ICS 3105
OBJECT ORIENTED SOFTWARE ENGINEERING

Chapter 5.4

Interaction Diagrams

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - Draw communication digrams and sequence diagrams.
  - Differentiate communication diagrams from object diagrams.

Introduction

- Class diagrams are used to build a static model of objects in a software system.
- Modeling system dynamics focus on two aspects:
  - Interactions and
  - Behavior.

Introduction

- An interaction model shows a set of actors and objects interacting by exchanging messages.
- A behavior model shows how an object or system changes state in reaction to a series of events.

Introduction

- Two types of UML interaction diagrams used to model detailed scenarios of system execution are:
  - Sequence diagrams and
  - Communication diagrams.

Introduction

- State and activity diagrams, are two other UML diagram types that are used to model the possible behavior of a system.

Interaction diagrams

- Interaction diagrams are used to model the dynamic aspects of a software system – they help to visualize how the system runs.
- They show how a set of actors and objects communicate with each other to perform the steps of a use case, or of some other piece of functionality.

Interaction diagrams

- The set of steps, taken together, is called an interaction.
- Interaction diagrams can show several different types of communication.
- These include messages exchanged over a network, simple procedure calls, and commands issued by an actor through the user interface.

Interaction diagrams

- Collectively, these are referred to as messages
- The following elements can be found in an interaction diagram:
  - Instances of classes or actors.
  - Messages.

Interaction diagrams

- Instances of classes or actors
  - Instances of classes (i.e. objects) are shown as boxes with the class and object identifier underlined.
  - Actors are shown using the same stick-person symbol as in use case diagrams.

Interaction diagrams

- Messages:
  - These are shown as arrows from actor to object, or from object to object. One of the main objectives of drawing interaction diagrams is to better understand the sequence of messages.

Interaction diagrams

- Since you need to know the actors and objects involved in an interaction, you should normally develop a class diagram and a use case model before starting to create an interaction diagram.

Interaction diagrams

- Two kinds of diagrams are used to show interactions:
  - Sequence diagrams and
  - Communication diagrams.

Interaction diagrams

- Both sequence and communication diagrams contain similar information about an interaction, although sequence diagrams have notations that make them somewhat more powerful.

Interaction diagrams

- Sequence diagrams explicitly show the sequence of events on a time line, whereas communication diagrams are more compact.

Sequence diagrams

- A sequence diagram shows the sequence of messages exchanged by the set of objects (and optionally an actor) performing a certain task.
- The objects are arranged from left to right across the diagram – an actor that initiates the interaction is often shown on the left.
- The vertical dimension represents time.

Sequence diagrams

- The top of the diagram is the starting point, and time progresses downwards towards the bottom of the diagram.
- A vertical dashed line, called a lifeline, is attached to each object or actor.

