UNIT-IV

BASIC BEHAVIORAL MODELING-I

CONTENTS

1. Interactions
    - Terms and Concepts
    - Modeling Techniques

2. Interaction Diagrams
    - Terms and Concepts
    - Modeling Techniques

Interactions:

Terms and Concepts:

An interaction is a behavior that comprises a set of messages exchanged among objects in a set of roles within a context to accomplish a purpose. A message is a specification of a Collaboration between objects that conveys information with the expectation that activity will ensue.

Context

You may find an interaction wherever objects are linked to one another. You'll find interactions in the collaboration of objects that exist in the context of your system or subsystem. You will also find interactions in the context of an operation. Finally, you'll find interactions in the context of a class.

Most often, you'll find interactions in the collaboration of objects that exist in the context of your system or subsystem as a whole. For example, in a system for Web commerce, you'll find objects on the client (such as instances of the classes BookOrder and OrderForm) interacting with one another. You'll also find objects on the client (again, such as instances of BookOrder) interacting with objects on the server (such as instances of BackOrderManager). These interactions therefore not only involve localized collaborations of objects (such as the interactions surrounding

OrderForm), but they may also cut across many conceptual levels of your system (such as the interactions surrounding BackOrderManager).

You'll also find interactions among objects in the implementation of an operation, in the context of a class. You can use interactions to visualize, specify, construct, and document the semantics of a class. An interaction may also be found in the representation of a component, node, or use case, each of which is really a kind of UML 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.

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

In a collaboration, the interactions are usually prototypical things that play particular roles, not specific objects in the real world, although it is sometimes useful to describe collaborations among particular objects.

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 represent instances of these things in an interaction. Such instances do not represent direct instances of the abstract class or of the interface, but may represent, respectively, indirect (or prototypical) instances of any concrete children of the abstract class of some concrete class that realizes that interface.

