Introduction

- Interactions
- Use-cases
- Use-Case Diagrams
- Interaction Diagrams
- Activity Diagrams
- State-chart Diagrams.

Interactions

- In every interesting system, objects don't just sit idle; they interact with one another by passing messages.
- An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose.
- We use interactions to model the dynamic aspect of collaborations, representing various objects playing specific roles, all working together to carry out some behavior that's bigger than the sum of the elements.
- You can model each interaction in two ways:
  - by emphasizing its time ordering of messages, **or** **(Sequence)**
  - by emphasizing its sequencing of messages in the context of some structural organization of objects **(Collaboration)**.

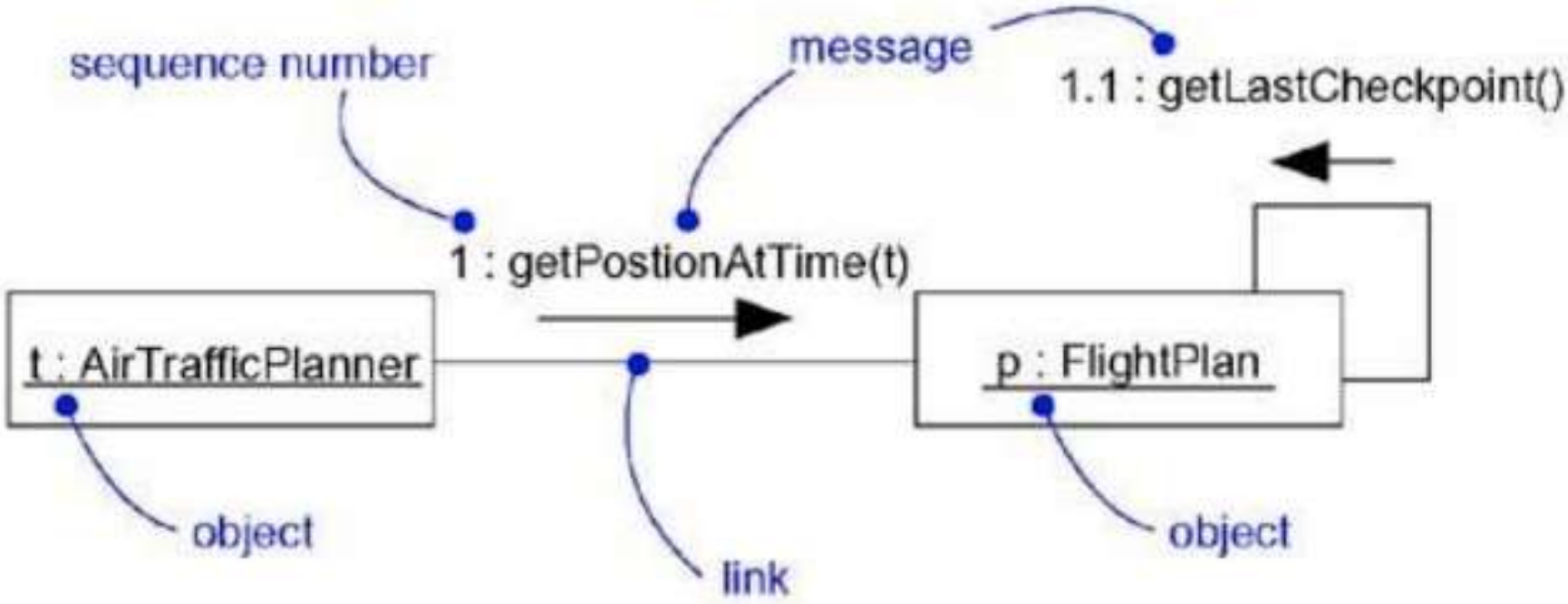
Interactions

- In the UML, dynamic aspects of a system are modelled by using interactions. Like an object diagram, an interaction statically sets the stage for its behavior by introducing all the objects that work together to carry out a specific action.
- Interactions also introduce messages that are dispatched from object to object. Often, messages involve the invocation of an operation or the sending of a signal; messages may also encompass the creation and destruction of other objects.
- We use interactions to model the flow of control within an operation, a class, a component, a use case, or the system as a whole.
- Using interaction diagrams, we can model the flows in two ways.
  - First, you can focus on how messages are dispatched across time.
  - Second, you can focus on the structural relationships among the objects in an interaction and then consider how messages are passed within the context of that structure.

