

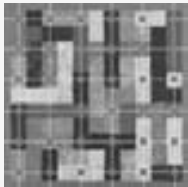
# CAPÍTULO 10

## Análisis y diseño de sistemas orientados a objetos mediante el uso de UML<sup>1</sup>

### OBJETIVOS DE APRENDIZAJE

Al completar este capítulo usted podrá:

1. Comprender qué es el análisis y diseño de sistemas orientados a objetos, y apreciar su utilidad.
2. Comprender los conceptos del lenguaje unificado de modelado (UML), la metodología estándar para modelar un sistema en el mundo orientado a objetos.
3. Aplicar los pasos utilizados en el UML para descomponer el sistema en un modelo de casos de uso y después en un modelo de clases.
4. Hacer diagramas de los sistemas con el conjunto de herramientas de UML, para poder describirlos y diseñarlos en forma apropiada.
5. Documentar y comunicar el sistema orientado a objetos recién modelado para los usuarios y demás analistas.



El análisis y diseño orientados a objetos pueden ofrecer una metodología que facilita los métodos lógicos, rápidos y detallados para crear sistemas que respondan a un panorama de negocios en evolución. Las técnicas orientadas a objetos funcionan bien en situaciones en las que los sistemas de información complicados pasan a través de un proceso continuo de mantenimiento,

adaptación y rediseño.

En este capítulo presentaremos el lenguaje unificado de modelado (UML), el estándar en la industria para modelar sistemas orientados a objetos. El conjunto de herramientas de UML incluye diagramas que permiten visualizar la construcción de un sistema orientado a objetos. Cada iteración aborda de manera cada vez más detallada el diseño del sistema, hasta que las cosas y las relaciones en el sistema estén definidas con claridad y precisión en documentos de UML. El UML es una potente herramienta que puede mejorar en forma considerable la calidad de nuestro análisis y diseño de sistemas, y en consecuencia puede ayudarnos a crear sistemas de información de mayor calidad.

Cuando se introdujo la metodología orientada a objetos, sus defensores citaban la reutilización de los objetos como su principal beneficio. Es lógico pensar que el reciclaje de las partes del programa reduce los costos del desarrollo en los sistemas basados en computadoras, y esto ha resultado un procedimiento muy efectivo en el desarrollo de GUI y bases de datos. Aunque la reutilización es el principal objetivo, también es muy importante el mantenimiento de los sistemas y, como la metodología orientada a objetos crea objetos que contienen tanto datos como código de programa, si se modifica un objeto habrá un mínimo impacto en los demás objetos.

<sup>1</sup> Por Julie E. Kendall, Kenneth E. Kendall y Allen Schmidt.

## CONCEPTOS ORIENTADOS A OBJETOS

La programación orientada a objetos difiere de la programación tradicional por procedimientos en cuanto a que examina los objetos que forman parte de un sistema. Cada objeto es una representación de alguna cosa o evento real. En esta sección presentaremos las descripciones generales de los conceptos orientados a objetos clave: objetos, clases y herencia. Más adelante en el capítulo presentaremos más detalles sobre otros conceptos de UML.

### Objetos

Los objetos son personas, lugares o cosas relevantes para el sistema a analizar. Los sistemas orientados a objetos describen las entidades como objetos. Algunos objetos comunes son clientes, artículos, pedidos, etcétera. Los objetos también pueden ser pantallas de GUI o áreas de texto en la pantalla.

### Clases

Por lo general, los objetos forman parte de un grupo de elementos similares, conocidos como clases. La intención de colocar elementos en clases no es nuevo. Describir el mundo como algo compuesto de animales, vegetales y minerales es un ejemplo de clasificación. La metodología científica incluye clases de animales (como mamíferos) y después divide esas clases en subclases (como animales que ponen huevos, y mamíferos marsupiales).

La idea subyacente es tener un punto de referencia y describir un objeto específico en términos de sus similitudes o diferencias en relación con los miembros de su propia clase. Al hacer esto es más eficiente para alguien decir: “El oso koala es un marsupial (o animal con bolsa) con una cabeza grande y redonda, y con oídos peludos” que tratar de describir un oso koala a través de todas sus características como mamífero. Es más eficiente describir las características, apariencia e incluso el comportamiento de esta manera. La palabra *reutilizable*, en el mundo orientado a objetos, significa que se puede ser más eficiente gracias a que no hay necesidad de comenzar desde el principio para describir cada objeto cada vez que se requiera en el desarrollo de software.

Los objetos se representan y agrupan mediante clases, las cuales son óptimas para la reutilización y la facilidad de mantenimiento. Una clase define el conjunto de atributos compartidos y comportamientos que se encuentran en cada objeto de la clase. Por ejemplo, los registros para los estudiantes en la sección de un curso tienen información similar almacenada para cada estudiante. Se dice que los estudiantes conforman una clase. Los valores pueden ser distintos para cada estudiante, pero el tipo de información es el mismo. Los programadores deben definir las diversas clases en el programa que escriben. Al ejecutarse el programa se pueden crear objetos a partir de la clase establecida. El término *instanciar* se utiliza cuando se crea un objeto a partir de una clase. Por ejemplo, un programa podría instanciar un estudiante llamado Peter Wellington como un objeto de la clase etiquetada como estudiante.

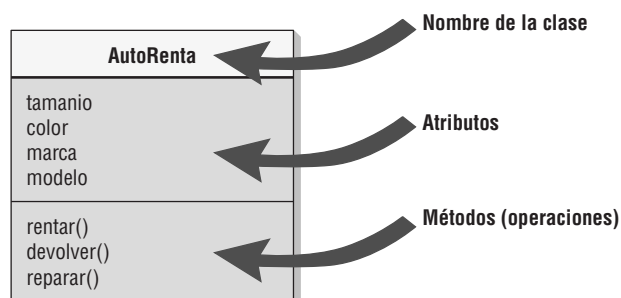
Lo que distingue a la programación orientada a objetos (y por ende al análisis y diseño orientado a objetos) de la programación clásica es la técnica de colocar todos los atributos y métodos de un objeto dentro de una estructura autocontenida, la clase en sí. Éste es un acontecimiento familiar en el mundo físico. Por ejemplo, la mezcla de un pastel en caja es análoga a una clase, ya que tiene los ingredientes así como las instrucciones sobre cómo mezclar y hornear el pastel. Un suéter de lana es similar a una clase, ya que tiene una etiqueta cosida con instrucciones sobre su cuidado, que le advierten lavarlo a mano y dejarlo secar extendido.

Cada clase debe tener un nombre que la distinga de las demás. Por lo general, los nombres de las clases son sustantivos o frases cortas y empiezan con mayúscula. En la figura 10.1 la clase se llama **AutoRenta**. En UML, una clase se dibuja como un rectángulo. Este rectángulo contiene otras dos características importantes: una lista de atributos y una serie de métodos. Estos elementos describen una clase, la unidad de análisis que forma una gran parte de lo que llamamos análisis y diseño orientado a objetos.

Un atributo describe cierta propiedad que poseen todos los objetos de la clase. Cabe mencionar que la clase **AutoRenta** posee los atributos de tamaño, color, marca y modelo. Todos los automóviles poseen estos atributos,

**FIGURA 10.1**

Un ejemplo de una clase de UML, la cual se describe como un rectángulo que consiste en el nombre de la clase, sus atributos y métodos.



pero cada automóvil tendrá distintos valores para sus atributos. Por ejemplo, un automóvil puede ser azul, blanco o de algún otro color. Más adelante demostraremos que puede ser más específico sobre el rango de valores para estas propiedades. Al especificar atributos, por lo general la primera letra va en minúscula.

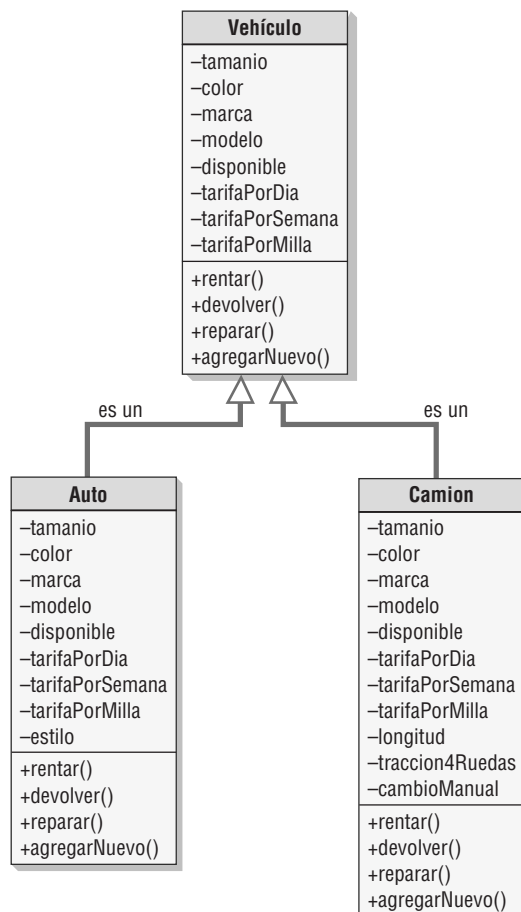
Un método es una acción que se puede solicitar de cualquier objeto de la clase. Los métodos son los procesos que una clase sabe cómo llevar a cabo. También se les conoce como operaciones. Para la clase **AutoRenta**, **rentar()**, **devolver()** y **reparar()** son ejemplos de métodos. Al especificar métodos, por lo general la primera letra está en minúscula.

## Herencia

Otro concepto clave de los sistemas orientados a objetos es la herencia. Las clases pueden tener hijos; es decir, se puede crear una clase a partir de otra. En UML, la clase original (o padre) se conoce como clase base; a la clase hija se le denomina clase derivada. Podemos crear una clase derivada de tal forma que herede todos los atributos y comportamientos de la clase base. Sin embargo, una clase derivada puede tener atributos y comportamientos adicionales. Por ejemplo, podría haber una clase **Vehículo** para una empresa de renta de automóviles que contenga atributos tales como **tamaño**, **color** y **marca**.

La herencia reduce la labor de programación al permitir que se utilicen los objetos comunes con facilidad. El programador sólo necesita declarar que la clase **Auto** hereda de la clase **Vehículo** y después proporcionar todos los detalles adicionales sobre los nuevos atributos o comportamientos que sean únicos para un automóvil. Todos los atributos y comportamientos de la clase **Vehículo** pasan de manera automática e implícita a formar parte de la clase **Auto** y no requieren de programación adicional. Esto permite al analista definir una vez pero usar muchas veces; algo similar a los datos que están en la tercera forma normal, que se definen sólo una vez en una tabla de la base de datos (como veremos en el capítulo 13).

Las clases derivadas que se muestran en la figura 10.2 son **Auto** o **Camión**. Aquí se antepone un signo negativo a los atributos y un signo positivo a los métodos. Más adelante en el capítulo hablaremos sobre esto con más detalle; por ahora tenga en cuenta que los signos negativos significan que estos atributos son privados (no se comparten con otras clases) y que los signos positivos significan que estos métodos son públicos (otras clases pueden invocarlos).



