UML (Sequence Diagrams)

- In every interesting systems, objects don't just sit idle, they interact with one another
- UML is used to model both static and dynamic aspects of a system
- Dynamic aspect is modeled using interactions
- An *interaction* is a set of messages exchanged among a set of objects in order to accomplish a specific goal
- Interaction diagrams describe how a group of objects collaborate in some behavior

Introduction

- *Interaction diagrams* ...
  - Aid the developer visualize the system as it is running.
  - Show selected sequences of message traffic between objects.
- After class diagrams, interaction diagrams are possibly the most widely used diagrams in UML.
- Interaction diagrams commonly contain participants (*objects*), *links* and *messages*.

Message

- Objects communicate with each other by sending *messages*.

- Sending a *message* is another name for a member function call.

- Some C++ examples of member function calls ...
  - objectName.messageName( );
  - objectPointer->messageName( );
  - (*objectPointer).messageName( );

Messages

- Graphically a message is shown as a directed line , almost always including the name of the operation

display

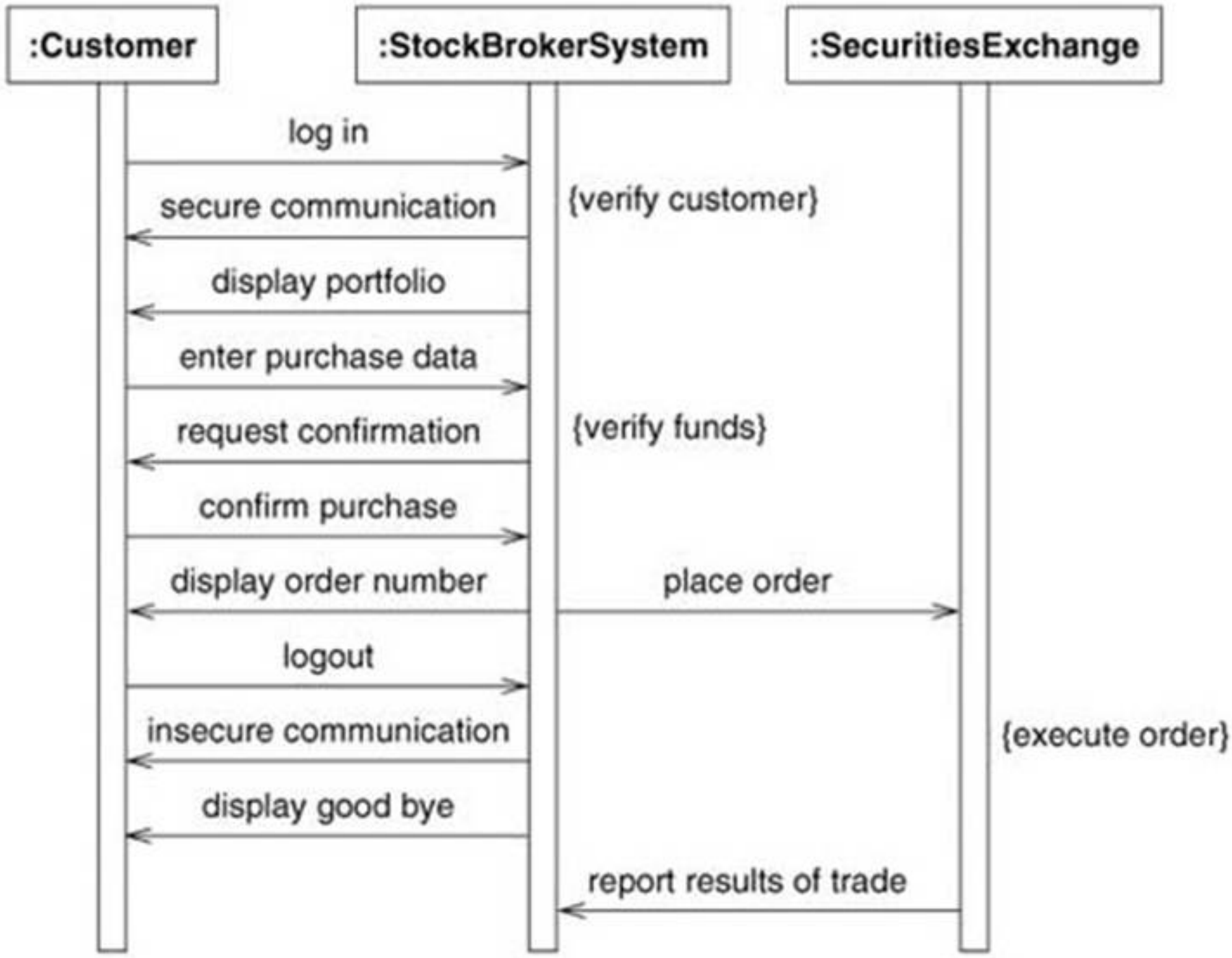
- A ***call*** is the most common type of message.
- The ***return*** of data as a result of a function call is also considered a message.
- A message may result in a change of state for the receiver of the message.
- The receipt of a message is considered an instance of an *event*.

