Vietnam National University of HCMC
International University
School of Computer Science and Engineering

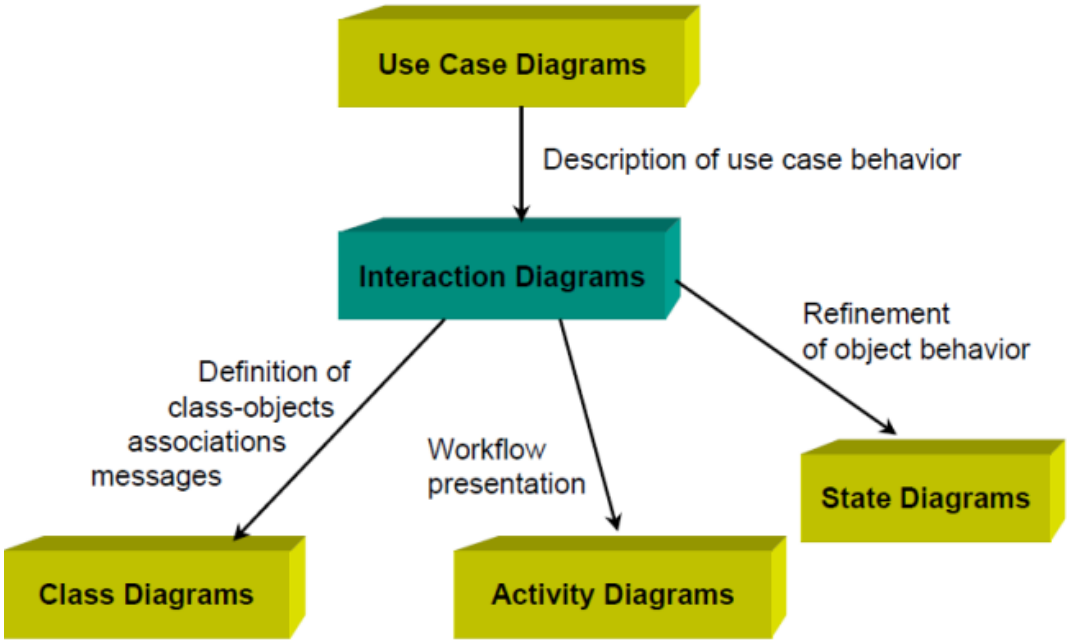
Object – Oriented Analysis and Design

Sequence Diagram

Instructor: Le Thi Ngoc Hanh, Ph.D

ltnhanh@hcmiu.edu.vn

Role of Interaction Diagram in UML



Interaction Diagrams

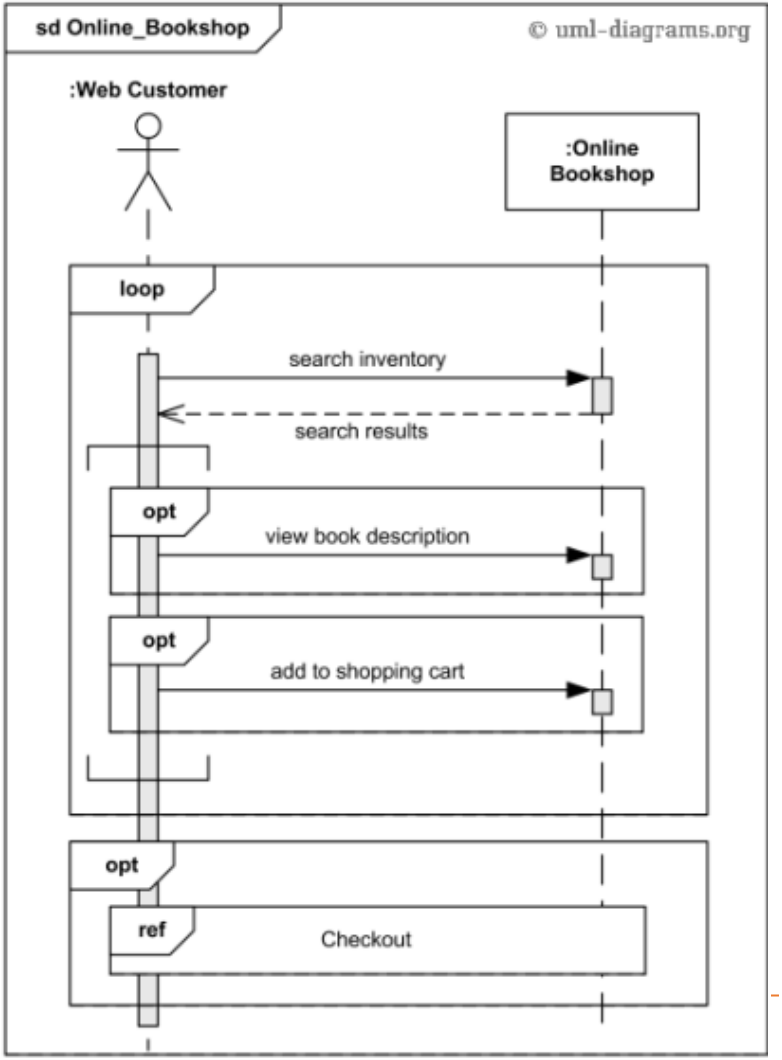
- An interaction diagram typically captures the behavior of a user goal use case.
- Sequence diagrams: emphasize the order or concurrency of the interactions
- Collaboration diagrams: emphasize the interacting objects

