

CHAPTER 9

Sequence Diagrams

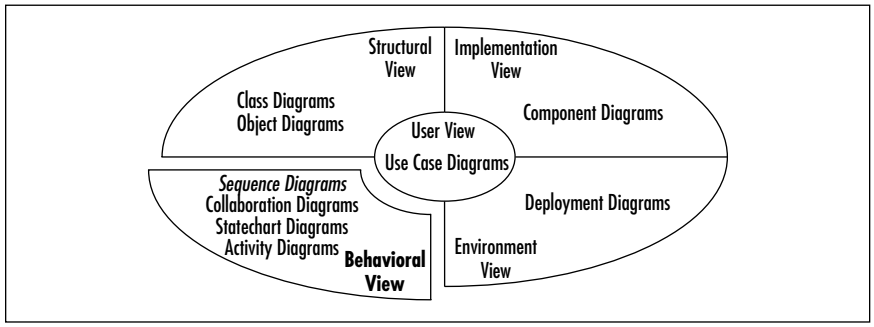


Figure 9-1: Behavioral Model View—Sequence Diagrams

Sequence diagrams (Figure 9-1) lie within the behavioral view of a system and render the specification of behavior. These diagrams describe the behavior provided by a system to interactors. These diagrams contain classes that exchange messages within an interaction arranged in a time sequence. They may be constructed using the following technique:

1. Identify behavior to be specified.
2. Model class roles.
3. Model lifelines.
4. Model activations.
5. Model messages.
6. Refine and elaborate as required.

Sequence diagrams

- May exist in generic form. These describe a set of message exchange sequences among a set of classes, that is, interactions among class roles.

- May exist in instance form. These describe one actual message exchange sequence consistent with the generic form, that is, a scenario among objects conforming to class roles. This form does not include repetition sequences (loops) or conditional sequence (branches).
- Are types of interaction diagrams that focus on one message exchange sequence or a set of such sequences involved in specifying behavior. This is depicted as a set of class roles involved in an interaction arranged in a time sequence.
- Show sequences of messages exchanged by class roles (horizontal dimension) within a time sequence (vertical dimension). The axes may be interchanged.
- Are particularly valuable for real-time specifications and for complex scenarios.
- Must have a vertical dimension representing the time over which an interaction occurs.
  - Time proceeds down the page. This may be reversed.
  - Only sequences are important, not the absolute span of time, but this dimension may have an actual metric.
  - Labels may be shown either in the margin or near the elements they reference.
- Must have a horizontal dimension specifying the different class roles participating in an interaction. There is no significance to the ordering of these class roles.
- May have the various expressions (activations, messages, guard conditions, iterations, and message signatures) they utilize be expressed using pseudocode or another language.

Interactions

Interactions are classes that define message exchange sequences or patterns of message exchanges among other classes participating to accomplish a specific purpose. Interactions are used to model communications between entities. Interactions

- Are defined in the context of collaborations (Chapter 10).
- May be associated with use cases (Chapter 8) or operations. They *specify* the behavior provided.
- Are illustrated by scenarios. Scenarios are specific sequences of message exchanges and actions.

Class Roles

Class roles (Figure 9-2) are classes that define roles or specific parts played by class participants in interactions or collaborations. They are used to model roles that entities may play within interactions or collaborations.

