Web Application Development

Indian Institute of Information Technology Sri City, India

March 17, 2021

Sequence Models

- The sequence model elaborates the themes of use cases.
- There are two kinds of sequence models: scenarios and a more structured format called sequence diagrams.

Scenarios

- A scenario is a sequence of events that occurs during one particular execution of a system, such as for a use case.
- There are two kinds of sequence models: scenarios and a more structured format called sequence diagrams.
- The scope of a scenario can vary; it may include all events in the system, or it may include only those events impinging on or generated by certain objects.
- A scenario can be the historical record of executing an actual system or a thought experiment of executing a proposed system.

Scenarios

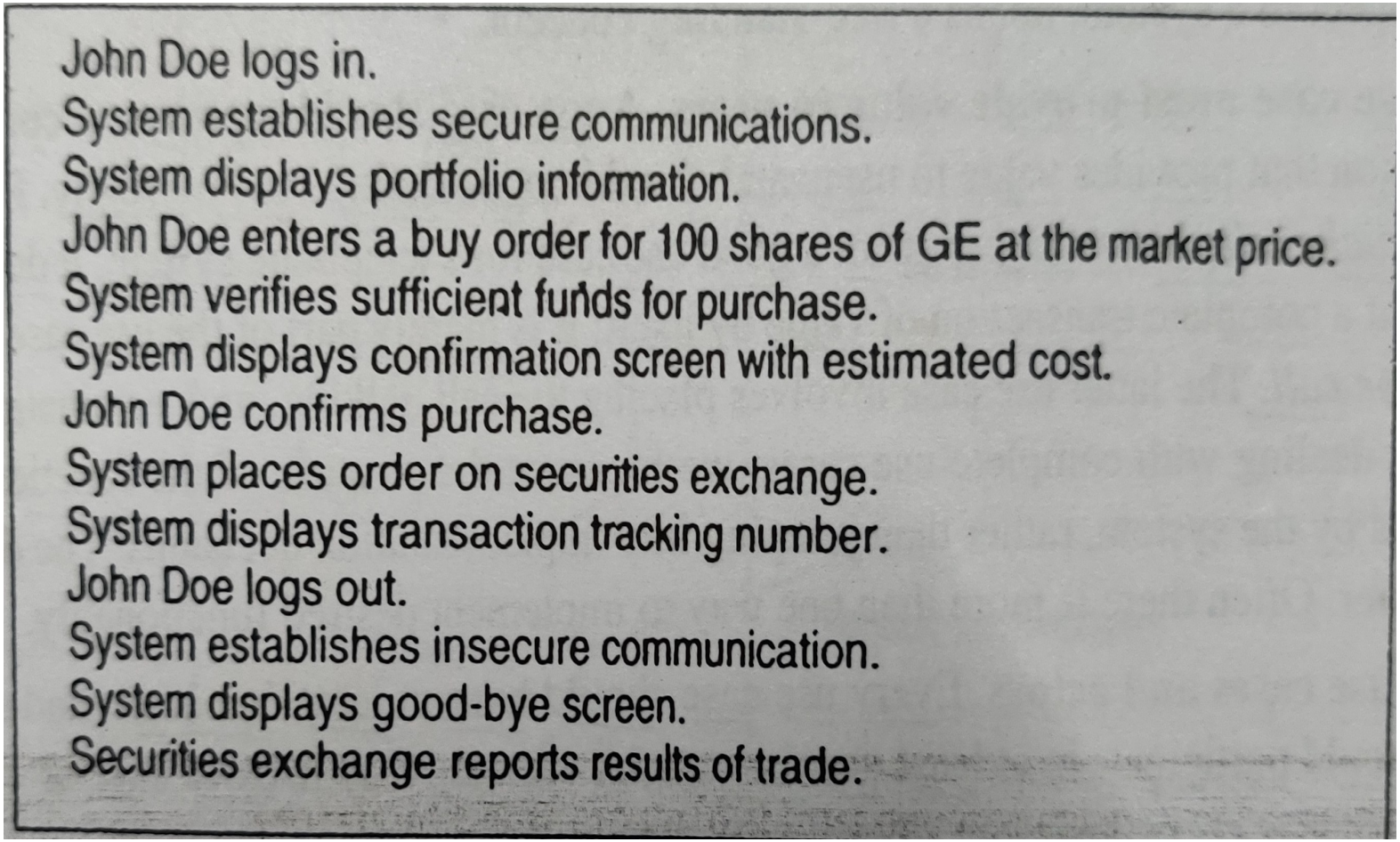
John Doe logs in.
System establishes secure communications.
System displays portfolio information.
John Doe enters a buy order for 100 shares of GE at the market price.
System verifies sufficient funds for purchase.
System displays confirmation screen with estimated cost.
John Doe confirms purchase.
System places order on securities exchange.
System displays transaction tracking number.
John Doe logs out.
System establishes insecure communication.
System displays good-bye screen.
Securities exchange reports results of trade.