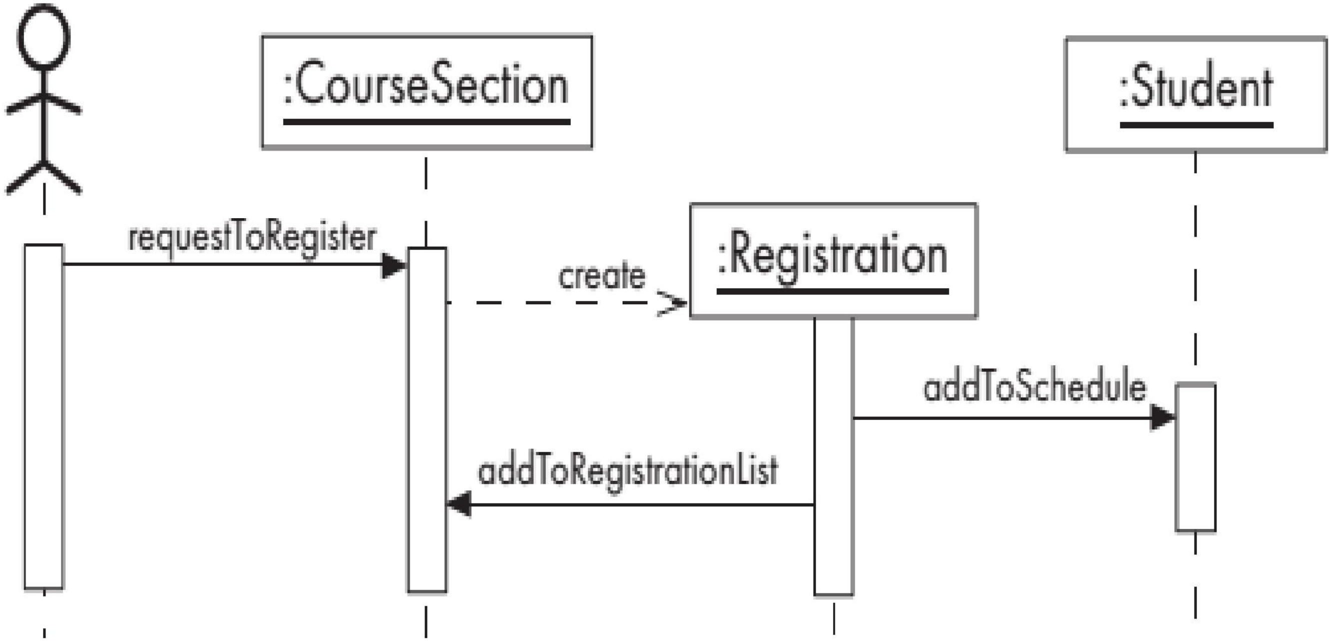
Sequence diagrams

- The lifeline becomes a box, called an activation box, during the period of time that the object is performing computations.
- The object is said to have live activation during these times.

Sequence diagrams

- A message is represented as an arrow between activation boxes of the sender and receiver.
- Each message is given a label; it can optionally have an argument list and a response.
- The complete message syntax is as follows: *response:=message(arg,...)*

Sequence diagrams of student registration process



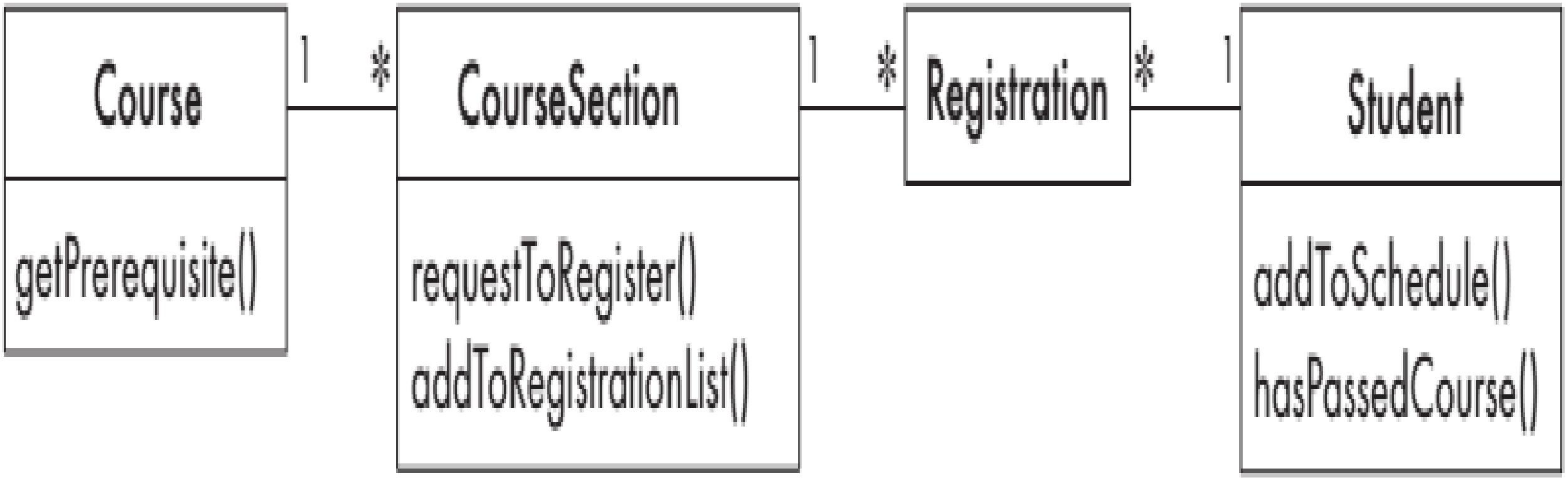
Sequence diagrams

- There are three objects and one actor involved in this interaction.
- A Student object and a CourseSection object exist initially; a Registration object is created as the interaction proceeds.

