Subject: Object Oriented Analysis and Design

Module Number- 3: Class and Object Diagrams

SME NAME: ENTER NAME
SUBMISSION DATE: MENTION DATE
VERSION CODE: TO BE FILLED AT HO
RELEASED DATE: TO BE FILLED AT HO

Class and Object Diagrams

CLASS & OBJECT DIAGRAMS

Class Diagrams:

- A class diagram is a diagram that shows a set of classes, interfaces, and collaborations and their relationships.
- Graphically, a class diagram is a collection of vertices and arcs.
- A class diagram is just a special kind of diagram and shares the same common properties as do all other diagrams name and graphical content that are a projection into a model.
- What distinguishes a class diagram from other kinds of diagrams is its particular content.

Class and Object Diagrams

Contents:

Class diagrams commonly contain the following things:

- Classes
- Interfaces
- Collaborations
- Dependency, generalization, and association relationships.
- Like all other diagrams, class diagrams may contain notes and constraints.

Class diagrams may also contain packages or subsystems, both of which are used to group elements of your model into larger chunks. Sometimes you'll want to place instances in your class diagrams as well, especially when you want to visualize the (possibly dynamic) type of an instance.

Class and Object Diagrams

Common Uses :

We use class diagrams to model the static design view of a system.

When we model the static design view of a system, we typically use class diagrams in one of three ways.

1. To model the vocabulary of a system

Modeling the vocabulary of a system involves making a decision about which abstractions are a part of the system under consideration and which fall outside its boundaries. We use class diagrams to specify these abstractions and their responsibilities.

Class and Object Diagrams

2. To model simple collaborations

A collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. For example, when you are modeling the semantics of a transaction in a distributed system, you can't just stare at a single class to understand what's going on. Rather, these semantics are carried out by a set of classes that work together. You use class diagrams to visualize and specify this set of classes and their relationships.

3. To model a logical database schema

Think of a schema as the blueprint for the conceptual design of a database. In many domains, you'll want to store persistent information in a relational database or in an object-oriented database. You can model schemas for these databases using class diagrams.

Class and Object Diagrams

COMMON MODELING TECHNIQUES

1.Modeling Simple Collaborations

To model a collaboration,

- Identify the mechanism you'd like to model. A mechanism represents some function or behavior of the part of the system you are modeling that results from the interaction of a society of classes, interfaces, and other things.
- For each mechanism, identify the classes, interfaces, and other collaborations that participate in this collaboration. Identify the relationships among these things as well.
- Use scenarios to walk through these things. Along the way, you'll discover parts of your model that were missing and parts that were just plain semantically wrong.
- Be sure to populate these elements with their contents. For classes, start with getting a good balance of responsibilities. Then, over time, turn these in to concrete attributes and operations.

Class and Object Diagrams

2. Modeling a Logical Database Schema

To model a schema,

- Identify those classes in your model whose state must transcend the lifetime of their applications.
- Create a class diagram that contains these classes. You can define your own set of stereotypes and tagged values to address database-specific details.
- Expand the structural details of these classes. In general, this means specifying the details of their attributes and focusing on the associations and their multiplicities that relate these classes.

Class and Object Diagrams

- Watch for common patterns that complicate physical database design, such as cyclic associations and one-to-one associations. Where necessary, create intermediate abstractions to simplify your logical structure.
- Consider also the behavior of these classes by expanding operations that are important for data access and data integrity. In general, to provide a better separation of concerns, business rules concerned with the manipulation of sets of these objects should be encapsulated in a layer above these persistent classes.
- Where possible, use tools to help you transform your logical design into a physical design.

Class and Object Diagrams

Object Diagrams:

Object diagrams model the instances of things contained in class diagrams. An object diagram shows a set of objects and their relationships at a point in time.

- An object diagram is a diagram that shows a set of objects and their relationships at a point in time. Graphically, an object diagram is a collection of vertices and arcs.
- An object diagram is a special kind of diagram and shares the same common properties as all other diagrams that is, a name and graphical contents that are a projection into a model.
- What distinguishes an object diagram from all other kinds of diagrams is its particular content.

Class and Object Diagrams

Contents

Object diagrams commonly contain

- Objects
- Links

Like all other diagrams, object diagrams may contain notes and constraints.

Sometimes you'll want to place classes in your object diagrams as well, especially when you want to visualize the classes behind each instance.

Class and Object Diagrams

Common Uses

You use object diagrams to model the static design view or static process view of a system just as you do with class diagrams, but from the perspective of real or prototypical instances. This view primarily supports the functional requirements of a system that is, the services the system should provide to its end users.

Object diagrams let you model static data structures.

When you model the static design view or static process view of a system, you typically use object diagrams in one way:

Class and Object Diagrams

To model object structures

- Modeling object structures involves taking a snapshot of the objects in a system at a given moment in time.
- An object diagram represents one static frame in the dynamic storyboard represented by an interaction diagram. You use object diagrams to visualize, specify, construct, and document the existence of certain instances in your system, together with their relationships to one another.

Class and Object Diagrams

COMMON MODELING TECHNIQUES

Modeling Object Structures

To model an object structure,

- Identify the mechanism you'd like to model. A mechanism represents some function or behavior of the part of the system you are modeling that results from the interaction of a society of classes, interfaces, and other things.
- Create a collaboration to describe a mechanism.

Class and Object Diagrams

- For each mechanism, identify the classes, interfaces, and other elements that participate in this collaboration; identify the relationships among these things as well.
- Consider one scenario that walks through this mechanism. Freeze that scenario at a moment in time, and render each object that participates in the mechanism.
- Expose the state and attribute values of each such object, as necessary, to understand the scenario.
- Similarly, expose the links among these objects, representing instances of associations among them.

Class and Object Diagrams

BASIC BEHAVIOURAL MODELLING

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.

Class and Object Diagrams

- 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.

Class and Object Diagrams

- 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**).