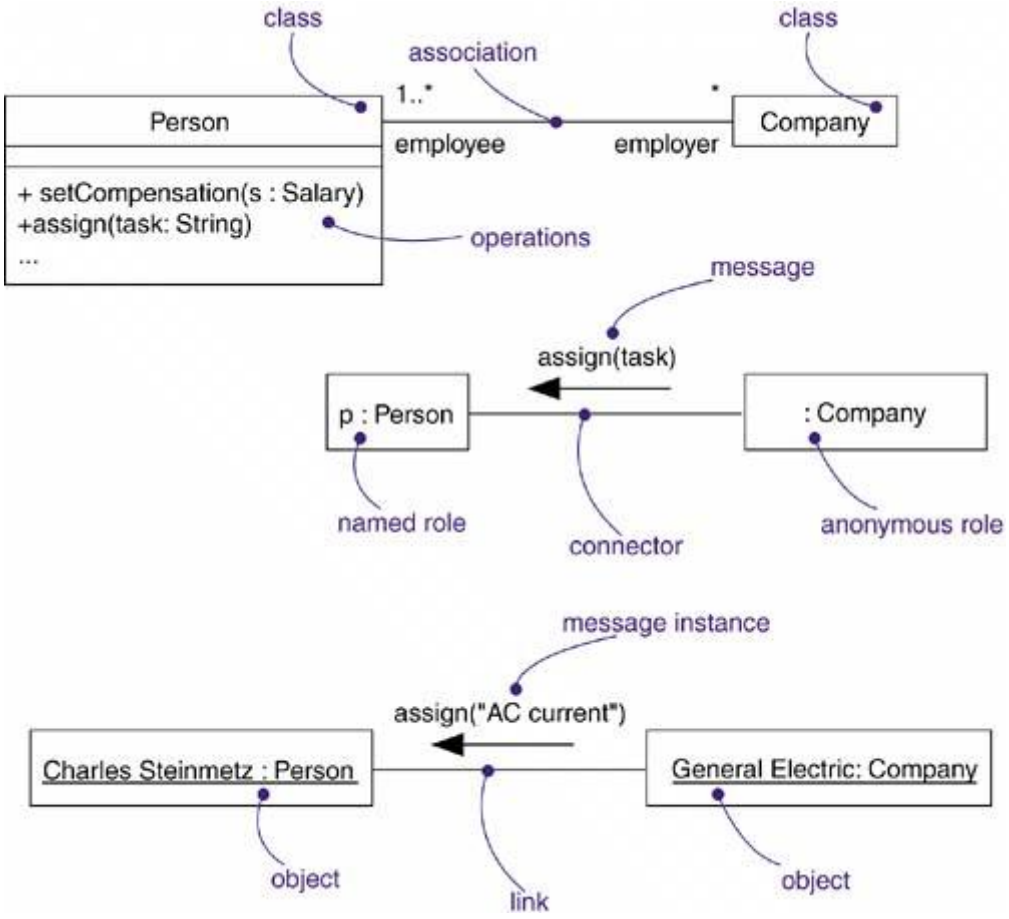
Links and Connectors

A link is a semantic connection among objects. In general, a link is an instance of an association. As Figure 16-2 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.

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. If you need to be more precise about how that path exists, you can adorn the appropriate end of the link with one of the following constraints:

association Specifies that the corresponding object is visible by association

association	Specifies that the corresponding object is visible by association
self	Specifies that the corresponding object is visible because it is the dispatcher of the operation
global	Specifies that the corresponding object is visible because it is in an enclosing scope
local	Specifies that the corresponding object is visible because it is in a local scope
parameter	Specifies that the corresponding object is visible because it is a parameter



Messages

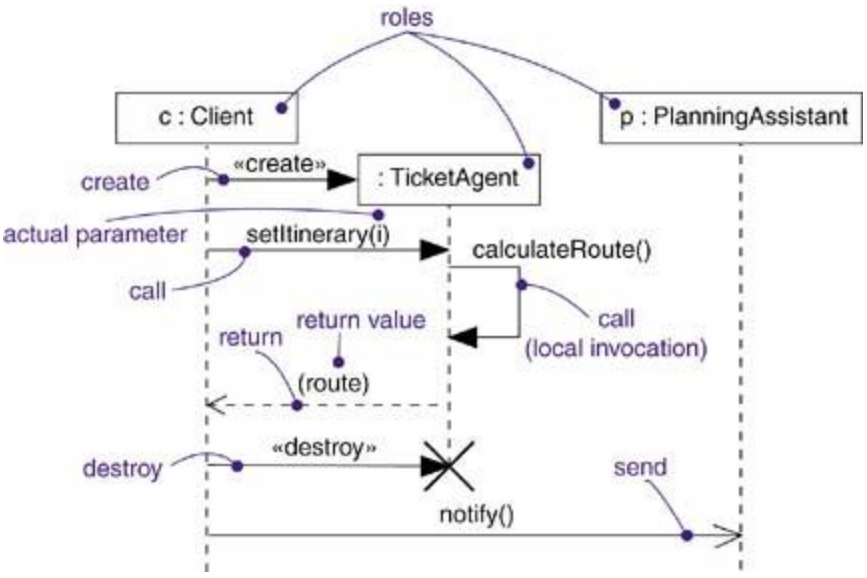
Suppose you have a set of objects and a set of links that connect those objects. If that's all you have, then you have a completely static model that can be represented by an object diagram. Object diagrams model the state of a society of objects at a given moment in time and are useful when you want to visualize, specify, construct, or document a static object structure.

A message is the specification of a Collaboration among objects that conveys information with the expectation that activity will ensue. The receipt of a message instance may be considered an occurrence of an event. (An occurrence is the UML name for an instance of an event.)

When you pass a message, an action usually results on its receipt. An action may result in a change in state of the target object and objects accessible from it.

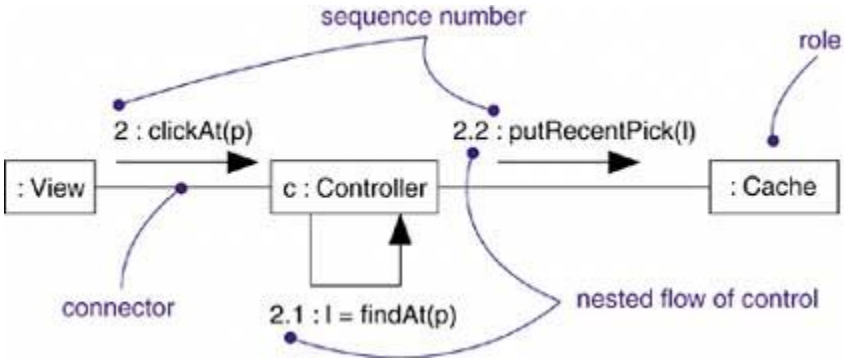
In the UML, you can model several kinds of messages.