Sequence diagrams

- A creation message is shown using a dashed line with the label create.
- Note the different types of arrowheads used by the create message and the others

Corresponding Class diagram



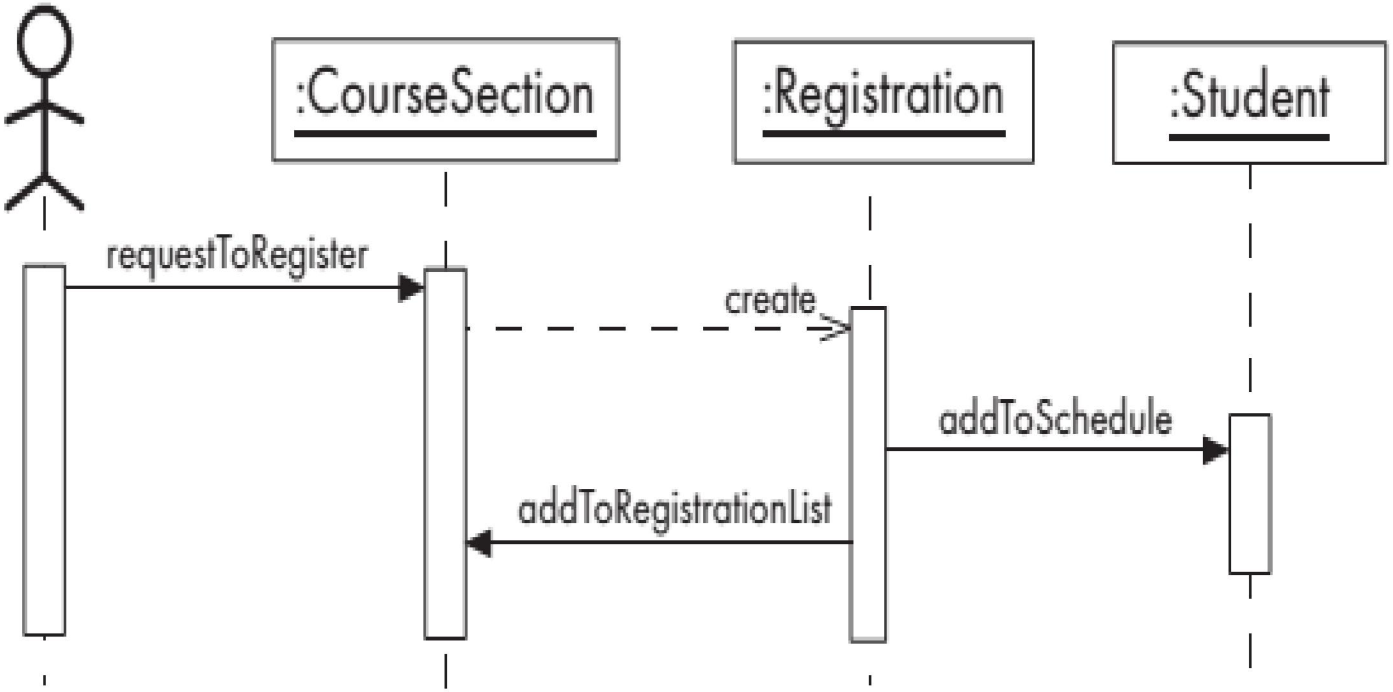
Sequence diagrams

- The objects that exist initially should be lined up along the top of the diagram.
- Since the Registration is created later, its box appears further down, at the time when it is created.

Sequence diagrams

- Unfortunately, many tools can only draw diagrams in which all the objects appear at the top.
- However, the create message still makes it clear when the object is created.

Similar Sequence diagram with all object as the top of the diagram



Sequence diagrams

- The actor initiates the interaction via the user interface; the user interface sends a requestToRegister message to the CourseSection, which in turn creates a Registration.