What do Sequence Diagram model?

- Capture the interaction between objects in a context of a collaboration.
- Show object instances that play the roles defined in a collaboration
- Show the order of the interaction visually by using the vertical axis of the diagram to represent time what message are sent and when.

Example: Online bookstore

1. The customer begins the interaction by searching for a book by title.
2. The system will return all books with that title.
3. The customer can look at the book description.
4. The customer can place a book in the shopping cart.
5. The customer can repeat the interaction as many times as desired.
6. The customer can purchase the items in the cart by checking out.


sd Online_Bookshop
© uml-diagrams.org
:Web Customer
:Online Bookshop
loop
search inventory
search results
opt
view book description
opt
add to shopping cart
opt
ref
Checkout

Participants in Sequence Diagram

- A sequence diagram is made up of a collection of participants.
- Participants – the system parts that interact each other during the sequence
- Classes or Objects: each object (class) in the interaction is represented by its named icon along the top of the diagram.

Lifeline

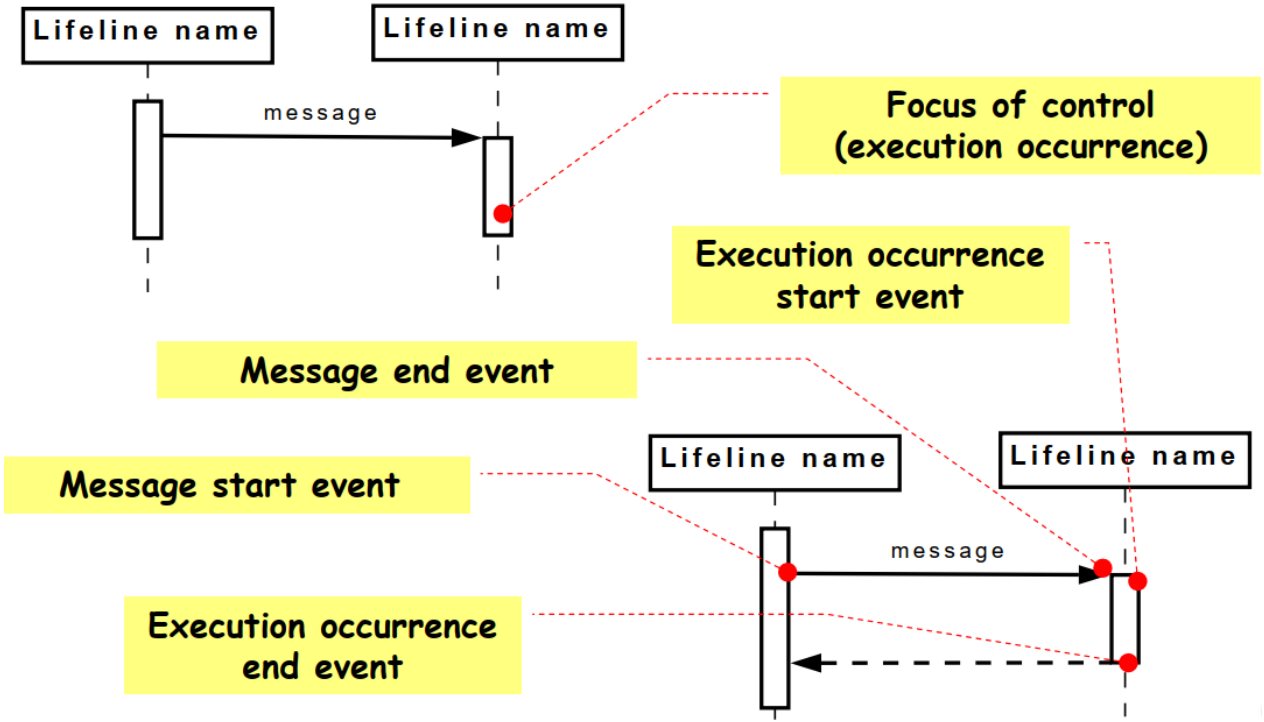
- Sequence diagrams are organized according to time
- Each participant has a corresponding lifeline
- **Lifelines**: each vertical dotted line is a lifeline, representing the time that an object exists