**FIGURA 10.2**

Un diagrama de clases que muestra la herencia. Auto y Camion son ejemplos específicos de vehículos y heredan las características de la clase más general, **Vehículo**.



## OPORTUNIDAD DE CONSULTORÍA 10.1

### Alrededor del mundo en 80 objetos

Como usted describió las ventajas de usar metodologías orientadas a objetos, Jules y Vern, dos ejecutivos de nivel superior en World's Trend desean que analice su empresa mediante el uso de esta metodología. En la figura 7.15 encontrará un resumen de las actividades de negocios de World's Trend. Observe también la serie de diagramas de flujo de datos en ese capítulo para que pueda conceptualizar el problema y empiece a realizar la transición al Pensamiento en objetos.

Como usted es buen amigo de Jules y Vern, y no tiene inconveniente en obtener un poco de experiencia práctica en cuanto al uso del pensamiento orientado a objetos, accede a aplicar lo que sabe para después enviarles un informe.

Una vez que haya vuelto a leer las actividades de negocios para World's Trend, provea una revisión oportuna completando las siguientes tareas:

Use la técnica de tarjetas CRC para listar las clases, responsabilidades y colaboradores.

Use la técnica de Pensamiento en objetos para listar lo que se conoce y los atributos correspondientes para los objetos en las clases identificadas en la etapa anterior.

Haga un informe usando ambos pasos y vaya a las oficinas generales de World's Trend con su informe en la mano. Sin duda Jules y Vern esperan un viaje fantástico en el nuevo mundo de los métodos orientados a objetos.

La reutilización de código de programa ha formado parte del desarrollo de sistemas estructurados y los lenguajes de programación (como COBOL) durante muchos años; además han existido subprogramas que encapsulan datos. Sin embargo, la herencia es una herramienta que se encuentra sólo en los sistemas orientados a objetos.

## TARJETAS CRC Y PENSAMIENTO EN OBJETOS

Ahora que hemos presentado los conceptos fundamentales del análisis y diseño orientados a objetos, necesitamos examinar formas de crear clases y objetos a partir de los problemas de negocios y los sistemas a los que nos enfrentamos. Una forma de empezar a poner en práctica la metodología orientada a objetos es empezar a pensar y hablar de esta manera. Un método útil es desarrollar tarjetas CRC.

CRC representa a las clases, responsabilidades y colaboradores. El analista puede usar estos conceptos cuando empieza a hablar sobre el sistema o a modelarlo a partir de una perspectiva orientada a objetos. Las tarjetas CRC se utilizan para representar las responsabilidades de las clases y las interacciones entre ellas. Los analistas crean las tarjetas con base en escenarios que describen los requerimientos del sistema. Estos escenarios modelan el comportamiento del sistema que se está estudiando. Si se van a usar en un grupo, las tarjetas CRC se pueden elaborar en forma manual en pequeñas tarjetas de cartulina por cuestión de flexibilidad, o se pueden crear mediante el uso de una computadora.

Hemos agregado dos columnas a la plantilla de tarjeta CRC original: la columna Pensamiento en objetos y la columna de propiedades. Los enunciados de Pensamiento en objetos están escritos en español común; el nombre de la propiedad o atributo está escrito en su lugar apropiado. El propósito de estas columnas es aclarar el pensamiento y ayudar a pasar a la creación de diagramas de UML.

### Interacción durante una sesión CRC

Las tarjetas CRC se pueden crear de manera interactiva con unos cuantos analistas que puedan trabajar en conjunto para identificar la clase en el dominio del problema presentado por la empresa. Una sugerencia es buscar todos los sustantivos y verbos en un enunciado del problema que se haya creado para capturarlo. Por lo general, los sustantivos indican las clases en el sistema; para encontrar las responsabilidades hay que identificar los verbos.

Haga una sesión de lluvia de ideas con su grupo de analistas para identificar todas las clases que pueda. Siga el formato estándar para la lluvia de ideas, que consiste en no criticar la respuesta de ninguno de los participantes en este punto, sino solicitar todas las respuestas posibles. Una vez identificadas todas las clases, los analistas pueden empezar a compilarlas, eliminar las ilógicas y escribir cada una en su tarjeta. Asigne una clase a cada persona en el grupo, que será su "propietaria" durante la sesión CRC.

A continuación, el grupo crea escenarios que en realidad son recorridos de las funciones del sistema, para lo cual se toma la funcionalidad deseada del documento de requerimientos creado con anterioridad. Primero hay que considerar los métodos de sistemas típicos, con excepciones tales como la recuperación de errores que se lleva a cabo después de haber cubierto los métodos de rutina.

Mientras el grupo decide sobre la clase que será responsable de una función específica, el analista propietario de la misma durante la sesión elige esa tarjeta y declara: "Necesito cumplir con mi responsabilidad". Cuando

| <b>Nombre de la clase:</b> Departamento |               |                                   |                         |
|---|---------------|-----------------------------------|-------------------------|
| <b>Superclases:</b>                     |               |                                   |                         |
| <b>Subclases:</b>                       |               |                                   |                         |
| Responsabilidades                       | Colaboradores | Pensamiento en objetos            | Propiedad               |
| Agregar un nuevo departamento           | Curso         | Conozco mi nombre                 | Nombre del departamento |
| Proveer información del departamento    |               | Conozco mi silla del departamento | Nombre de la silla      |
|   |               |                                   |                         |
|   |               |                                   |                         |

| <b>Nombre de la clase:</b> Curso |                |                               |                       |
|----------------------------------|----------------|-------------------------------|-----------------------|
| <b>Superclases:</b>              |                |                               |                       |
| <b>Subclases:</b>                |                |                               |                       |
| Responsabilidades                | Colaboradores  | Pensamiento en objetos        | Propiedad             |
| Agregar un nuevo curso           | Departamento   | Conozco el número de mi curso | Número de curso       |
| Modificar información del curso  | Libro de texto | Conozco mi descripción        | Descripción del curso |
| Mostrar información del curso    | Asignación     | Conozco mi número de créditos | Créditos              |
|                                  | Examen         |                               |                       |
|                                  |                |                               |                       |

| <b>Nombre de la clase:</b> Libro de texto |               |                        |             |
|---|---------------|------------------------|-------------|
| <b>Superclases:</b>                       |               |                        |             |
| <b>Subclases:</b>                         |               |                        |             |
| Responsabilidades                         | Colaboradores | Pensamiento en objetos | Propiedad   |
| Agregar un nuevo libro de texto           | Curso         | Conozco mi ISBN        | ISBN        |
| Cambiar la información del libro de texto |               | Conozco mi autor       | Autor       |
| Buscar información del libro de texto     |               | Conozco mi título      | Título      |
| Eliminar libros de texto obsoletos        |               | Conozco mi edición     | Edición     |
|   |               | Conozco mi editorial   | Editorial   |
|   |               | Sé si soy obligatorio  | Obligatorio |
|   |               |                        |             |

| <b>Nombre de la clase:</b> Asignación |               |                                 |                      |
|---------------------------------------|---------------|---------------------------------|----------------------|
| <b>Superclases:</b>                   |               |                                 |                      |
| <b>Subclases:</b>                     |               |                                 |                      |
| Responsabilidades                     | Colaboradores | Pensamiento en objetos          | Propiedad            |
| Agregar una nueva asignación          | Curso         | Conozco mi número de asignación | Número de tarea      |
| Cambiar una asignación                |               | Conozco mi descripción          | Descripción de tarea |
| Ver una asignación                    |               | Sé cuántos puntos valgo         | Puntos               |
|                                       |               | Sé cuando me deben entregar     | Fecha de entrega     |
|                                       |               |                                 |                      |
|                                       |               |                                 |                      |

FIGURA 10.3

Cuatro tarjetas CRC para los ofrecimientos de cursos, las cuales muestran cómo los analistas llenan los detalles para las clases, responsabilidades y colaboradores, así como para los enunciados de pensamiento en objetos y los nombres de las propiedades.

alguien sostiene una tarjeta en el aire, se considera un objeto y puede hacer cosas. Después el grupo procede a refinar la responsabilidad en tareas cada vez más pequeñas, de ser posible. Si es apropiado, el objeto puede cumplir con estas tareas o el grupo puede decidir que se cumplan mediante la interacción con otras cosas. Si no existen otras clases apropiadas, tal vez el grupo tenga que crear una.

Las cuatro tarjetas CRC descritas en la figura 10.3 muestran cuatro clases de ofrecimientos de cursos. Observe que en una clase llamada **Curso**, el analista de sistemas es referido a cuatro colaboradores: el departa-

mento, el libro de texto, la asignación del curso y el examen del curso. Después, estos colaboradores se describen como clases por sí solas en las otras tarjetas CRC.

Las responsabilidades que se enlistan evolucionarán en un momento dado en lo que se conoce como métodos en UML. Los enunciados de Pensamiento en objetos parecen rudimentarios, pero son conversacionales, de tal forma que alientan a los analistas durante una sesión CRC a describir tantos de estos enunciados como puedan. Como vimos en el ejemplo, todo el diálogo durante una sesión CRC se lleva a cabo en primera persona, de manera que hasta el **libro de texto** habla: “Conozco mi ISBN”, “Conozco mi autor”. Después, estos enunciados se pueden utilizar para describir atributos en UML. Podemos llamar a estos atributos según los nombres de sus variables, como **edicion** y **editorial**.

CONCEPTOS Y DIAGRAMAS DEL LENGUAJE UNIFICADO DE MODELADO (UML)

Es muy conveniente investigar y comprender la metodología del UML debido a su amplia aceptación y uso. UML provee un conjunto estandarizado de herramientas para documentar el análisis y diseño de un sistema de software. El conjunto de herramientas de UML incluye diagramas que permiten a las personas visualizar la construcción de un sistema orientado a objetos, algo similar a la forma en que los planos de construcción permiten a las personas visualizar la construcción de un edificio. Ya sea que usted trabaje de manera independiente o con un extenso equipo de desarrollo de sistemas, la documentación que puede crear con UML provee un medio efectivo de comunicación entre el equipo de desarrollo y el equipo de negocios en un proyecto.