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

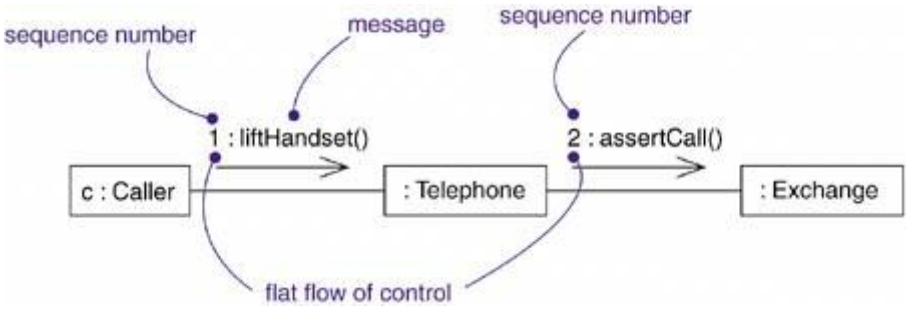


Sequencing

When an object passes a message to another object (in effect, 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 Collaboration diagram shows message flow between roles within a collaboration. Messages flow along connections of the collaboration.



Procedural Sequence



Flat Sequence

Creation, Modification, and Destruction

Most of the time the objects you show participating in an interaction exist for the entire duration of the interaction. However, in some interactions objects may be created (specified by a create message) and destroyed (specified by a destroy message). The same is true of links: The relationships among objects may come and go. To specify if an object or link enters and/or leaves during an interaction, you can attach a note to its role within a Collaboration diagram.

During an interaction, an object typically changes the values of its attributes, its state, or its roles. You can represent the modification of an object in a sequence diagram by showing the state or the values on the lifeline.

Within a sequence diagram, the lifetime, creation, and destruction of objects or roles are explicitly shown by the vertical extent of their lifelines. Within a Collaboration diagram, creation and destruction must be indicated using notes. Use sequence diagrams if object lifetimes are important to show.

Representation

When you model an interaction, you typically include both roles (each one representing objects that appear in an instance of the interaction) and messages (each one representing the Collaboration between objects, with some resulting action).

You can visualize those roles and messages involved in an interaction in two ways: by emphasizing the time ordering of its messages, and by emphasizing the structural organization of the roles that send and receive messages. In the UML, the first kind of representation is called a sequence diagram; the second kind of representation is called a Collaboration diagram. Both sequence diagrams and Collaboration diagrams are kinds of interaction diagrams. (UML also has a more specialized kind of interaction diagram called a timing diagram, which shows the exact times at which messages are exchanged by roles. This diagram is not covered in this book. See the UML Reference Manual for more information.)

Sequence diagrams and Collaboration diagrams are similar, meaning that you can take one and transform it into the other, although they often show different information, so it may not be so useful to go back and forth. There are some visual differences. First, sequence diagrams permit you to model the lifeline of an object. An object's lifeline represents the existence of the object at a particular time, possibly covering the object's creation and destruction. Second, Collaboration diagrams permit you to model the structural links that may exist among the objects in an interaction.