Class roles

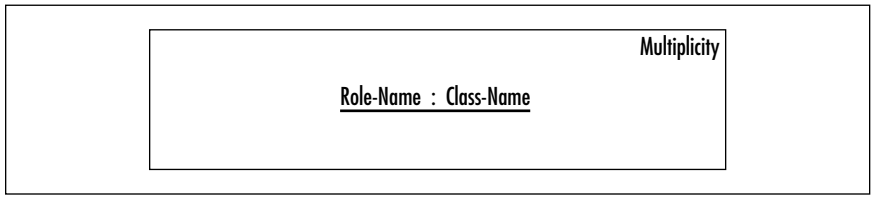


Figure 9-2: Class Role

- Are denoted as class rectangles.
- Specify the types of objects that may participate within interactions and collaborations.
- Specify a restricted view of classes. That is, they define what is required of a class for its participation in the interaction or collaboration.
- May use other notation of classes and objects.
- Represent roles that bind to actual objects when interactions or collaborations are used.
- Must have a name compartment.
- May have a role name or identifier string that represents the name of the role. The name must be underlined. When omitted, the colon remains and the role is played by an anonymous object.
- May have a colon followed by a class name or identifier string that represents the class of the role. The name must be underlined.
- May have a multiplicity expression that indicates that a set of objects participates within a role. When suppressed, it indicates a role for a single object.
- May have other compartments, which may be used for other model properties (business rules, responsibilities, variations, events, exceptions, etc.). These are not normally shown.

Lifelines

Lifelines (Figure 9-3) are graphical constructs that represent the existence of class roles over a period of time. They are used to model the existence of entities over time. Lifelines

- Are denoted as dashed lines.
- Must have their class roles identified.
- Must start at the top of the diagram, above the first message, if they exist when the interaction starts.
- Must end at the bottom of the diagram, below the last message, if they exist when the interaction ends.
- Must start when the class role object is created if it is created during the interaction. The message that creates it must be drawn with its arrowhead to the class role.

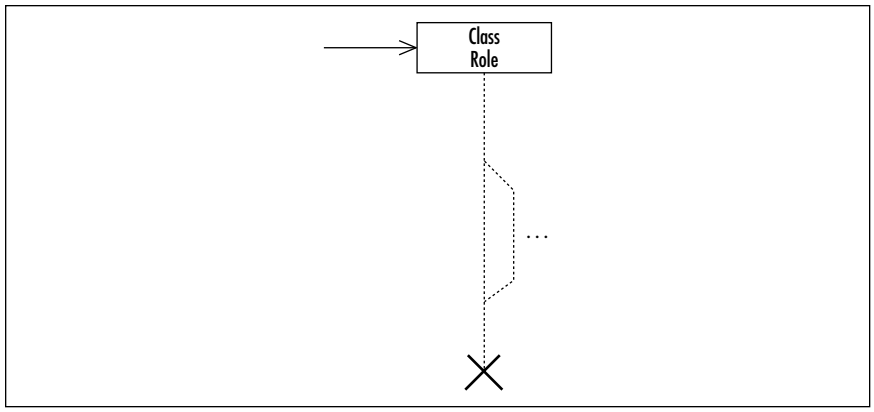


Figure 9-3: Lifeline

- Must end when the class role object is destroyed if it is destroyed during the interaction. The destruction is marked with a large "X" at the message that causes the destruction or at the final return message from the class role.
- May split into two or more concurrent lifelines to show conditionality. Each track corresponds to a conditional branch. They may merge together at some subsequent point in the class role's lifeline.
- May subsume an entire set of objects on a diagram representing a high-level view.

Activations

Activations (Figure 9-4) are graphical constructs that represent the time during which a class role is performing an action or when it is active and has focus of control (thread). They are used to model the time during which entities are active or performing some operation and to model control relationships among entities. Activations

- Are denoted as thin rectangles.
- Must have an initiation time. This coincides and is aligned with a rectangle's top. It may have an incoming message.
- Must have a completion time. This coincides and is aligned with a rectangle's bottom. It may be the tail of a return message.
- May be labeled to indicate their actions or operations. Labels
  - May be placed next to activation symbols.
  - May be placed in the margins of the diagram. Other information regarding the activation may also be placed in the margins of the diagram.
  - May be attached to incoming messages (and omitted from the activation symbol itself).