El UML consiste en cosas, relaciones y diagramas, como se muestra en la figura 10.4. Los primeros componentes (o elementos primarios) de UML se llaman cosas. Tal vez usted prefiera otra denominación, como

**FIGURA 10.4**  
Una vista general de UML y sus componentes: cosas, relaciones y diagramas.

| Categoría de UML | Elementos de UML             | Detalles específicos de UML  |
|------------------|------------------------------|--|
| Cosas            | Cosas estructurales          | Clases<br>Interfaces<br>Colaboraciones<br>Casos de uso<br>Clases activas<br>Componentes<br>Nodos                                   |
|                  | Cosas de comportamiento      | Interacciones<br>Máquinas de estado  |
|                  | Cosas de agrupamiento        | Paquetes   |
|                  | Cosas de anotaciones         | Notas  |
| Relaciones       | Relaciones estructurales     | Dependencias<br>Agregaciones<br>Asociaciones<br>Generalizaciones   |
|                  | Relaciones de comportamiento | Comunica<br>Incluye<br>Extiende<br>Generaliza  |
| Diagramas        | Diagramas estructurales      | Diagramas de clases<br>Diagramas de componentes<br>Diagramas de despliegue   |
|                  | Diagramas de comportamiento  | Diagramas de casos de uso<br>Diagramas de secuencia<br>Diagramas de comunicación<br>Diagramas de estados<br>Diagramas de actividad |

objeto, pero en UML se les llama cosas. Las cosas estructurales son las más comunes. Las cosas estructurales son clases, interfaces, casos de uso y muchos otros elementos que proveen la forma de crear modelos. Las cosas estructurales permiten al usuario describir las relaciones. Las cosas de comportamiento describen la forma en que funcionan las cosas. Algunos ejemplos de cosas de comportamiento son las interacciones y las máquinas de estado. Las cosas de grupo se utilizan para definir límites. El paquete es un ejemplo de cosa de grupo. Por último tenemos las cosas de anotaciones, para poder agregar notas en los diagramas.

Las relaciones son el pegamento que mantiene las cosas unidas entre sí. Es conveniente pensar en la relaciones de dos formas. Las relaciones estructurales se utilizan para unir las cosas en los diagramas estructurales. Las relaciones estructurales incluyen dependencias, agregaciones, asociaciones y generalizaciones. Por ejemplo, las relaciones estructurales muestran herencia. Las relaciones de comportamiento se utilizan en los diagramas de comportamiento. Los cuatro tipos básicos de relaciones de comportamiento son comunica, incluye, extiende y generaliza.

Hay dos tipos principales de diagramas en UML: diagramas estructurales y diagramas de comportamiento. Los diagramas estructurales se utilizan, por ejemplo, para describir las relaciones entre las clases. Éstos se dividen en diagramas de clases, diagramas de objetos, diagramas de componentes y diagramas de despliegue. Por otro lado, los diagramas de comportamiento se pueden utilizar para describir la interacción entre las personas (actores en UML) y lo que denominamos caso de uso, o la forma en que los actores utilizan el sistema. Los diagramas de comportamiento se dividen en diagramas de casos de uso, diagramas de secuencia, diagramas de comunicación, diagramas de estados y diagramas de actividad.

En el resto del capítulo nos enfocaremos primero en el modelado de casos de uso, la base de todas las técnicas de UML. Después analizaremos la forma en que se utiliza un caso de uso para derivar las actividades, secuencias y clases: los diagramas de UML que se utilizan con más frecuencia. Como hay libros completos dedicados a la sintaxis y el uso del UML (el documento oficial de especificaciones de UML tiene más de 800 páginas), proveeremos sólo un breve resumen de los aspectos más valiosos y más utilizados del UML.

Los seis diagramas de UML que se utilizan con más frecuencia son:

1. Un diagrama de casos de uso, que describe la forma en que se utiliza el sistema.
2. Un escenario de caso de uso (aunque técnicamente no es un diagrama). Este escenario es una articulación verbal de excepciones para el comportamiento principal descrito por el caso de uso principal.
3. Un diagrama de actividad, que ilustra el flujo de actividades en general. Cada caso de uso puede crear un diagrama de actividad.
4. Los diagramas de secuencia, que muestran la secuencia de las actividades y las relaciones entre las clases. Cada caso de uso puede crear uno o más diagramas de secuencia. El diagrama de comunicación es la alternativa a un diagrama de secuencia, el cual contiene la misma información pero enfatiza la comunicación en vez de la sincronización.
5. Los diagramas de clases, que muestran las clases y sus relaciones. Los diagramas de secuencia se utilizan (junto con las tarjetas CRC) para determinar las clases. El diagrama de generalización/especialización (gen/spec) es un derivado del diagrama de clases.
6. Los diagramas de estados, que muestran las transiciones de estado. Cada clase puede crear un diagrama de estados, el cual es útil para determinar los métodos de la clase.

En la figura 10.5 se ilustra la forma en que se relacionan estos diagramas entre sí. En las siguientes secciones hablaremos sobre cada uno de estos diagramas.

## MODELADO DE CASOS DE USO

UML se basa fundamentalmente en una técnica de análisis orientado a objetos conocida como modelado de casos de uso, la cual se presentó en el capítulo 2. Un modelo de casos de uso muestra una vista del sistema desde la perspectiva del usuario, por lo cual describe *qué* hace el sistema sin describir *cómo* lo hace. Podemos utilizar UML para analizar el modelo de casos de uso y derivar los objetos del sistema junto con sus interacciones entre sí y con los usuarios del sistema. Al utilizar técnicas de UML podemos analizar con más detalle los objetos y sus interacciones para derivar su comportamiento, atributos y relaciones.

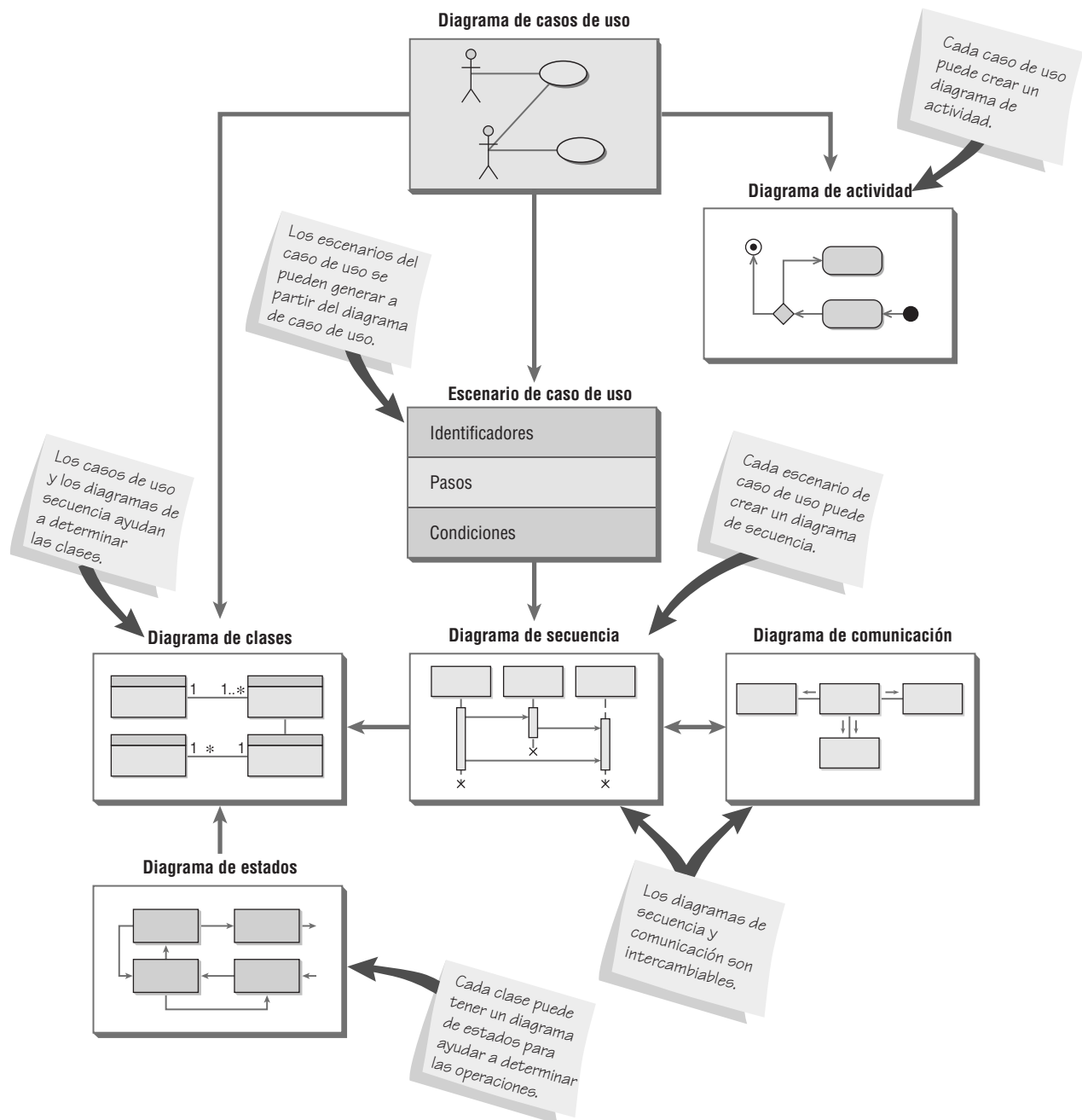
Un caso de uso provee a los desarrolladores un panorama sobre lo que desean los usuarios. Está libre de detalles técnicos o de implementación. Podemos pensar en un caso de uso como una secuencia de transacciones en un sistema. El modelo de casos de uso se basa en las interacciones y relaciones de los casos de uso individuales.

Un caso de uso siempre describe tres cosas: un actor que inicia un evento, el evento que desencadena un caso de uso y el caso de uso que realiza las acciones desencadenadas por el evento. En un caso de uso, un actor que utiliza el sistema inicia un evento que a su vez genera una serie relacionada de interacciones en el sistema. Los casos de uso se utilizan para documentar una transacción o evento individual. Se introduce un evento en el sistema, el cual ocurre en un tiempo y lugar específicos para provocar que el sistema haga algo. Para obtener más información sobre los símbolos de los casos de uso y cómo dibujar diagramas de casos de uso, vea el capítulo 2.



FIGURA 10.5

Una vista general de los diagramas de UML que muestra cómo cada diagrama conduce al desarrollo de otros diagramas de UML.



La figura 10.6 es el ejemplo de un caso de uso de la inscripción de estudiantes en una universidad. Cabe mencionar que se representan sólo las funciones más importantes. El caso de uso **Agregar estudiante** no indica cómo agregar estudiantes, el método de implementación. Los estudiantes se pueden agregar en persona, a través de Web, mediante el uso de un teléfono de tonos o a través de una combinación de todos estos métodos. El caso de uso **Agregar estudiante** incluye el caso de uso **Verificar identidad** para verificar la identidad del estudiante. El caso de uso **Comprar libro de texto** extiende el caso de uso **Inscribirse en clase** y puede formar parte de un sistema para inscribir a los estudiantes en un curso en línea.

