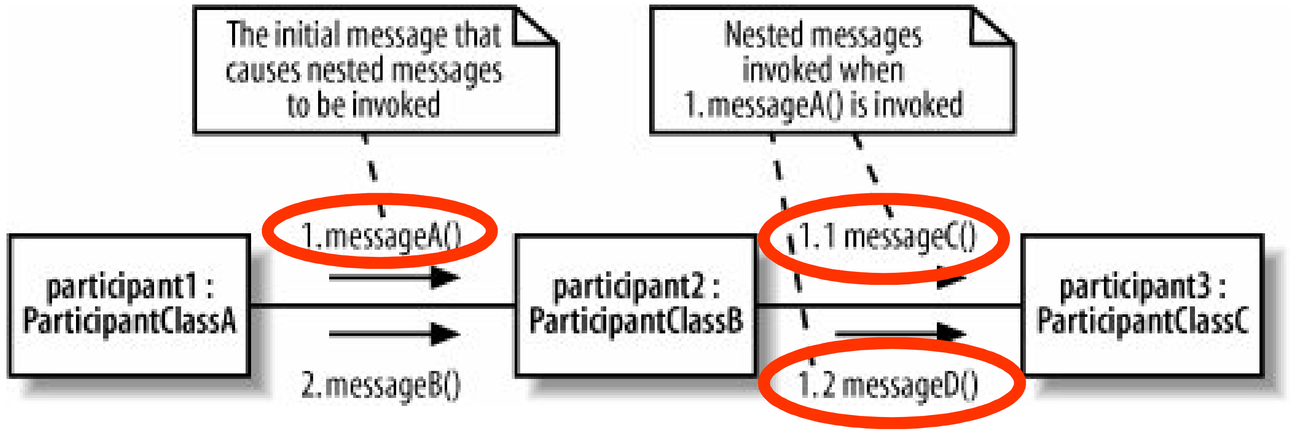


Sample Collaboration Diagram



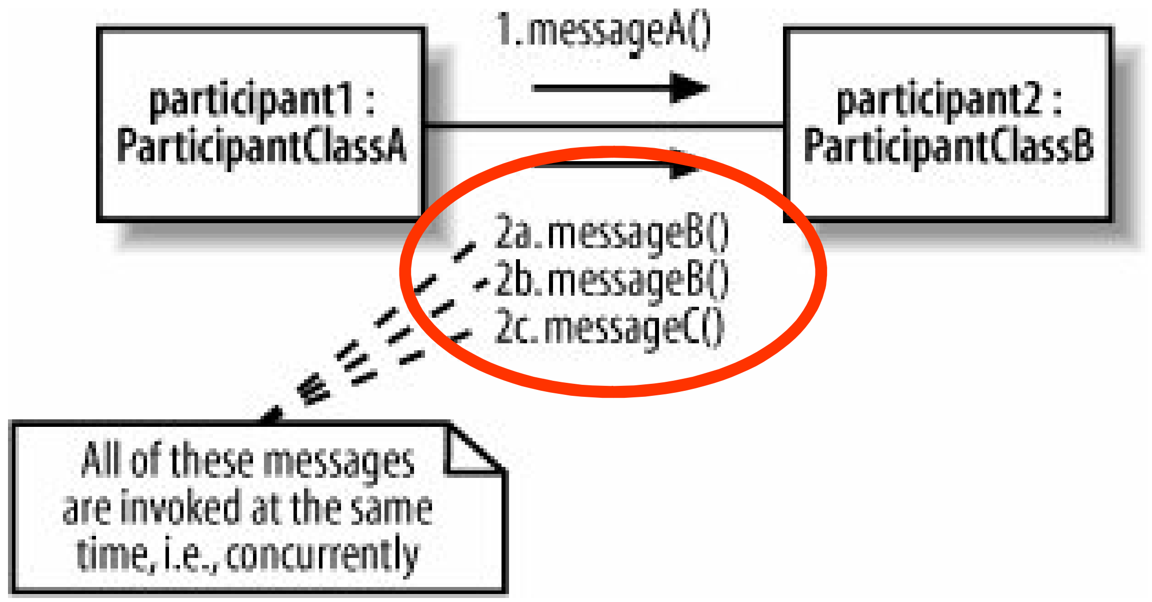
Nested messages




The initial message that causes nested messages to be invoked
Nested messages invoked when 1.messageA() is invoked
1.messageA()
2.messageB()
1.1 messageC()
1.2 messageD()
participant1 : ParticipantClassA
participant2 : ParticipantClassB
participant3 : ParticipantClassC

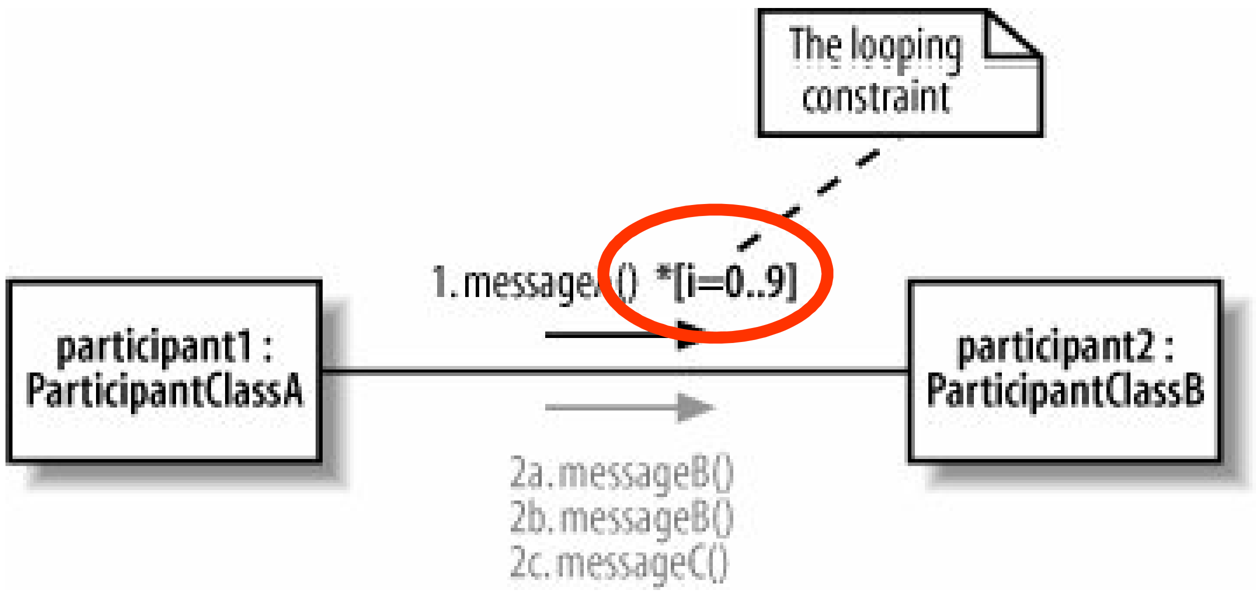
Concurrence messages

- **All concurrence messages have the same number, but different in the subsequent alphabetical character**


participant1 : ParticipantClassA
participant2 : ParticipantClassB
1.messageA()
2a.messageB()
2b.messageB()
2c.messageC()
All of these messages are invoked at the same time, i.e., concurrently