- We can model each interaction by emphasizing its
  - time ordering of messages, or
  - sequencing of messages in the context of some structural organization of objects.

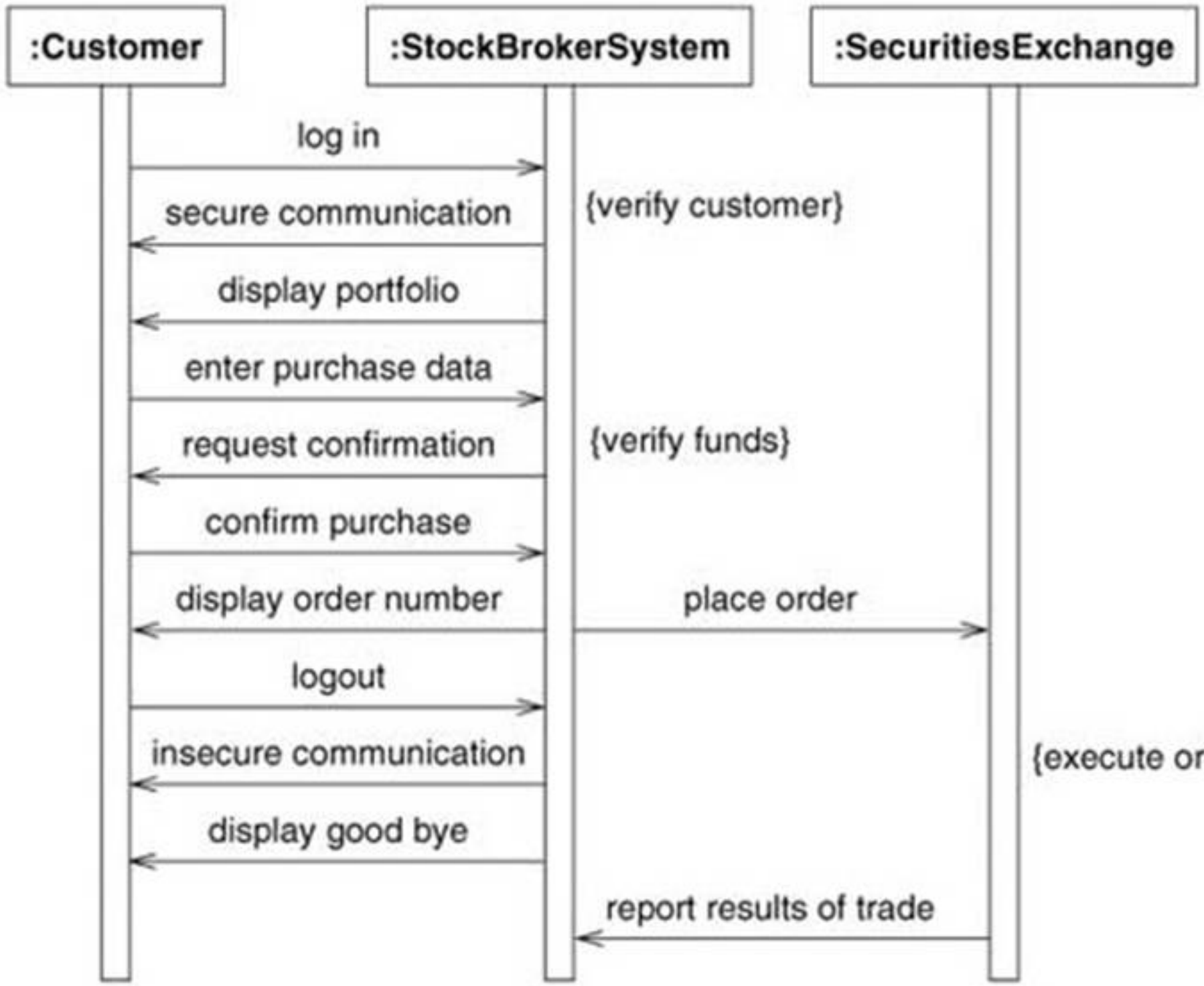
- The *sequence diagram* is the most commonly used UML interaction diagram
- A *sequence diagram* captures the behavior of a group of objects in a single *scenario*.
- A *sequence diagram is an* interaction diagram that emphasizes the time ordering of messages