Examples of Lifeline names

Syntax	Explanation
seoclecturer	An object named secolecturer
seoclecturer : Lecturer	An object names seoclecturer of class Lectuer.
:Lecturer	An anonymous object of class Lecturer
lecturer[i]	The object lecturer that is selected by the index value *i*.
s ref sd3	A subsystem s whose internal interaction is shown in sequence diagram sd3 (decomposition).
self	The connectable element that owns the interaction shown in the sequence diagram

Types of Lifelines

Message and Focus of Control



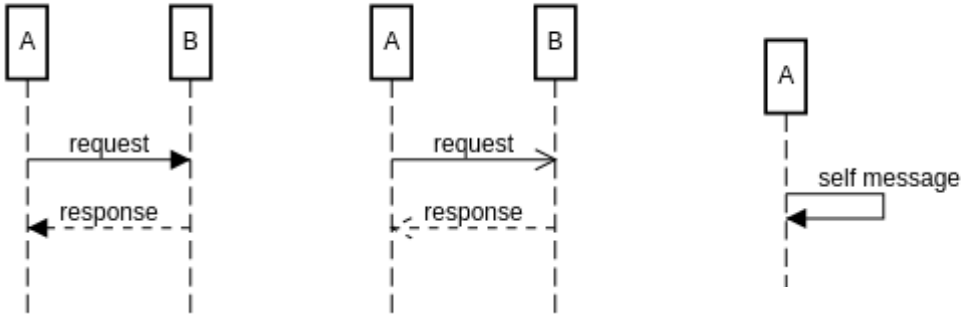
Messages

- Messages (or signals) on a sequence diagram are specified using an arrow from the participant (message caller) that wants to pass the message to the participant (message receiver) that is to receive the message.
- A message is represented as an arrow going from the sender to the top of the focus of control (i.e., execution occurrence) of the message on the receiver's lifeline.

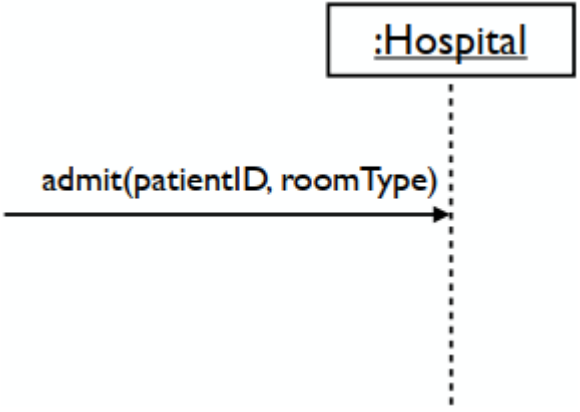
Message type notations

Synchronous Or Call	Asynchronous	Creation	Reply (Return)