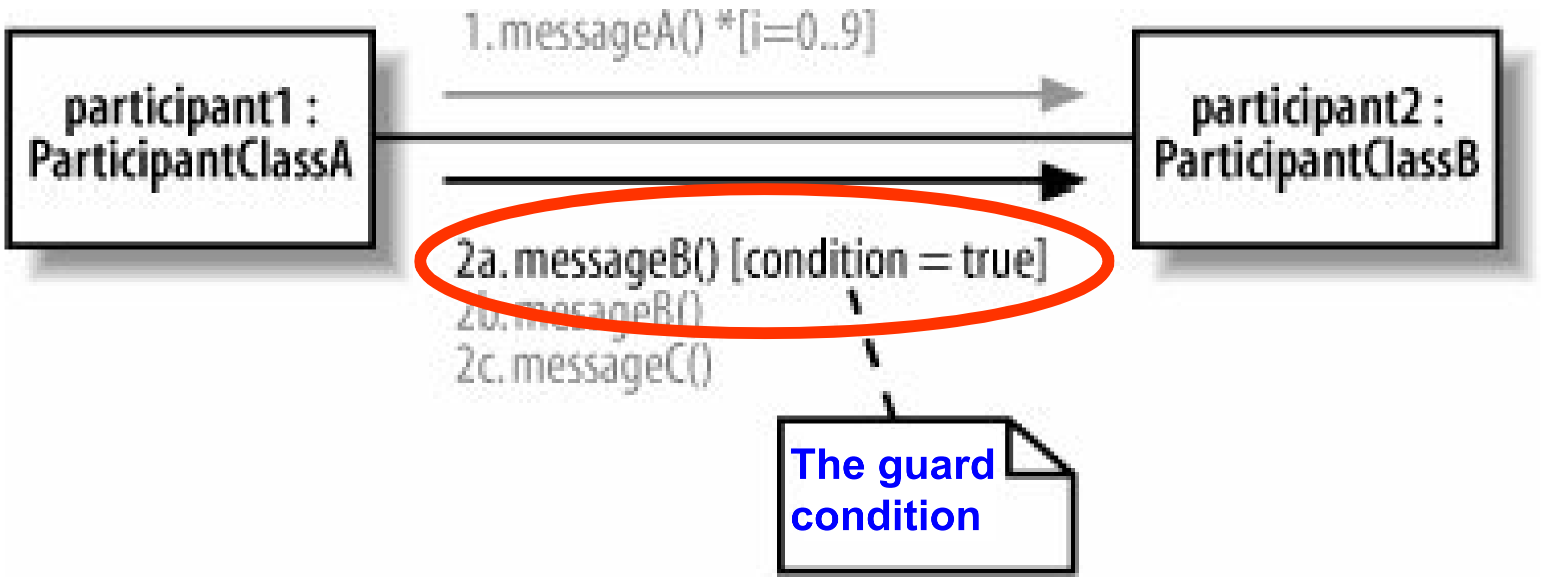
Messages invoked multiple times

- **Using multiplicity after the message**



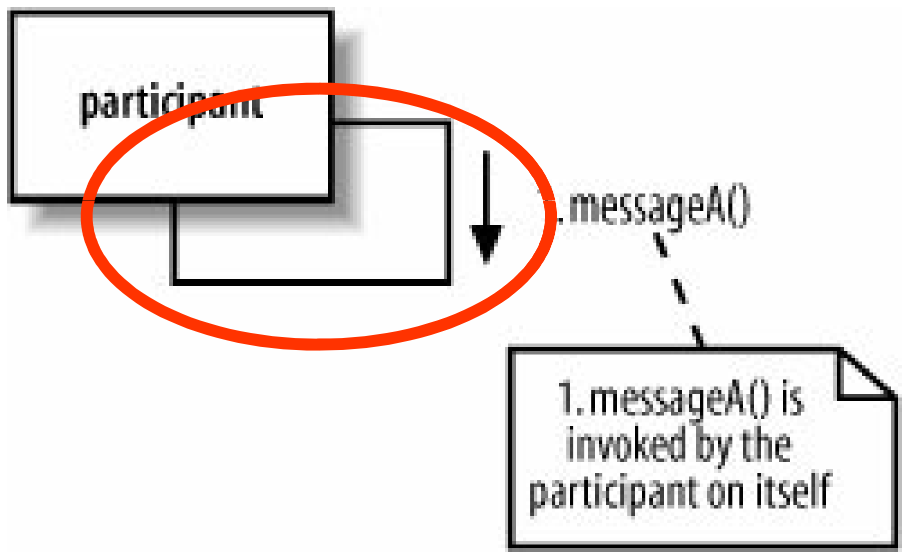
Messages based on a condition

- **The message can only be invoked if the condition is true**

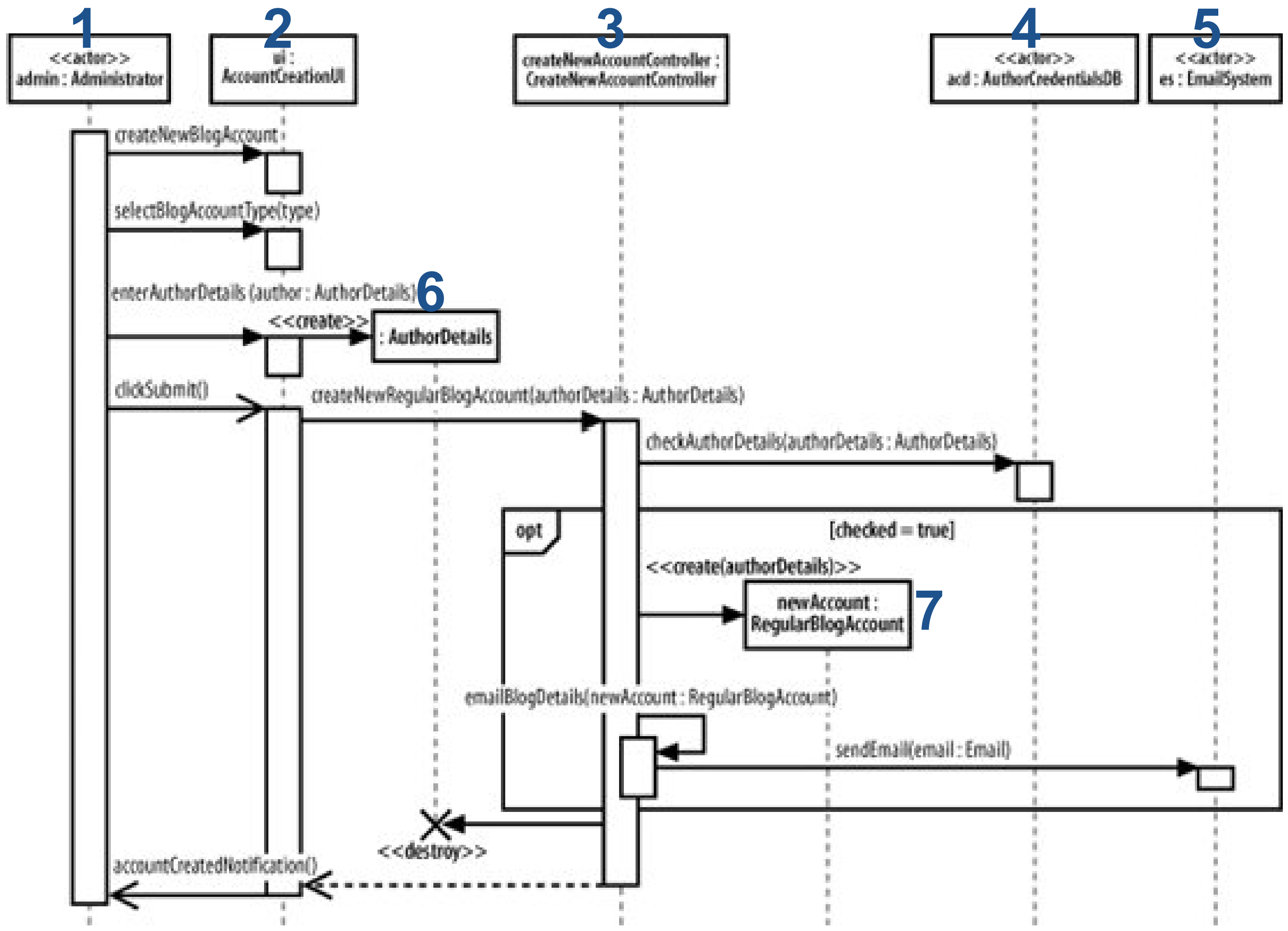

1.messageA() *[i=0..9]
participant1 : ParticipantClassA
participant2 : ParticipantClassB
2a.messageB() [condition = true]
2b.messageB()
2c.messageC()
The guard condition

Self messages

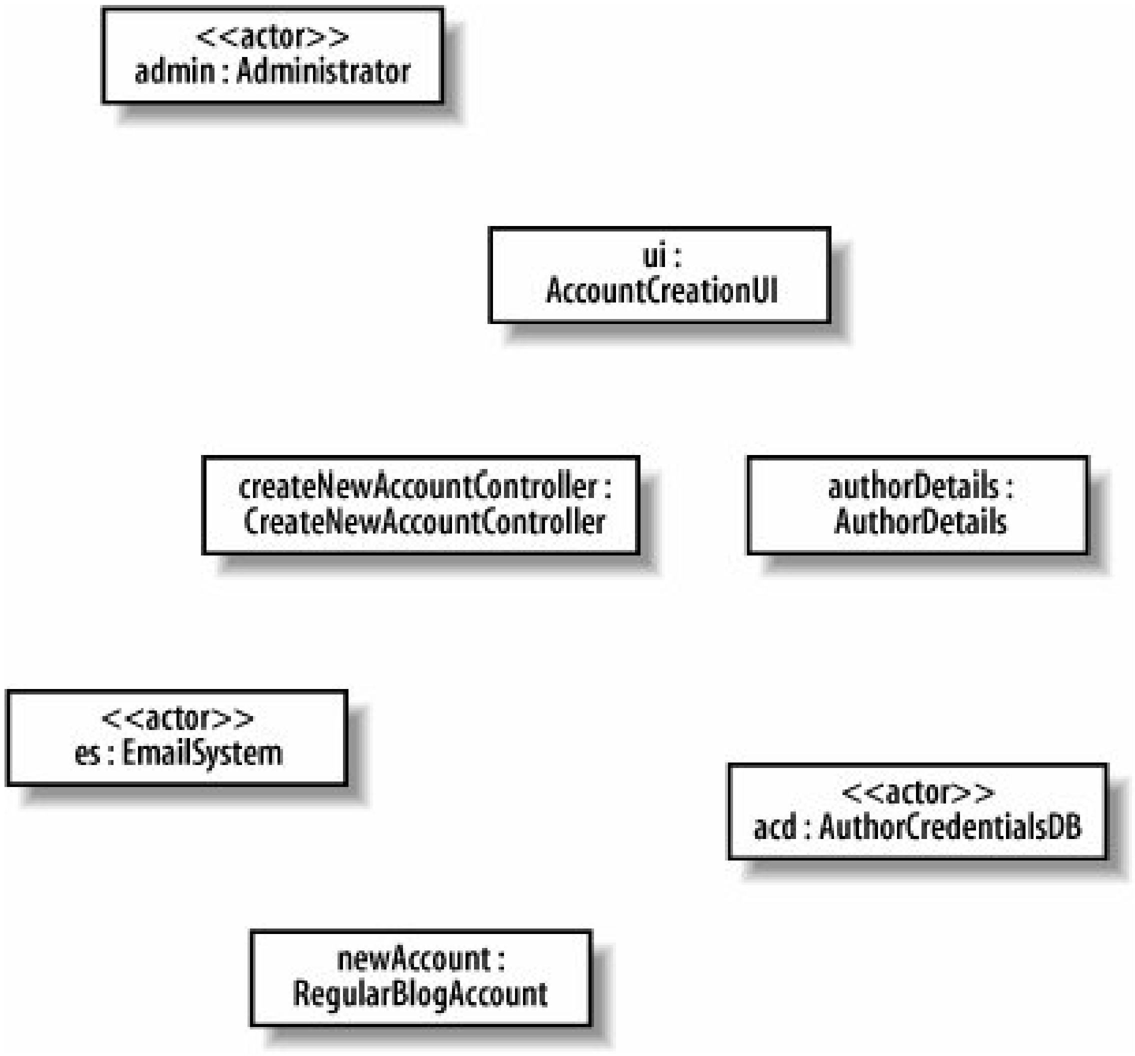
- **An object may send a message to itself**