Interactions

- The UML provides a graphical representation of messages, as Figure next shows. This notation permits you to visualize a message in a way that lets you emphasize its most important parts: its name, parameters (if any), and sequence.
- Graphically, a message is rendered as a directed line and almost always includes the name of its operation.

Figure 15-1 Messages, Links, and Sequencing



Interactions

- An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose.
- A message is a specification of a communication between objects that conveys information with the expectation that activity will ensue.
- Interactions between objects are found in the collaboration of objects, in the implementation of operations, and in the context of a class.
- **Note**
- An interaction may also be found in the representation of a component, node, or use case, each of which, in the UML, is really a kind of classifier.
- In the context of a use case, an interaction represents a scenario that, in turn, represents one thread through the action of the use case.

Interactions

- **Objects and Roles**
- The objects that participate in an interaction are either concrete things or prototypical things. As a concrete thing, an object represents something in the real world.
- For example, p, an instance of the class Person, might denote a particular human. Alternately, as a prototypical thing, p might represent any instance of Person.
- **Note**
- This is what distinguishes a collaboration. In a collaboration, the objects you find are prototypical things that play particular roles, not specific objects in the real world.

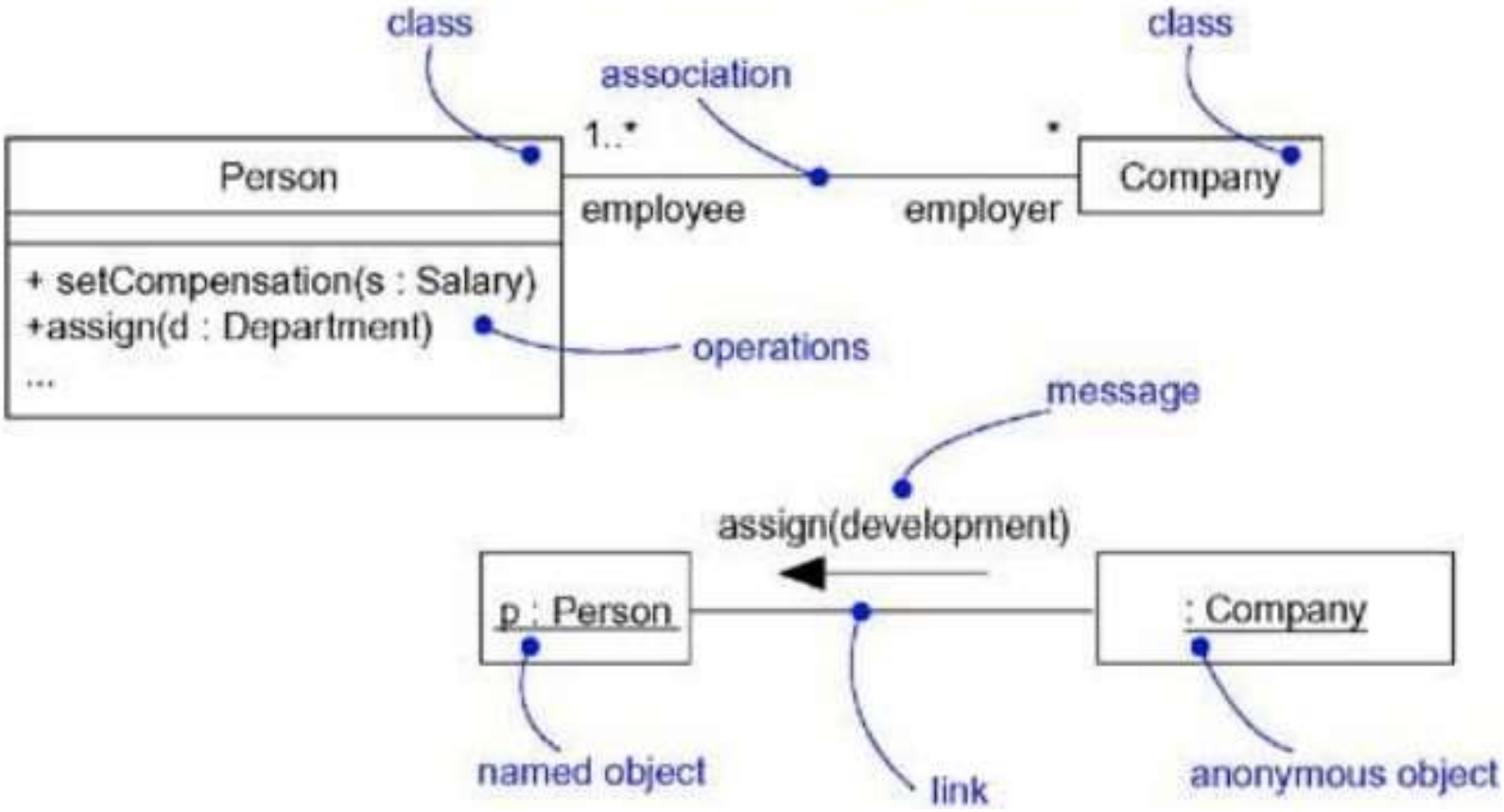
Interactions

- In the context of an interaction, you may find instances of classes, components, nodes, and use cases.
- Although abstract classes and interfaces, by definition, may not have any direct instances, you may find instances of these things in an interaction.