Sequence Diagrams

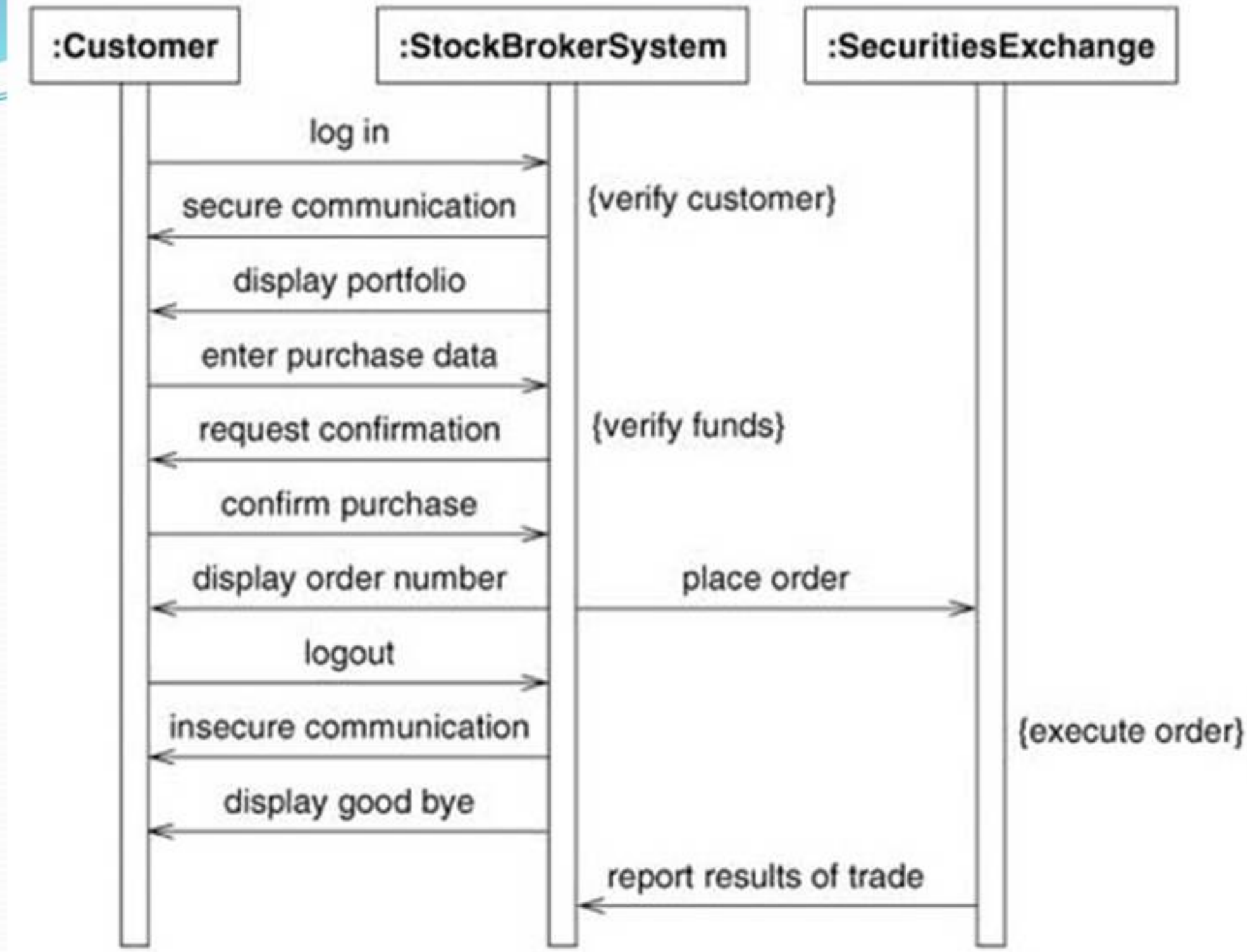
- Sequence diagrams are build around an X-Y axis.