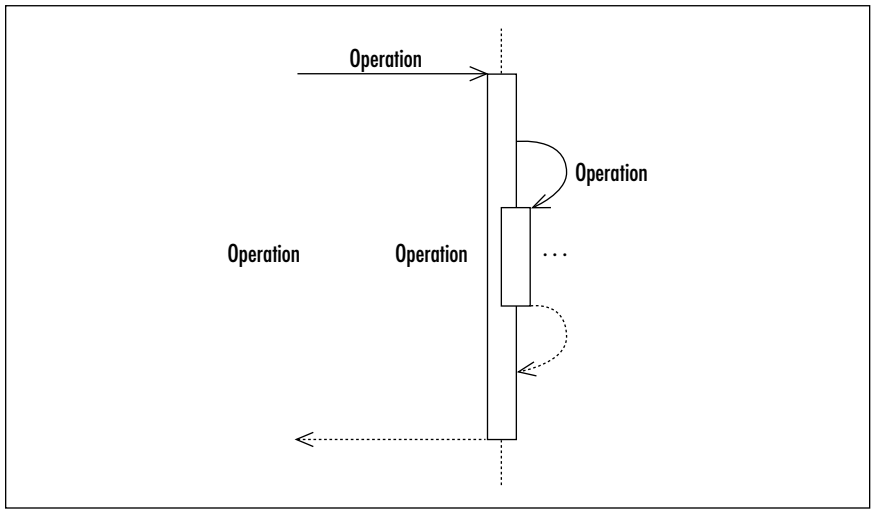


Figure 9-4: Activation

- May be used with concurrent objects or multiple active objects.
  - Direct computation involves a thread owner performing an operation.
  - Indirect computation involves a non–thread owner performing an operation via a nested thread.
  - An activation shows the time during which each object is performing an operation. Operations of other objects or other threads are not relevant.
  - If the distinction between direct and indirect computation is unimportant, the entire lifeline may be shown as an activation, or all activations may be omitted and only lifelines shown.
- May be used with nonconcurrent objects.
  - All objects share one thread of control. The thread may be nested or passed among objects as they request services from one another and perform operations to fulfill their responsibilities.
  - An activation shows the time when an object is performing an operation or another object is performing an operation via the thread being nested. That is, all of the active nested activations may be summarized by an object at a given time.
- May be recursive (self-delegating); that is, an operation may invoke itself. This is depicted using an activation symbol slightly to the right of the first activation symbol. Nesting may occur to an arbitrary depth.
- May be distinguished as actually computing or being live.
  - *Computing* activations are actually performing an operation. They are involved in direct computation. This may be indicated by shading the activation region.

– *Live* activations call or pass control to another object for actual processing. They are involved in indirect computation. This may be indicated by not shading the activation region.

Messages

Messages (Figure 9-5) are classes that define information that is exchanged in interactions and collaborations. They are used to model the content of communication between entities. They are used to convey information between entities and enable entities to request services of other entities. Class roles communicate by sending and receiving messages. A request is a specification of a communication, and a message shows a request in an interaction. When two instances communicate, an instance of a message class is passed between the two instances. The message instance has a sender, a receiver, and possibly other information according to the characteristics of the request. An event within the receiver will signal the receipt of the message from the sender.