Tal vez parezca que el caso de uso **Cambiar información de estudiante** fuera una característica menor del sistema y no debiera incluirse en el diagrama de casos de uso, pero como esta información cambia con frecuencia, la ad-



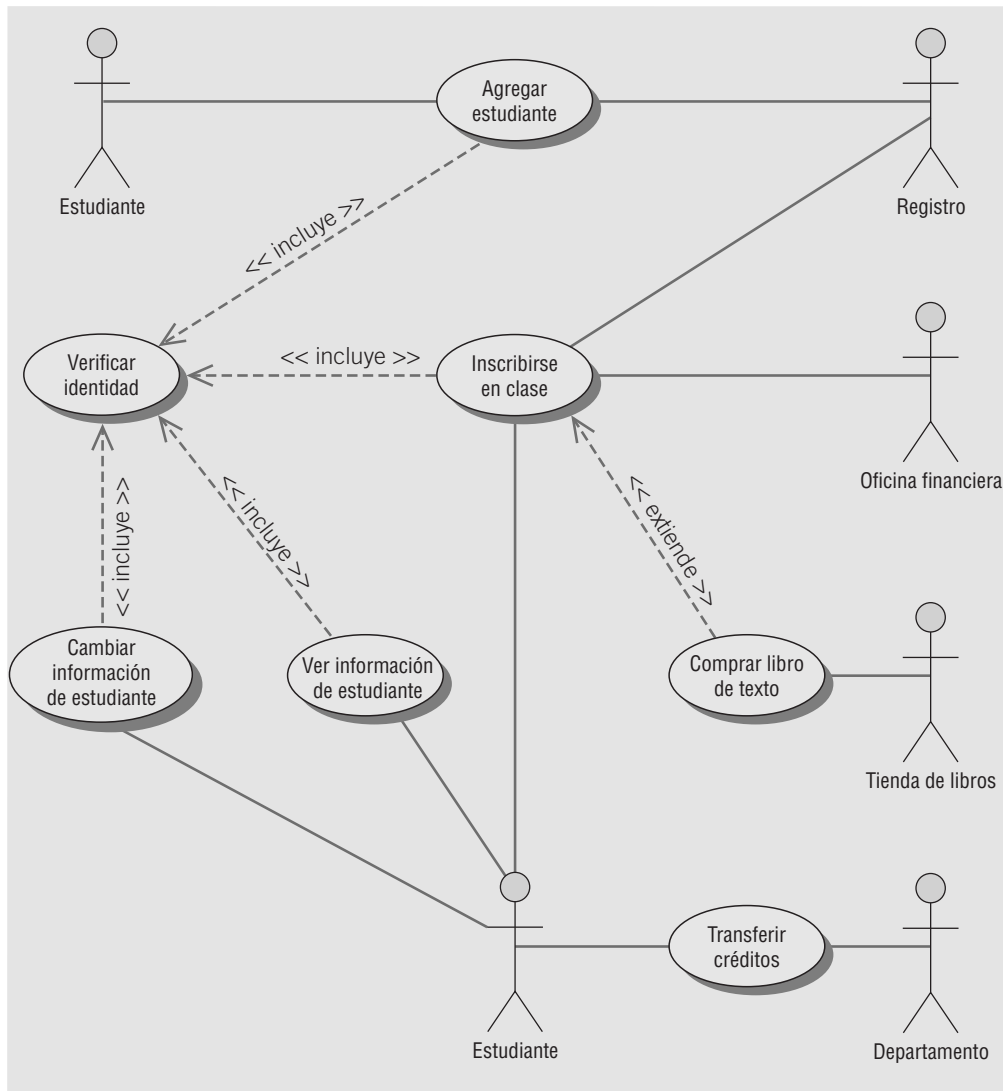


FIGURA 10.6

Ejemplo de casos de uso sobre la inscripción de estudiantes.

ministración tiene mucho interés en permitir a los estudiantes cambiar su propia información personal. El hecho de que los administradores consideren esto importante no sólo justifica, sino también exige, que se elabore el caso de uso.

No se debe permitir a los estudiantes cambiar el promedio de su calificación, las cuotas pendientes u otra información relacionada. Este caso de uso también incluye el caso de uso **Verificar identidad**, que en esta situación significa que el estudiante tiene que introducir un ID de usuario y una contraseña para poder acceder al sistema. **Ver información de estudiante** permite a los estudiantes ver su información personal, así como los cursos y las calificaciones.

En la figura 10.7 se muestra el ejemplo de un escenario de caso de uso. Algunas de las áreas incluidas son opcionales ya que tal vez no todas las organizaciones las utilicen. Las tres áreas principales son:

1. Un área de encabezado que contiene los identificadores e iniciadores de casos.
2. Los pasos realizados.
3. Un área al pie que contiene las precondiciones, suposiciones, preguntas y demás información.

En la primera área el caso de uso se identifica por su nombre, **Cambiar información de estudiante**; el actor se identifica como **Estudiante** y se describen el Caso de uso y Evento desencadenador. La segunda área contiene una serie de pasos que se realizan siempre y cuando no haya errores. Por último, en la tercera área se identifican todas las pre- y post-condiciones, además de las suposiciones. Algunas de éstas son obvias, como la pre-condición de que el estudiante esté en la página Web correcta y la suposición de que el estudiante tenga un ID y contraseña válidos. Otras no son tan obvias, como la cuestión pendiente relacionada con las veces que se permite a un estudiante iniciar sesión en el sistema.

|   |  |  |
|---|--|--|
| <b>Nombre del caso de uso:</b> Cambiar información de estudiante  |  | <b>ID única:</b> Estudiante UC 005                                 |
| <b>Área:</b>  | Sistema de estudiantes   |  |
| <b>Actor(es):</b>   | Estudiante   |  |
| <b>Descripción:</b>   | Permitir al estudiante cambiar su propia información tal como el nombre, la dirección de su casa, el número telefónico de su casa, la dirección del campus, el número telefónico del campus, el número telefónico celular y demás información mediante el uso de un sitio Web. |  |
| <b>Evento desencadenador:</b>   | El estudiante usa el sitio Web Cambiar información de estudiante, introduce el ID y contraseña de estudiante y hace clic en el botón <b>Enviar</b> .   |  |
| <b>Tipo de desencadenador:</b>  | <input checked="" type="checkbox"/> Externo <input type="checkbox"/> Temporal  |  |
| <b>Pasos realizados (ruta principal)</b>  |  |  |
|   |  | <b>Información para los pasos</b>                                  |
| 1. El estudiante inicia sesión en el servidor Web seguro.   |  | ID de estudiante, contraseña                                       |
| 2. Se lee el registro del estudiante y se verifica la contraseña.   |  | Registro de estudiante, ID de estudiante, contraseña               |
| 3. La información personal actual del estudiante se muestra en la página Web Cambiar datos de estudiante.                       |  | Registro de estudiante   |
| 4. El estudiante introduce los cambios en el formulario Web Cambiar datos de estudiante y hace clic en el botón <b>Enviar</b> . |  | Formulario Web Cambiar datos de estudiante                         |
| 5. Los cambios se validan en el servidor Web.   |  | Formulario Web Cambiar datos de estudiante                         |
| 6. Se escribe el registro en el Diario de cambios de estudiantes.   |  | Formulario Web Cambiar datos de estudiante                         |
| 7. Se actualiza el registro del estudiante en el Archivo maestro de estudiantes.  |  | Formulario Web Cambiar datos de estudiante, Registro de estudiante |
| 8. Se envía la página Web de confirmación al estudiante.  |  | Formulario Web Cambiar datos de estudiante                         |
| <b>Pre-condiciones:</b>   | El estudiante se encuentra en la página Web Cambiar información de estudiante.   |  |
| <b>Post-condiciones:</b>  | El estudiante cambió con éxito su información personal.  |  |
| <b>Suposiciones:</b>  | El estudiante tiene un navegador y un ID de usuario y contraseña válidos.  |  |
| <b>Requerimientos cumplidos:</b>  | Permitir que los estudiantes puedan cambiar su información personal mediante el uso de un sitio Web seguro.  |  |
| <b>Cuestiones pendientes:</b>   | ¿Hay que controlar el número de veces que se permite a un estudiante iniciar sesión?   |  |
| <b>Prioridad:</b>   | Media  |  |
| <b>Riesgo:</b>  | Medio  |  |

FIGURA 10.7

Un escenario de caso de uso se divide en tres secciones: identificación e iniciación, pasos realizados y las condiciones, suposiciones y preguntas.

Los diagramas de casos de uso proveen la base para crear otros tipos de diagramas, como los diagramas de clases y los diagramas de actividad. Los escenarios de casos de uso son útiles para dibujar diagramas de secuencia. Tanto los diagramas de casos de uso como los escenarios de casos de uso son potentes herramientas para ayudarnos a comprender la forma en que un sistema funciona en general.

DIAGRAMAS DE ACTIVIDAD

Los diagramas de actividad muestran la secuencia de actividades en un proceso, incluyendo las actividades secuenciales y paralelas, además de las decisiones que se toman. Por lo general se crea un diagrama de actividad para un caso de uso y puede mostrar los distintos escenarios posibles.

En la figura 10.8 se muestran los símbolos en los diagramas de actividad. Un rectángulo con esquinas redondas representa una actividad, ya sea manual —como firmar un documento— o automatizada —como un método o programa—.

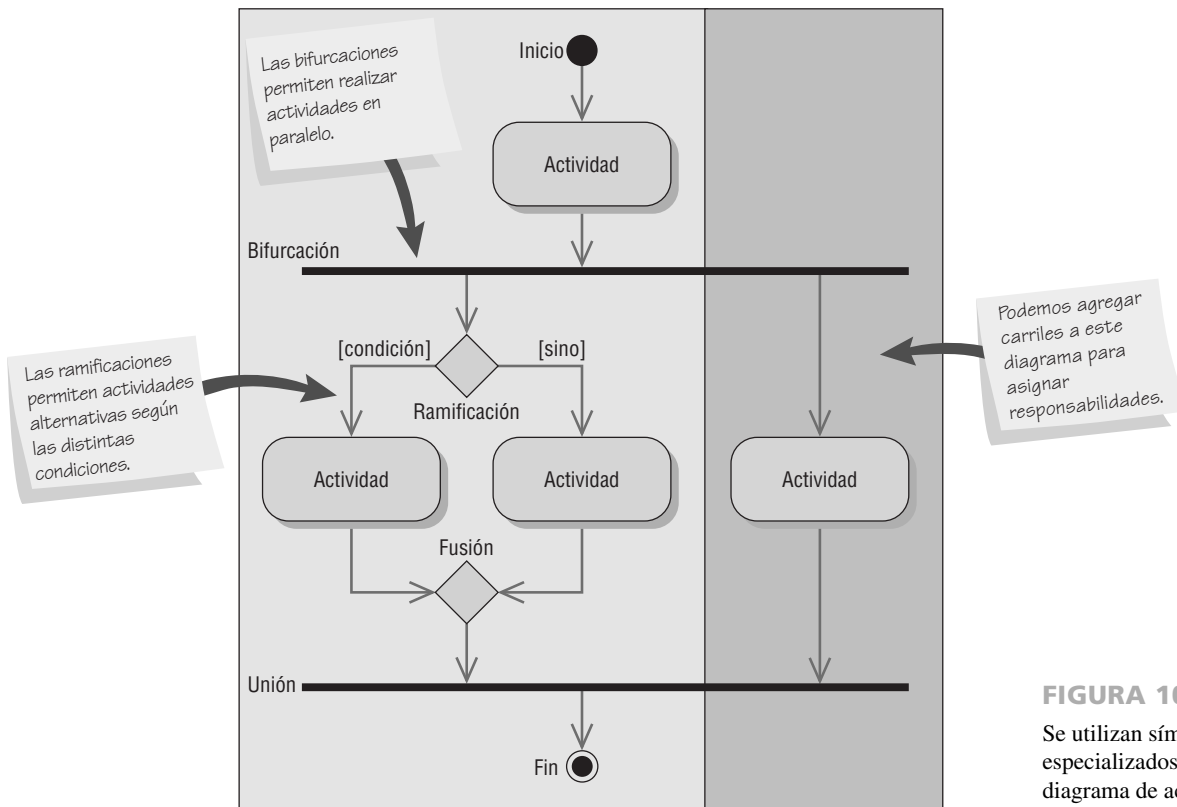


FIGURA 10.8

Se utilizan símbolos especializados para dibujar un diagrama de actividad.