Class and Object Diagrams

- You'll also find interactions among objects in the implementation of an operation. The parameters of an operation, any variables local to the operation, and any objects global to the operation (but still visible to the operation) may interact with one another to carry out the algorithm of that operation's implementation.
- **For example,** invoking the operation **moveToPosition (p :Position)** defined for a class in a mobile robot will involve the interaction of a parameter **(p),** an object global to the operation (such as the object **currentPosition**), and possibly several local objects (such as local variables used by the operation to calculate intermediate points in a path to the new position).

Class and Object Diagrams

- Finally, you will find interactions in the context of a class. You can use interactions to visualize, specify, construct, and document the semantics of a class.

- **For example,** to understand the meaning of a class **RayTraceAgent**, you might create interactions that show how the attributes of that class collaborate with one another (and with objects global to instances of the class and with parameters defined in the class's operations).

Class and Object Diagrams

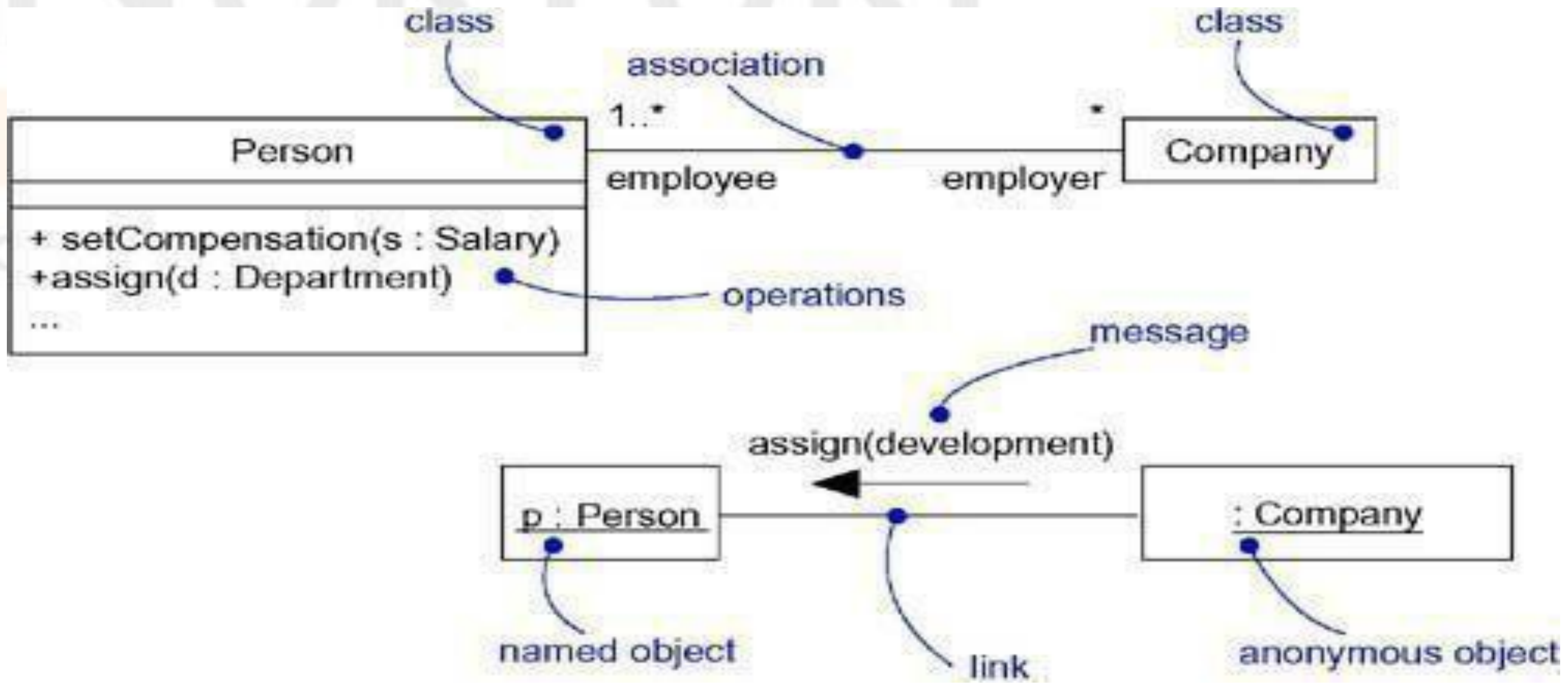
Objects and Roles

- 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.
- 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. 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.

Class and Object Diagrams

Links

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



Class and Object Diagrams

- 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 any of the following standard stereotypes.