Figure 1. Scenario for session with an online stock broker: A scenario is as sequence of events that occurs during one particular execution of a system.

Scenarios

- A scenario can be displayed as a list of text statements as Figure illustrates.
- In this example, John Doe logs on with an online stock broker system, places an order for GE stock, and then logs off.
- Sometime later, after the order is executed, the securities exchange reports the results of the trade to the broker system.
- John Doe will see the results on the next login, but that is not part of this scenario.

Scenarios

- A scenario contains messages between objects as well as activities performed by objects.
- Each message transmits information from one object to another.
- The first step of writing a scenario is to identify the objects exchanging messages.
- Then you must determine the sender and receiver of each message, as well as the sequence of the messages.
- Finally, you can add activities for internal computations as scenarios are reduced to code.

Sequence Diagram

- A text format is convenient for writing, but it does not clearly show the sender and receiver of each 'message, especially if there are more than two objects.
- A sequence diagram shows the participants in an interaction and the sequence of messages among them.
- A sequence diagram shows the interaction of a system with its actors to perform all or part of a use case.

Sequence Diagram

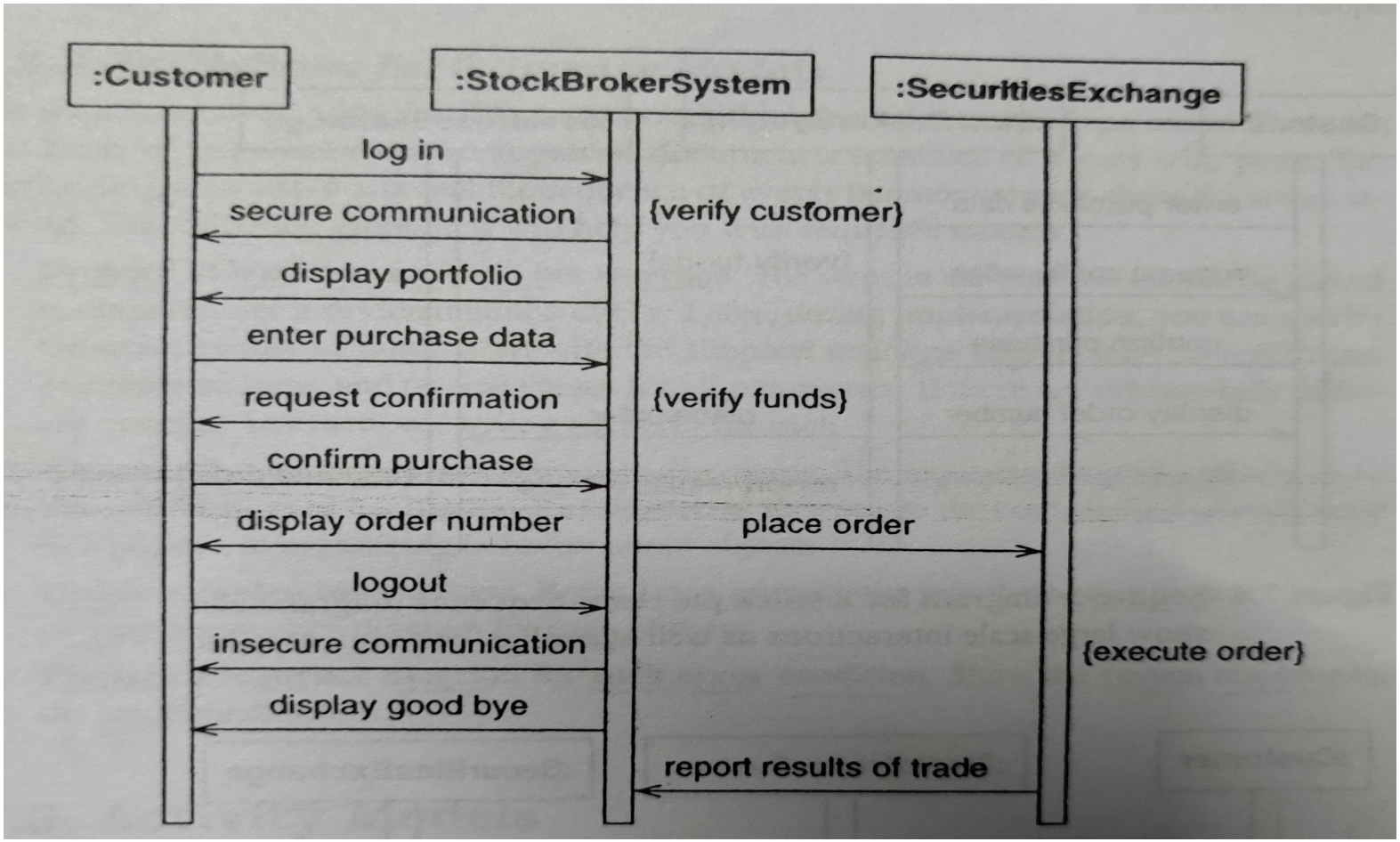

Figure 2. Sequence Diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and a sequence of message among them