Una flecha representa a un evento. Los eventos representan cosas que ocurren en cierto momento y lugar.

Un diamante representa una decisión (también conocida como ramificación) o una fusión. En las decisiones hay una flecha que entra al diamante y varias que salen de él. Se puede incluir una condición de guardia, que muestra los valores de la condición. Las fusiones muestran varios eventos que se combinan para formar un evento.

Un rectángulo largo y plano representa una barra de sincronización. Estas barras se utilizan para mostrar las actividades paralelas, donde puede haber un evento que entre a la barra de sincronización y varios eventos que salgan de ella, a lo cual se le denomina bifurcación. Una sincronización en la que varios eventos se fusionan en uno solo se denomina unión.

Hay dos símbolos que muestran el inicio y fin del diagrama. El estado inicial se muestra como un círculo relleno. El estado final se muestra como un círculo negro rodeado por un círculo blanco.

Los rectángulos que rodean otros círculos se denominan carriles (*swimlanes*). Estos carriles indican particionamiento y se utilizan para mostrar qué actividades se realizan en cada plataforma, como un navegador, servidor o computadora mainframe; también muestran las actividades que realizan distintos grupos de usuarios. Los carriles son zonas que pueden describir tanto la lógica como la responsabilidad de una clase.

En la figura 10.9 podemos ver un ejemplo de carriles. En esta figura se ilustra un diagrama de actividad para el caso de uso **Cambiar información de estudiante**. Empieza con el estudiante al iniciar sesión en el sistema mediante el proceso de llenar un formulario Web y hacer clic en el botón **Enviar**. El formulario se transmite al servidor Web, el cual pasa a su vez los datos a la computadora mainframe. La mainframe accede a la base de datos de ESTUDIANTES y pasa el mensaje “No se encontró” o los datos del estudiante seleccionado al servidor Web.

El diamante debajo del estado **Obtener registro de estudiante** indica esta decisión. Si no se encontró el registro del estudiante, el servidor Web muestra un mensaje de error en la página Web. Si se encontró el registro del estudiante, el servidor Web da formato a una nueva página Web que contiene los datos del estudiante en un formulario Web. El estudiante puede cancelar el cambio a través de los estados **Iniciar sesión en sistema** o **Introducir cambios**, y la actividad se detiene.

Si el estudiante introduce los cambios en el formulario y hace clic en el botón **Enviar**, los datos del cambio se transmiten al servidor y se empieza a ejecutar un programa que valida los cambios. Si hay errores se envía un mensaje de error a la página Web; si los datos son válidos se actualiza el registro de estudiantes y se escribe un Registro en el Diario de cambios de estudiantes. Después de una actualización válida, se envía una página Web de confirmación al navegador y termina la actividad.

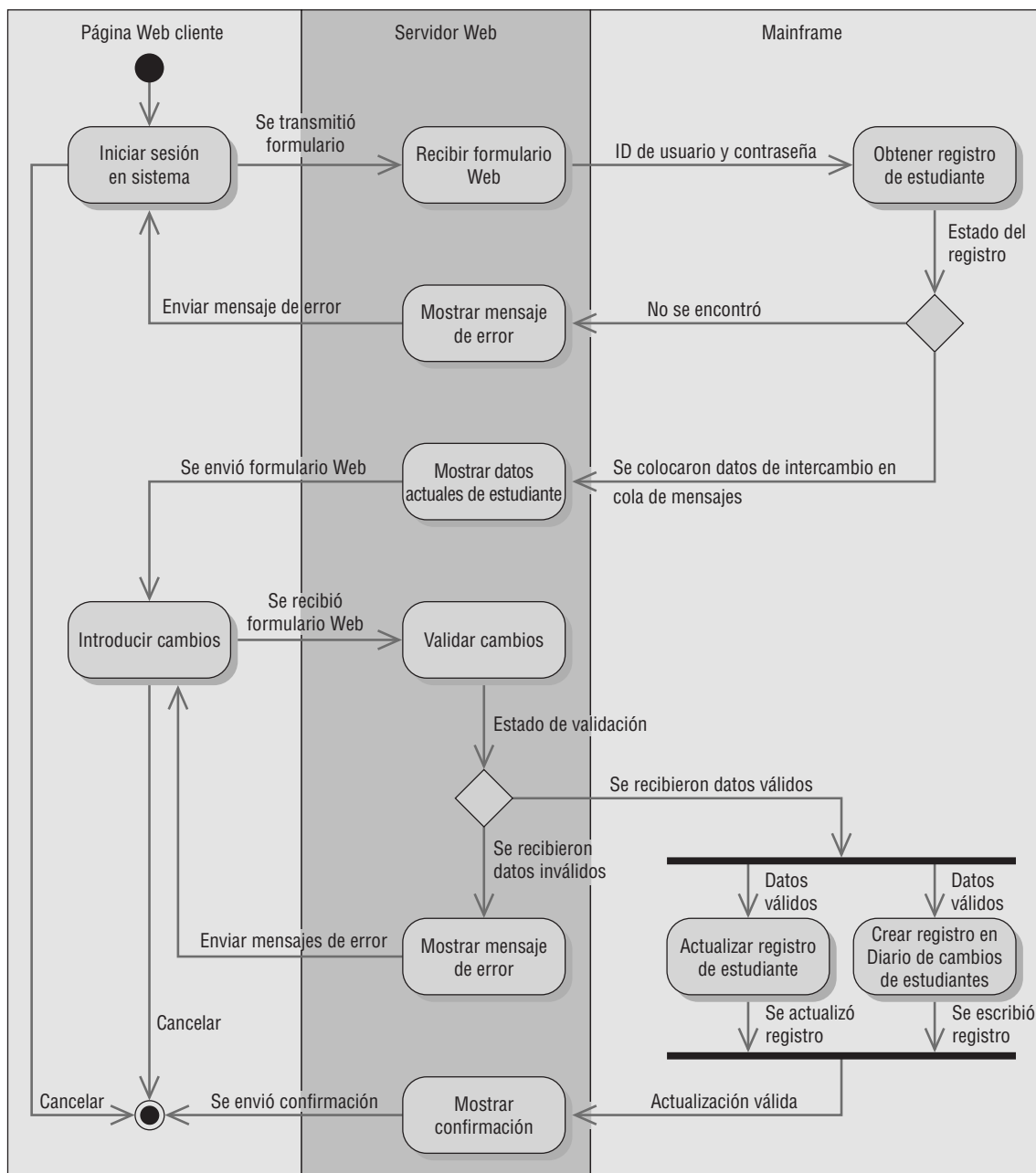


FIGURA 10.9

Este diagrama de actividad muestra tres carriles: Página Web cliente, Servidor Web y Mainframe.

### Creación de diagramas de actividad

Para crear diagramas de actividad hay que preguntarse qué ocurre primero y qué ocurre después. Debemos determinar si las actividades se realizarán en secuencia o en paralelo. Si se crearon diagramas de flujo de datos físicos (como vimos en el capítulo 7), hay que examinarlos para determinar la secuencia de las actividades. Busque los lugares en donde se toman decisiones y pregunte qué ocurre en cada uno de los resultados de esas decisiones. También podemos crear diagramas de actividad al examinar todos los escenarios de un caso de uso.

Cada ruta a través de las diversas decisiones incluidas en el caso de uso es un escenario distinto. En la ruta principal estarían **Iniciar sesión en sistema**, **Recibir formulario Web**, **Obtener registro de estudiante**, **Mostrar datos actuales de estudiante**, **Introducir cambios**, **Validar cambios**, **Actualizar registro de estudiante**, **Crear registro en Diario de cambios de estudiantes** y **Mostrar confirmación**.

Éste no es el único escenario que surge de este caso de uso. Puede haber otros. Una posibilidad podría ser **Iniciar sesión en sistema**, **Recibir formulario Web**, **Obtener registro de estudiante** y **Mostrar mensaje de error**. Otro escenario podría ser **Iniciar sesión en sistema**, **Recibir formulario Web**, **Obtener registro de estudiante**, **Mostrar datos actuales de estudiante**, **Introducir cambios**, **Validar cambios** y **Mostrar mensaje de error**.



## OPORTUNIDAD DE CONSULTORÍA 10.2

### Reciclando el entorno de programación

“Creo que estoy escribiendo el mismo código una y otra vez”, dice Benito Pérez, un programador que trabaja en el nuevo diseño de un almacén automatizado. “Últimamente he escrito tantos programas relacionados con cosas de robots que se controlan a sí mismas: carritos automatizados para la sala de correo, robots de vigilancia de edificios, limpiadores de albercas automáticos, podadores de césped automáticos, trenes de monorriel y ahora carritos de almacén. Todas son variaciones del mismo tema”.

Lisa Bernoulli, gerente del proyecto, ha escuchado este tipo de quejas durante años. Ella le contesta: “Oh, vamos Ben. Estas cosas en realidad no están tan relacionadas. ¿Cómo puedes comparar un robot de la sala de correo, un almacén automatizado y un tren de monorriel? Apuesto a que menos del 10 por ciento del código es el mismo”.

“Mira”, dice Benito. “Los tres involucran máquinas que tienen que buscar un punto inicial, seguir una ruta tortuosa, hacer paradas para cargar y descargar y, en un momento dado, llegar a un punto final. Los tres tienen que tomar decisiones en las ramificaciones de sus rutas. Los tres tienen que evitar chocar con cosas. Estoy cansado de rediseñar código que es bastante familiar para mí”.

“Hmmm”, reflexiona Lisa mientras da un vistazo a los requerimientos básicos del sistema del almacén y recuerda el sistema de monocarril en el que Benito y ella trabajaron el año pasado. Los requerimientos se referían a una pequeña empresa de fabricación de aparatos electrónicos en lotes pequeños que deseaba automatizar su sistema de almacén y movimiento de productos. El almacén contiene piezas entrantes, trabajo en progreso y productos terminados. El almacén automatizado utiliza un carrito robot de plataforma, similar a un ca-

rrito de golf, con la excepción de que no tiene asientos. Los carritos de robot de plataforma tienen una superficie de carga plana de  $6 \times 4$  pies<sup>2</sup>, aproximadamente a 3 pies por encima del nivel del suelo. Estos carritos tienen un dispositivo de radio comunicaciones que provee un enlace de datos en tiempo real a la computadora central del almacén. Los carritos de plataforma tienen dos sensores: un sensor de ruta que detecta un tipo especial de pintura y un sensor de movimiento que detecta el movimiento. Estos carritos siguen rutas pintadas en el piso de la fábrica. Los códigos especiales de pintura marcan bifurcaciones y ramificaciones en las rutas, los puntos de inicio y final del carrito, además de los puntos de ubicaciones en general.