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

Class and Object Diagrams

Messages

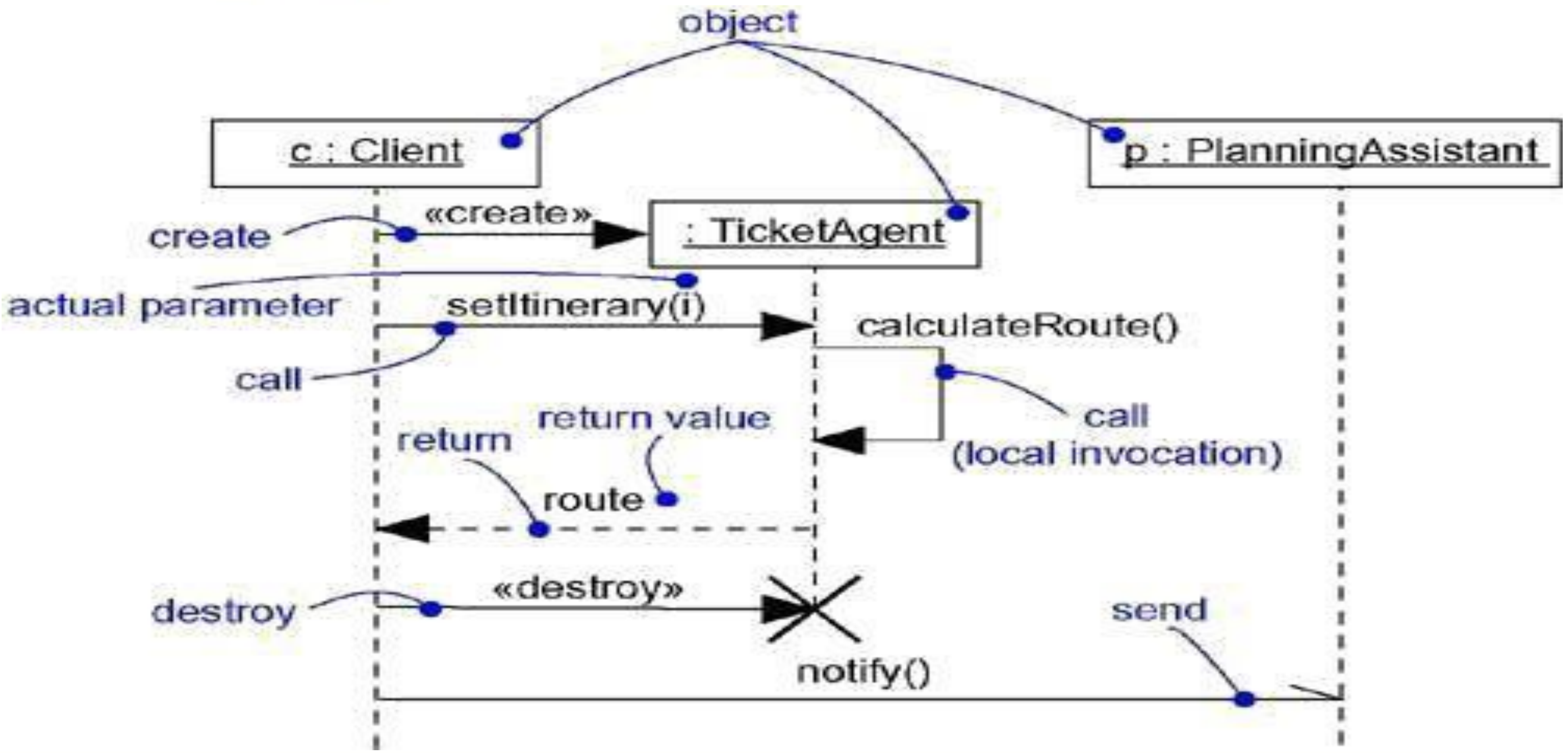
A message is the specification of a communication among objects that conveys information with the expectation that activity will ensue. The receipt of a message instance may be considered an instance of an event.

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

Class and Object Diagrams

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

The UML provides a visual distinction among these kinds of messages, as Figure shows.



Class and Object Diagrams

- 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. If an object, such as **c** in the example above, calls the operation **setItinerary** on an instance of the class **TicketAgent**, the operation setItinerary must not only be defined for the class **TicketAgent** (that is, it must be declared in the class **TicketAgentor** one of its parents), it must also be visible to the caller **c**.

- 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.

Class and Object Diagrams

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 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.

Class and Object Diagrams

- 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.

- Most commonly, you can specify a procedural or nested flow of control, rendered using a filled solid arrowhead, as Figure shows. In this case, the message **findAt** is specified as the first message nested in the second message of the sequence (2.1).

Class and Object Diagrams

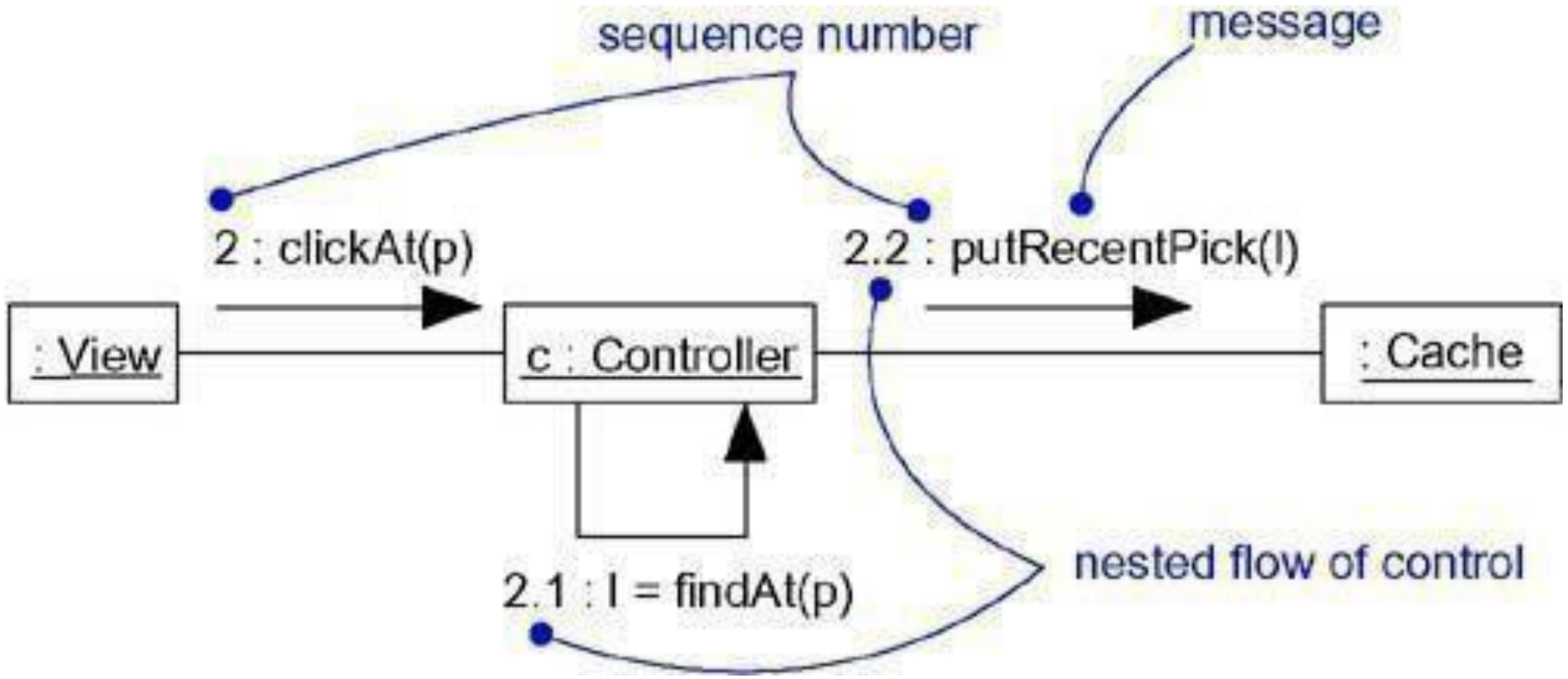


Figure: Procedural Sequence

Class and Object Diagrams

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 one of the following constraints to the element:

new	Specifies that the instance or link is created during execution of the enclosing interaction
destroyed	Specifies that the instance or link is destroyed prior to completion of execution of the en losing interaction
transient	Specifies that the instance or link is created during execution of the enclosing interaction but is destroyed before completion of execution

Class and Object Diagrams

Representation

- When you model an interaction, you typically include both objects (each one playing a specific role) and messages (each one representing the communication between objects, with some resulting action).
- You can visualize those objects 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 objects 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**.

Class and Object Diagrams

- Sequence diagrams and collaboration diagrams are largely isomorphic, meaning that you can take one and transform it into the other without loss of information.
- There are some visual differences, however.
- 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.

Class and Object Diagrams

Common Modeling Techniques

Modeling a Flow of Control

When you model an interaction, you essentially build a storyboard of the actions that take place among a set of objects. Techniques such as CRC cards are particularly useful in helping you to discover and think about such interactions.

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.

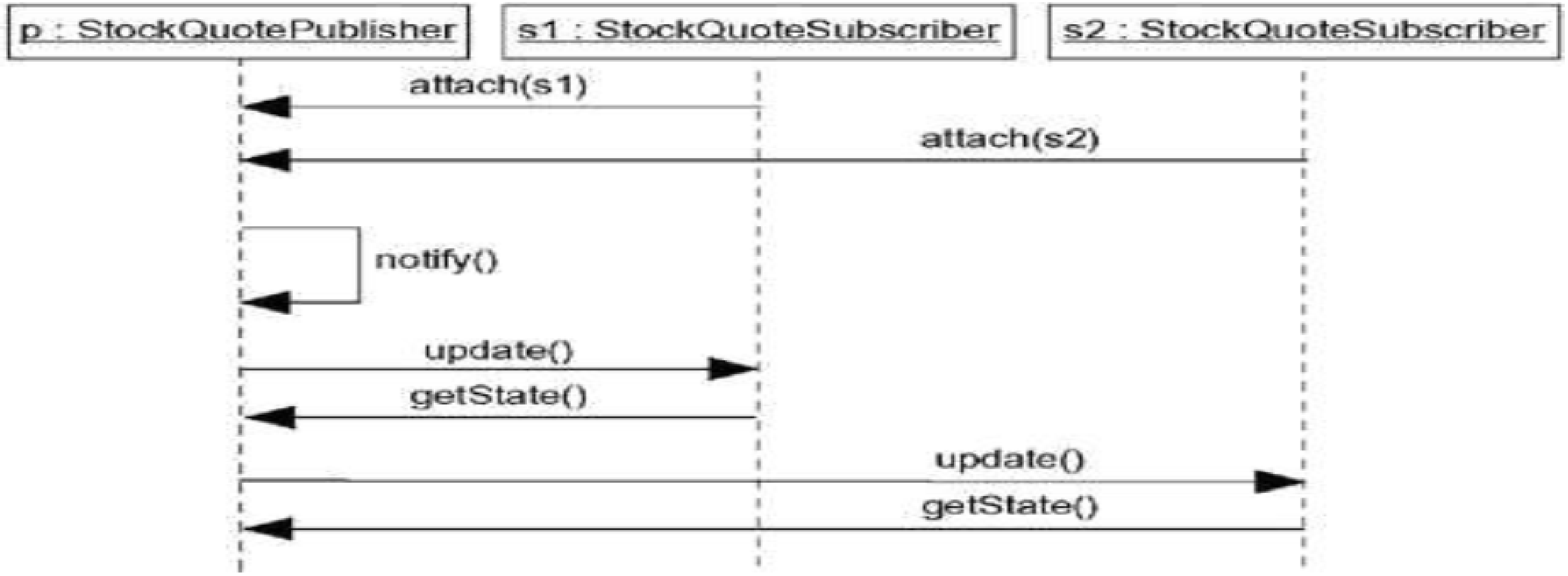
Class and Object Diagrams

- 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.

For example, following Figure shows a set of objects that interact in the context of a publish and subscribe mechanism (an instance of the observer design pattern). This figure includes three objects: **p** (a **StockQuotePublisher**), **s1**, and **s2** (both instances of **StockQuoteSubscriber**). This figure is an example of a sequence diagram, which emphasizesthe time order of messages.