Building a communication diagram

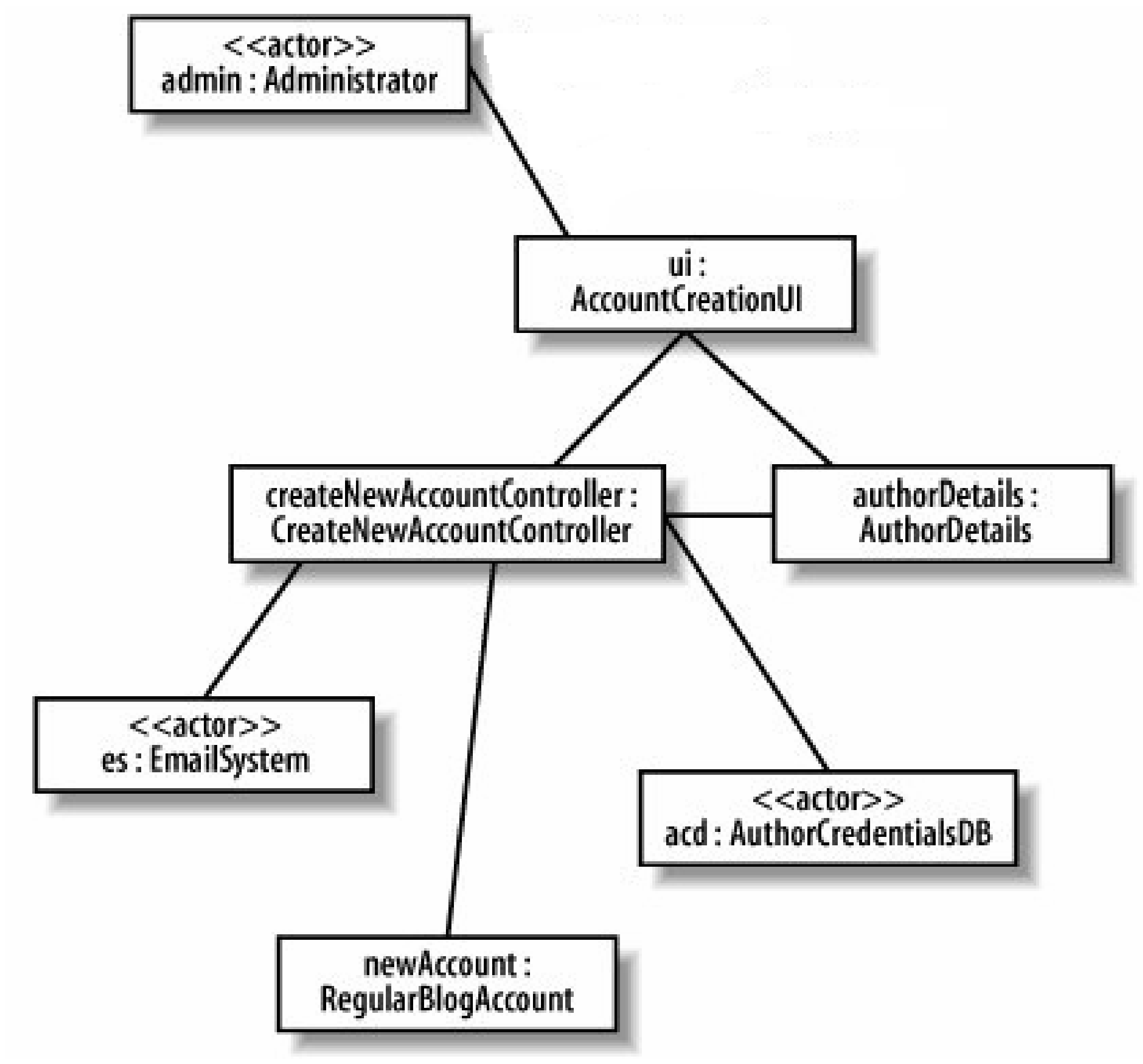

1
2
3
4
5
6
7
<<actor>> admin : Administrator
ui : AccountCreationUI
createNewAccountController : CreateNewAccountController
<<actor>> acd : AuthorCredentialsDB
<<actor>> es : EmailSystem
createNewBlogAccount
selectBlogAccountType(type)
enterAuthorDetails (author : AuthorDetails)
<<create>>
: AuthorDetails
clickSubmit()
createNewRegularBlogAccount(authorDetails : AuthorDetails)
checkAuthorDetails(authorDetails : AuthorDetails)
opt
[checked = true]
<<create(authorDetails)>>
newAccount : RegularBlogAccount
emailBlogDetails(newAccount : RegularBlogAccount)
sendEmail(email : Email)
<<destroy>>
accountCreatedNotification()

Building a collaboration diagram



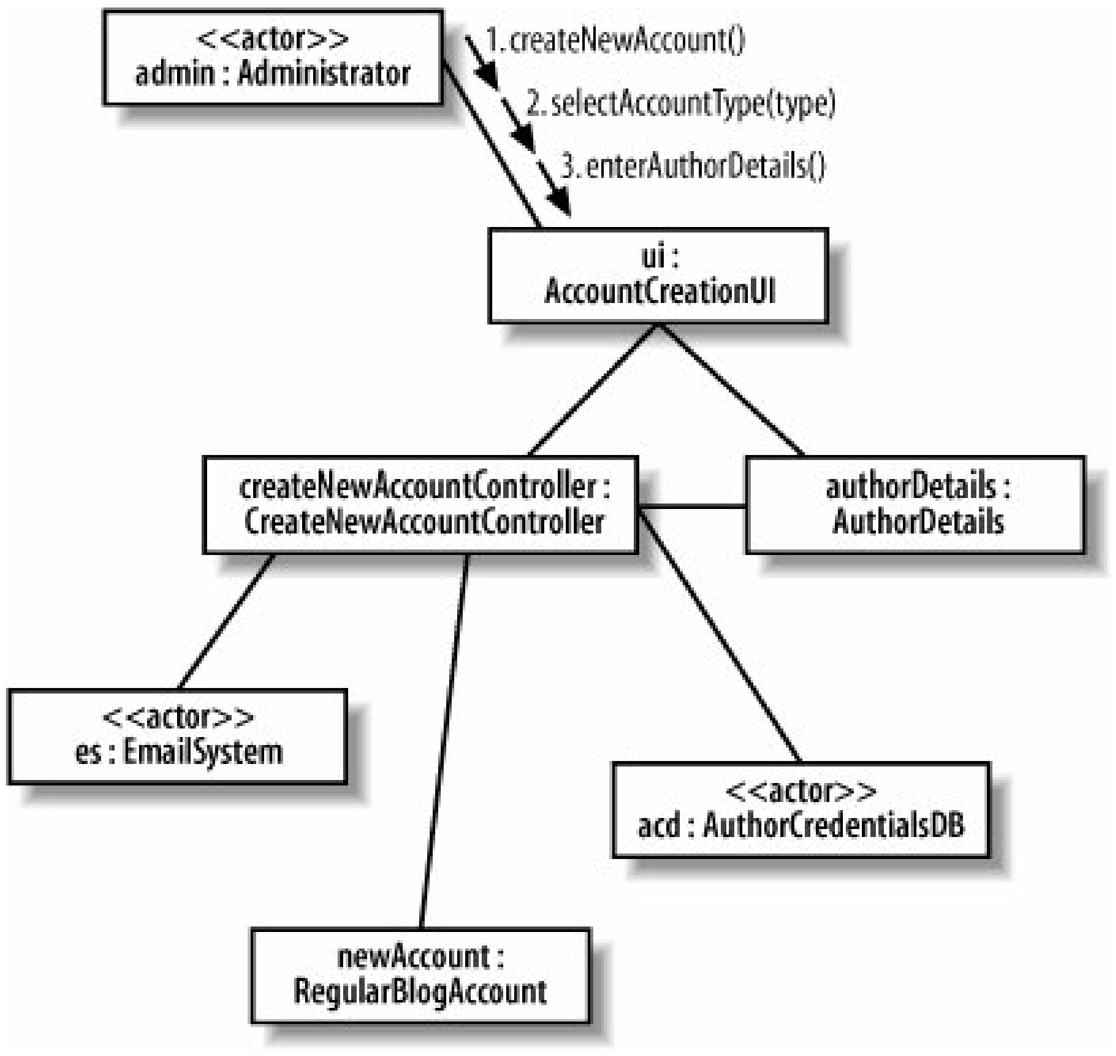
- **Adding the participating objects into the communication diagram**

Building a collaboration diagram


<<actor>>
admin : Administrator
ui : AccountCreationUI
createNewAccountController : CreateNewAccountController
authorDetails : AuthorDetails
<<actor>>
es : EmailSystem
<<actor>>
acd : AuthorCredentialsDB
newAccount : RegularBlogAccount