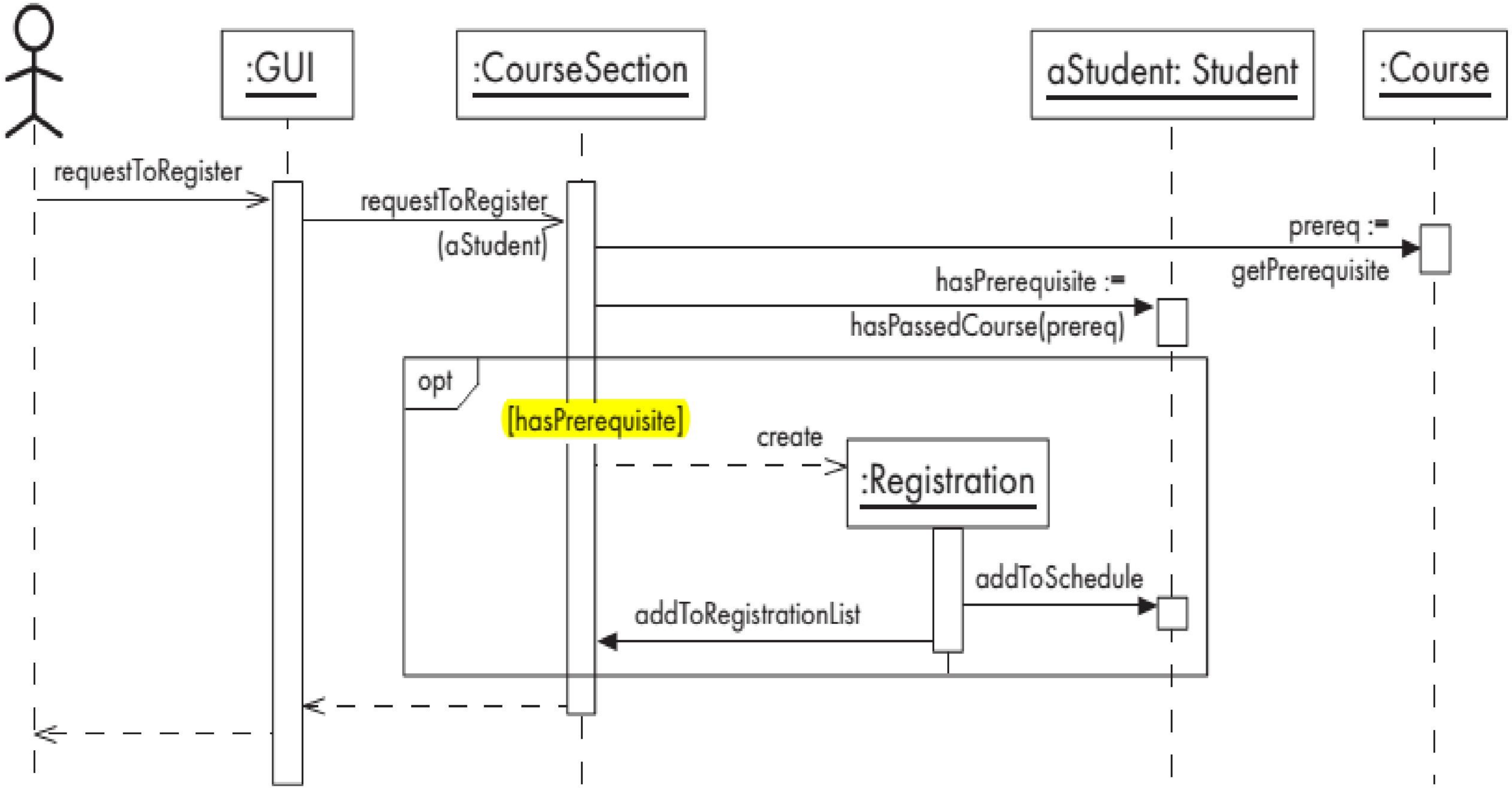
Sequence diagrams

- The Registration object then asks the Student to add it to the list of courses the student is taking, and also asks the CourseSection to add it to the list of registered students.
- The labels on the messages correspond to operations.

Sequence diagrams

- Often, when an actor interacts with a system, a corresponding object will exist that contains information about that actor.
- As with class diagrams, interaction diagrams can be drawn at various levels of detail.
- The level of detail you choose depends on what you wish to communicate.

More detailed sequence diagram



Sequence diagrams

- The first sequence diagram showed the user directly interacting with a CourseSection object.
- In reality, the user interacts with the user interface, which in turn interacts with the rest of the data in the system.

Detailed Sequence diagrams

- More detailed Sequence diagram gives the arguments and return values of certain messages.
- For example, the requestToRegister message has aStudent as an argument.

Detailed Sequence diagrams

- This same object is also the destination of two messages, therefore the second-to-right object has been labeled aStudent:Student to make this clear.

Detailed Sequence diagrams

- Use of a combined fragment marked ‘opt’.
- A combined fragment is a subsequence of an interaction that is special in some way, and is shown within a box.
- The ‘opt’ label means that it may or may not occur.

Detailed Sequence diagrams

- A Boolean condition, written within square brackets, describes the circumstances when it will occur.
- In this case, the condition is written over the :CourseSection lifeline, and indicates that the subsequence in the combined fragment will only occur if the hasPrerequisitevariable (the return value of the previous message) is true.

Detailed Sequence diagrams

- Sometimes a message is sent, but the reply to that message is sent back after considerable delay.
- A dashed line from the CourseSection to the GUI indicates when the reply to the original requestToRegister message is sent.