Common Modeling Techniques

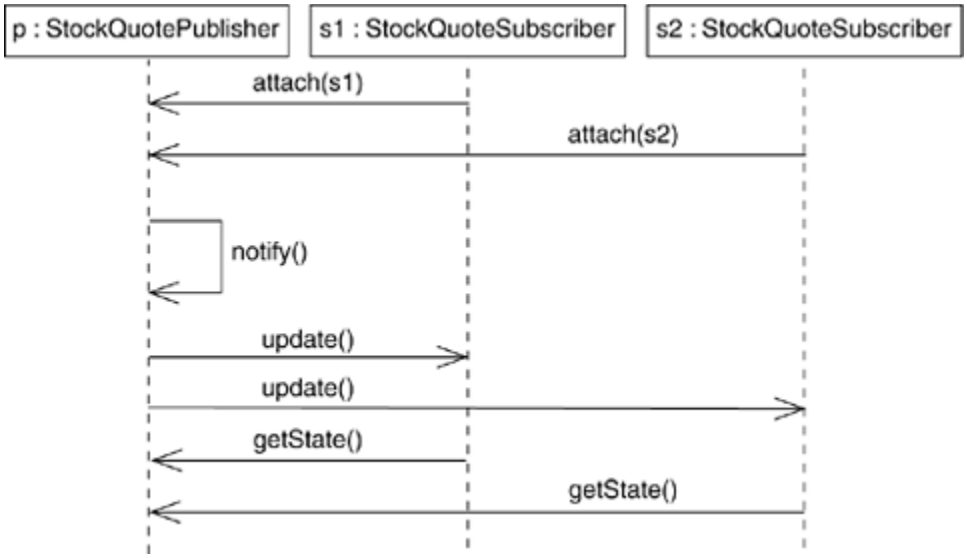
Modeling a Flow of Control

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.

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. Name the roles.
- If your model emphasizes the structural organization of these objects, identify the links that connect them, relevant to the paths of Collaboration 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.



Modeling flow of control by time ordering

The below figure is semantically equivalent to the previous one, but it is drawn as a Collaboration diagram, which emphasizes the structural organization of the objects. This figure shows the same flow of control, but it also provides a visualization of the links among these objects.

Modeling Flows of Control by Organization

Interaction Diagrams:

erms and Concepts

An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis. A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Graphically, a Collaboration diagram is a collection of vertices and arcs.

Common Properties

An interaction diagram is just a special kind of diagram and shares the same common properties as do all other diagrams - a name and graphical contents that are a projection into a model. What distinguishes an interaction diagram from all other kinds of diagrams is its particular content.

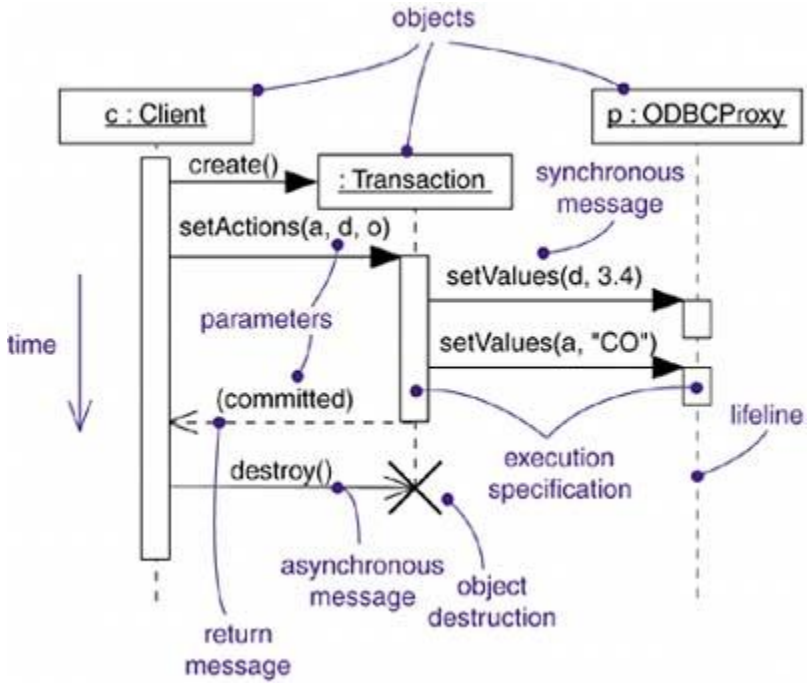
Contents

Interaction diagrams commonly contain

- Roles or objects
- Collaborations or links
- Messages
- Like all other diagrams, interaction diagrams may contain notes and constraints.

Sequence Diagrams

A sequence diagram emphasizes the time ordering of messages. you form a sequence diagram by first placing the objects or roles that participate in the interaction at the top of your diagram, across the horizontal axis. Typically, you place the object or role that initiates the interaction at the left, and increasingly more subordinate objects or roles to the right. Next, you arrange the messages that these objects send and receive along the vertical axis in order of increasing time from top to bottom. This gives the reader a clear visual cue to the flow of control over time.



Sequence diagrams have two features that distinguish them from Collaboration diagrams.