Class and Object Diagrams

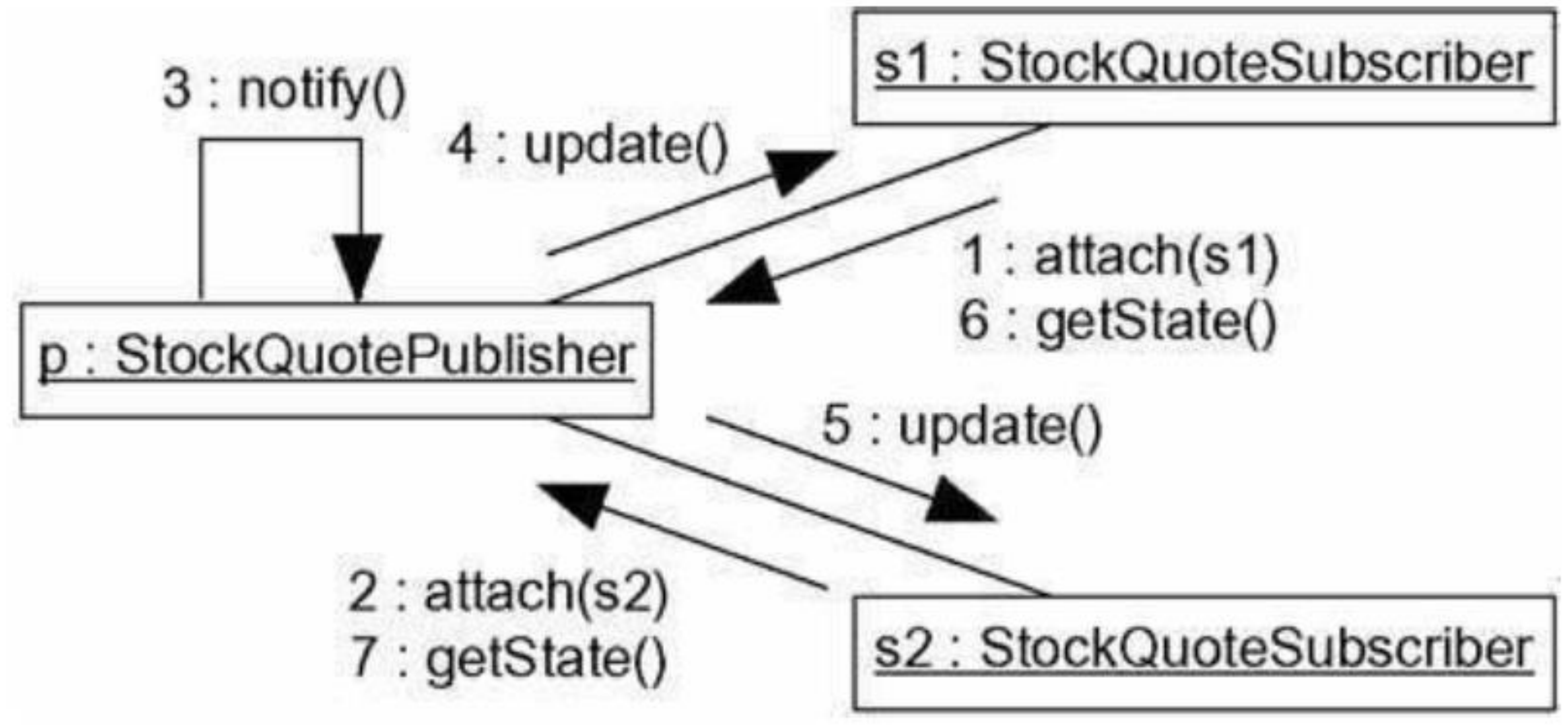
Figure Flow of Control by Time



Class and Object Diagrams

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.

Figure Flow of Control by Organization



Class and Object Diagrams

Interaction Diagrams

Terms 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.

Class and Object Diagrams

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

- Objects
- Links
- Messages

Like all other diagrams, interaction diagrams may contain notes and constraints.