- You can think of an object diagram as a representation of the static aspect of an interaction, setting the stage for the interaction by specifying all the objects that work together.
- An interaction goes further by introducing a dynamic sequence of messages that may pass along the links that connect these objects.

Interactions

- **Links**
- A link is a semantic connection among objects. In general, **a link is an instance of an association**.
- As Figure 15-2 shows, wherever a class has an association to another class, there may be a link between the instances of the two classes; wherever there is a link between two objects, one object can send a message to the other object.

Figure 15-2 Links and Associations



Interactions

- A link specifies a path along which one object can dispatch a message to another (or the same) object. Most of the time, it is sufficient to specify that such a path exists.
- **Note**
- As an instance of an association, a link may be rendered with most of the adornments appropriate to associations, such as a name, association role name, navigation, and aggregation.
- *Multiplicity, however, does not apply to links, since they are instances of an association.*

Interactions

- **Messages**
- Suppose you want to model the changing state of a group of objects over a period of time. (Think of it as taking a motion picture of a set of objects, each frame representing a successive moment in time.)
- If these objects are not totally idle, you'll see objects passing messages to other objects, sending events, and invoking operations.
- A message is the specification of a communication among objects that conveys information with the expectation that activity will ensue. The receipt of a message instance may be considered an instance of an event.
- When you pass a message, the action that results is an executable statement that forms an abstraction of a computational procedure. An action may result in a change in state.

Interactions

- **Messages**
- In the UML, you can model several kinds of actions.