Las instalaciones incluyen tres estaciones de puertos de carga y 10 estaciones de trabajo. Cada estación tiene una terminal de video o computadora conectada a la computadora central. Cuando se requieren productos o cuando están listos para ser recolectados de una estación, el trabajador en la estación informa a la computadora central. Después, la computadora central despacha los carritos según sea apropiado. Cada estación tiene un punto de entrega y un punto de recolección. Los carritos de plataforma se desplazan por la fábrica recogiendo el trabajo en los puntos de recolección y dejan el trabajo en los puntos de entrega. El programa que opere los carritos deberá interactuar mucho con el programa existente para programar tareas que ayuda a establecer los horarios para las tareas de las estaciones de trabajo.

¿Cómo debería Lisa proceder en cuanto a reutilizar el trabajo que hizo Benito Pérez en el monocarril para adaptarlo a su tarea actual de crear un objeto carrito? Explique en dos párrafos.

Los carriles son útiles para mostrar cómo se debe transmitir o convertir datos, tales como los que van del sitio Web al servidor, o del servidor a la mainframe. Por ejemplo, el diagrama de actividad de **Cambiar registro de estudiante** tiene tres carriles.

El carril de la izquierda muestra las actividades que ocurren en el navegador cliente. Hay que crear páginas Web para estas actividades. El carril de en medio muestra las actividades que ocurren en el servidor. Los eventos como **Se transmitió formulario** representan los datos que se transmiten del navegador al servidor; debe haber programas en el servidor para recibir y procesar los datos cliente.

El carril de la derecha representa la computadora mainframe. En empresas grandes es común para muchas aplicaciones Web trabajar con una computadora mainframe. Gran parte de los datos en las grandes organizaciones residen en bases de datos de equipos mainframe y hay una gran cantidad de programas para mainframe en existencia.

Cuando un evento cruza el carril del servidor a la computadora mainframe, debe haber un mecanismo para transmitir los datos del evento entre las dos plataformas. Los servidores usan un formato distinto para representar datos (ASCII) al que usan las computadoras mainframe (uno llamado EBCDIC). Debe haber middleware presente para hacerse cargo de la conversión. A menudo, las computadoras IBM utilizan una cola de mensajes (*mqueue*). La cola de mensajes recibe datos de los programas del servidor, los coloca en un área de retención y llama a un programa de la mainframe, que por lo general está escrito en un lenguaje conocido como CICS. Este programa recupera o actualiza los datos y envía los resultados de vuelta a la cola de mensajes.

En el diagrama de actividad de ejemplo antes mostrado, la decisión debajo del estado **Obtener registro de estudiante** se toma en la computadora mainframe. Esto significa que la cola de mensajes recibe un mensaje “No se encontró” o recibe el registro de la base de datos para el estudiante. Si la mainframe simplemente colocara **Se recibió estado de registro** en la cola de mensajes y la decisión se evaluara en el servidor, éste tendría que llamar a la mainframe de nuevo para obtener los datos válidos. Esto provocaría que la respuesta llegara más tarde a la persona que espera en el navegador.

Los carriles también ayudan a dividir las tareas en un equipo. Se requieren diseñadores Web para las páginas Web que se muestran en el navegador cliente. Otros miembros tendrían que trabajar con lenguajes de programación tales como Java, PHP, Ruby on Rails, PERL o .NET en el servidor. Los programadores de CICS de la main-frame escribirían programas que funcionaran con la cola de mensajes. El analista debe asegurar que los datos que necesitan los diversos miembros del equipo estén disponibles y definidos en forma correcta. Algunas veces los datos en la cola de mensajes están en forma de un documento de XML. Si hay una organización externa involucrada, los datos también podrían ser un documento de XML.

El diagrama de actividad provee un mapa de un caso de uso; además permite al analista experimentar al mover partes del diseño a distintas plataformas y preguntar: “¿Qué pasaría si?” para una variedad de decisiones. El uso de símbolos únicos y carriles hace de este diagrama algo que las personas desean usar para comunicarse con los demás.

Se pueden usar diagramas de actividad para construir planes de prueba. Hay que evaluar cada evento para ver si el diagrama de actividad pasa al siguiente estado. Hay que evaluar cada decisión para ver si se toma la ruta correcta cuando ocurren las condiciones de decisión.

Los diagramas de actividad no se utilizan para todos los casos. Debe usar diagramas de actividad cuando:

- 1. Le ayude a comprender las actividades de un caso de uso.
- 2. El flujo de control sea complejo.
- 3. Exista la necesidad de modelar el flujo de trabajo.
- 4. Haya que mostrar todos los escenarios.

El analista no necesita un diagrama de actividad cuando el caso de uso es simple, o cuando hay que modelar el cambio de estado.

También se pueden usar los diagramas de actividad para modelar un método de nivel más bajo, en el que se muestre la lógica detallada.

Entradas en el repositorio para un diagrama de actividad

Cada estado y evento se pueden definir con más detalle mediante el uso de una descripción de texto en el repositorio, el cual es una colección de descripciones de texto para el proyecto. Hay que describir los estados con información sobre el estado, tal como el nombre de la página Web, los elementos de la página Web, y otros datos. Hay que describir los eventos con la información requerida para comunicarse con el siguiente estado, como los datos del formulario Web, los datos que se colocan en una cola de mensajes o mediante una descripción del evento que ocasionó la transición, como el clic de un botón.

DIAGRAMAS DE SECUENCIA Y DE COMUNICACIÓN

Un diagrama de interacción puede ser un diagrama de secuencia o un diagrama de comunicación, ambos de los cuales muestran esencialmente la misma información. Estos diagramas, junto con los diagramas de clases, se utilizan para la realización de un caso de uso, lo cual es una forma de lograr o realizar un caso de uso.

Diagramas de secuencia

Los diagramas de secuencia pueden ilustrar una sucesión de interacciones entre clases o instancias de objetos a través del tiempo. A menudo, los diagramas de secuencia se utilizan para ilustrar el procesamiento descrito en los escenarios de casos de uso. En la práctica, los diagramas de secuencia se derivan del análisis de casos de uso y se utilizan en el diseño de sistemas para derivar las interacciones, las relaciones y los métodos de los objetos en el sistema. Los diagramas de secuencia se utilizan para mostrar el patrón general de las actividades o interacciones en un caso de uso. Cada escenario de caso de uso puede crear un diagrama de secuencia, aunque éstos no siempre se crean para escenarios de menor importancia.

En la figura 10.10 se muestran los símbolos utilizados en los diagramas de secuencia. Los actores y las clases o instancias de objetos se muestran en cuadros en la parte superior del diagrama. El objeto de más a la izquierda es el objeto inicial y puede ser una persona (para la cual se utiliza un símbolo de actor de caso de uso), ventana, cuadro de diálogo u otra interfaz de usuario. Algunas de las interacciones son sólo físicas, como la firma de un contrato. Los rectángulos de la parte superior utilizan indicadores en el nombre para indicar si el rectángulo representa un objeto, una clase, o una clase y un objeto.

|                    |  |
|--------------------|--|
| nombreObjeto:      | Un nombre seguido de un signo de dos puntos representa a un objeto.                                |
| :clase             | Un signo de dos puntos seguido de un nombre representa a una clase.                                |
| nombreObjeto:clase | Un nombre seguido de un signo de dos puntos y de otro nombre, representa a un objeto en una clase. |



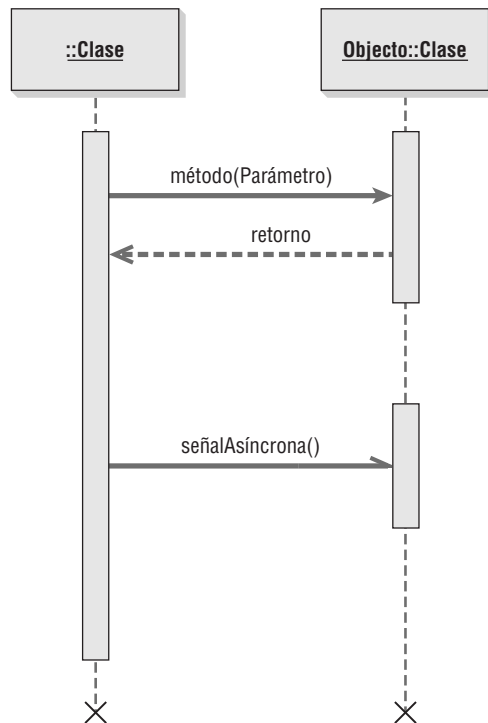


FIGURA 10.10

Símbolos especializados que se utilizan para dibujar un diagrama de secuencia.

Una línea vertical representa la línea de vida de la clase u objeto, que corresponde al tiempo a partir del que se creó hasta el momento en que se destruye. Una X en la parte inferior de la línea de vida representa el momento en que se destruye el objeto. Una barra lateral o un rectángulo vertical en la línea de vida muestran el foco de control cuando el objeto está ocupado haciendo cosas.

Las flechas horizontales muestran mensajes o señales que se envían entre las clases. Los mensajes pertenecen a la clase receptora. Hay algunas variaciones en las flechas de los mensajes. Las puntas de flecha sólidas representan llamadas sincrónicas, que son las más comunes. Éstas se utilizan cuando la clase emisora espera una respuesta de la clase receptora y el control se devuelve a la clase emisora cuando la clase receptora que recibe el mensaje termina de ejecutarse. Las medias puntas de flecha (o abiertas) representan llamadas asíncronas: aquellas que se envían sin esperar que la clase emisora las devuelva. Un ejemplo sería el uso de un menú para ejecutar un programa. El retorno se muestra como una flecha, algunas veces con una línea punteada. Los mensajes se etiquetan mediante el uso de uno de los siguientes formatos:

- El nombre del mensaje seguido de paréntesis vacíos: **nombreMensaje()**.
- El nombre del mensaje seguido de parámetros entre los paréntesis: **nombreMensaje(parámetro1, parámetro2 ...)**.
- El nombre del mensaje seguido del tipo de parámetro, nombre del parámetro y cualquier valor predeterminado para el parámetro entre paréntesis: **nombreMensaje(tipoParámetro:nombreParámetro-(valorPredeterminado))**. Los tipos de los parámetros indican el tipo de datos, como cadena, número o fecha.
- El mensaje puede ser un estereotipo tal como **<<Crear>>** para indicar que se va a crear un nuevo objeto como resultado del mensaje.