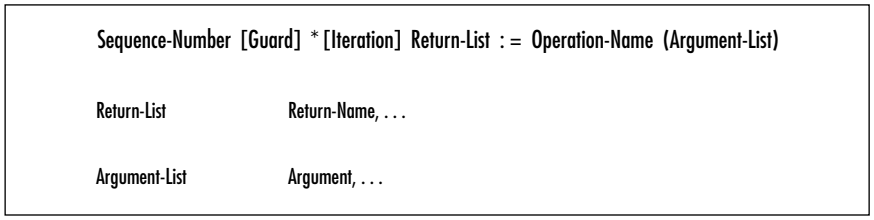


Figure 9-5: Message

Messages

- Are denoted as labeled horizontal arrows between lifelines. The sender will send the message and the receiver will receive the message.
- May have a sequence number. This is an integer that indicates the sequence of the message in the overall interaction. It starts at 1 and increments by 1. They are often omitted in sequence diagrams since the physical location of arrows shows the relative sequence of messages, but are necessary for collaboration diagrams (Chapter 10).
- May have square brackets containing a guard condition. This is a Boolean condition that must be satisfied to enable the message to be sent.
- May have an asterisk followed by square brackets containing an iteration specification. This specifies the number of times the message is sent. An asterisk without a specified iteration (and without square brackets) indicates that the message is repeatedly sent an unspecified number of times.
- May have a return list consisting of a comma-separated list of names that designate the values returned by the operation. These values may be used as actual parameters in subsequent messages. The return names must match the order, number, and types specified in the formal return list of the operation. When the return list is omitted, the assignment operator (:=) is omitted.

- Must have a name or identifier string that represents the name of the operation. The operation must be in the class role receiving the message.
- May have parentheses containing an argument list consisting of a comma-separated list of actual parameters passed to a method. The actual parameters must match the order, number, and types specified in the formal parameter list of the operation.

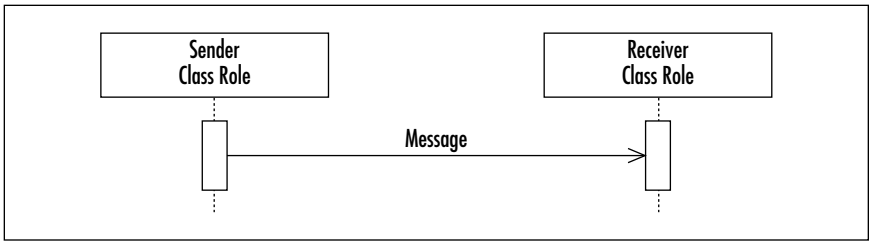


Figure 9-6: Simple Message

- May be simple (Figure 9-6).
  - This represents a flat flow of control.
  - This type of message is depicted using a stick arrowhead.
  - Control is passed from the sender to the receiver.
  - No other details about the communication are provided, either because they are not known or are not relevant.