Sequence Diagram

- Figure shows a sequence diagram corresponding to the previous stock broker scenario.
- Each actor as well as the system is represented by a vertical line called a lifeline and each message by a horizontal arrow from the sender to the receiver.
- Time proceeds from top to bottom, but the spacing is irrelevant; the diagram shows only the sequence of messages, not their exact timing.
- Note that sequence diagrams can show concurrent signals—stock broker system sends messages to customer and securities exchange concurrently and signals between participants need not alternate stock broker system sends secure communication followed by display portfolio.

Sequence Diagram

- Each use case requires one or more sequence diagrams to describe its behavior.
- Each sequence diagram shows a particular behavior sequence of the use case.
- It is best to show a specific portion of a use case and not attempt to be too general.
- Sequence diagrams can show large-scale interactions, such as an entire session with the stock broker system, but often such interactions contain many independent tasks that can be combined in various ways.

Sequence Diagram

- Rather than repeating information, you can draw a separate sequence diagram for each task.
- For example, Figure 3 and Figure 4 show an order to purchase a stock and a request for a quote on a stock.
- These and various other tasks (not shown) would fit within an entire stock trading session.

Sequence Diagram

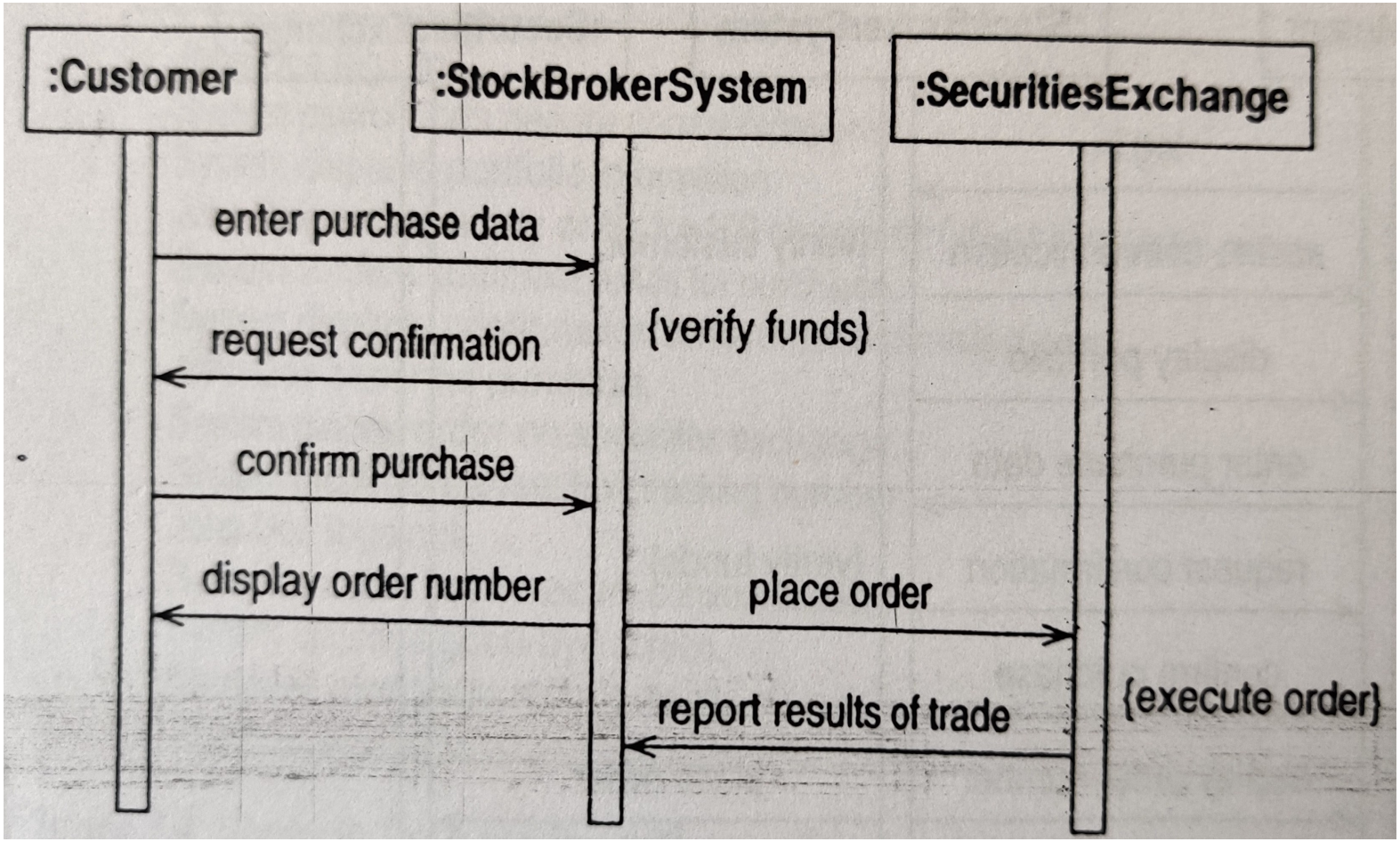

Figure 3. Sequence Diagram for a stock purchase. Sequence diagram can show large-scale interactions as well as smaller, constitute tasks

Sequence Diagram

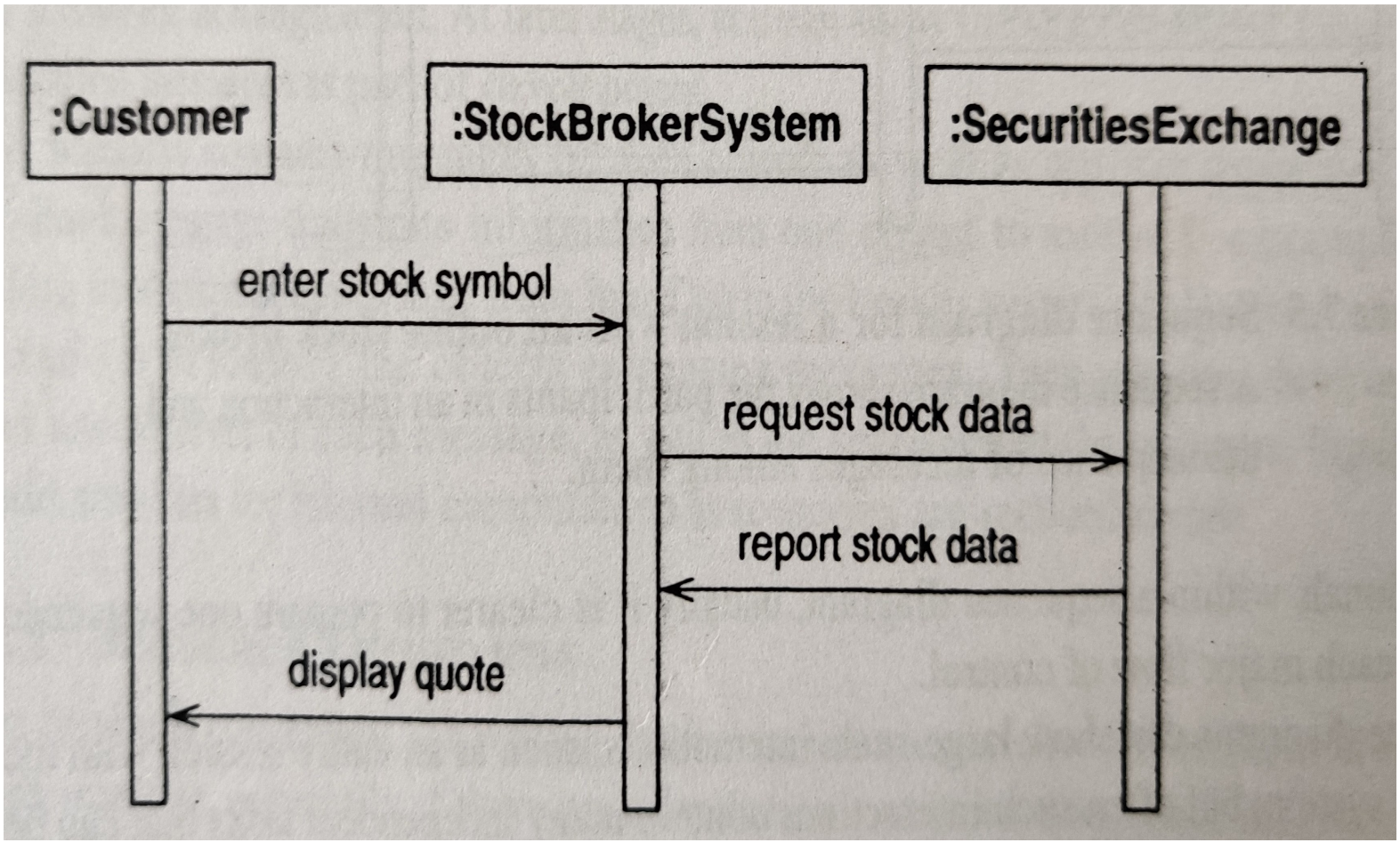

Figure 4. Sequence Diagram for a stock quote

Sequence Diagram

- You should also prepare a sequence diagram for each exception condition within the use case.
- For example, Figure 4 shows a variation in which the customer does not have sufficient funds to place the order.
- In this example, the customer cancels the order.