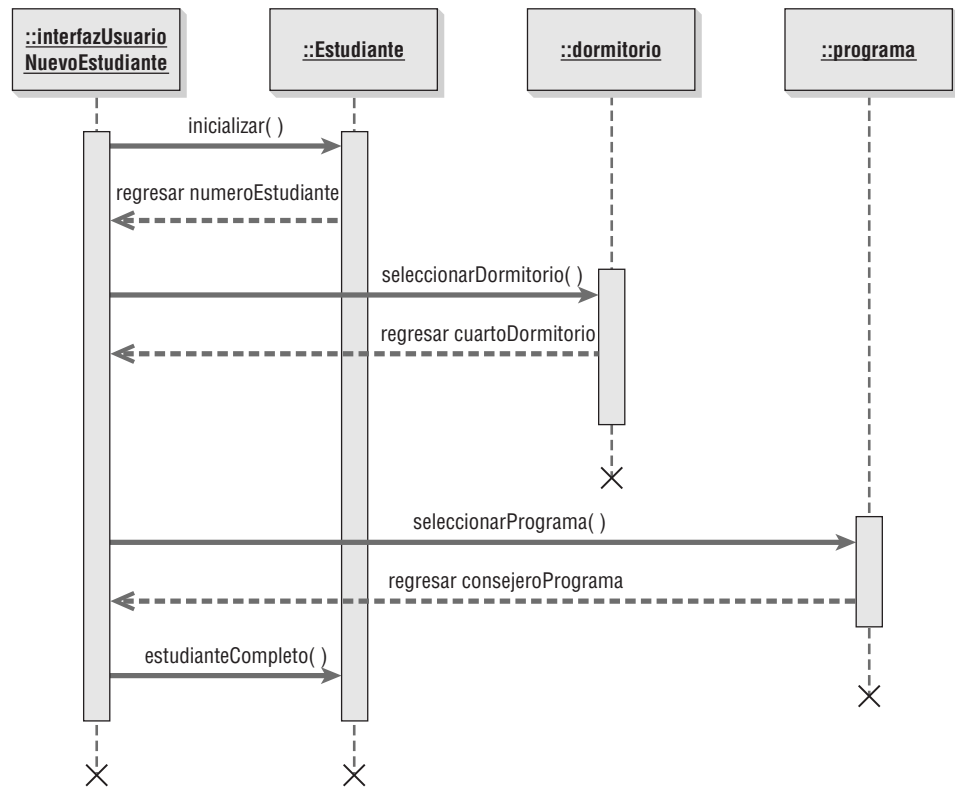
La sincronización en el diagrama de secuencia se muestra de arriba hacia abajo; la primera interacción se dibuja en la parte superior del diagrama y la interacción que ocurre al último se dibuja en la parte inferior del diagrama. Las flechas de interacción empiezan en la barra del actor u objeto que inicia la interacción y terminan apuntando a la barra del actor u objeto que recibe la solicitud de interacción. El actor inicial u objeto se muestra a la izquierda. Éste puede ser el actor que inicia la actividad o una clase que represente la interfaz de usuario.

La figura 10.11 es un ejemplo simplificado de un diagrama de secuencia para un caso de uso que admite a un estudiante en una universidad. A la izquierda está la clase **interfazUsuarioNuevoEstudiante** que se utiliza para obtener la información del estudiante. El mensaje **inicializar()** se envía a la clase **Estudiante**, la cual crea un nuevo registro de estudiante y devuelve el número de estudiante. Para simplificar el diagrama omitimos los parámetros que se envían a la clase **Estudiante**, pero éstos deben incluir el nombre del estudiante, su dirección y otros. La siguiente actividad es enviar un mensaje **seleccionarDormitorio** a la clase



**FIGURA 10.11**

Un diagrama de secuencia para admitir a un estudiante. Los diagramas de secuencia hacen énfasis en el orden de los mensajes en el tiempo.



**Dormitorio.** Este mensaje debe incluir la información de selección del dormitorio, como un dormitorio para el cuidado de la salud u otros requerimientos del estudiante. La clase **Dormitorio** devuelve el nombre del dormitorio y el número de habitación. La tercera actividad es enviar un mensaje **seleccionarPrograma** a la clase **Programa**, incluyendo el nombre del programa y demás información sobre el curso de estudio. Se devuelve el nombre del consejero del programa a la clase **interfazUsuarioNuevoEstudiante**. Se envía un mensaje **estudianteCompleto** a la clase **Estudiante** con el nombre del dormitorio, del consejero y demás información pertinente.

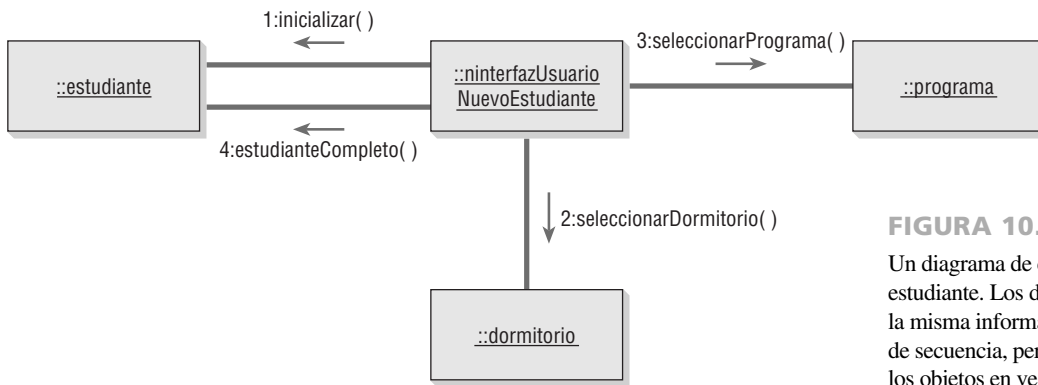
Los diagramas de secuencia se pueden utilizar para traducir el escenario de caso de uso en una herramienta visual para el análisis de sistemas. El diagrama de secuencia inicial utilizado en el análisis de sistemas muestra los actores y las clases en el sistema, así como las interacciones entre ellos para un proceso específico. Usted puede utilizar esta versión del diagrama de secuencia para verificar los procesos con los expertos del área de negocios que le hayan ayudado a desarrollar los requerimientos del sistema. Un diagrama de secuencia hace énfasis en el orden de los mensajes (secuencia) en el tiempo.

Durante la fase de diseño de sistemas, los diagramas de secuencia se refinan para derivar los métodos y las interacciones entre las clases. Los mensajes de una clase se utilizan para identificar las relaciones de las clases. Los actores en los primeros diagramas de secuencia se traducen en interfaces y las interacciones de las clases se traducen en métodos de clase. Los métodos de clase que se utilizan para crear instancias de otras clases y realizar otras funciones internas del sistema se revelan en el diseño del sistema mediante el uso de diagramas de secuencia.

### Diagramas de comunicación

Los diagramas de comunicación se introdujeron en el UML 2.0. Su nombre original en el UML 1.x era diagramas de colaboración. Los diagramas de comunicación describen las interacciones entre dos o más cosas en el sistema que desempeñan un comportamiento mayor a lo que cualquiera de las dos cosas pueden hacer por su cuenta. Por ejemplo, un automóvil se puede descomponer en varios miles de piezas individuales. Las piezas se conectan para formar los subsistemas principales del vehículo: el motor, la transmisión, el sistema de frenos, etcétera. Las piezas individuales del automóvil se pueden considerar como clases, ya que tienen distintos atributos y funciones. Las piezas individuales del motor forman una colaboración, ya que se “comunican” entre sí para hacer que el motor funcione cuando el conductor pisa el acelerador.

Un diagrama de comunicación consta de tres partes: los objetos (también llamados participantes), los enlaces de comunicación y los mensajes que se pueden pasar a través de esos enlaces. Los diagramas de comunica-

**FIGURA 10.12**

Un diagrama de comunicación para admitir a un estudiante. Los diagramas de comunicación muestran la misma información que se describe en un diagrama de secuencia, pero hacen énfasis la organización de los objetos en vez del orden en el tiempo.

ción muestran la misma información que un diagrama de secuencia, pero pueden ser más difíciles de leer. Para poder mostrar el orden en el tiempo, debemos indicar un número de secuencia y describir el mensaje.

Un diagrama de comunicación hace énfasis en la organización de los objetos, mientras que un diagrama de secuencia hace énfasis en el orden de los mensajes en el tiempo. Un diagrama de comunicación mostrará una ruta para indicar cómo está un objeto enlazado con otro.

Cierto software de modelado de UML, como Rational Rose de IBM, convierte con el clic de un botón un diagrama de secuencia en un diagrama de comunicación, o un diagrama de comunicación en un diagrama de secuencia. En la figura 10.12 se muestra el diagrama de comunicación para admitir a un estudiante. Cada rectángulo representa a un objeto o una clase. Las líneas conectoras muestran las clases que necesitan colaborar o trabajar entre sí. Los mensajes que se envían de una clase a otra se muestran a lo largo de las líneas conectoras. Los mensajes están numerados para mostrar la secuencia en el tiempo. También se pueden incluir valores de retorno y se pueden enumerar para indicar cuándo se devuelven dentro de la secuencia de tiempo.

## DIAGRAMAS DE CLASES

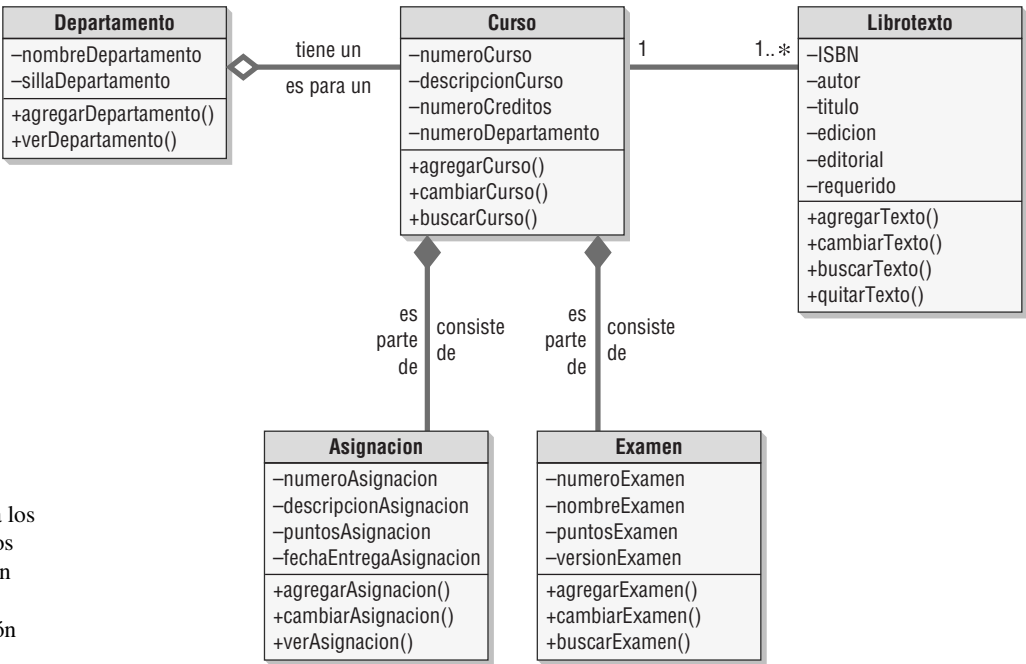
Las metodologías orientadas a objetos trabajan para descubrir las clases, atributos, métodos y relaciones entre las clases. Como la programación ocurre a nivel de clase, definir clases es una de las tareas más importantes del análisis orientado a objetos. Los diagramas de clases muestran las características estáticas del sistema y no representan ningún procesamiento en especial. Un diagrama de clases también muestra la naturaleza de las relaciones entre las clases.