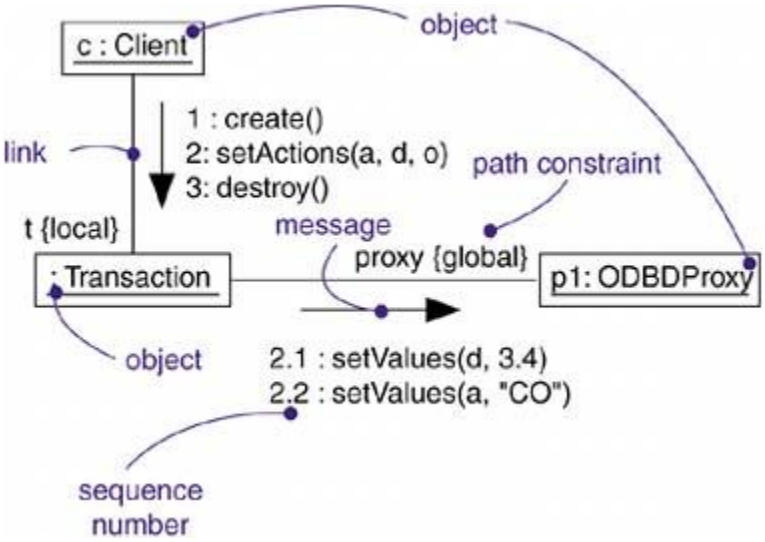
First, there is the lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence for the duration of the interaction, so these objects are all aligned at the top of the diagram, with their lifelines drawn from the top of the diagram to the bottom. Objects may be created during the interaction. Their lifelines start with the receipt of the message create (drawn to box at the head of the lifeline). Objects may be destroyed during the interaction. Their lifelines end with the receipt of the message destroy (and are given the visual cue of a large X, marking the end of their lives).

Second, there is the focus of control. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure. The top of the rectangle is aligned with the start of the action; the bottom is aligned with its completion (and can be marked by a return message). You can show the nesting of a focus of control (caused by recursion, a call to a self-operation, or by a callback from another object) by stacking another focus of control slightly to the right of its parent (and can do so to an arbitrary depth).

Collaboration Diagrams

A Collaboration diagram emphasizes the organization of the objects that participate in an interaction. you form a Collaboration diagram by first placing the objects that participate in the interaction as the vertices in a graph. Next, you render the links that connect these objects as the arcs of this graph. The links may have role names to identify them. Finally, you adorn these links

with the messages that objects send and receive. This gives the reader a clear visual cue to the flow of control in the context of the structural organization of objects that collaborate.



Collaboration diagram have two features that distinguish them from sequence diagrams.

First, there is the path. You render a path corresponding to an association. You also render paths corresponding to local variables, parameters, global variables, and self access. A path represents a source of knowledge to an object.

Second, there is the sequence number. To indicate the time order of a message, you prefix the message with a number (starting with the message numbered 1), increasing monotonically for each new message in the flow of control (2, 3, and so on). To show nesting, you use Dewey decimal numbering (1 is the first message, which contains message 1.1 and message 1.2, and so on). You can show nesting to an arbitrary depth. Note also that, along the same link, you can show many messages (possibly being sent from different directions), and each will have a unique sequence number.

Semantic Equivalence

Because they both derive from the same information in the UML's metamodel, sequence diagrams and communication diagrams are semantically equivalent. As a result, you can take a diagram in one form and convert it to the other without any loss of information, as you can see in the previous two figures, which are semantically equivalent. However, this does not mean that both diagrams will explicitly visualize the same information.

For example, in the previous two figures, the communication diagram shows how the objects are linked (note the {local} and {global} annotations); the corresponding sequence

diagram does not. Similarly, the sequence diagram shows message return (note the return value committed), but the corresponding communication diagram does not. In both cases, the two diagrams share the same underlying model, but each may render some things the other does not. However, a model entered in one format may lack some of the information shown on the other format, so although the underlying model can include both kinds of information, the two kinds of diagrams may lead to different models.

Common Uses

You use interaction diagrams to model the dynamic aspects of a system. These dynamic aspects may involve the interaction of any kind of instance in any view of a system's architecture, including instances of classes (including active classes), interfaces, components, and nodes.

When you use an interaction diagram to model some dynamic aspect of a system, you do so in the context of the system as a whole, a subsystem, an operation, or a class. You can also attach interaction diagrams to use cases (to model a scenario) and to collaborations (to model the dynamic aspects of a society of objects).