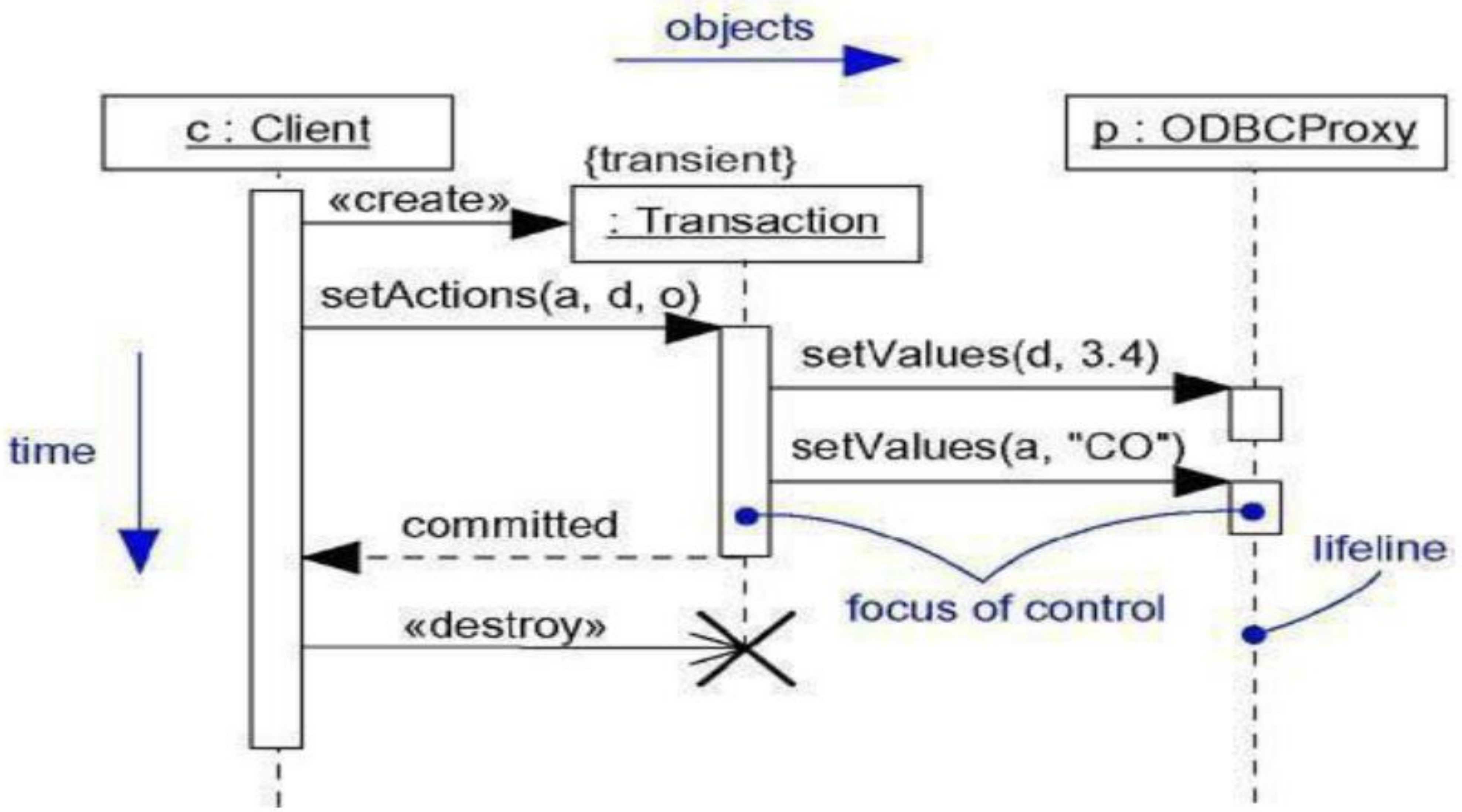
Class and Object Diagrams

Sequence Diagrams

- A sequence diagram emphasizes the time ordering of messages. As Figure shows, you form a sequence diagram by first placing the objects that participate in the interaction at the top of your diagram, across the X axis.
- Typically, you place the object that initiates the interaction at the left, and increasingly more subordinate objects to the right.
- Next, you place the messages that these objects send and receive along the Y 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.

Class and Object Diagrams

Figure Sequence Diagram



Class and Object Diagrams

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

- First, there is the object 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 stereotyped as create. Objects may be destroyed during the interaction. Their lifelines end with the receipt of the message stereotyped as destroy (and are given the visual cue of a large X, marking the end of their lives).

Class and Object Diagrams

- 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). If you want to be especially precise about where the focus of control lies, you can also shade the region of the rectangle during which the object's method is actually computing (and control has not passed to another object).

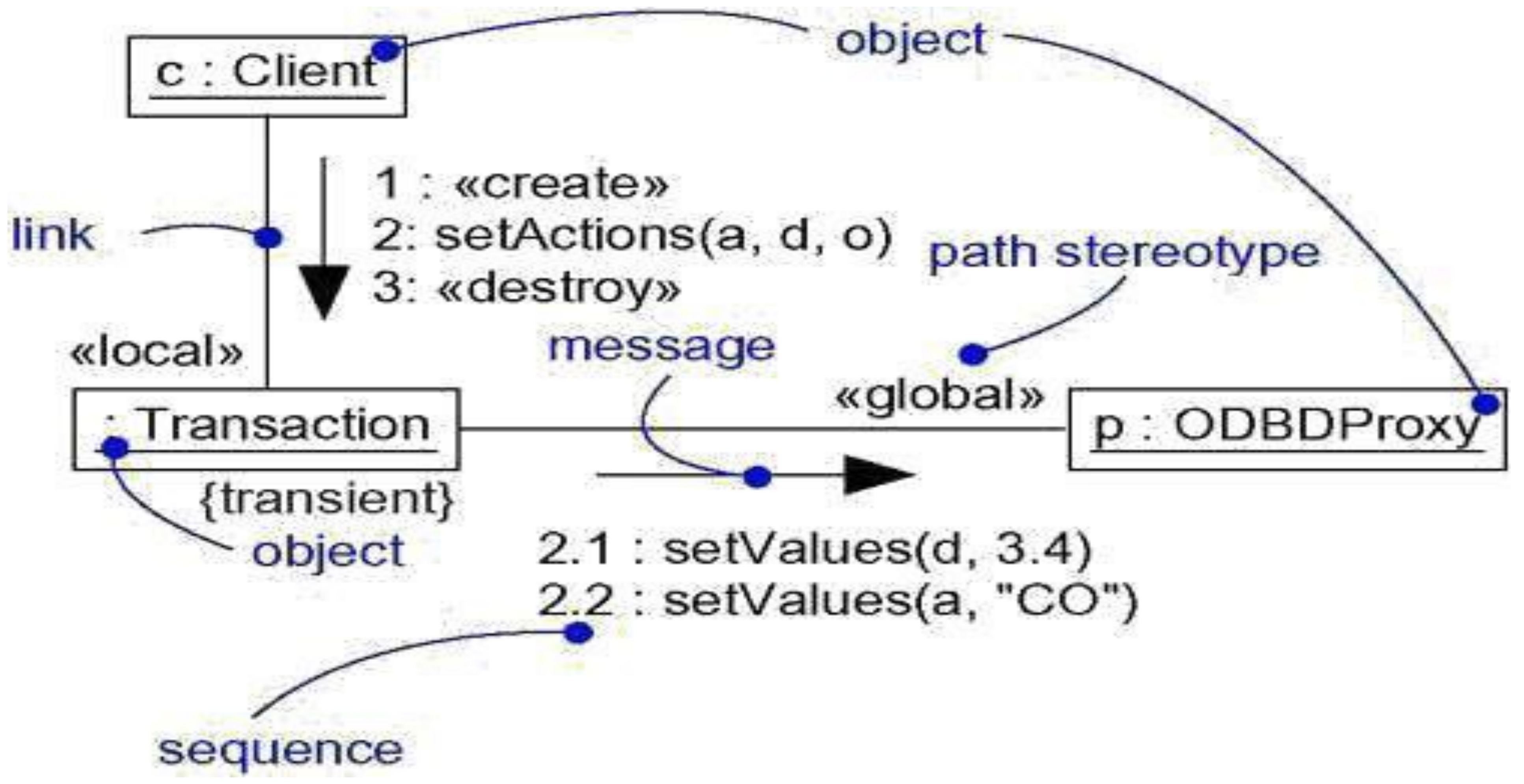
Class and Object Diagrams

Collaboration Diagrams

- A collaboration diagram emphasizes the organization of the objects that participate in an interaction. As Figure shows, 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. 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.

Class and Object Diagrams

Figure Collaboration Diagram



Class and Object Diagrams

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

- First, there is the path. To indicate how one object is linked to another, you can attach a path stereotype to the far end of a link (such as **»local**, indicating that the designated object is local to the sender). Typically, you will only need to render the path of the link explicitly for **local**, **parameter**, **global**, and **self** (but not **association**) paths.
- 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; **1.1** is the first message nested in message **1**; **1.2** is the second message nested in message **1** ; 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.