Message

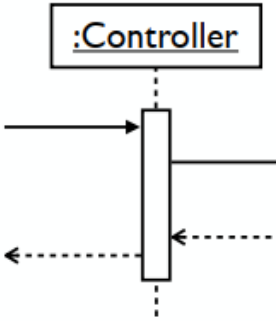


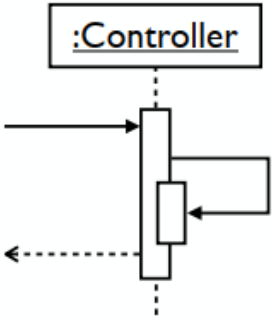
Message



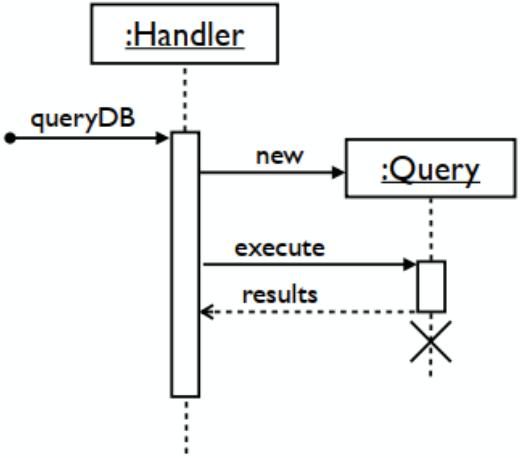
Focus of control

- Activation: thick box over object's lifeline.
- Nest activations to indicate an object calling itself.