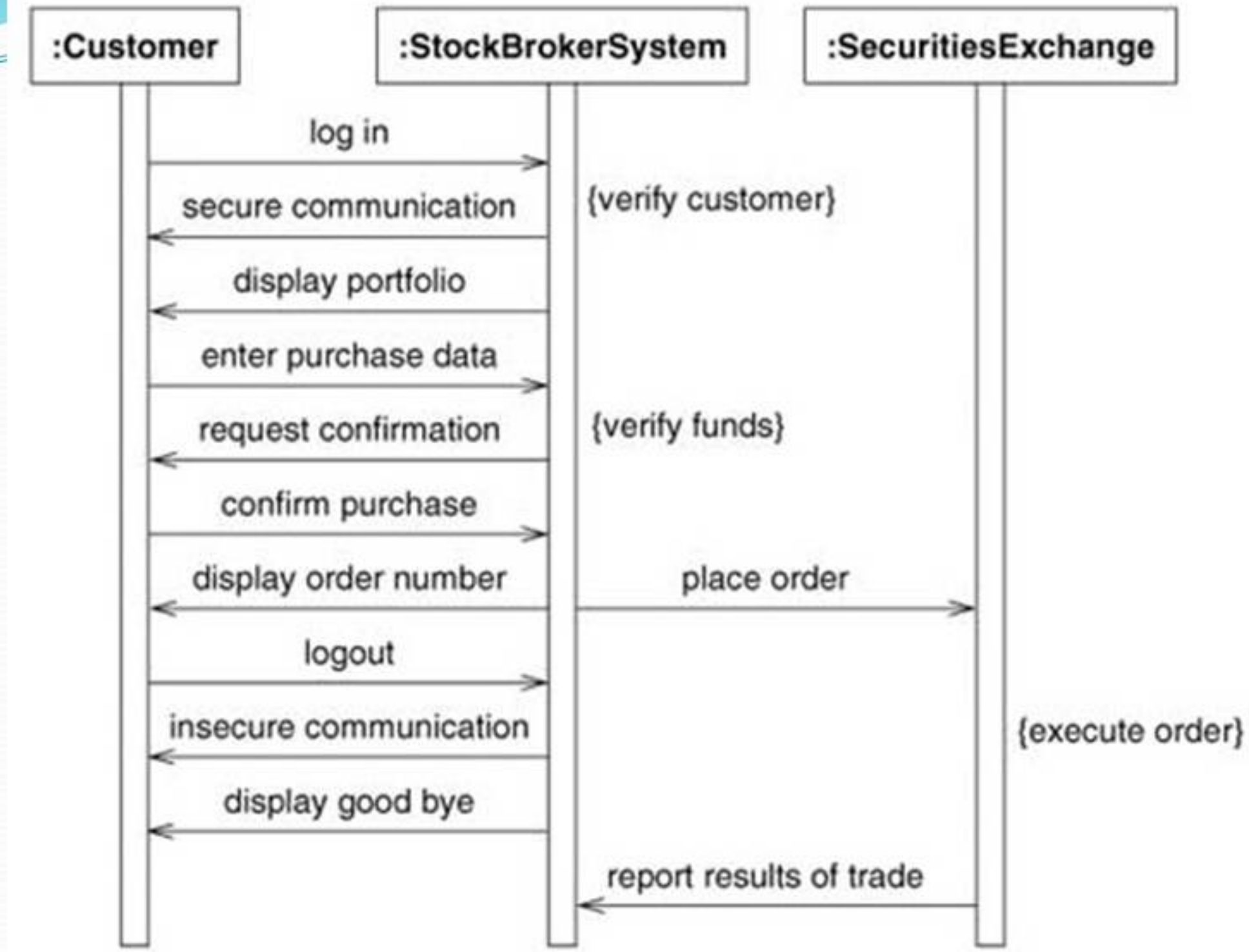
- Participants (Objects) are aligned (in most cases) at the top of the diagram, parallel to the X axis.