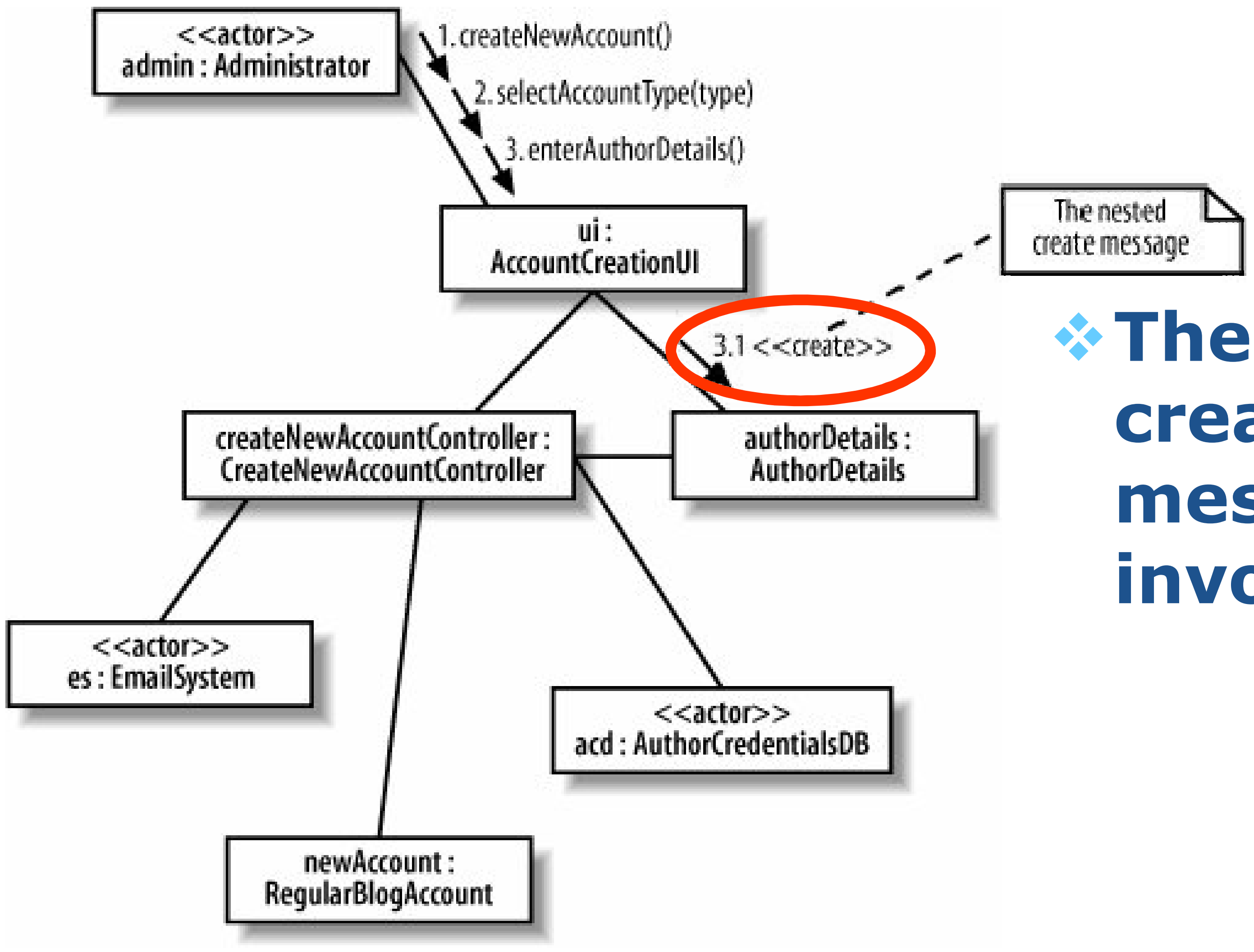
- **Adding links required for the message passing to the communication diagram**

Building a collaboration diagram



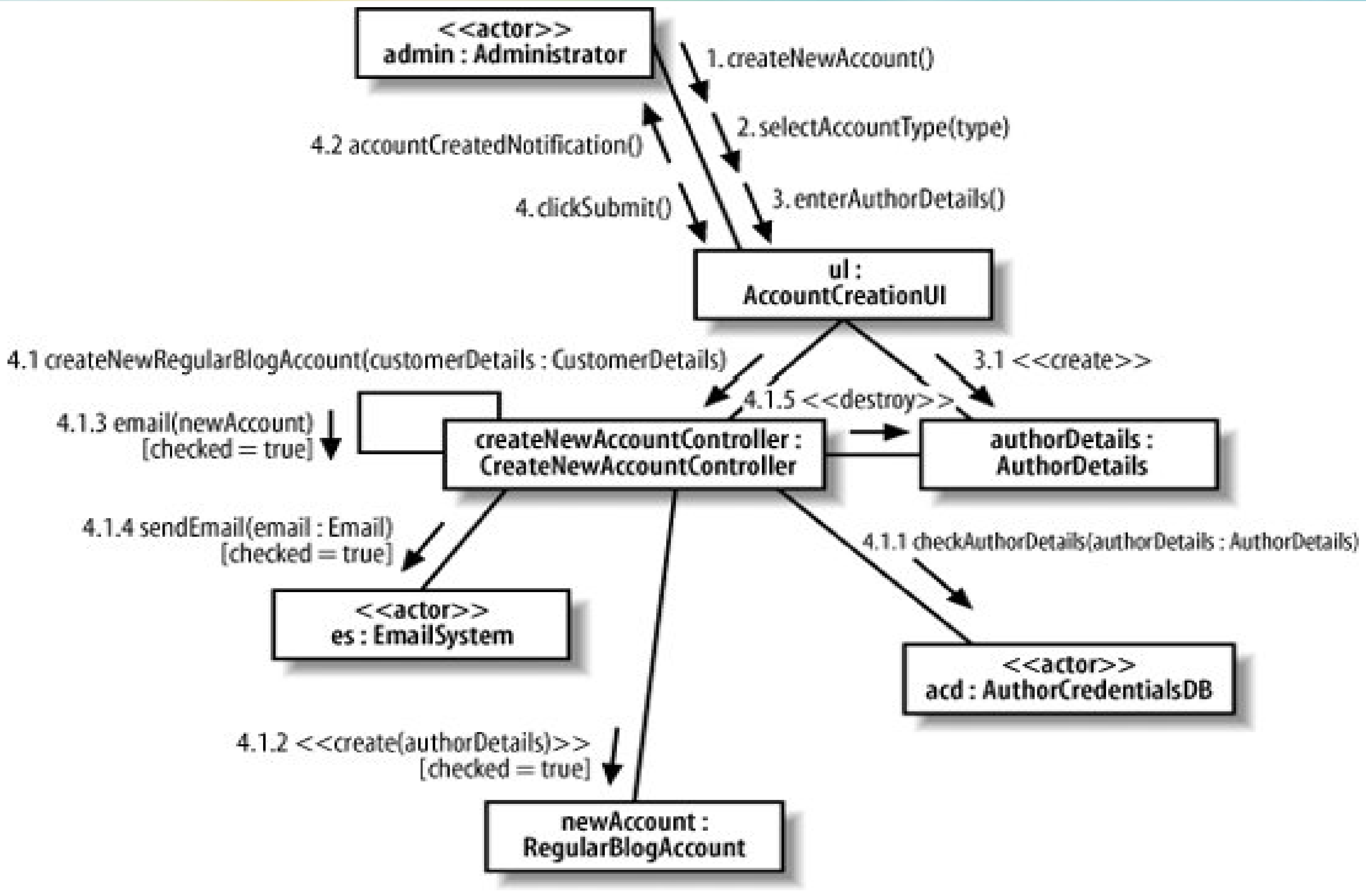
- **3 first separate messages are passed to the ui object, one after another**

Building a collaboration diagram


<<actor>> admin : Administrator
1. createNewAccount()
2. selectAccountType(type)
3. enterAuthorDetails()
ui : AccountCreationUI
The nested create message
3.1 <<create>>
createNewAccountController : CreateNewAccountController
authorDetails : AuthorDetails
<<actor>> es : EmailSystem
<<actor>> acd : AuthorCredentialsDB
newAccount : RegularBlogAccount

- **The nested create message is invoked**

Building a collaboration diagram


<<actor>>
admin : Administrator
1. createNewAccount()
2. selectAccountType(type)
3. enterAuthorDetails()
4.2 accountCreatedNotification()
4. clickSubmit()
ul :
AccountCreationUI
4.1 createNewRegularBlogAccount(customerDetails : CustomerDetails)
3.1 <<create>>
4.1.5 <<destroy>>
4.1.3 email(newAccount)
[checked = true]
createNewAccountController :
CreateNewAccountController
authorDetails :
AuthorDetails
4.1.4 sendEmail(email : Email)
[checked = true]
4.1.1 checkAuthorDetails(authorDetails : AuthorDetails)
<<actor>>
es : EmailSystem
<<actor>>
acd : AuthorCredentialsDB
4.1.2 <<create(authorDetails)>>
[checked = true]
newAccount :
RegularBlogAccount