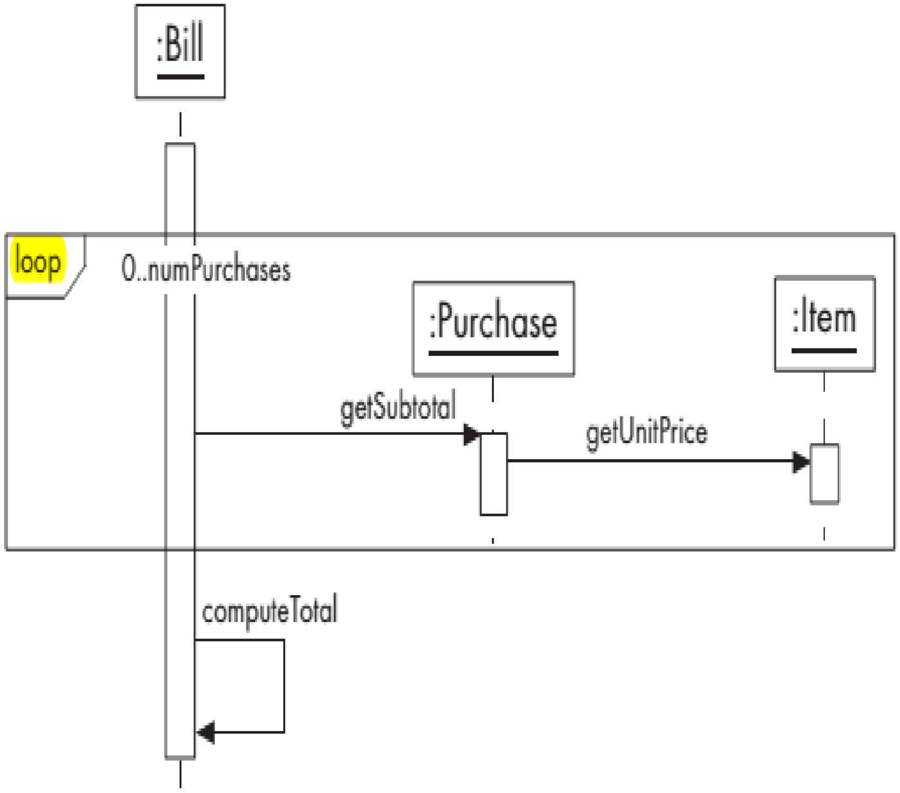
Detailed Sequence diagrams

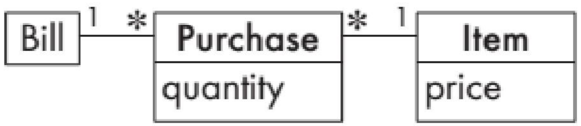
- In some cases, a sequence of messages must be repeated – in other words iteration must occur.
- You show iteration using a combined fragment marked ‘loop’,

Detailed Sequence diagrams

- The number of times to loop is specified using the syntax min..max
- The getSubtotal message will be sent to numPurchase different Purchase objects.