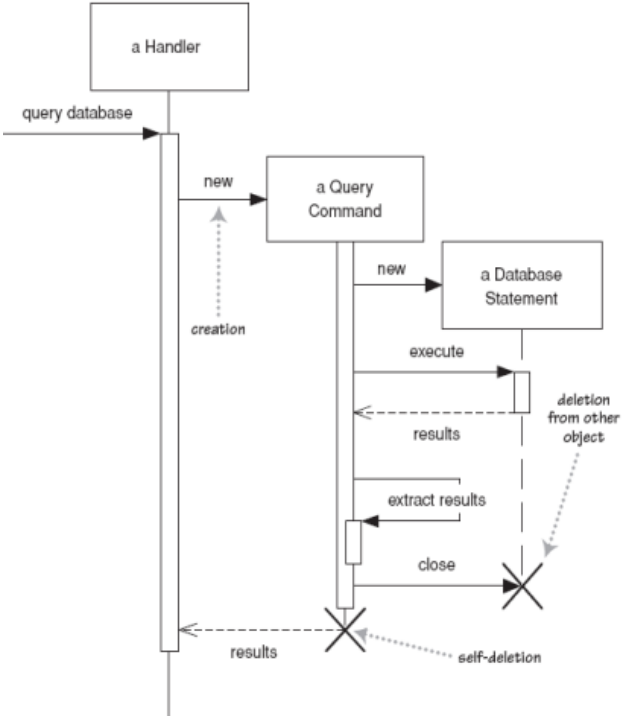




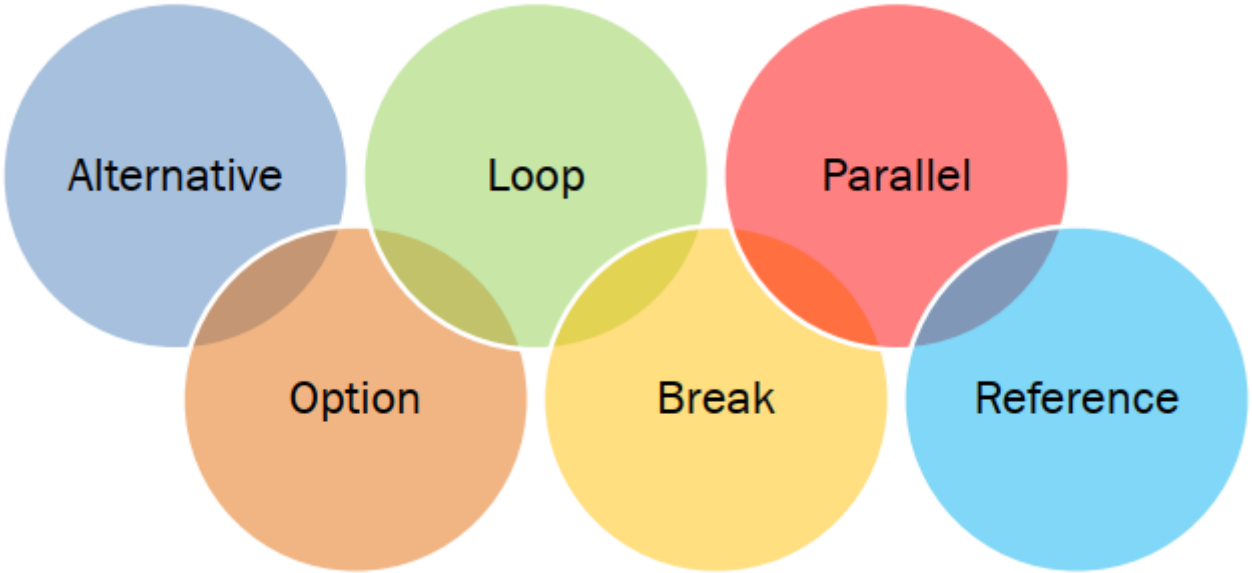
Object creation/deletion



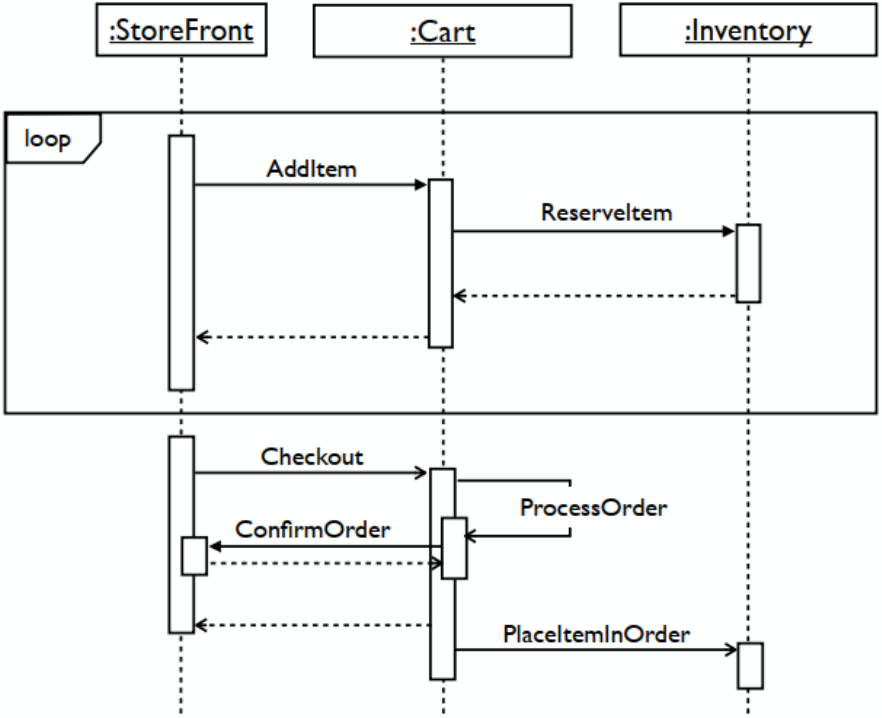
Example


a Handler
query database
new
a Query Command
creation
new
a Database Statement
execute
deletion from other object
results
extract results
close
results
self-deletion

Control Structure

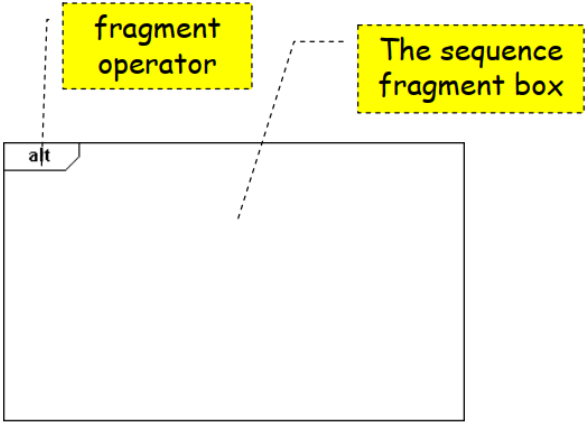


Example


:StoreFront
:Cart
:Inventory
loop
AddItem
ReserveItem
Checkout
ProcessOrder
ConfirmOrder
PlaceItemInOrder