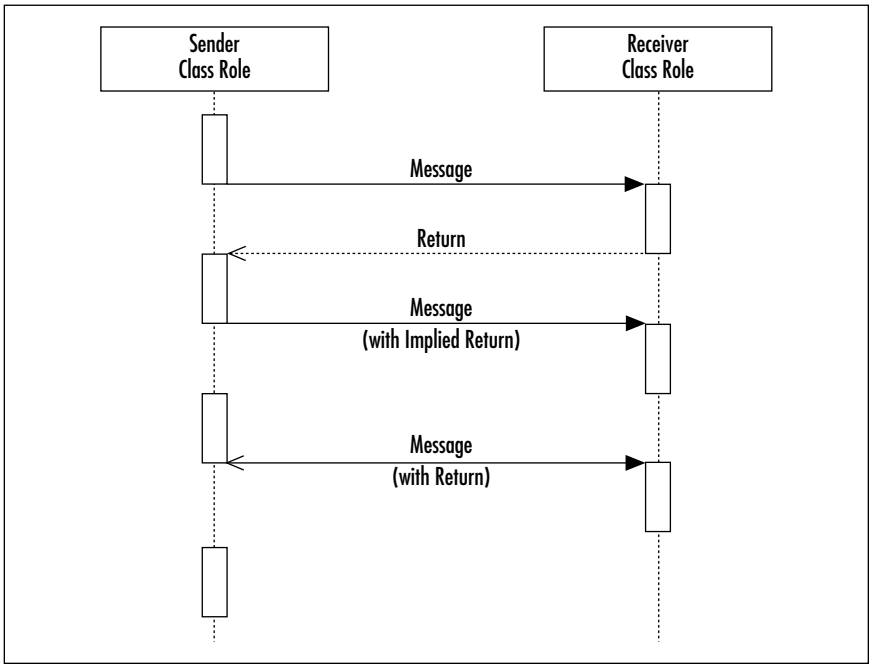


Figure 9-7: Synchronous Messages

- May be synchronous (Figure 9-7).
  - Are denoted as filled solid arrowheads.
  - This represents a nested flow of control via an operation call. The operation call invokes an operation synchronously.
  - This type of message is depicted using a filled arrowhead.
  - The sender passes control to the receiver via the message and pauses to wait for the receiver to relinquish or return control.
  - The receiver performs the operation, may pass control to other class roles if necessary, and returns control to the caller once the operation is completed. The receiver may also return or send information back to the sender.
  - Nested sequences complete before their outer sequences resume.
  - The receiver must be a passive object.
  - The return message may be omitted since it is implicit at the end of an activation, or it may be shown on the message line as a dashed arrow with a stick arrowhead.

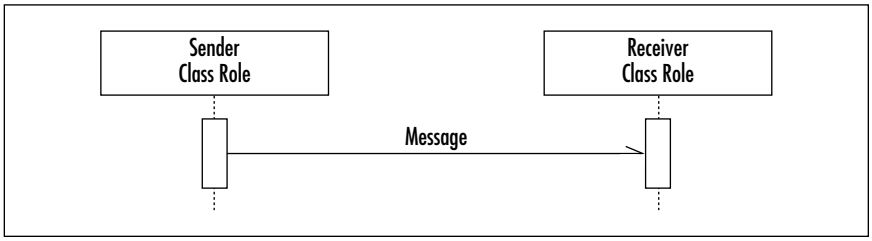


Figure 9-8: Asynchronous Message

- May be asynchronous (Figure 9-8).
  - Are denoted as half stick arrowheads.
  - This represents a nonnested flow of control via a signal. The signal invokes an operation asynchronously.
  - This type of message is depicted using a half-arrowhead.
  - The sender signals the receiver via the message and continues with the sequence without pausing to wait for the receiver.
  - The receiver performs the operation and may return or send information back to the sender.
  - The sender and receiver sequences are concurrent.
  - The receiver must be an active object.
  - If the receiver returns information to the sender, the information must be shown explicitly.
- May require time to travel between the sender and receiver (Figure 9-9).
  - Horizontal messages indicate that the time required to send the message is atomic, or brief compared with the activation. That is, the message transmission is negligible.

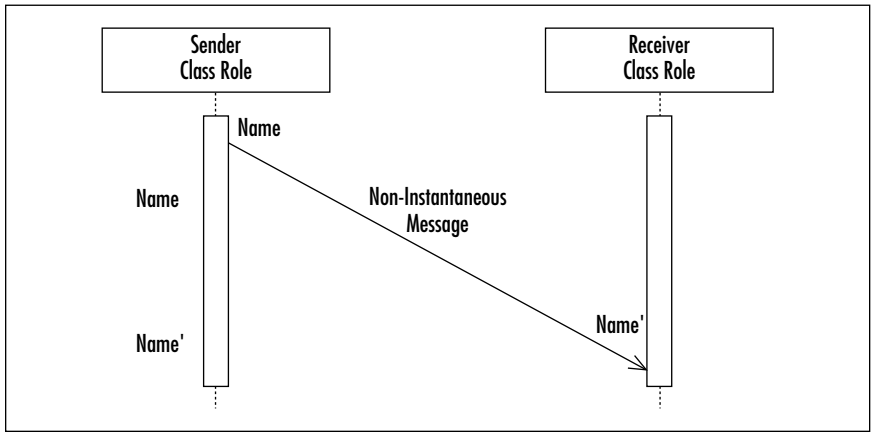


Figure 9-9: Noninstantaneous Message

  - Slanted messages indicate that the time required to send the message is nonatomic, or requires some time. That is, the message transmission requires some time and other occurrences may take place, including messages in the opposite direction.