A sequence diagram showing a loop fragment

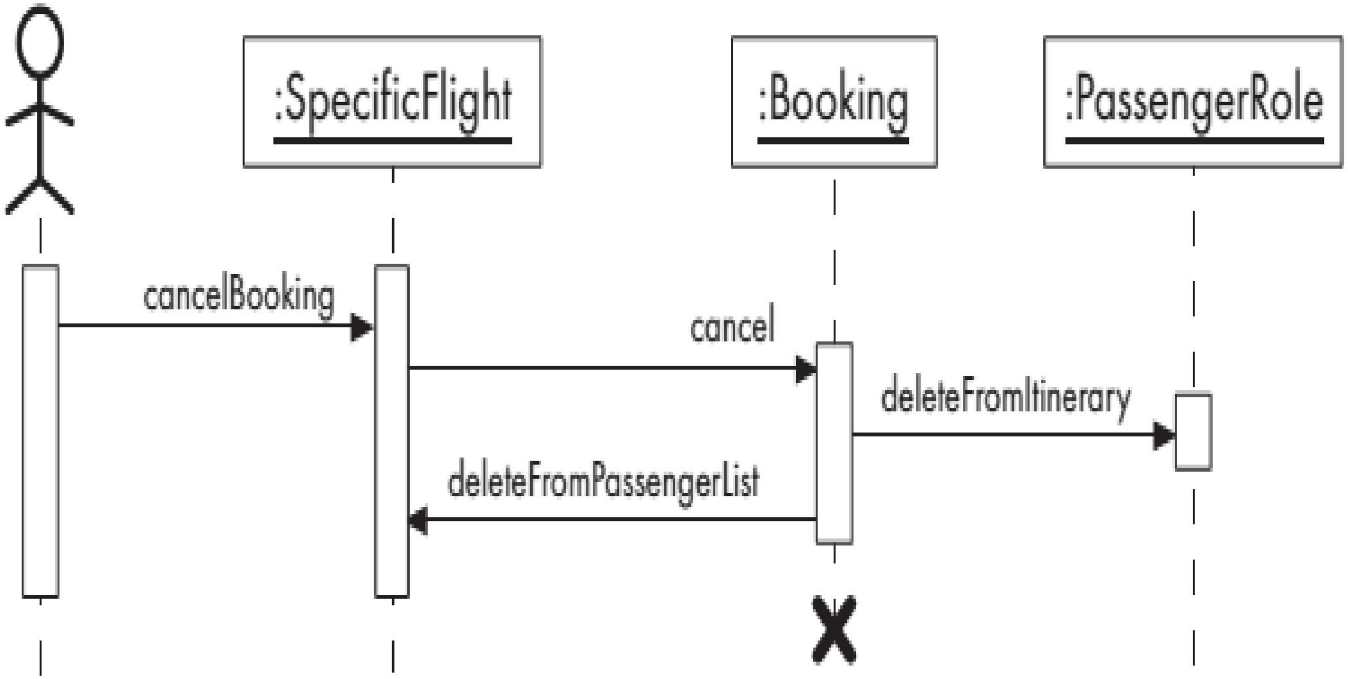




Deleting an object.

- A sequence diagram can show the destruction of an object using a big X symbol on a lifeline.

booking in the airline system is canceled



COMMUNICATION DIAGRAMS

Communication diagrams

- A communication diagram shows several objects working together.
- It appears as a graph with a set of objects and actors as the vertices.

Communication diagrams

- A communication diagram is very much like an object diagram except that, it shows communication links instead of links of associations.
- It also has much in common with a sequence diagram, except that lifelines, activation boxes and combined fragments are absent.

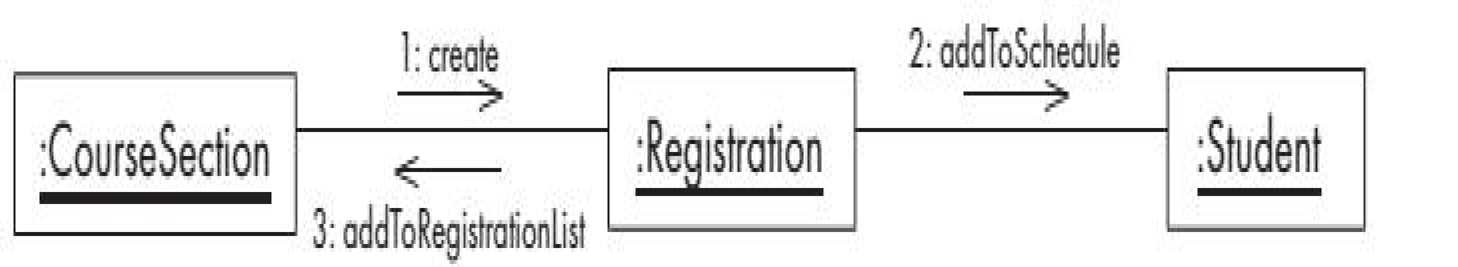
Communication diagrams

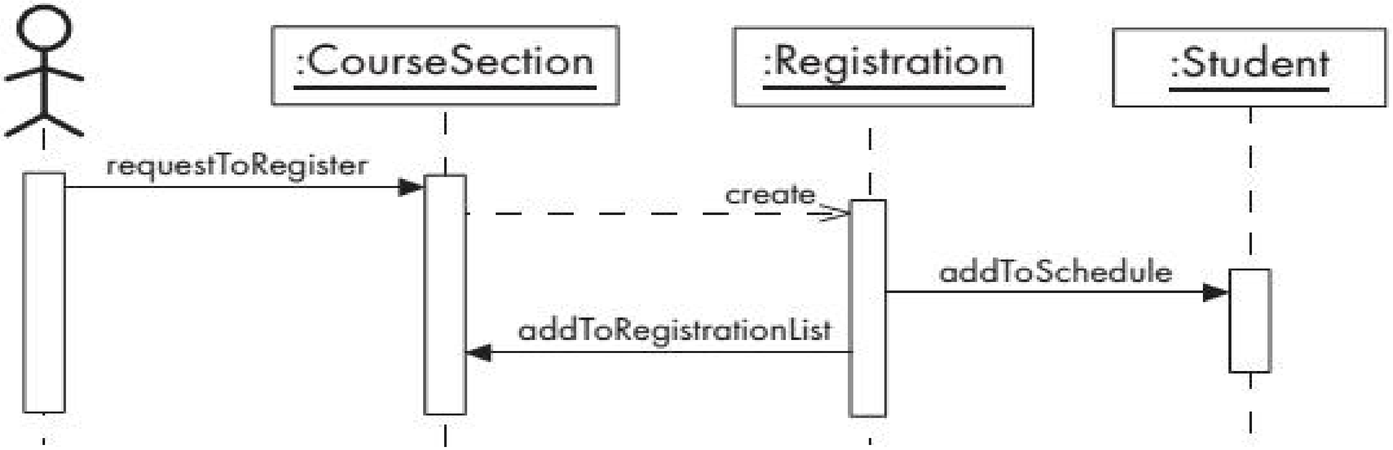
- A communication link is drawn between each pair of objects involved in the sending of a message; the messages themselves are attached to this link.
- A message using is represented by an arrow, labeled with the message name and optional arguments.