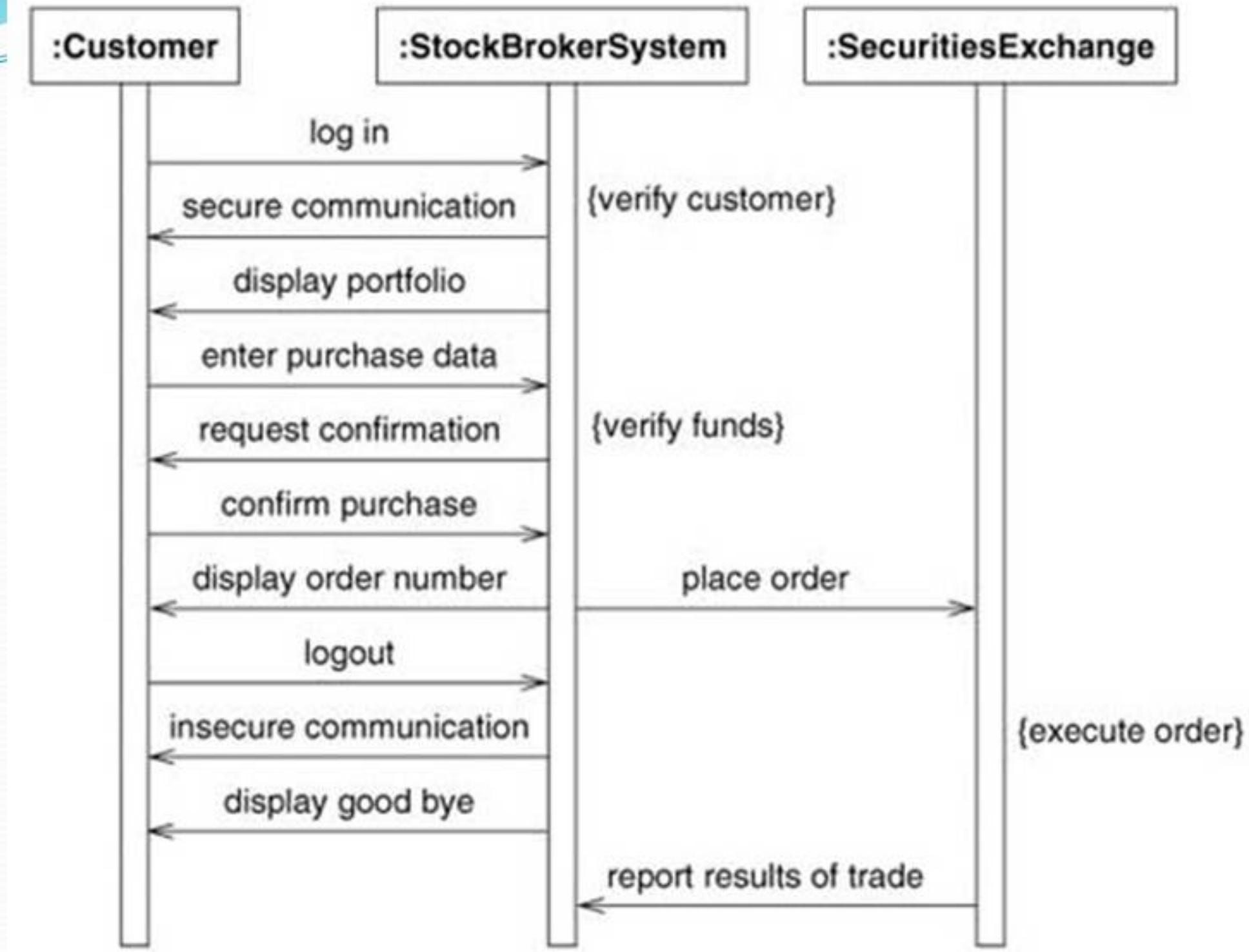
- Messages travel (in most cases) parallel to the X axis.
- Time passes from top to bottom along the Y axis.



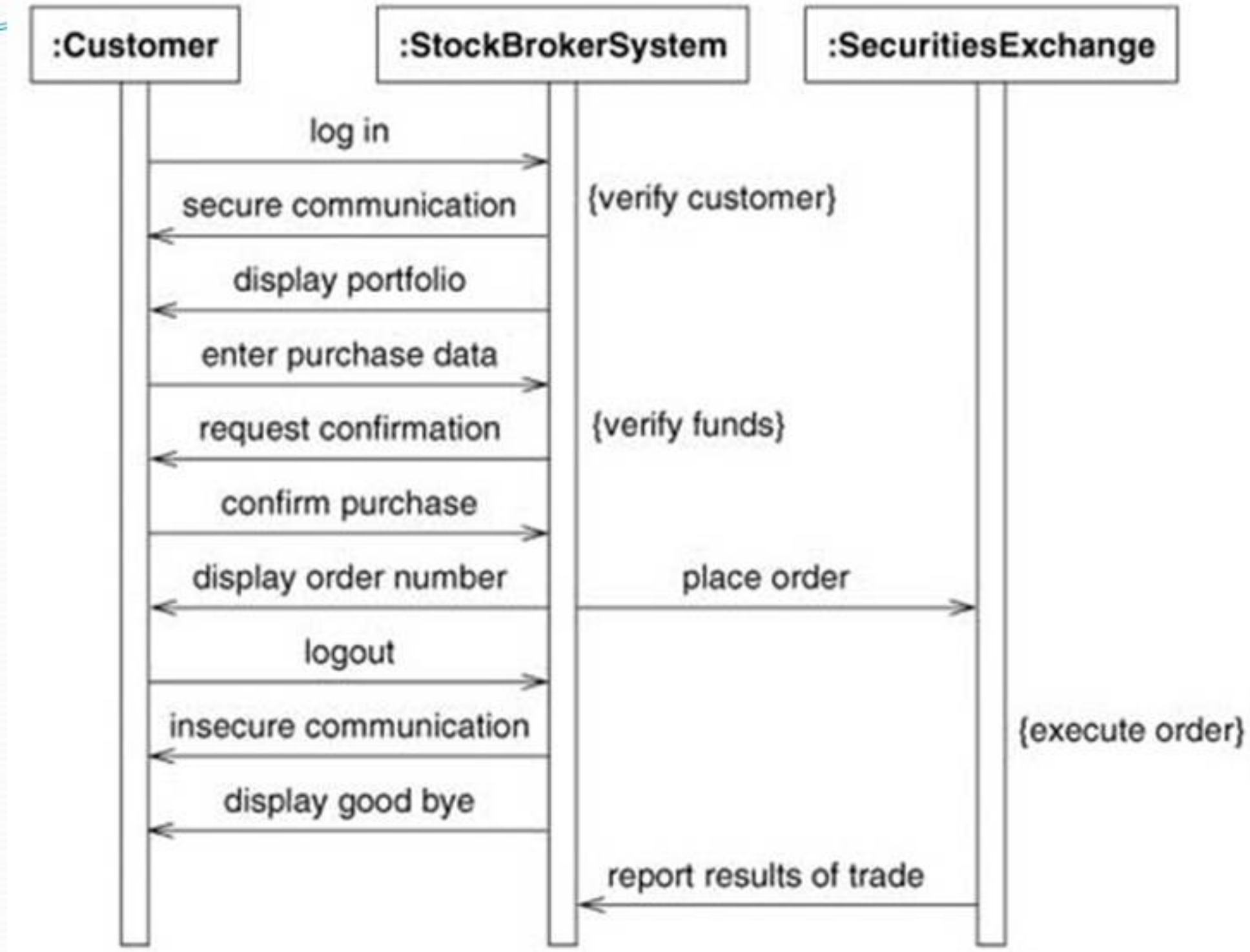
- Where a message arrow appears relative to the Y axis and other message arrows, determines the relative time the message is sent.