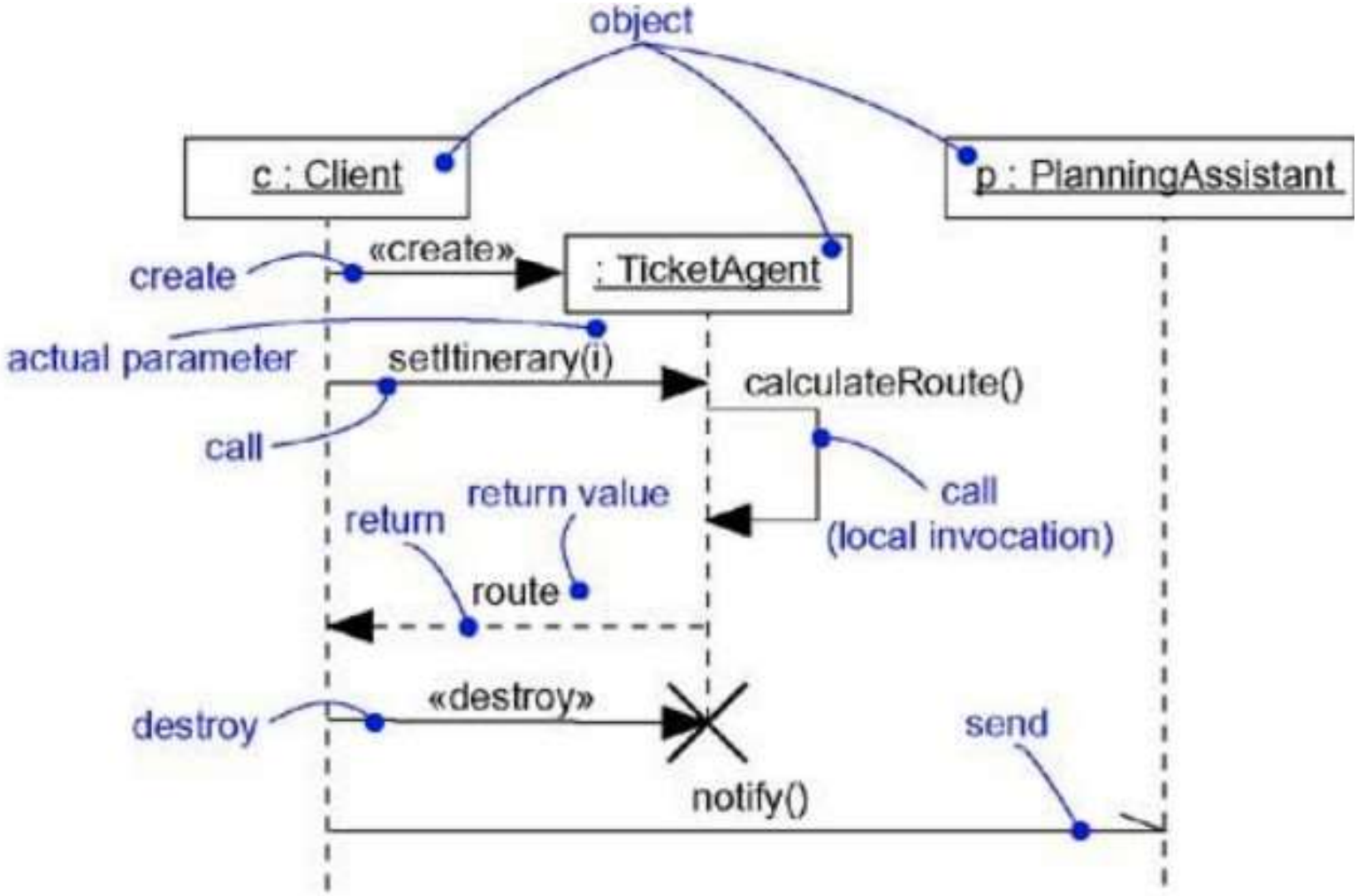
• Call	Invokes an operation on an object; an object may send a message to itself, resulting in the local invocation of an operation
• Return	Returns a value to the caller
• Send	Sends a signal to an object
• Create	Creates an object
• Destroy	Destroys an object; an object may commit suicide by destroying itself

- **Note**
- You can model complex actions in the UML, as well. In addition to the five basic kinds of actions listed above, you can attach an arbitrary string to a message, in which you can write complex expressions.
- The UML does not specify the syntax or semantics of such strings.
- The UML provides a visual distinction among these kinds of messages as shown in the following figure.
- The most common kind of message you'll model is the call, in which one object invokes an operation of another (or the same) object. An object can't just call any random operation.