Contents

I. Communication Diagrams

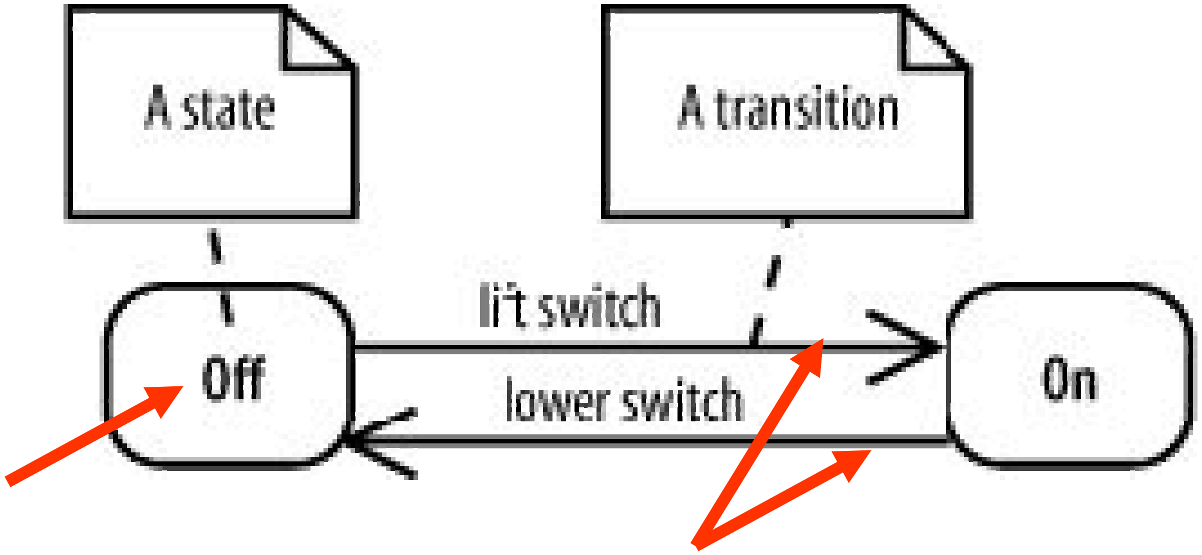
II Statechart Diagrams

Introduction

- Also called state diagrams or statechart diagrams.
- Are part of the logical view
- Used to model states of an object and the events causing state changes.
- Somehow similar to the activity diagrams

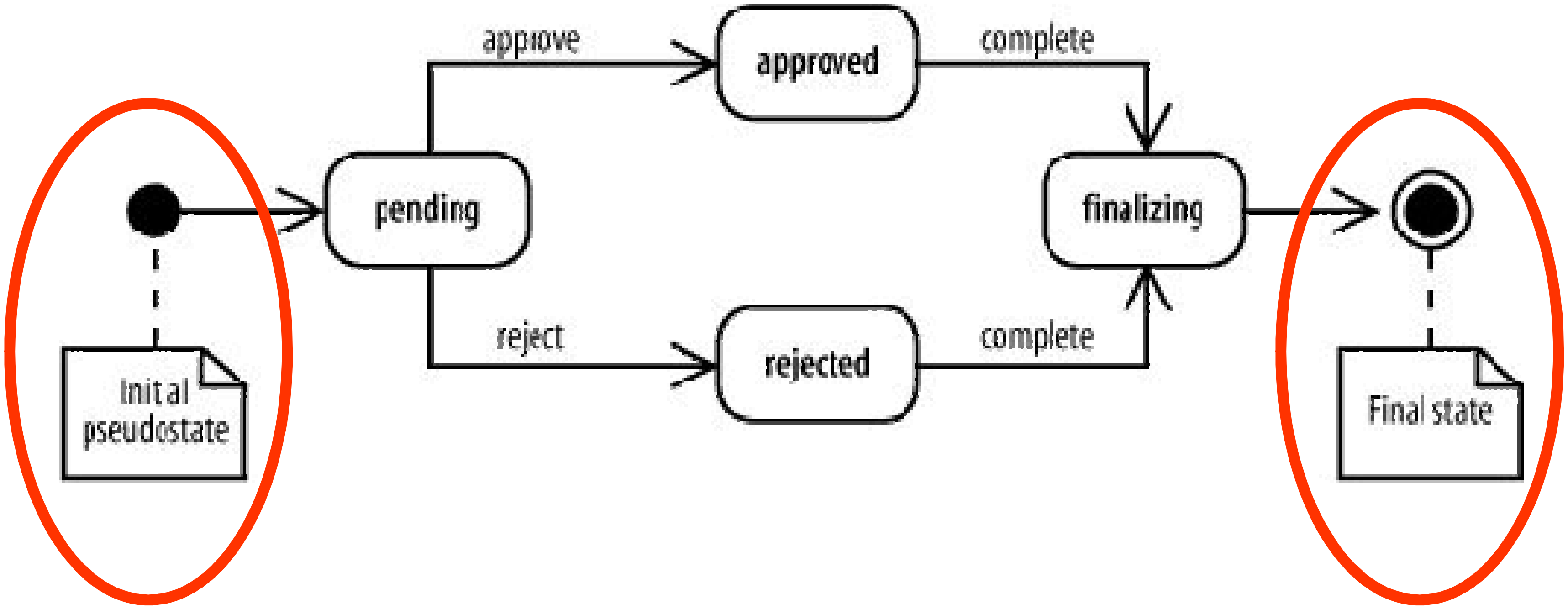
Notations

- **A state is represented by a rounded rectangle**
- **A transition is denoted by an arrow, with its trigger above**



Initial pseudostate & final state

- Used to mark **the life-time of the object** to be examined.

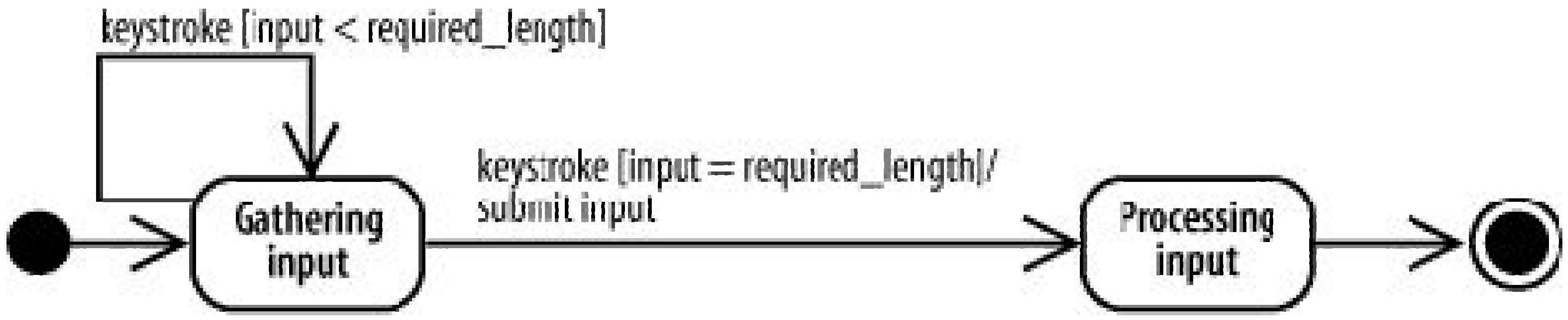


Transition Notation

- **The full notation for transition descriptions is trigger[guard]/behavior**
- **A *trigger* is an event that may cause a transition.**
- **A *guard* is a boolean condition that permits or blocks the transition.**
- **Transition *behavior* is an activity that executes during the transition.**

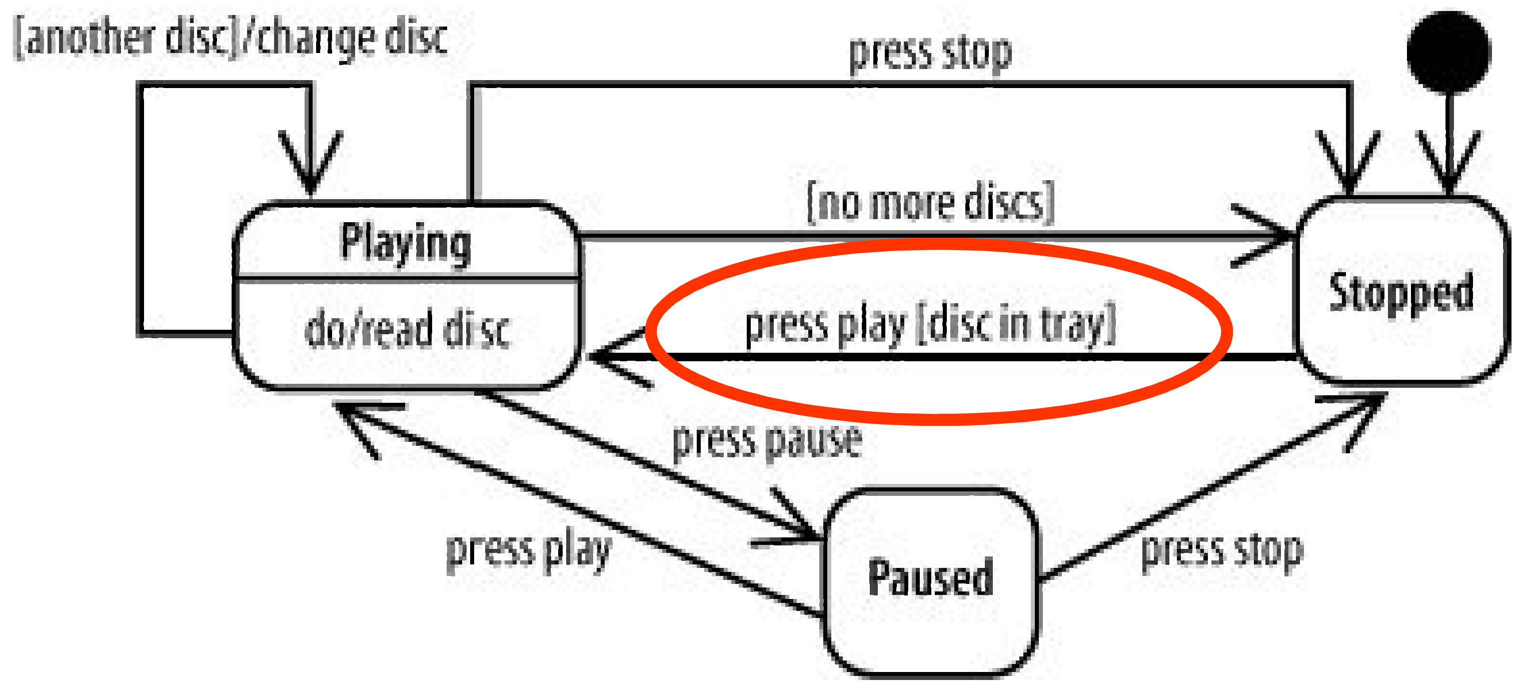
Transition Notation

- **An example about the user input process:**


keystroke [input < required_length]
Gathering input
keystroke [input = required_length]/ submit input
Processing input