- Sequence diagrams show relative timings, not absolute timings.



- Links between objects are implied by the existence of a message.



- Concurrent messages
- Messages between participants need not alternate

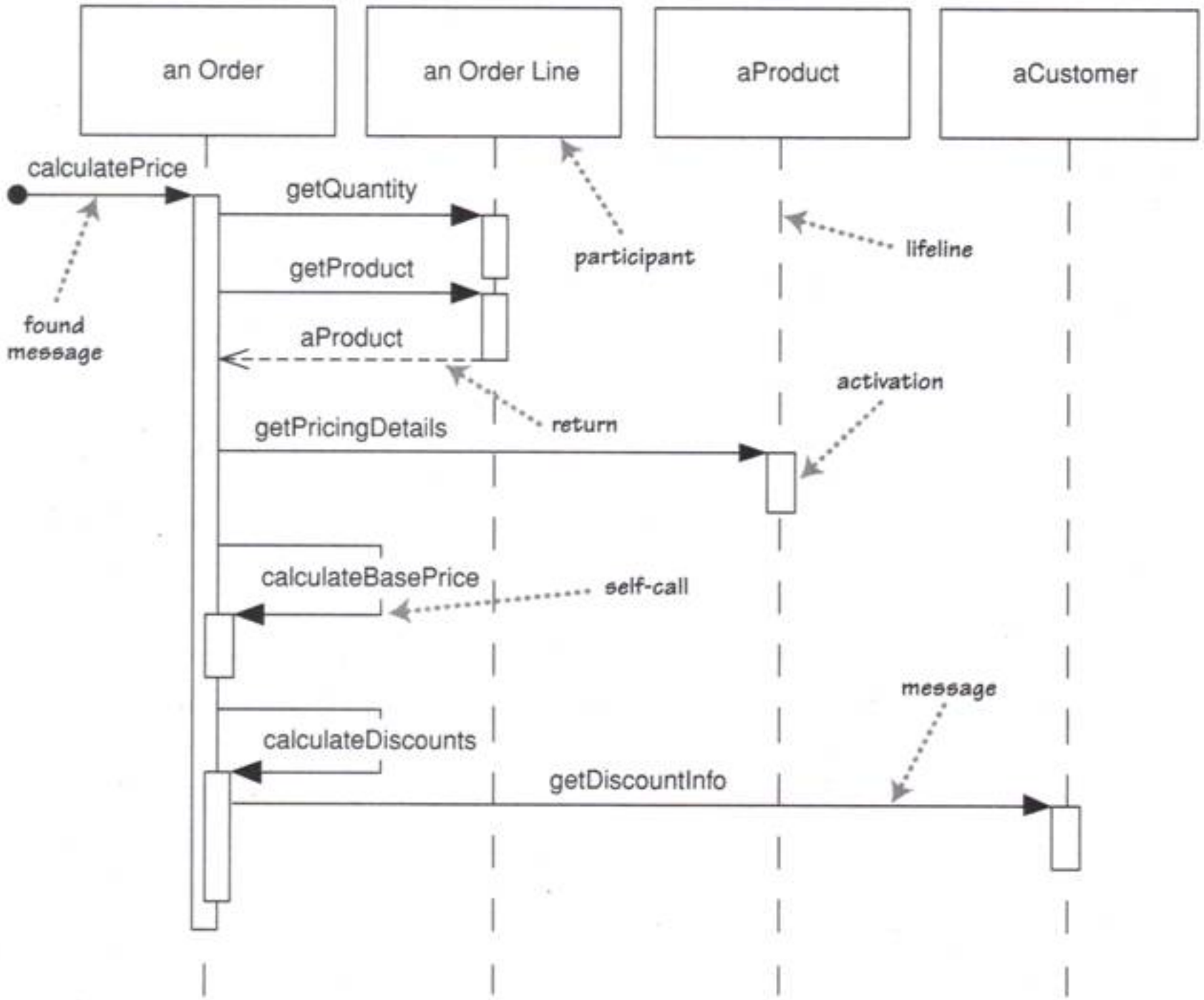
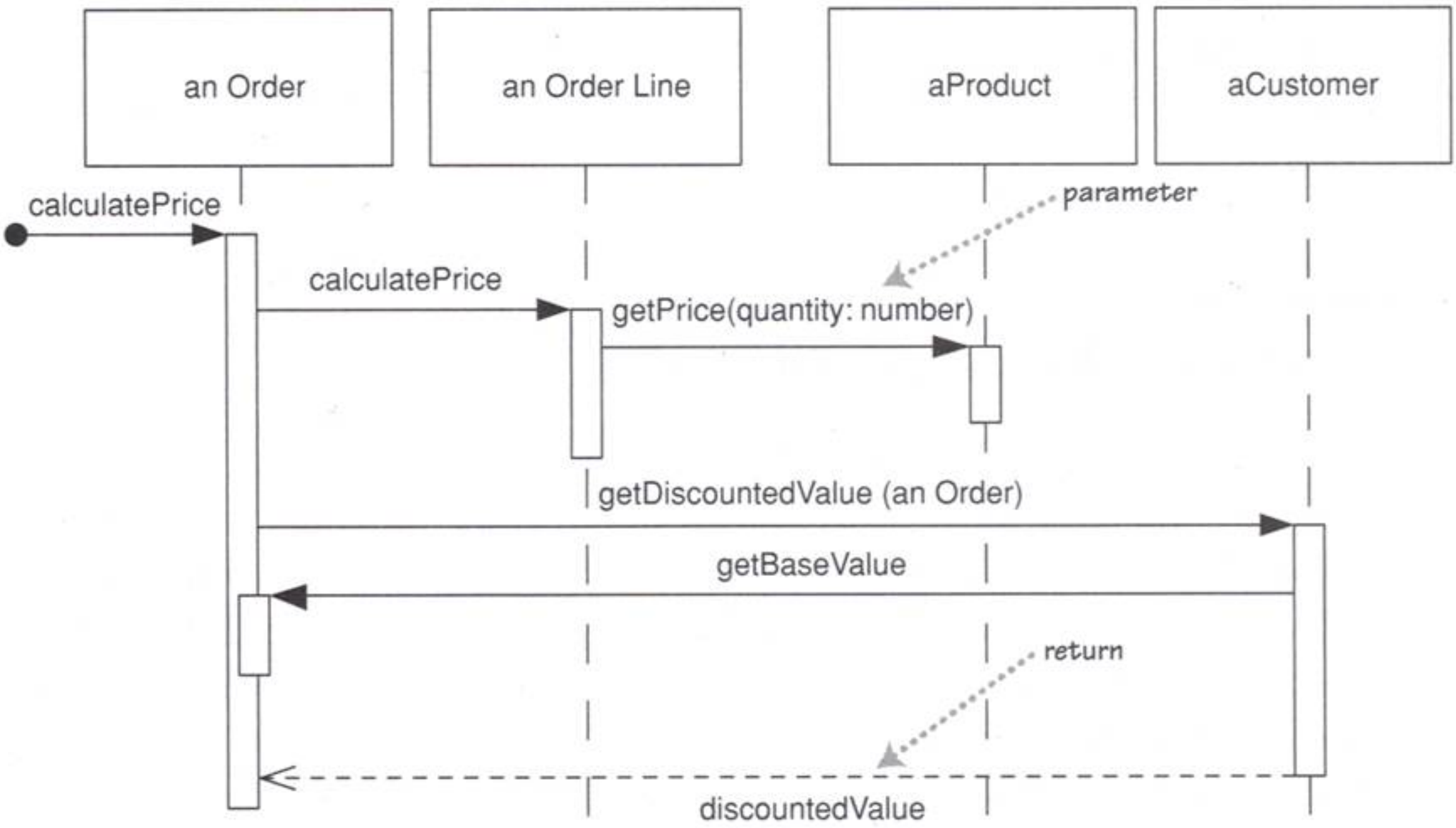