Sequence Diagram

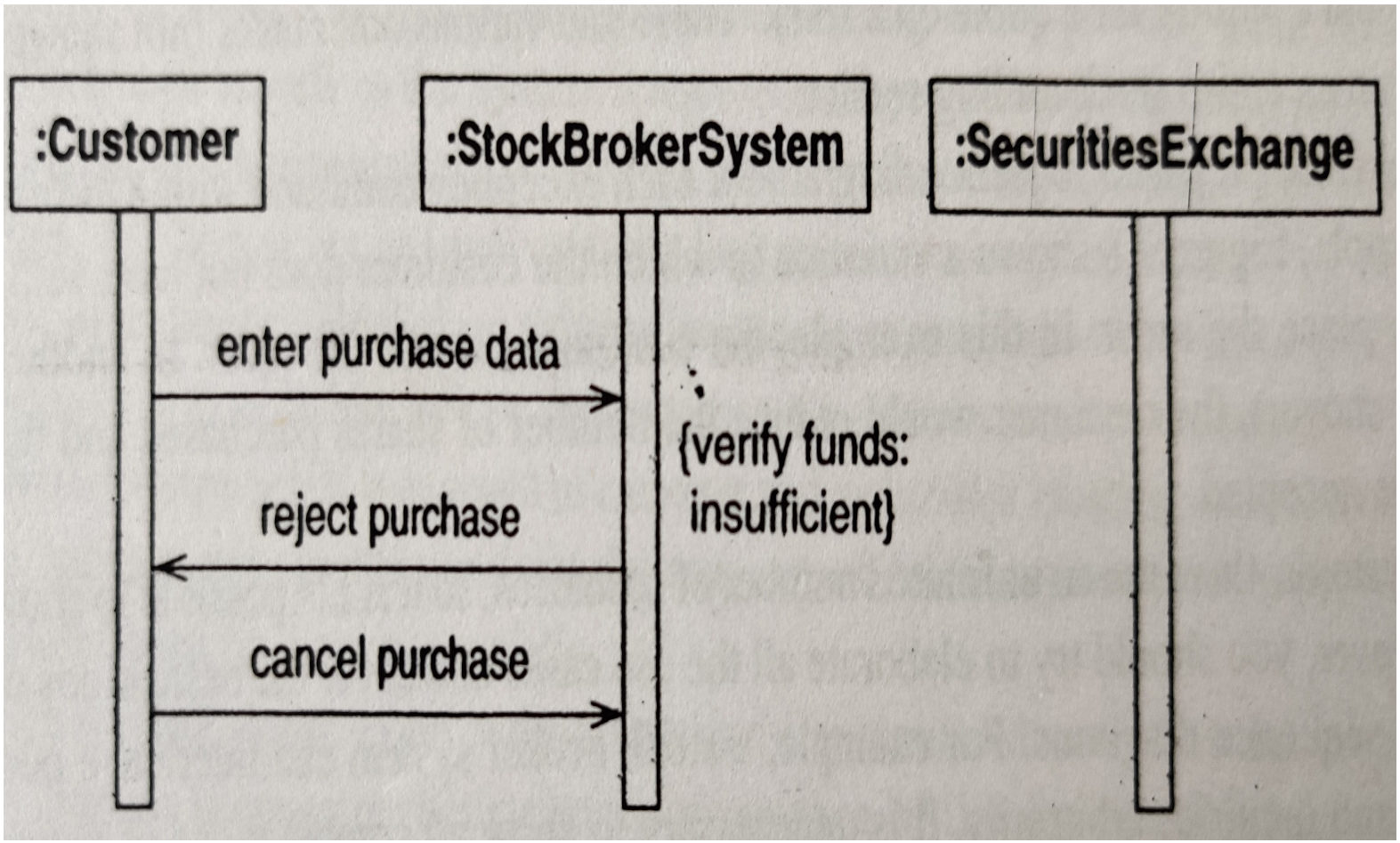

Figure 5. Sequence Diagram for a stock purchase that fails

Guidelines for Sequence Models

- The sequence model adds detail and elaborates the informal themes of use cases.
- There are two kinds of sequence models.
- Scenarios document a sequence of events in the form of spoken or written language.
- Sequence diagrams also document the sequence of events but more clearly show the actors involved.

Guidelines for Sequence Models

- The following guidelines will help you with sequence models.
- **Prepare at least one scenario per use case**:
- The steps in the scenario should be logical commands, not individual button clicks.
- Later, during implementation, you can specify the exact syntax of input.
- Start with the simplest mainline interaction-no repetitions, one main activity, and typical values for all parameters.