Interactions

- **Messages**
- Figure 15-3

Messages



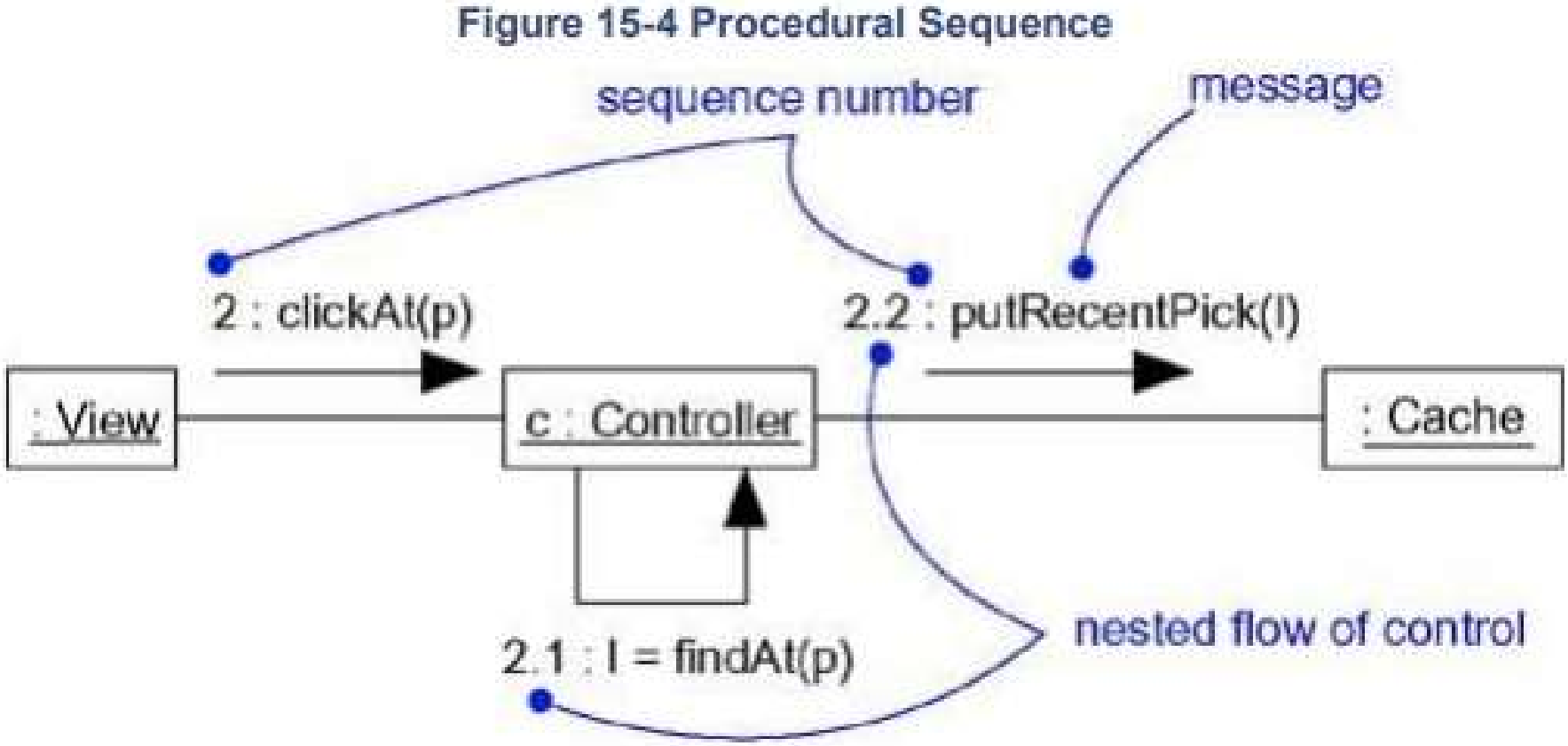
- When an object calls an operation or sends a signal to another object, you can provide actual parameters to the message. Similarly, when an object returns control to another object, you can model the return value, as well.
- You can also qualify an operation by the class or interface in which it is declared.

Interactions

- **Sequencing**
- When an object passes a message to another object (Like, delegating some action to the receiver), the receiving object might in turn send a message to another object, which might send a message to yet a different object, and so on.
- This stream of messages forms a sequence. Any sequence must have a beginning; the start of every sequence is rooted in some process or thread.
- Furthermore, any sequence will continue as long as the process or thread that owns it lives. A nonstop system, such as you might find in real time device control, will continue to execute as long as the node it runs on is up.
- Each process and thread within a system defines a distinct flow of control, and within each flow, messages are ordered in sequence by time.
- *To better visualize the sequence of a message, you can explicitly model the order of the message relative to the start of the sequence by prefixing the message with a sequence number set apart by a colon separator.*