Communication diagrams

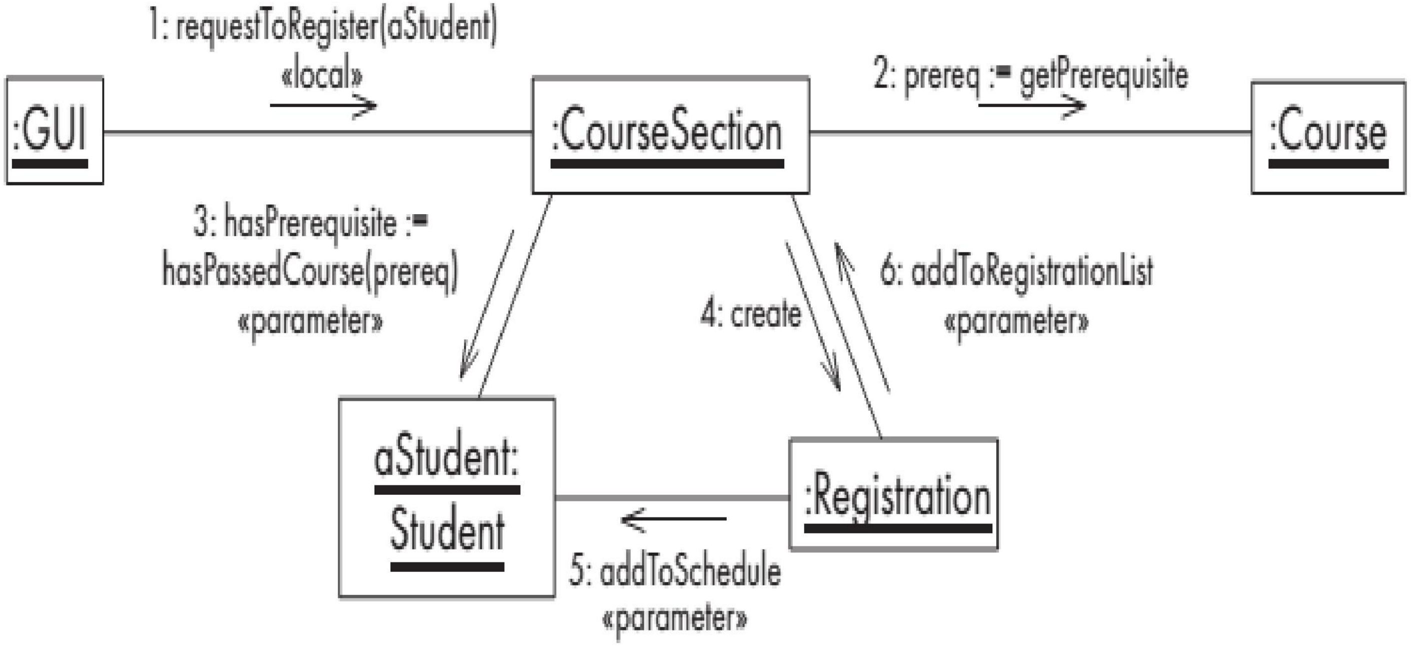
- The order in which messages are sent are specified by prefixing each message using some numbering scheme.