Guidelines for Sequence Models

- **Abstract the scenarios into sequence diagrams:**
- The sequence diagrams clearly show the contribution of each actor.
- It is important to separate the contribution of each actor as a event to organizing behavior about objects.

Guidelines for Sequence Models

- **Divide complex interactions:**
- Break large interactions into their constituent task and prepare a sequence diagram for each of them
- **Prepare a sequence diagram for each error condition:**
- Show the system response to the error condition.

Activity Models

- An activity diagram shows the sequence of steams that make up a complex process, such as an algorithm or workflow.
- An activity diagram shows flow of control, similar to a sequence diagram, but focuses on operations rather than on objects.
- Activity diagrams are most useful during the early stages of designing algorithms and workflows.

Activity Models

- Figure shows an activity diagram for the processing of a stock trade order that has been received by an online stock broker.
- The elongated ovals show activities and the arrows show their sequencing.
- The diamond shows a decision point and the heavy bar shows split¬ting or merging of concurrent threads.

Activity Models

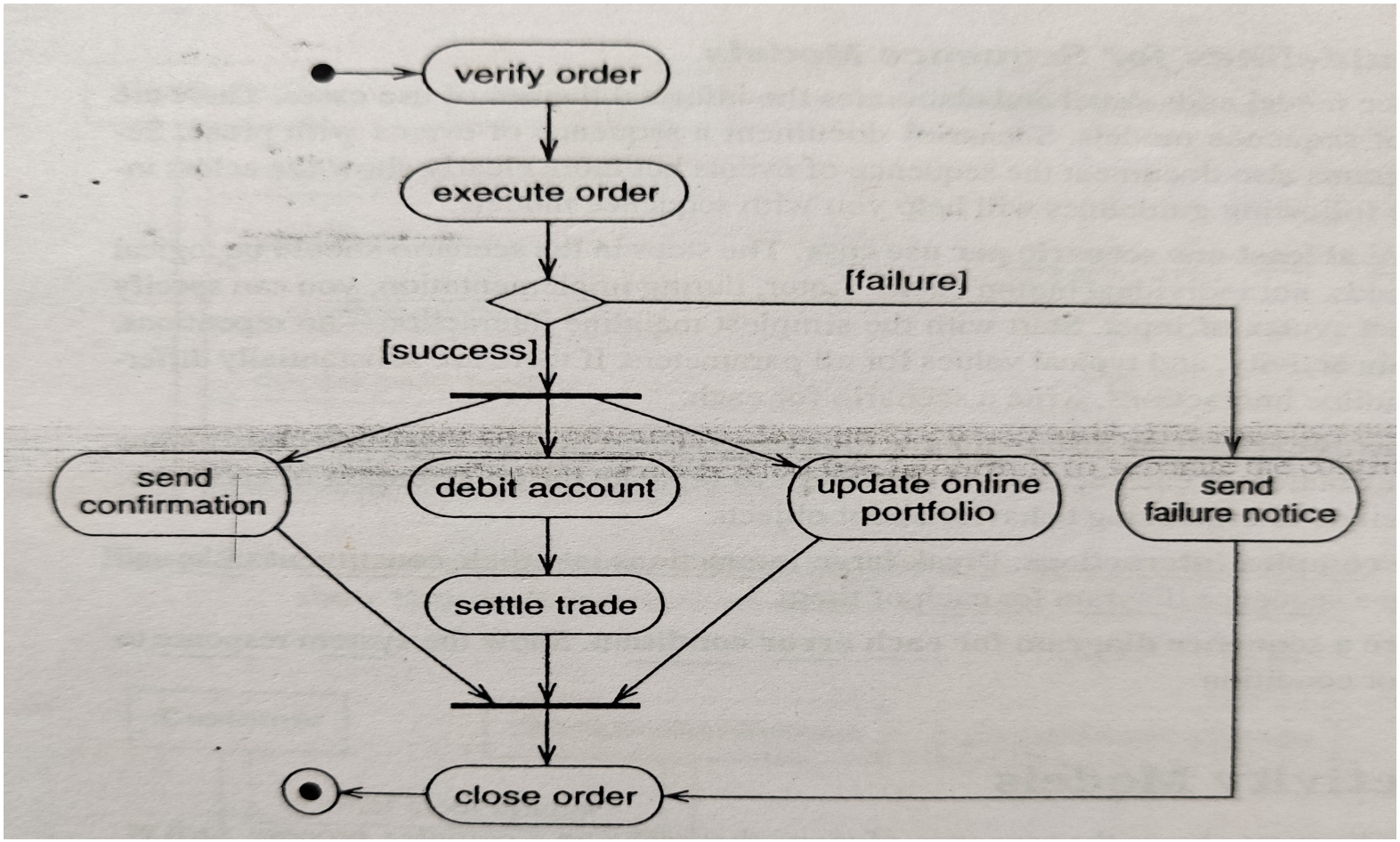

Figure 6. Activity Diagram for the stock trade processing. An activity diagram shows the sequence of steps that make up a complex process

Activity Models

- The online stock broker first verifies the order against the customer's account, then executes it with the stock exchange.
- If the order executes.successfully, the system does three things concurrently: mails trade confirmation to the customer, updates the online portfolio to reflect the results of the trade, and settles the trade with the other party by debiting the account and transferring cash or securities.
- When all three concurrent threads have been completed, the system merges control into a single thread and closes the order.
- If the order execution fails, then the system sends a failure notice to the customer and closes the order.