When you model the dynamic aspects of a system, you typically use interaction diagrams in two ways.

1. To model flows of control by time ordering

Here you'll use sequence diagrams. Modeling a flow of control by time ordering emphasizes the passing of messages as they unfold over time, which is a particularly useful way to visualize dynamic behavior in the context of a use case scenario. Sequence diagrams do a better job of visualizing simple iteration and branching than do communication diagrams.

2. To model flows of control by organization

Here you'll use communication diagrams. Modeling a flow of control by organization emphasizes the structural relationships among the instances in the interaction, along which messages may be passed.

Common Uses

You use interaction diagrams to model the dynamic aspects of a system. These dynamic aspects may involve the interaction of any kind of instance in any view of a system's architecture, including instances of classes (including active classes), interfaces, components, and nodes.

When you use an interaction diagram to model some dynamic aspect of a system, you do so in the context of the system as a whole, a subsystem, an operation, or a class. You can also attach interaction diagrams to use cases (to model a scenario) and to collaborations (to model the dynamic aspects of a society of objects).

When you model the dynamic aspects of a system, you typically use interaction diagrams in two ways.

1. To model flows of control by time ordering

Here you'll use sequence diagrams. Modeling a flow of control by time ordering emphasizes the passing of messages as they unfold over time, which is a particularly useful way to visualize dynamic behavior in the context of a use case scenario. Sequence diagrams do a better job of visualizing simple iteration and branching than do communication diagrams.

2. To model flows of control by organization

Here you'll use communication diagrams. Modeling a flow of control by organization emphasizes the structural relationships among the instances in the interaction, along which messages may be passed.

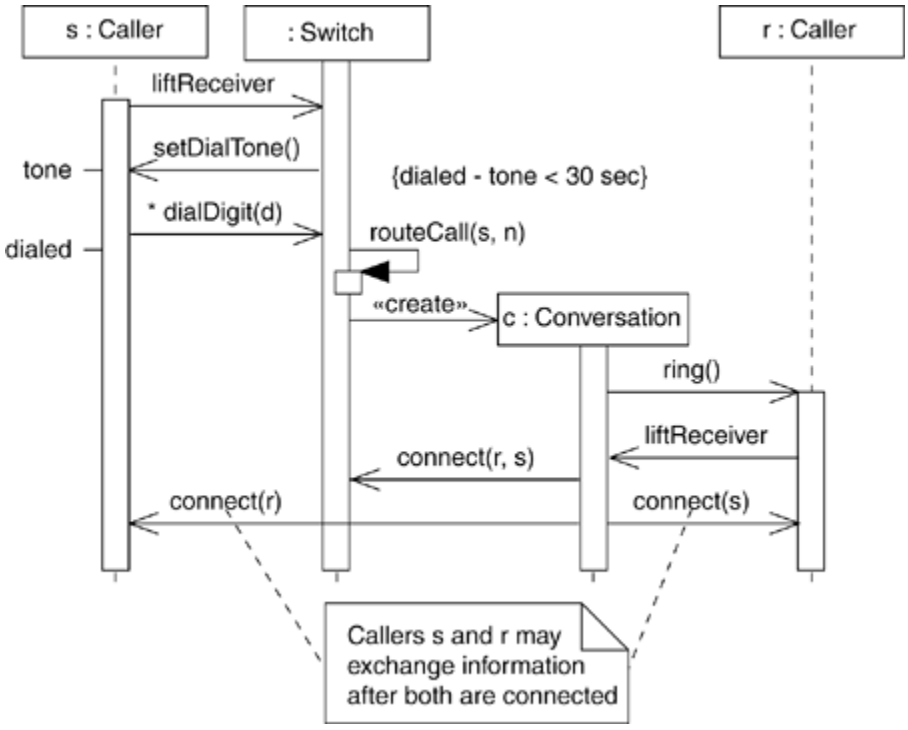
Common Modeling Techniques

Modeling Flows of Control by Time Ordering

Consider the objects that live in the context of a system, subsystem, operation or class. Consider also the objects and roles that participate in a use case or collaboration. To model a flow of control that winds through these objects and roles, you use an interaction diagram; to emphasize the passing of messages as they unfold over time, you use a sequence diagram, a kind of interaction diagram.