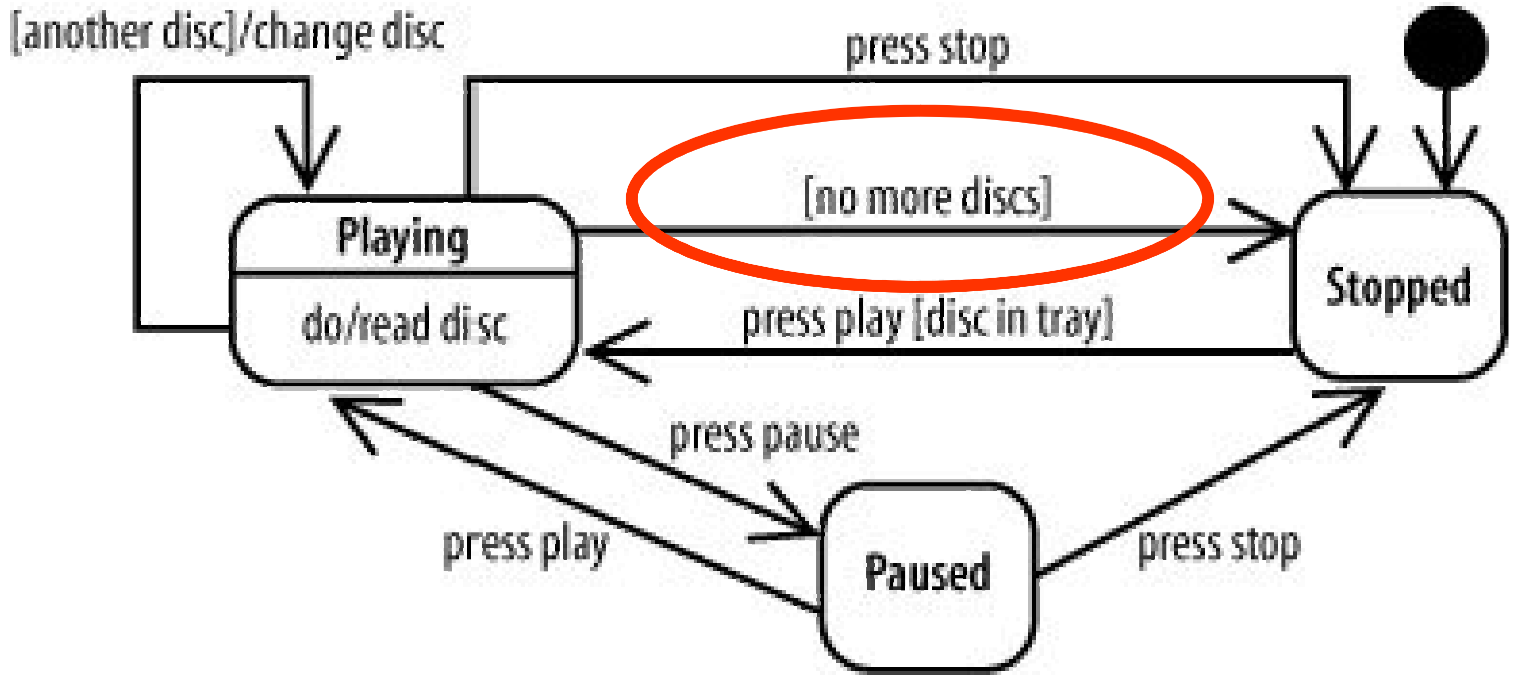
Transition variations

- **With a trigger name and a guard condition. A guard will block a transition if it evaluates to false**



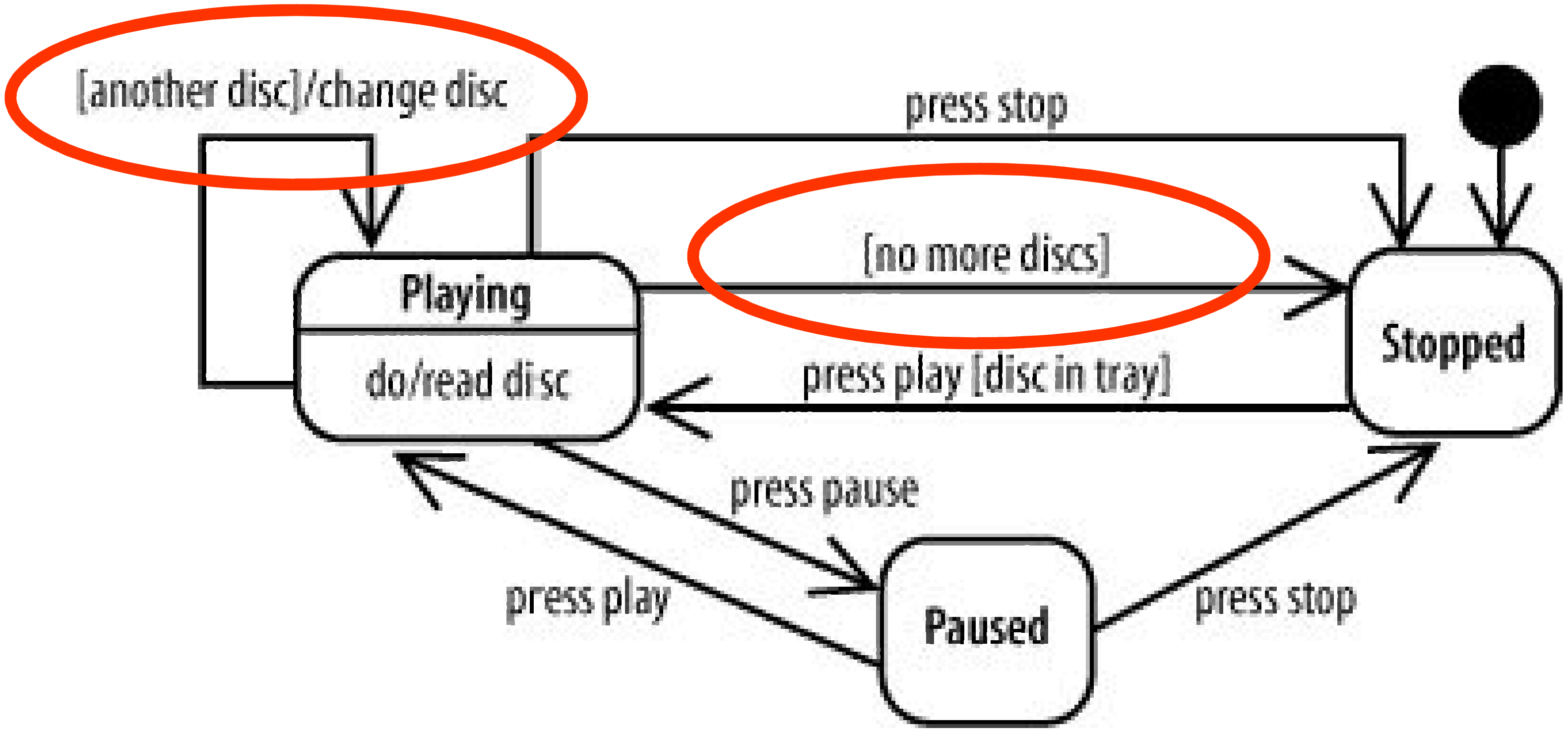
Transition variations

- **No trigger name: the transition is caused by the completion of the internal behavior.**