Activity Models

- An activity diagram is like a traditional flowchart in that it shows the flow of control from step to step.
- Unlike a traditional flowchart, however activity diagram can show both sequential and concurrent flow of control
- This distinction is important for a distributed system.
- Activity diagrams are often used for modeling human organizations because they involve man objects—persons and organizational units that perform operations concurrently.

Activity Models

- Activities
- Branches
- Initiation and Termination
- Concurrent Activities

Activities

- The steps of an activity diagram are operations, specifically activities from the state model.
- The purpose of an activity diagram is to show the steps within a complex process and the sequencing constraints among them.
- Some activities run forever until an outside event interrupts them, but most activities eventually complete their work and terminate by themselves.
- The completion of an activity is a completion event and usually indicates that the next activity can be started.

Activities

- An unlabeled arrow from one activity to another in an activity diagram indicates that the first activity must complete before the second activity can begin.
- An activity may be decomposed into finer activities.
- For example, Figure 7 expands the execute order activity of Figure 6.
- It is important that the activities on a diagram be at the same level of detail.

Activity Models

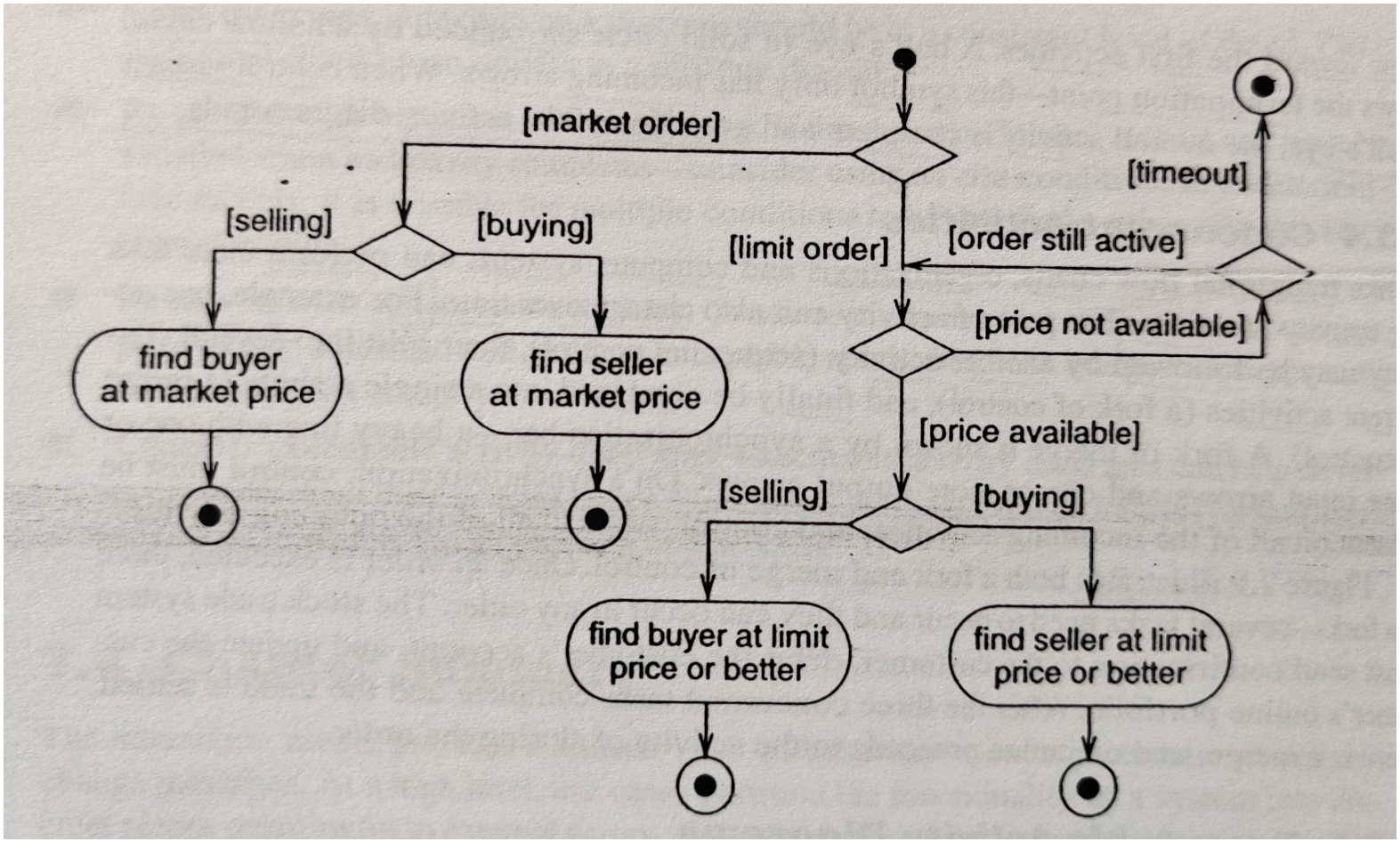

Figure 7. Activity Diagram for execute order. An activity may be decomposed into finer activities

Activities

- For example, in Figure 1 execute order and settle trade are similar in detail; they both express a high-level operation without showing the underlying mechanisms.
- If one of these activities were replaced in the activity diagram by its more detailed steps, the other activities should be replaced as well to maintain balance.
- Alternatively, balance can be preserved by elaborating the activities in separate diagrams.

Branches

- If there is more than one successor to an activity, each arrow may be labeled with a condition in square brackets, for example, [failure].
- All subsequent conditions are tested when an activity completes.
- If one condition is satisfied, its arrow indicates the next activity to perform.
- If no condition is satisfied, the diagram is badly formed and the system will hang unless it is interrupted at some higher level.

Branches

- To avoid this danger, you can use the else condition; it is satisfied in case no other condition is satisfied.
- If multiple conditions are satisfied, only one successor activity executes, but there is no guarantee which one it will be.
- As a notational convenience, a diamond shows a branch into multiple successors, but it means the same thing as arrows leaving an activity symbol directly.
- In Figure 6 the diamond has one incoming arrow and two outgoing arrows, each with a condition. A particular execution chooses only one path of control.

Initiation and Termination

- A solid circle with an outgoing arrow shows the starting point of an activity diagram.
- When an activity diagram is activated, control starts at the solid circle and proceeds via the outgoing arrow toward the first activities.