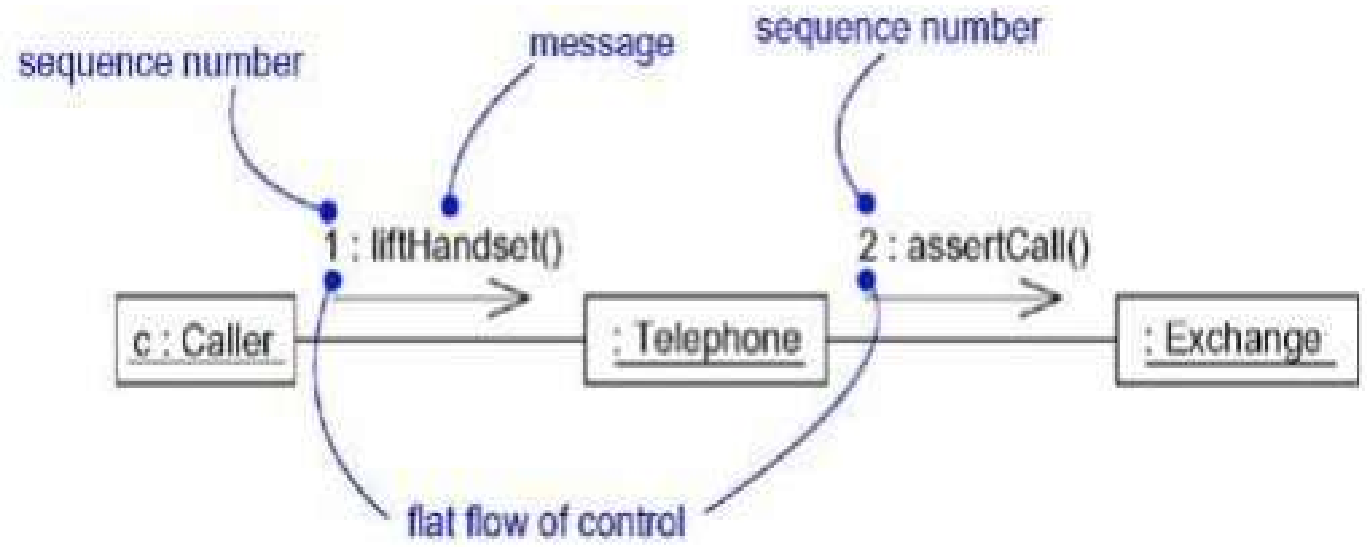
Interactions

- **Sequencing**
- Most commonly, you can specify a procedural or nested flow of control, rendered using a filled solid arrowhead.
- **Figure 15-4 Procedural Sequence**



Interactions

- **Sequencing**



- Use flat sequences only when modeling interactions in the context of use cases that involve the system as a whole, together with actors outside the system.
- Such sequences are often flat because control simply progresses from step to step, without any consideration for nested flows of control.
- In just about all other circumstances, you'll want to use procedural sequences, because they represent ordinary, nested operation calls of the type you find in most programming languages.

Interactions

- **Sequencing**
- When you are modeling interactions that involve multiple flows of control, it's especially important to identify the process or thread that sent a particular message.
- In the UML, you can distinguish one flow of control from another by prefixing a message's sequence number with the name of the process or thread that sits at the root of the sequence.
- For example, the expression D5 : ejectHatch(3) specifies that the operation ejectHatch is dispatched (with the actual argument 3) as the fifth message in the sequence rooted by the process or thread named.
- Not only can you show the actual arguments sent along with an operation or a signal in the context of an interaction, you can show the return values of a function as well.