Sequence Fragment

- UML 2.0 introduces Sequence (or Interaction) Frames
- A sequence fragment is represented as a box, called a **combined fragment**, which encloses a portion of the interactions within a sequence diagram
- The **fragment operator** (in the top left cornet) indicates the type of fragment
- Fragment types: **ref**, **assert**, **loop**, **break**, **alt**, **opt**, **neg**

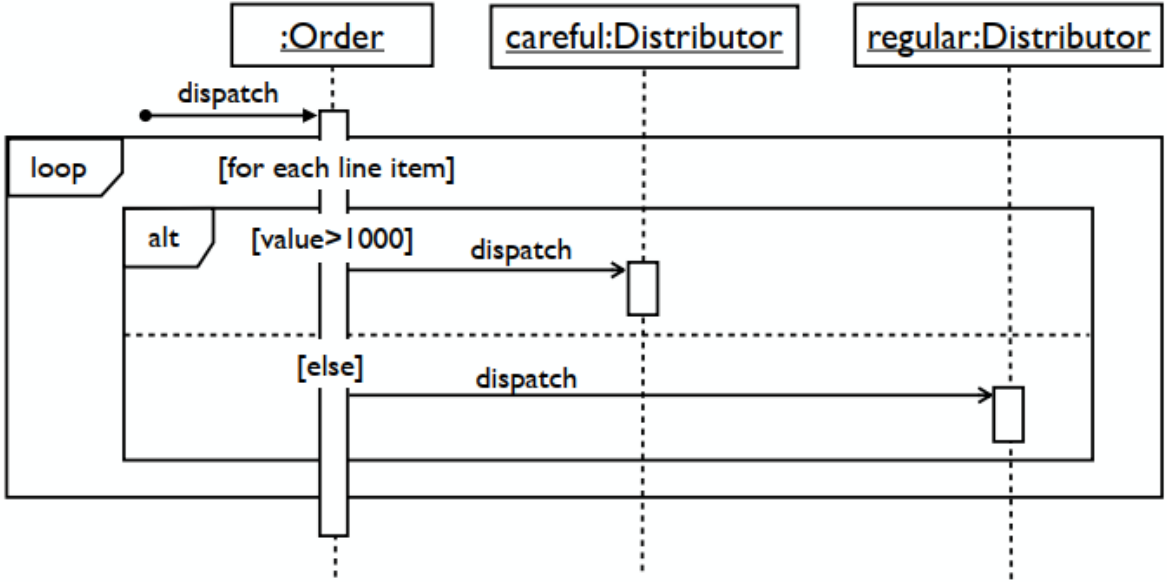


- Sequence fragments make it easier to create and maintain accurate sequence diagrams

Common fragment types

Operator	Meaning
alt	**Alternative multiple fragments:** only the one whose condition is true will execute.
opt	**Optional:** the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	**Parallel:** each fragment is run in parallel.
loop	**Loop:** the fragment may execute multiple times, and the guard indicates the basis of iteration.
region	**Critical region:** the fragment can have only one thread executing it at once.
neg	**Negative:** the fragment shows an invalid interaction.
ref	**Reference:** refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return value.
sd	**Sequence diagram:** used to surround an entire sequence diagram.