Communication diagram

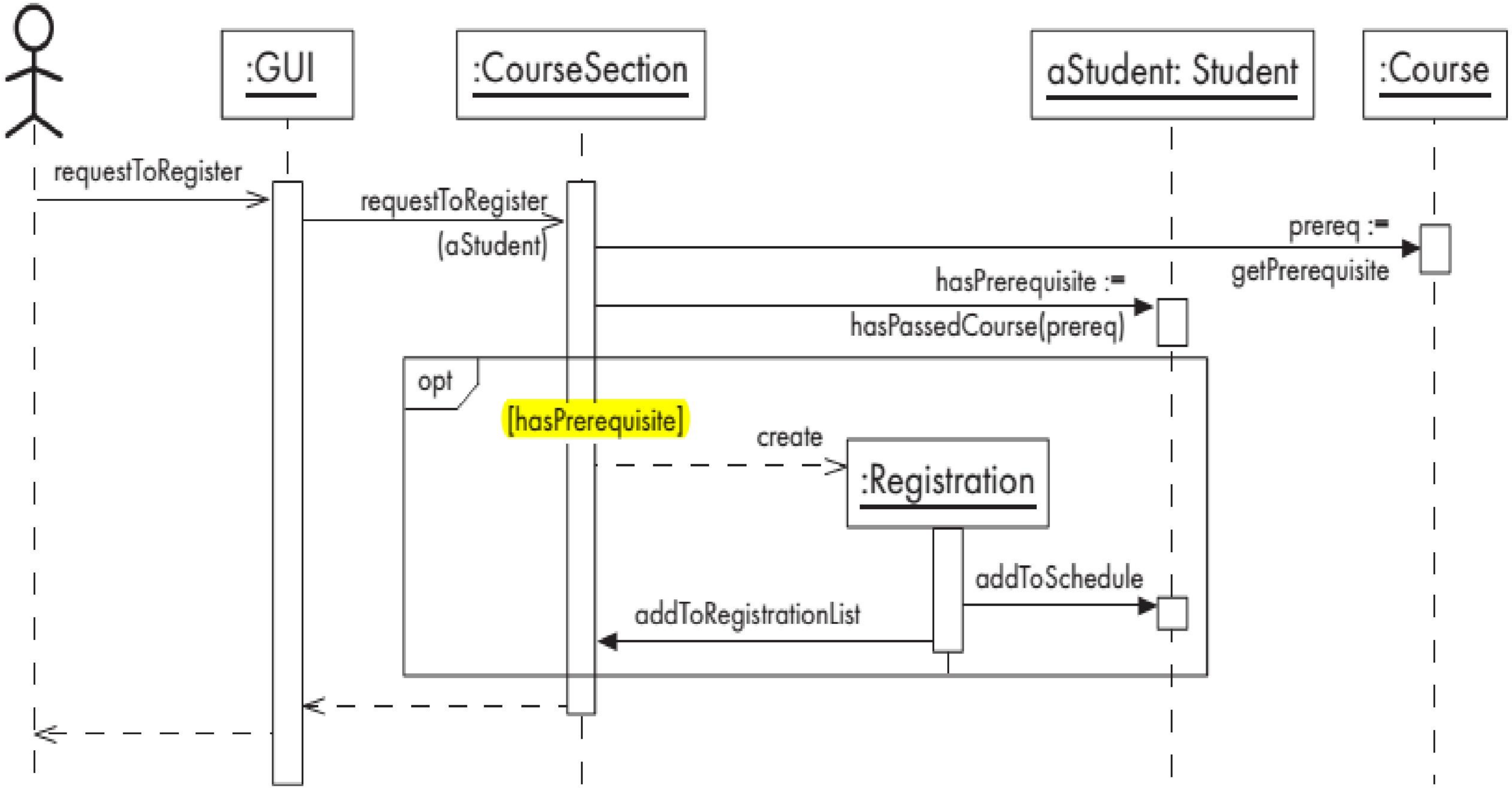




Communication diagram



Corresponding sequence diagram



Communication links between objects

- Communication links can exist between two objects whenever it is possible for one object to send a message to the other one.
- Several situations can make this possible.

Several situations can make link possible

- The classes of the two objects are joined by an association.
- The receiving object is stored in a local variable of the sending method (but the objects are not yet joined by an association).

Several situations can make link possible

- A reference to the receiving object has been received as a parameter of an earlier message to the sender.
- The receiving object is global.
- The objects communicate over a network

Choosing between a sequence or a communication diagram

- Since sequence and communication diagrams contain much the same information, software engineer may have to decide which of the two he/she should draw.

Preferred situations for Sequence diagrams

- Software engineer want the reader to be able to easily see the order in which messages occur.
- Software engineer want to build an interaction model from a use case. Use cases already have a sequence of steps; sequence diagrams expand on these to show which objects are involved.

Preferred situations for Sequence diagrams

- Software engineer need to show details of messages, such as parameters and return values. Doing so on communication diagrams can result in clutter.
- Software engineer need to show loops, optional sequences and other things that can only be properly expressed using combined fragments.

Preferring communication diagrams

- On the other hand, software engineers may prefer a communication diagram when they are deriving an interaction diagram from a class diagram.
- This is because communication diagrams are effectively object diagrams with communication links instead of association links.

Preferring communication diagrams

- Communication diagrams can in fact be used to help validate class diagrams – a communication diagram might suggest, for example, that you should add a new association in order to make the interaction possible.

End of chapter 5.4