- May have sending times and receiving times (Figure 9-9). These are formal names that may be used in constraint expressions.
  - These formal names are useful for real-time applications.
  - The transition is given a name that represents the time the message is sent.
  - The transition name with a prime sign appended to it represents the time the message is received.
  - Transition times may be shown in the margin or attached to the message.
- May be part of branching (Figure 9-10). Branching involves multiple messages originating at the same time from a single class role.
  - The branch represents conditionality if the guard conditions on all the messages are mutually exclusive. Thus, only one message is sent.
  - The branch represents concurrency if the guard conditions on the messages are mutually inclusive. Thus, multiple messages are sent.
- May be part of iterations (Figure 9-11). Iterations involve a set of messages being sent multiple times.
  - Iterations may have square brackets containing a continuation condition specifying the condition that must be satisfied in order to exit the iteration and continue with the sequence.

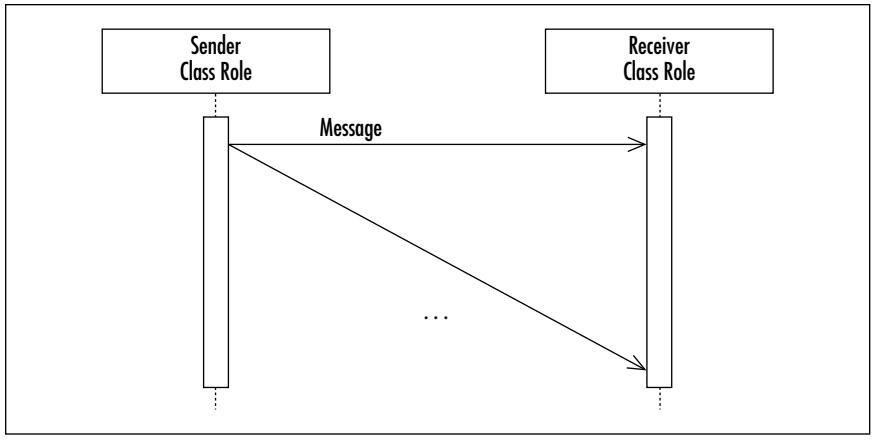


Figure 9-10: Branching and Concurrency

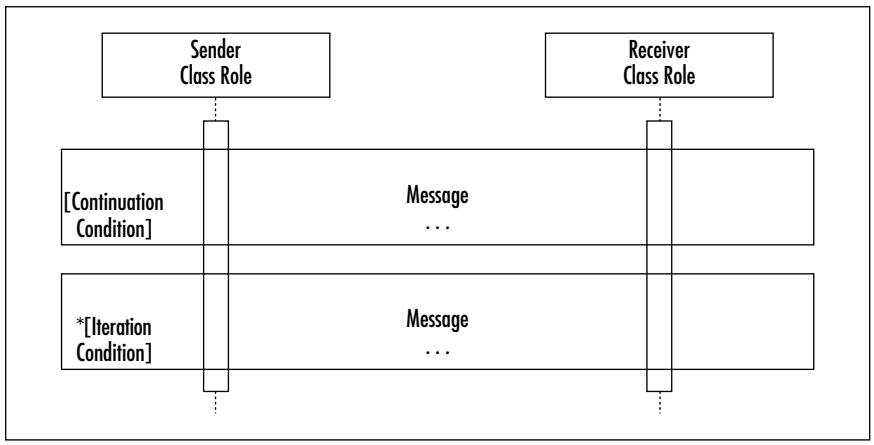


Figure 9-11: Iteration

- Iterations may have an asterisk followed by square brackets containing an iteration expression specifying the number of iterations.
- Iterations may be specified by placing square brackets containing a continuation condition or an asterisk followed by square brackets containing an iteration expression in a constraint expression (Chapter 15) that is attached to the messages within the iteration. This notation does not use the surrounding rectangle to enclose the messages.