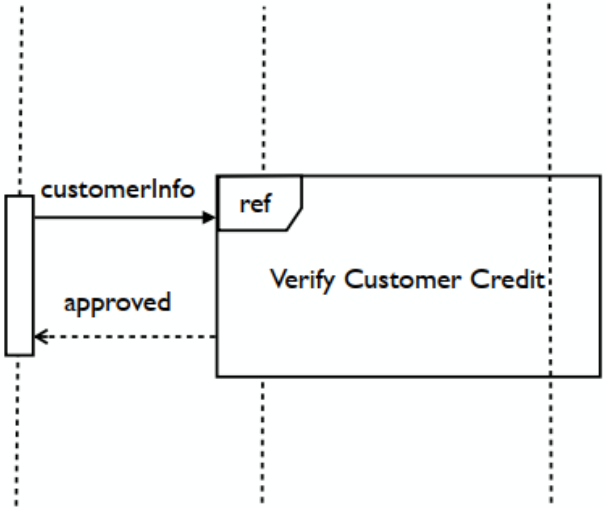
Examples of fragments

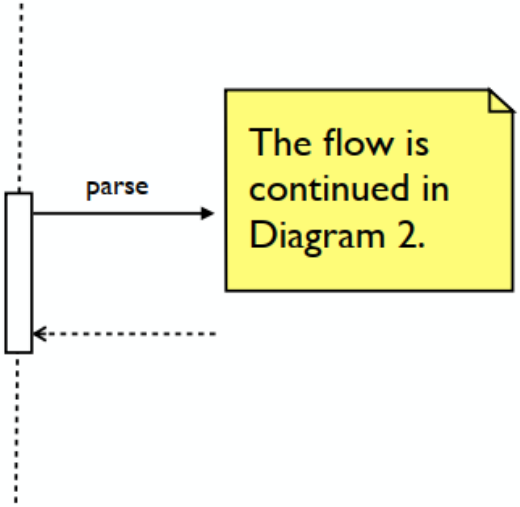


- **Frame**: a box around part of a sequence diagram
  - if → (opt) [condition]
  - if/else → (alt) [condition], separated by horizontal dashed line
  - loop → (loop) [condition or items to loop over]

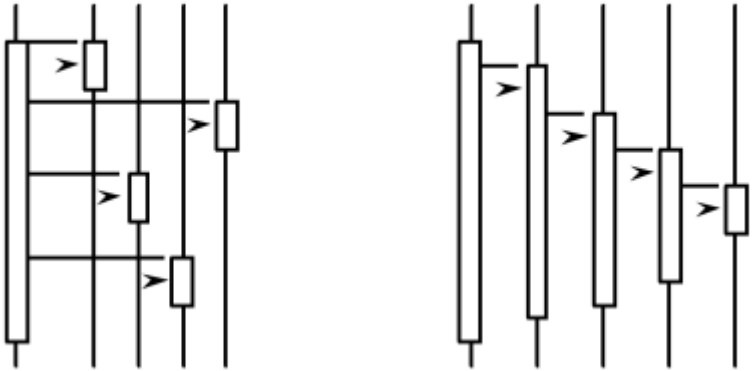
Linking sequence diagrams

- If one sequence diagram is too large or refers to another diagram:
  - An unfinished arrow and comment.
  - A **ref** frame that names the other diagram.