Figure 4.1 *A sequence diagram for centralized control*

- Objects are not always active

- Each lifeline has an activation bar that shows when the participant is active in the interaction.
- This corresponds to one of the participant's methods being on the stack.
- Activation bars are optional in UML
- The first message doesn't have a participant that sent it, as it comes from an undetermined source.
- It's called a **found message.**

Sequence Diagrams

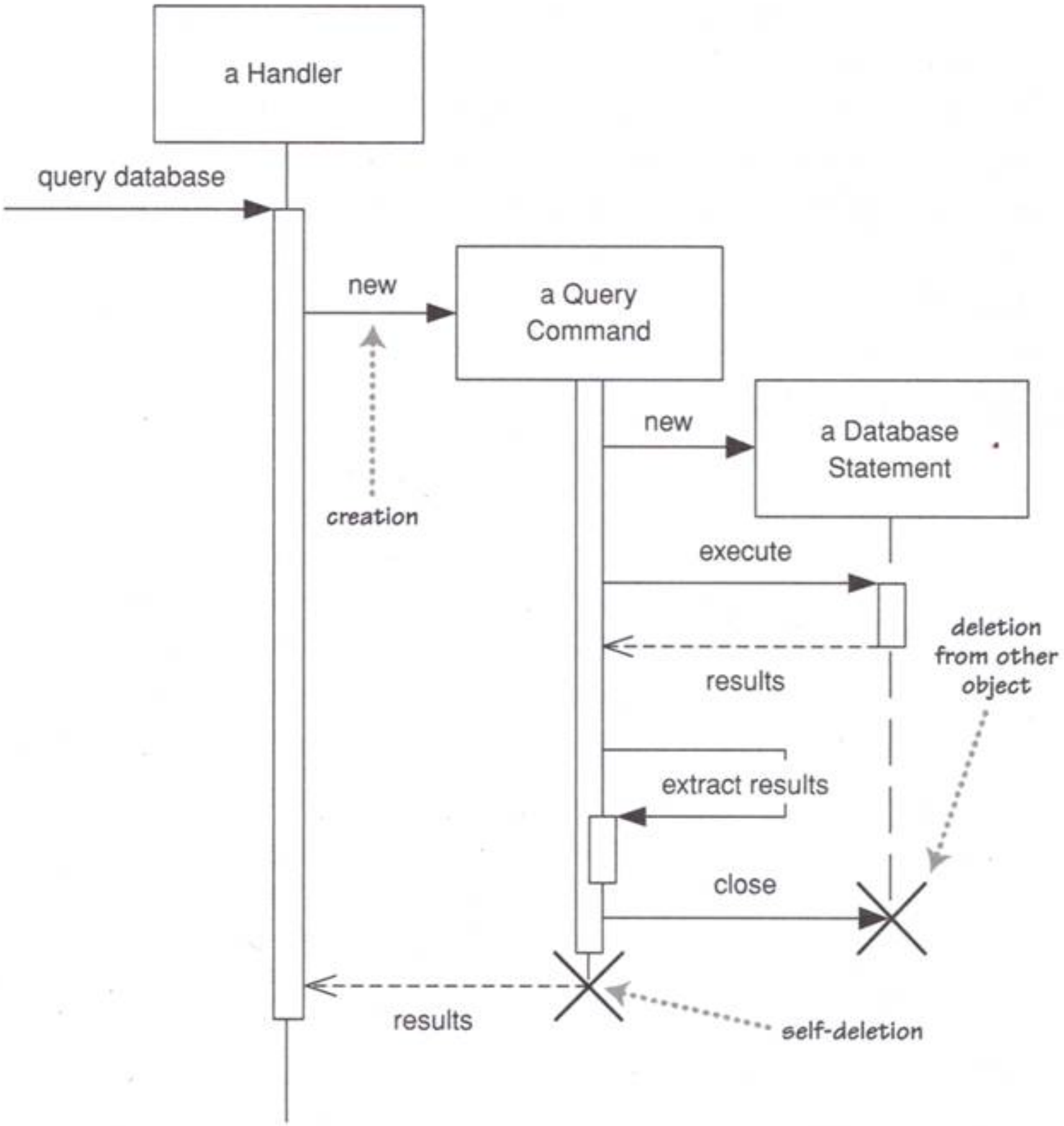


A sequence diagram for distributed control

Strengths and weaknesses (Centralized versus Distributed)

- Centralized approach is simpler, as all the processing is in one place;
- Distributed approach
  - Localizes the effects of change
  - Creates more opportunities for using polymorphism rather using conditional logic.

Transient Objects: Creating and Deleting Participants



Creation and deletion of participants

Loops and Conditionals

```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end for
  if (needsConfirmation) messenger.confirm
end procedure
```

Loops and Conditionals