Transition variations

- **Guards are used to model choices between paths**

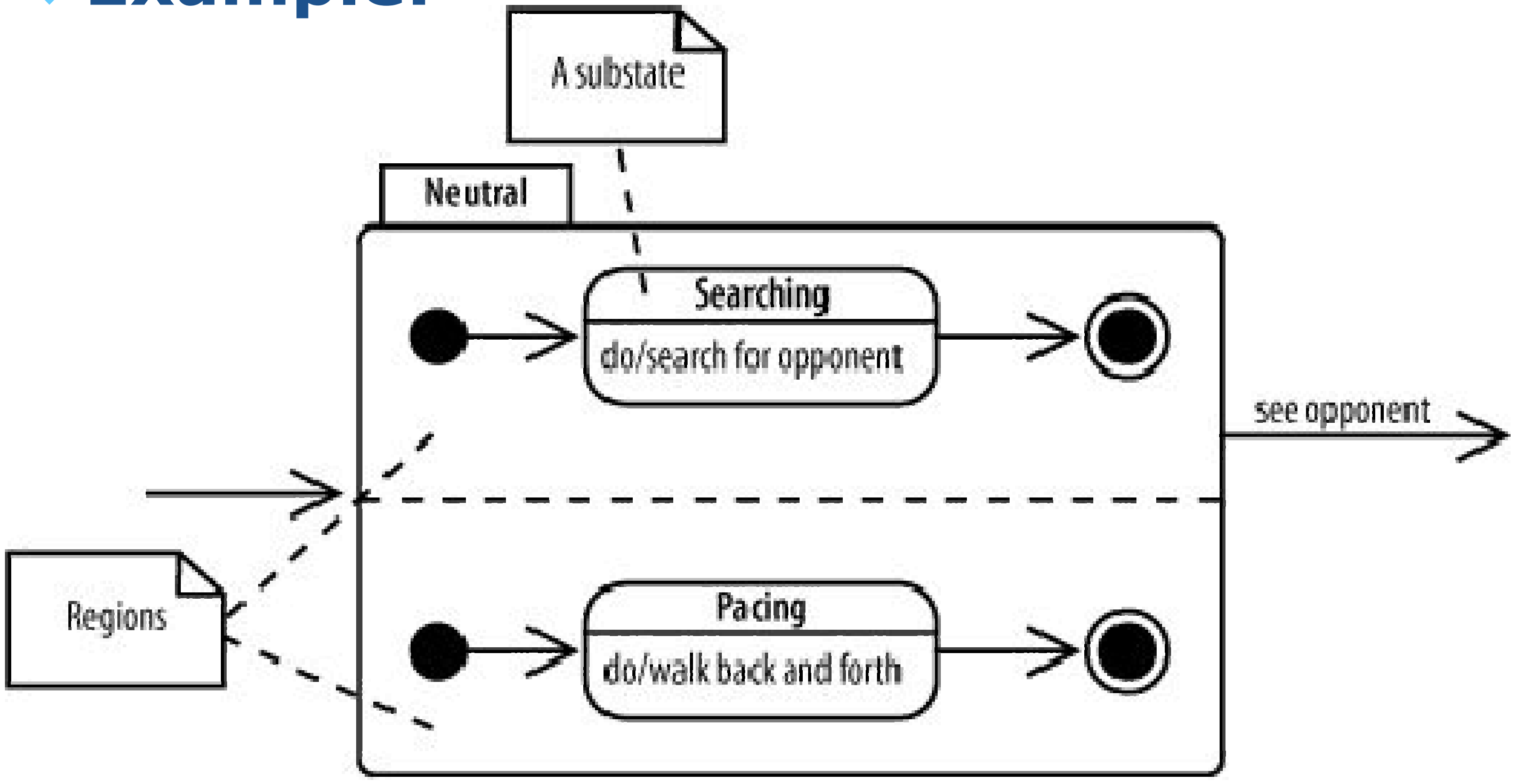


Composite states

- **UML allows concurrent states - being in multiple states at the same time.**
- **Composite states enable modeling this situation.**
- **A composite state is a state that contains one or more state diagrams.**
- **Each diagram belongs to a *region*, and regions are divided by a dotted line.**

Composite states

❖ **Example:**

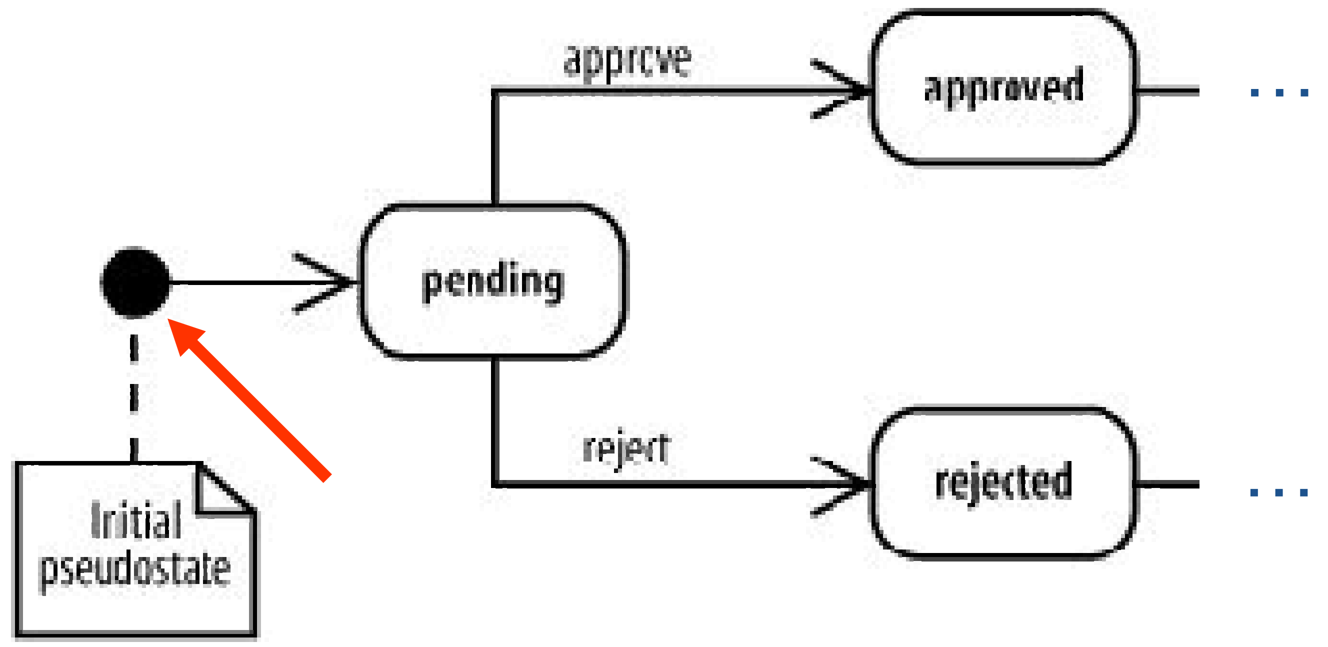


Composite states

- **How composite states work:**
  - When the composite state becomes active, the initial pseudostate of each region becomes active.
  - The contained state diagrams begin executing.
  - The contained state diagrams are interrupted if a trigger on the composite state occurs.