Class and Object Diagrams

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).

Class and Object Diagrams

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

1. To model flows of control by time ordering

Here we 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.

2. To model flows of control by organization

Here we use collaboration diagrams. Modeling a flow of control by organization emphasizes the structural relationships among the states in the interaction, along which messages may be passed.

Class and Object Diagrams

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,

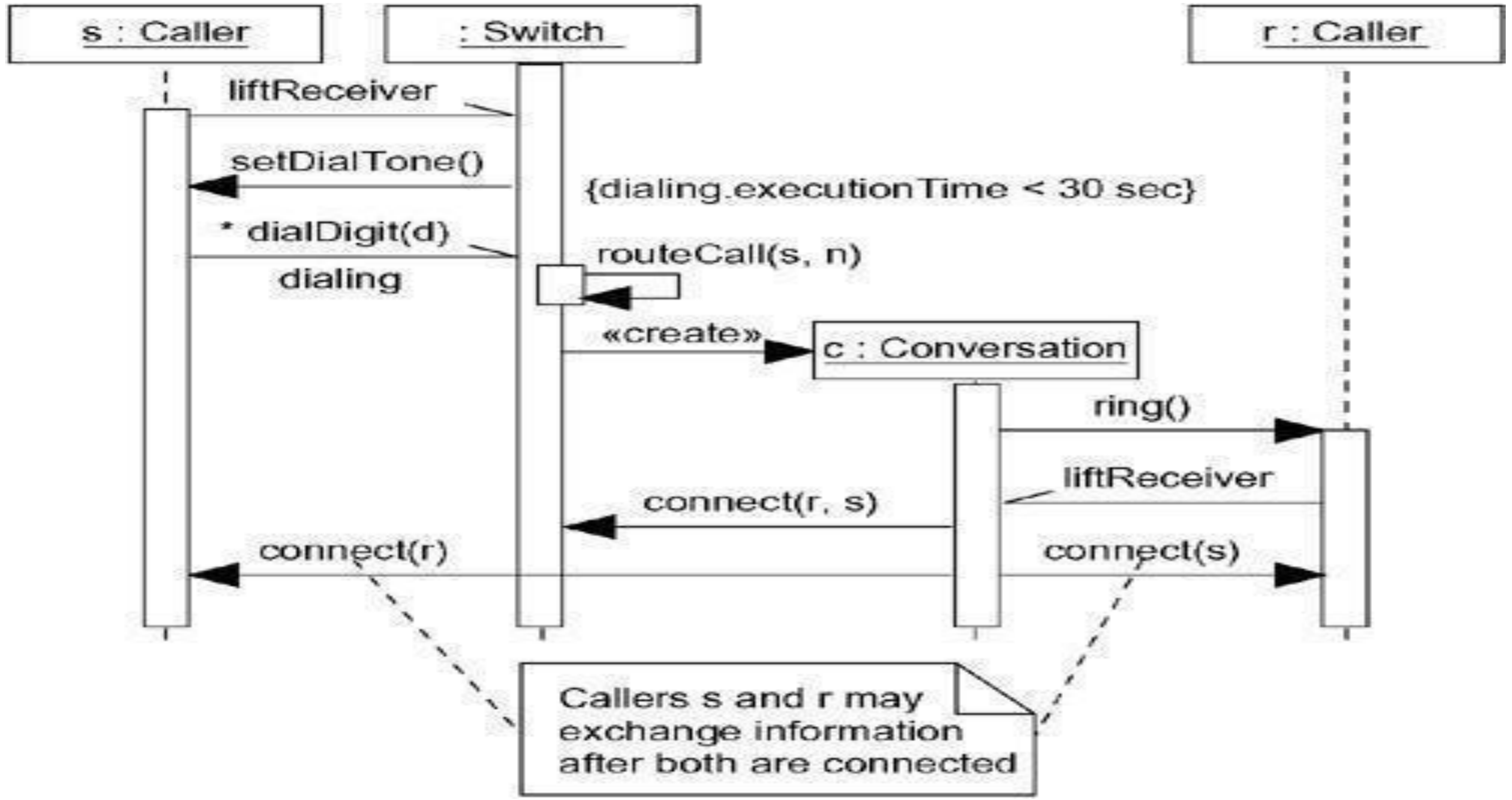
- Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.
- Set the stage for the interaction by identifying which objects play a role in the interaction. Lay them out on the sequence diagram from left to right, placing the more important objects to the left and their neighboring objects to the right.

Class and Object Diagrams

- 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 post-conditions to each message.

Class and Object Diagrams

Figure: Modeling Flows of Control by Time Ordering



Class and Object Diagrams

- Above Figure shows a sequence diagram that specifies the flow of control initiating a simple, two-party phone call.
- At this level of abstraction, there are four objects involved: two **Callers** (**s** and **r**), an unnamed telephone **Switch**, and **c**, the verification of the **Conversation** between the two parties.
- The sequence begins with one **Caller (s)** dispatching a signal (**liftReceiver**) to the Switch object.
- In turn, the **Switch** calls **setDialTone** on the **Caller**, and the **Caller** iterates on the message **dialDigit**.
- Note that this message has a timing mark (**dialing**) that is used in a timing constraint (its **executionTime** must be less than 30 seconds).
- This diagram does not indicate what happens if this time constraint is violated. For that you could include a branch or a completely separate sequence diagram.

Class and Object Diagrams

- The **Switch** object then calls itself with the message **routeCall**.
- It then creates a **Conversation** object **(c)**, to which it delegates the rest of the work.
- Although not shown in this interaction, **c** would have the additional responsibility of being a party in the switch's billing mechanism (which would be expressed in another interaction diagram).
- The **Conversation** object (**c**) rings the **Caller (r)**, who asynchronously sends the message **liftReceiver**.
- The **Conversation** object then tells the **Switch** to **connect** the call, then tells both **Caller** objects to **connect**, after which they may exchange information, as indicated by the attached note.