- A bull's-eye (a solid circle surrounded by a hollow circle) shows the termination point—this symbol only has incoming arrows.
- When control reaches a bull's-eye, the overall activity is complete and execution of the activity diagram ends

Concurrent Activities

- Unlike traditional flow charts, organizations and computer systems can perform more than one activity at a time.
- The pace of activity can also change over time.
- For example, one activity may be followed by another activity (sequential control), then split into several concurrent activities (a fork of control), and finally be combined into a simile activity (a merge of control).
- A fork or merge is shown by a synchronization bar — a heavy line with one or more input arrows and one or more output arrows.

Concurrent Activities

- On a synchronization, control must be present on all of the incoming activities, and control passes to all of the outgoing activities.
- Figure illustrates both a fork and merge of control.
- Once an order is executed, there is a fork—several tasks need to occur and they can occur in any order.
- The stock trade system must send confirmation to the customer, debit the customer's account, and update the customer's online portfolio.
- After the three concurrent tasks complete and the trade is settled, there is a merge, and execution proceeds to the activity of closing the order.

Guidelines for Activity Models

- Activity diagrams elaborate the details of computation, thus documenting the steps needed to implement an operation or a business process.
- In addition, activity diagrams can help developers understand complex computations by graphically displaying the progression through intermediate execution steps.
- Here is some advice for activity models.

Guidelines for Activity Models

- Don't misuse activity diagrams
- Level diagrams
- Be careful with branches and conditions
- Be careful with concurrent activities
- Consider executable activity diagrams

Guidelines for Activity Models

- **Don't misuse activity diagrams:**
- Activity diagrams are intended to elaborate use case and sequence models so that a developer can study algorithms and workflow.
- Activity diagrams supplement the object-oriented focus of UML models and should not be used as an excuse to develop software via flowcharts.
- **Level Diagrams:**
- Activities on a diagram should be at a consistent level of detail. Place additional detail for an activity in a separate diagram.

Guidelines for Activity Models

- **Be careful with branches and conditions:**
- If there are conditions, at least one must be satisfied when an activity completes—consider using an else condition.
- In undeterministic models, it is possible for multiple conditions to be satisfied—otherwise this is an error condition.

Guidelines for Activity Models

- **Be careful with concurrent activities:**
- Concurrency means that the activities can complete in any order and still yield an acceptable result. Before a merge can happen, all inputs must first complete.
- **Consider executable activity diagrams:**
- Executable activity diagrams can help developers understand their systems better. Sometimes they can even be helpful for end users who want to follow the progression of a process.

Book

- "Object-Oriented Modeling and Design with UML" by Michael Blaha and James Rumbaugh