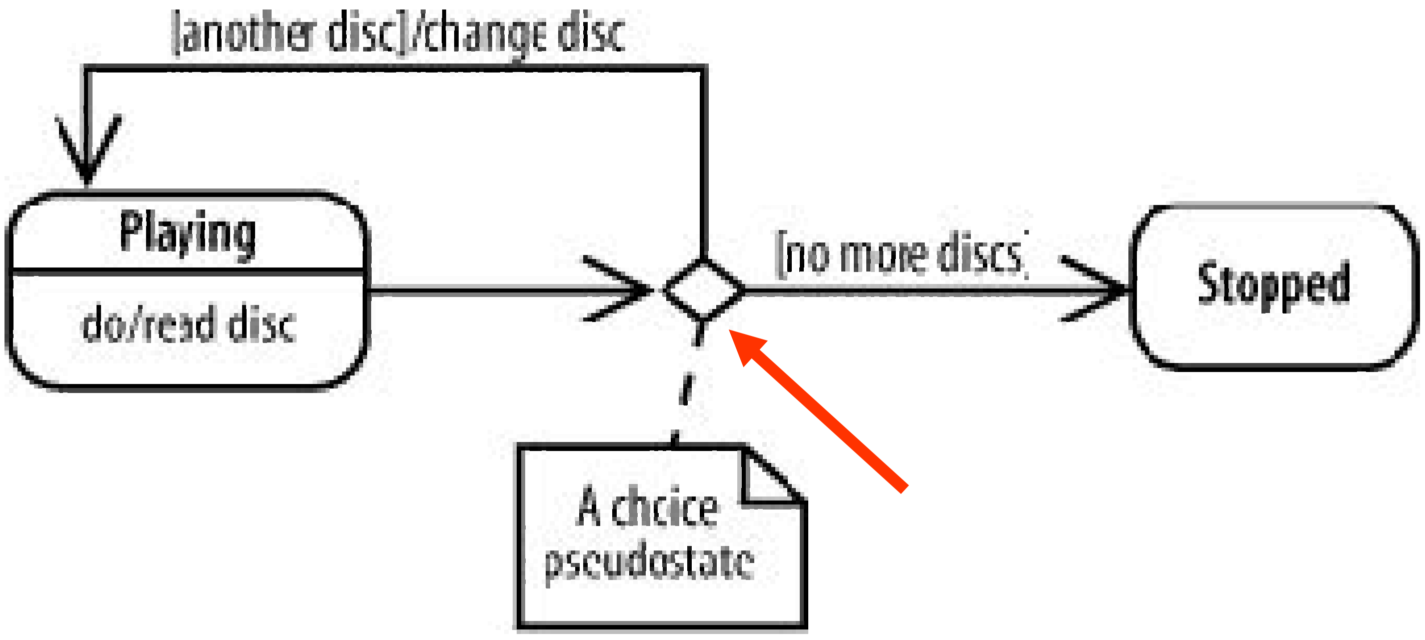
Pseudo states

- **Initial pseudostate: marks the beginning of the object's lifetime**



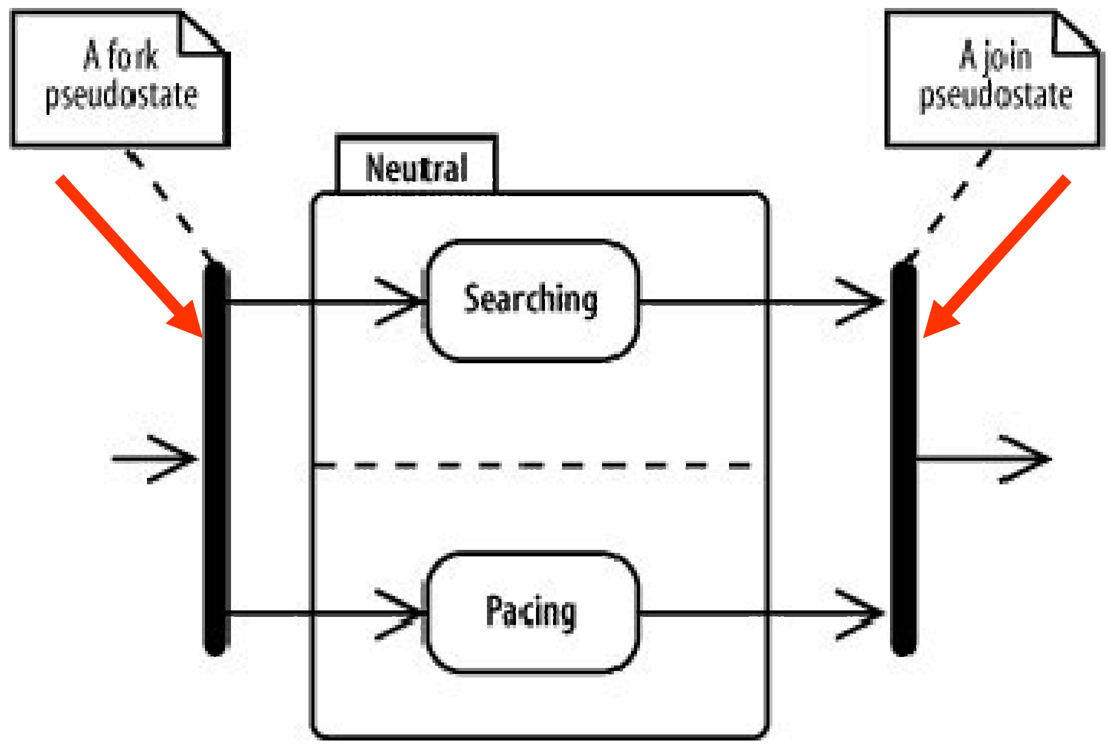
Pseudo states

- Choice **pseudostate: emphasizes that a Boolean condition determines which transition is followed.**

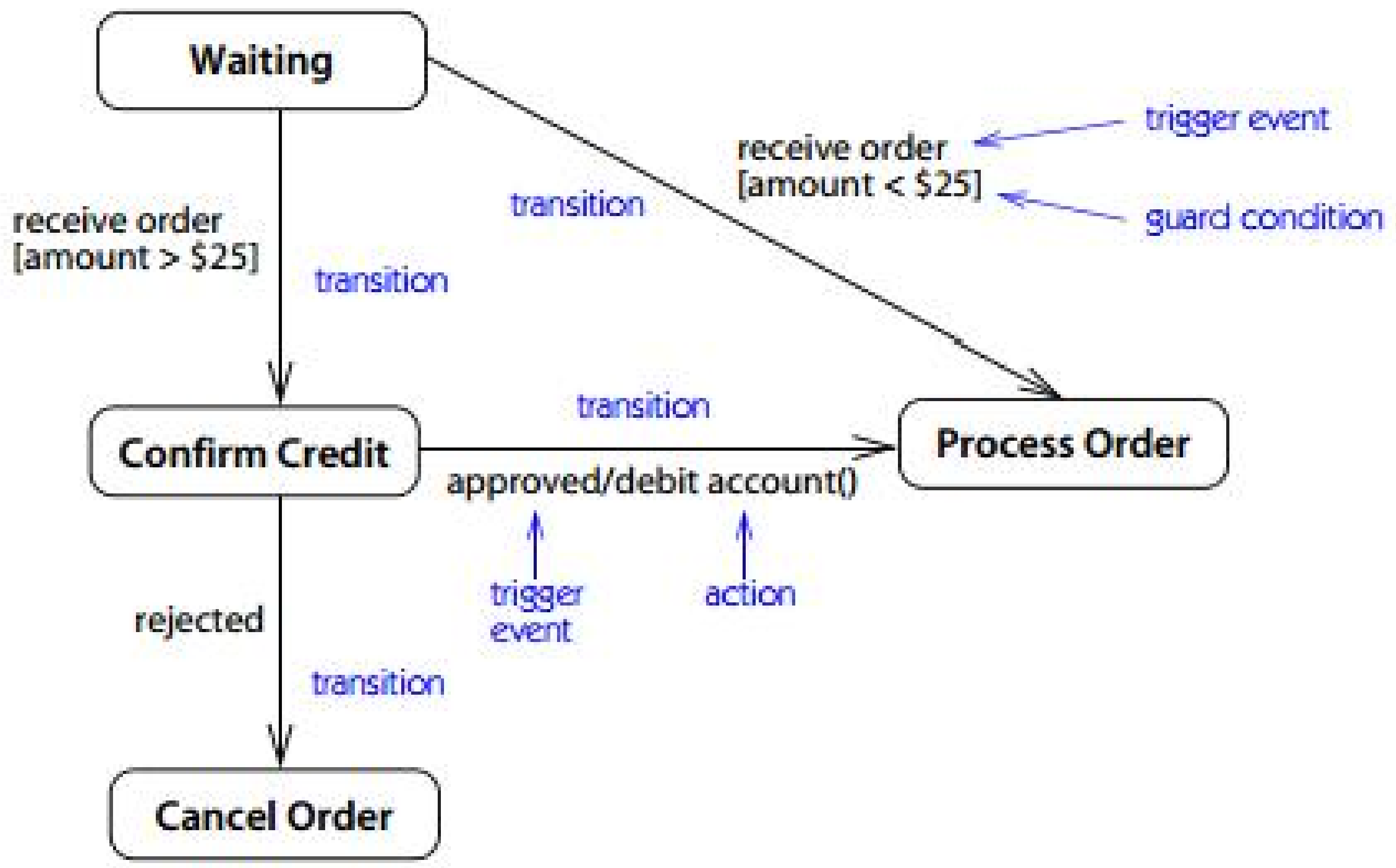


Pseudo states

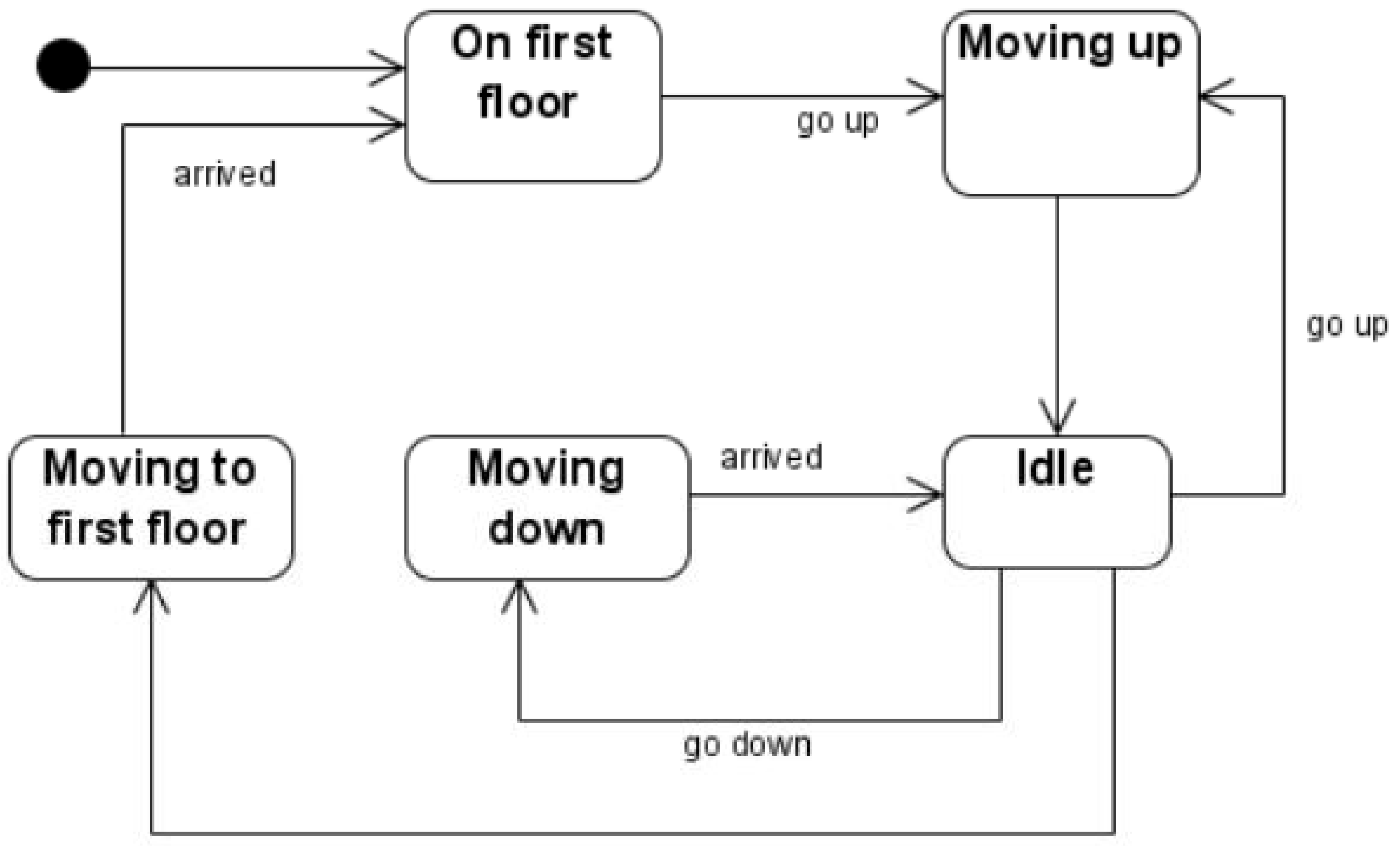
❖ Forks & joins **pseudostate: shows concurrent states.**



Example


Waiting
trigger event
receive order
[amount < $25]
guard condition
transition
receive order
[amount > $25]
transition
transition
Confirm Credit
Process Order
approved/debit account()
trigger
event
action
rejected
transition
Cancel Order

Example



A state machine diagram for a lift

Bài tập

- **Vẽ 1-2 lược đồ Statechart cho ứng dụng cụ thể (để mô tả các trạng thái của một đối tượng), ví dụ:**
  - Thang máy
  - Xe