To model a flow of control by time ordering,

- Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.
- Set the stage for the interaction by identifying which objects play a role in the interaction. Lay them out on the sequence diagram from left to right, placing the more important objects to the left and their neighboring objects to the right.
- Set the lifeline for each object. In most cases, objects will persist through the entire interaction. For those objects that are created and destroyed during the interaction, set their lifelines, as appropriate, and explicitly indicate their birth and death with appropriately stereotyped messages.
- Starting with the message that initiates this interaction, lay out each subsequent message from top to bottom between the lifelines, showing each message's properties (such as its parameters), as necessary to explain the semantics of the interaction.
- If you need to visualize the nesting of messages or the points in time when actual computation is taking place, adorn each object's lifeline with its focus of control.
- If you need to specify time or space constraints, adorn each message with a timing mark and attach suitable time or space constraints.
- If you need to specify this flow of control more formally, attach pre- and postconditions to each message.



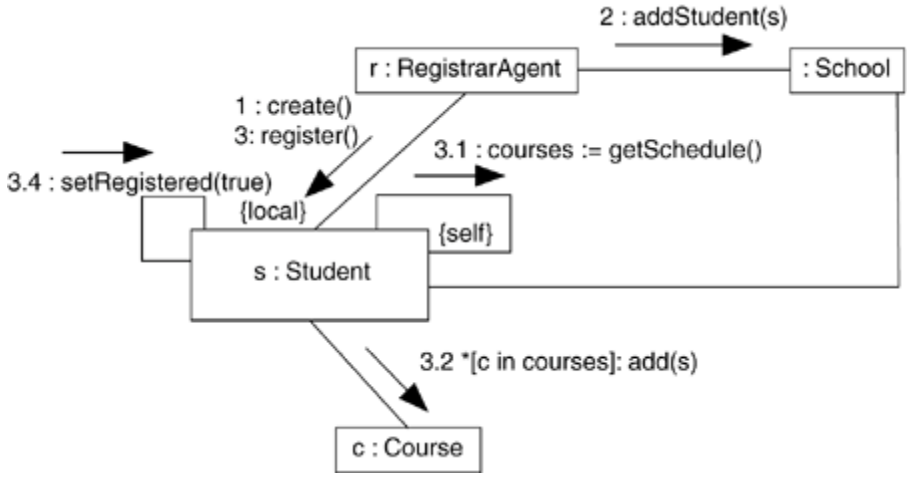
Modeling Flows of Control by Organization

Consider the objects that live in the context of a system, subsystem, operation, or class. Consider also the objects and roles that participate in a use case or collaboration. To model a flow of control that winds through these objects and roles, you use an interaction diagram; to show the passing of messages in the context of that structure, you use a communication diagram, a kind of interaction diagram.

o model a flow of control by organization,

- Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.
- Set the stage for the interaction by identifying which objects play a role in the interaction. Lay them out on the communication diagram as vertices in a graph, placing the more important objects in the center of the diagram and their neighboring objects to the outside.
- Specify the links among these objects, along which messages may pass.
    1. Lay out the association links first; these are the most important ones, because they represent structural connections.
    2. Lay out other links next, and adorn them with suitable path annotations (such as global and local) to explicitly specify how these objects are related to one another.
- Starting with the message that initiates this interaction, attach each subsequent message to the appropriate link, setting its sequence number, as appropriate. Show nesting by using Dewey decimal numbering.
- If you need to specify time or space constraints, adorn each message with a timing mark and attach suitable time or space constraints.

- If you need to specify this flow of control more formally, attach pre- and postconditions to each message.



Forward and Reverse Engineering

Forward engineering (the creation of code from a model) is possible for both sequence and communication diagrams, especially if the context of the diagram is an operation. For example, using the previous communication diagram, a reasonably clever forward engineering tool could generate the following Java code for the operation register, attached to the Student class.

```
public void register() {
 CourseCollection courses = getSchedule();
 for (int i = 0; i < courses.size(); i++)
  courses.item(i).add(this);
 this.registered = true;
}
```

Reverse engineering (the creation of a model from code) is also possible for both sequence and communication diagrams, especially if the context of the code is the body of an operation. Segments of the previous diagram could have been produced by a tool from a prototypical execution of the register operation.