Interactions

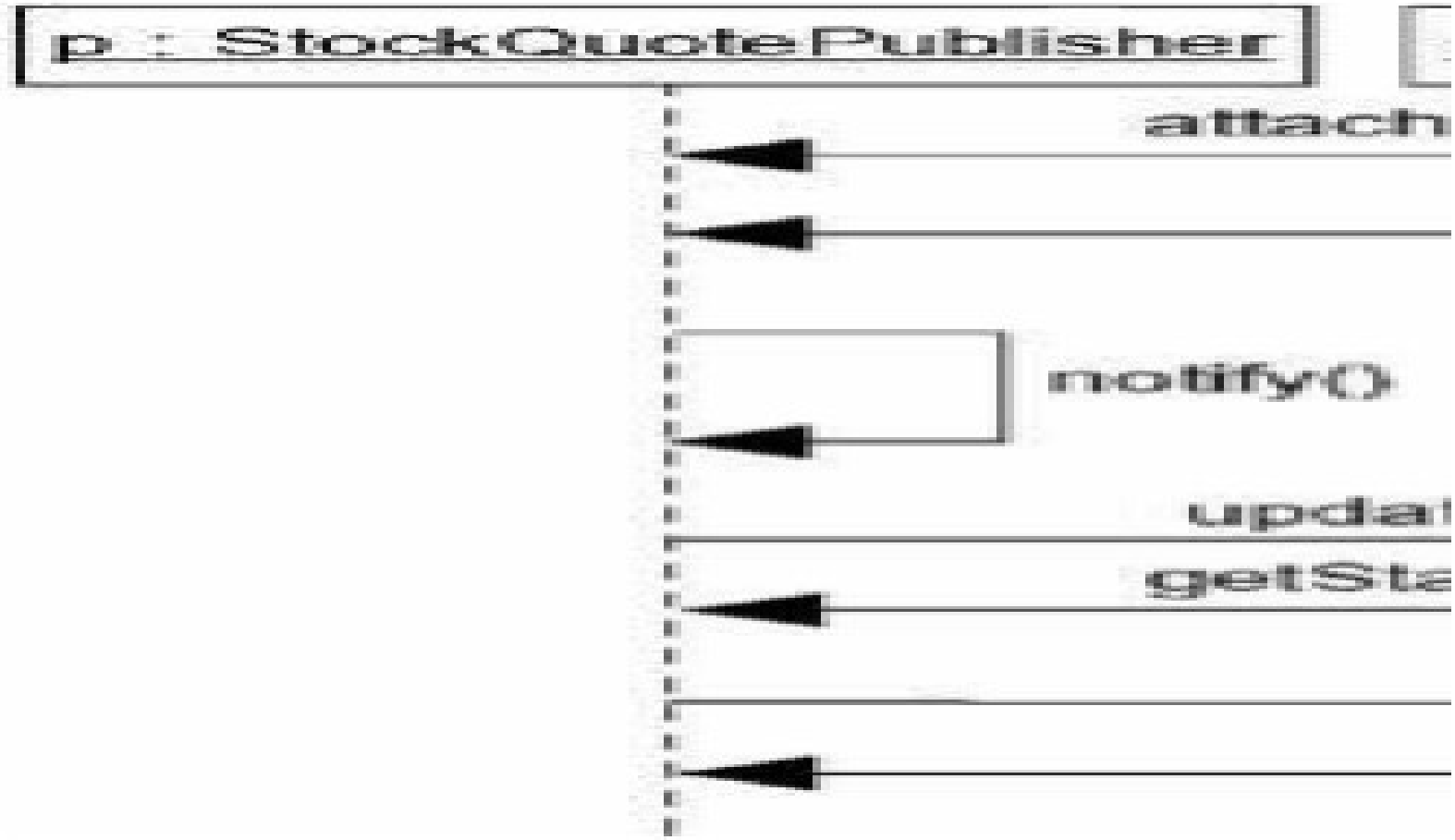
- **Modeling a Flow of Control**
- The most common purpose for which you'll use interactions is to model the flow of control that characterizes the behavior of a system as a whole, including use cases, patterns, mechanisms, and frameworks, or the behavior of a class or an individual operation.
- Whereas classes, interfaces, components, nodes, and their relationships model the static aspects of your system, interactions model its dynamic aspects.
-

Interactions

- **Modeling a Flow of Control**
- To model a flow of control,
  - Set the context for the interaction, whether it is the system as a whole, a class, or an individual operation.
  - Set the stage for the interaction by identifying which objects play a role; set their initial properties, including their attribute values, state, and role.
  - If your model emphasizes the structural organization of these objects, identify the links that connect them, relevant to the paths of communication that take place in this interaction. Specify the nature of the links using the UML's standard stereotypes and constraints, as necessary.
  - In time order, specify the messages that pass from object to object. As necessary, distinguish the different kinds of messages; include parameters and return values to convey the necessary detail of this interaction.
  - Also to convey the necessary detail of this interaction, adorn each object at every moment in time with its state and role.

Interactions

- **Modeling a Flow of Control**
- To model a flow of control,
  - Figure 15-6 Flow of Control by Time (A Sequence Diagram)



Interactions

- **Modeling a Flow of Control**
- To model a flow of control,
  - Figure 15-7 Flow of Control by Organization (A Collaboration Diagram)

Figure 15-7 Flow of Control by Organization