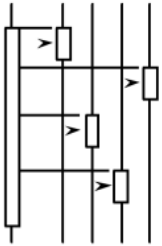


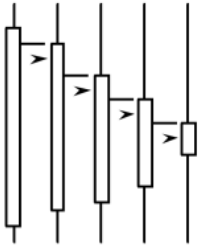


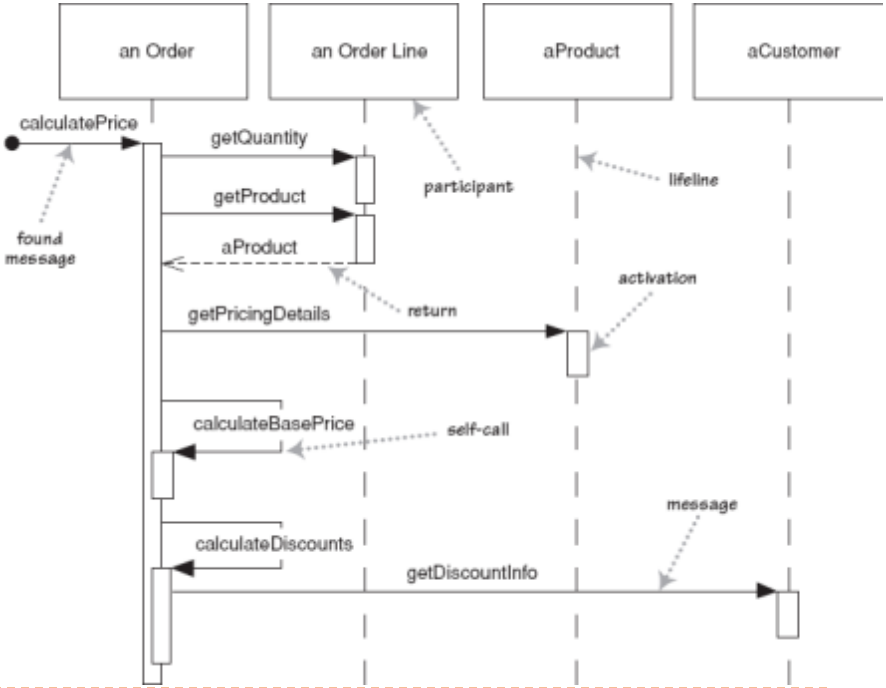
Forms of System Control

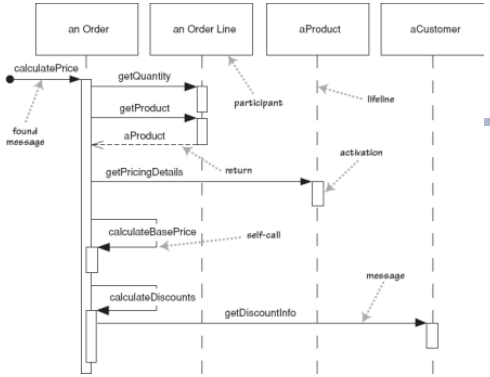


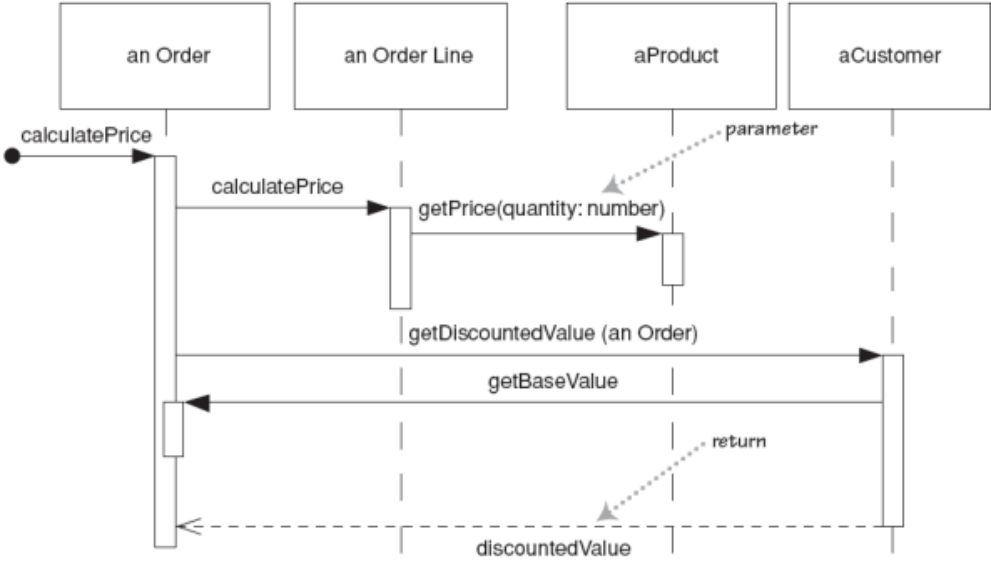
Distributed vs Centralized






an Order
an Order Line
aProduct
aCustomer
calculatePrice
getQuantity
getProduct
aProduct
getPricingDetails
calculateBasePrice
calculateDiscounts
getDiscountInfo
found message
participant
lifeline
return
activation
self-call
message


an Order
an Order Line
aProduct
aCustomer
calculatePrice
getQuantity
getProduct
aProduct
getPricingDetails
calculateBasePrice
calculateDiscounts
getDiscountInfo
found message
participant
lifeline
return
activation
self-call
message


an Order
an Order Line
aProduct
aCustomer
calculatePrice
calculatePrice
parameter
getPrice(quantity: number)
getDiscountedValue (an Order)
getBaseValue
return
discountedValue

How to produce sequence diagram?

- Decide on context: Identify behavior (or use case) to be specified
- Identify structural elements:
  - Model objects (classes)
  - Model lifelines
  - Model activations
  - Model message

How do interaction diagrams help?

- Check use cases
- Check class can provide an operation
  - Showing how a class realize some operation by interacting with other objects
- Describe design pattern
  - Parameterizing by class provides a scheme for a generic interaction
- Describe how to use a components
  - Capturing how components can interact

UML Sequence Diagram