Class and Object Diagrams

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 collaboration diagram, a kind of interaction diagram.

To 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 collaboration diagram as vertices in a graph, placing the more important objects in the center of the diagram and their neighboring objects to the outside.

Class and Object Diagrams

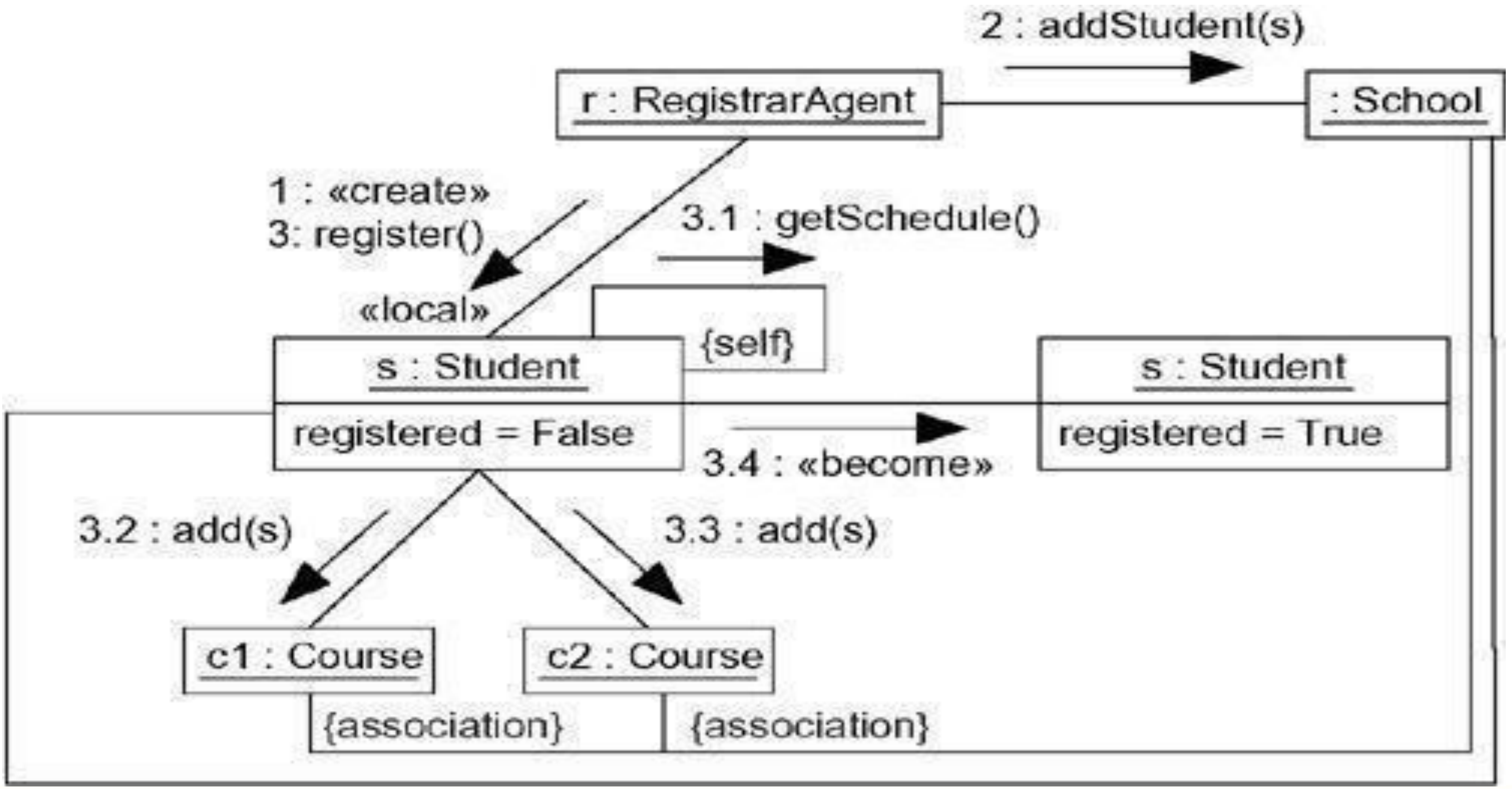
- Set the initial properties of each of these objects. If the attribute values, tagged values, state, or role of any object changes in significant ways over the duration of the interaction, place a duplicate object on the diagram, update it with these new values, and connect them by a message stereotyped as become or copy (with a suitable sequence number).
- 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 stereotypes (such as **global** and **local**) to explicitly specify how these objects are related to one another.

Class and Object Diagrams

- 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 post-conditions to each message.

Class and Object Diagrams

Figure Modeling Flows of Control by Organization



Class and Object Diagrams

- For example, above Figure shows a collaboration diagram that specifies the flow of control involved in registering a new student at a school, with an emphasis on the structural relationships among these objects.
- You see five objects: a **RegistrarAgent (r)**, a **Student (s)**, **two Course objects (c1** and **c2)**, and an unnamed **School** object.
- The flow of control is numbered explicitly. Action begins with the **RegistrarAgent** creating a **Student** object, adding the student to the school (the message **addStudent**), then telling the **Student** object to register itself.
- The **Student** object then invokes **getSchedule** on itself, presumably obtaining the **Course** objects for which it must register.
- The **Student** object then adds itself to each **Course** object.
- The flow ends with **s** rendered again, showing that it has an updated value for its **registered** attribute.

Class and Object Diagrams

Classes and Objects

- Object-oriented

Subject (Helvetica Bold- 24pt)

Self Assessment Question

1. Question (Times New Roman-20pt- Line spacing 1.5pt)

    a. Options

    b. Options

    c. Options

    d. Options

    Answer: Options

Subject (Helvetica Bold- 24pt)

Document Link

Topic	URL	Notes
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT

Subject (Helvetica Bold- 24pt)

Video Link

Topic	URL	Notes
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT

Subject (Helvetica Bold- 24pt)

E- Book Link

Ebook name	Chapter	Page No.	Notes	URL
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT
Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT	Times New Roman-14PT