En un diagrama de clases, las clases se representan mediante un rectángulo. En el formato más simple, el rectángulo puede incluir sólo el nombre de la clase, pero también puede incluir atributos y métodos. Los atributos son lo que la clase conoce sobre las características de los objetos, y los métodos (también llamados operaciones) son lo que la clase sabe acerca de cómo hacer las cosas. Los métodos son pequeñas secciones de código que trabajan con los atributos.

La figura 10.13 muestra un diagrama de clases para los ofrecimientos de cursos. Cabe mencionar que el nombre está centrado en la parte superior de la clase, por lo general en negrita. El área justo debajo del nombre muestra los atributos y la porción inferior lista los métodos. El diagrama de clases muestra los requerimientos de almacenamiento de datos, así como los requerimientos de procesamiento. Más adelante en el capítulo hablaremos sobre el significado de los símbolos de diamante que se muestran en esta figura.

Por lo general los atributos (o propiedades) se designan como privados, o que sólo están disponibles en el objeto. En un diagrama de clases esto se representa con un signo negativo al inicio del nombre del atributo. Los atributos también pueden ser protegidos, lo cual se indica con un símbolo (#). Estos atributos están ocultos para todas las clases, excepto las subclasses inmediatas. Bajo raras circunstancias un atributo se hace público, lo cual significa que otros objetos fuera de su clase pueden verlo. Hacer los atributos privados implica que serán visibles sólo para los objetos externos a través de los métodos de la clase, una técnica que se conoce como encapsulamiento u ocultamiento de la información.

Un diagrama de clase puede mostrar sólo el nombre de la clase, el nombre de la clase y los atributos o el nombre de la clase, los atributos y los métodos. Es útil mostrar sólo el nombre de la clase cuando el diagrama es muy complejo e incluye muchas clases. Si el diagrama es más simple, se pueden incluir los atributos y los métodos. Cuando se incluyen los atributos hay tres formas de mostrar la información de cada uno. La más simple es incluir sólo el nombre del atributo, lo cual ocupa la menor cantidad de espacio.



**FIGURA 10.13**  
Un diagrama de clases para los ofrecimientos de cursos. Los diamantes rellenos muestran agregación y los diamantes vacíos muestran una relación entre un todo y sus partes.

Se puede incluir el tipo de datos (como cadena, doble, entero o fecha) en el diagrama de clases. Las descripciones más completas incluyen un signo de igual (=) después del tipo de datos, seguido del valor inicial del atributo. La figura 10.14 ilustra los atributos de las clases.

Si el atributo debe tomar un valor de un número finito de ellos, como un tipo de estudiante con valores de C para tiempo completo, P para tiempo parcial y N para no matriculado, éstos se pueden incluir entre llaves separados por comas: **tipoEstudiante: char{F,P,N}**.

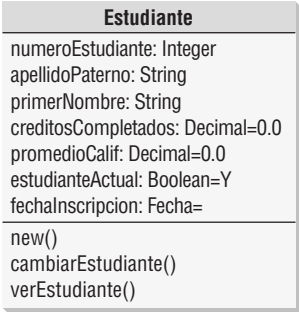
El ocultamiento de información significa que los métodos de los objetos deben estar disponibles para otras clases, por lo que comúnmente los métodos son públicos, lo cual significa que se pueden invocar desde otras clases. En un diagrama de clases, los mensajes públicos (al igual que los atributos públicos) se muestran con un signo positivo (+) al inicio del nombre correspondiente. Los métodos también tienen paréntesis después de su nombre, lo cual indica que se pueden pasar datos como parámetros junto con el mensaje. En el diagrama de clases se pueden incluir los parámetros del mensaje, así como el tipo de datos.

Hay dos tipos de métodos: estándar y personalizados. Los métodos estándar son las cosas básicas que todas las clases de objetos saben cómo hacer, como crear una nueva instancia de un objeto. Los métodos personalizados están diseñados para una clase específica.

**Sobrecarga de métodos**

La sobrecarga de métodos se refiere a incluir el mismo método (u operación) varias veces en una clase. La firma del método incluye su nombre y los parámetros que incluye. El mismo método se puede definir más de una vez en una clase dada, siempre y cuando los parámetros que se envían como parte del mensaje sean distintos; es decir, debe haber una firma de mensaje distinta. Puede haber un número distintos de parámetros, o éstos pueden ser de un tipo distinto, como un número en un método y una cadena de texto en otro método. Podemos encontrar un

**FIGURA 10.14**  
Una clase **Estudiante** extendida que muestra el tipo de datos y, en algunos casos, su valor inicial o valor predeterminado.



ejemplo de sobrecarga de métodos en el uso de un signo de suma en muchos lenguajes de programación. Si los atributos en ambos lados del signo de suma son números, se suman los dos números. Si los atributos son cadenas de caracteres, las cadenas se concatenan para formar una sola cadena extensa.

En un ejemplo de un depósito bancario, un recibo de depósito podría contener sólo el monto a depositar, en cuyo caso el banco depositaría el monto completo, o podría contener el monto a depositar y la cantidad de efectivo a devolver. Ambas situaciones utilizarían un método de verificación de depósito, pero los parámetros (una de las situaciones también requeriría devolver el monto de efectivo) serían distintos.

## Tipos de clases

Las clases se dividen en cuatro categorías: de entidad, de interfaz, abstracta y de control. A continuación explicaremos estas categorías.

**CLASES DE ENTIDAD** Las clases de entidad representan elementos del mundo real como personas o cosas, por ejemplo. Las clases de entidad son las entidades representadas en un diagrama de entidad-relación. Las herramientas CASE como Visible Analyst le permiten crear una clase de entidad de UML a partir de una entidad en un diagrama E-R.

El analista necesita determinar qué atributos debe incluir en las clases. Cada objeto tiene muchos atributos, pero la clase debe incluir sólo aquellos que la organización utilice. Por ejemplo, al crear una clase de entidad para un estudiante en una universidad necesitamos conocer los atributos que identifican al estudiante, como la dirección de su hogar y del campus al que pertenece, así como el promedio de sus calificaciones, los créditos totales, etcétera. Si tuviera que llevar el registro del mismo estudiante para una tienda de ropa en línea, tendría que conocer la información básica de identificación además de otros atributos descriptivos como las medidas o preferencias de color.

**CLASES DE LÍMITE O DE INTERFAZ** Las clases de límite o de interfaz proveen los medios para que los usuarios trabajen con el sistema. Hay dos amplias categorías de clases de interfaz: humana y de sistema.

Una interfaz humana puede ser una pantalla, una ventana, un formulario Web, un cuadro de diálogo, un menú, un cuadro de lista u otro control de visualización. También puede ser un teléfono de tonos, código de barras o cualquier otra forma en que los usuarios puedan interactuar con el sistema. Hay que crear prototipos de las interfaces humanas (como vimos en el capítulo 6) y a menudo se utiliza un guión gráfico para modelar la secuencia de interacciones.

Las interfaces del sistema necesitan enviar o recibir datos de otros sistemas. Esto puede incluir a las bases de datos en la organización. Si se envían datos a una organización externa, por lo general se hace en la forma de archivos de XML u otras interfaces reconocidas con mensajes y protocolos claramente definidos. Las interfaces externas son las menos estables, ya que a menudo hay muy poco o nada de control sobre un socio externo capaz de alterar el formato del mensaje o de los datos.

El XML ayuda a proveer estandarización, ya que un socio externo puede agregar nuevos elementos al documento de XML, aunque una corporación que transforme los datos en un formato que se pueda utilizar para agregar información a una base de datos interna puede ignorar los elementos adicionales sin ningún problema.

Los atributos de estas clases son los que se encuentran en la pantalla o informe. Los métodos son los que se requieren para trabajar con la pantalla o para producir el informe.

**CLASES ABSTRACTAS** Las clases abstractas son clases que no se pueden instanciar en forma directa. Las clases abstractas están enlazadas a clases concretas en una relación de generalización/especialización (gen/spec). Por lo general el nombre de una clase abstracta se escribe en cursiva.

**CLASES DE CONTROL** Las clases de control o activas se utilizan para controlar el flujo de actividades; actúan como un coordinador a la hora de implementar las clases. Para obtener clases que se puedan reutilizar, un diagrama de clases puede incluir muchas clases de control pequeñas. A menudo las clases de control se derivan durante el diseño del sistema.

Es común crear una clase de control sólo para poder reutilizar otra clase. Un ejemplo de ello sería el proceso de inicio de sesión. Podría haber una clase de control que se encargue de la interfaz de usuario de inicio de sesión, que contiene la lógica para verificar el ID y la contraseña del usuario. El problema que surge es que la clase de control del inicio de sesión está diseñada para una pantalla de inicio de sesión específica. Al crear una clase de control de inicio de sesión que maneje sólo la pantalla de inicio de sesión única, los datos se pueden pasar a una clase de control de validación más general, la cual verifica los ID y contraseñas de usuario que recibe de muchas otras clases de control que reciben mensajes de interfaces de usuario específicas. Esto incrementa la reutilización y aísla los métodos de verificación de inicio de sesión de los métodos para manejar la interfaz de usuario.

Las reglas para crear diagramas de secuencia son que todas las clases de interfaz deben estar conectadas a una clase de control. De manera similar, todas las clases de entidad deben estar conectadas a una clase de control. Las clases de interfaz, a diferencia de las otras dos, nunca se conectan de manera directa a las clases de entidad.

## Definición de mensajes y métodos

Podemos definir cada mensaje mediante una notación similar a la que se describe para el diccionario de datos (como vimos en el capítulo 8). La definición debe incluir una lista de los parámetros que se pasan con el mensaje, así como los elementos contenidos en el mensaje de retorno. Los métodos pueden tener lógica definida mediante español estructurado, una tabla de decisión o un árbol de decisión, como vimos en el capítulo 9.

El analista puede usar las técnicas de balanceo horizontal con cualquier método de clase. Todos los datos que se devuelven de una clase de entidad se deben obtener de los atributos almacenados en la clase de entidad, de los parámetros que se pasan en el mensaje que se envía a la clase o como resultado de un cálculo realizado por el método de la clase. Hay que examinar la lógica y los parámetros del método para asegurar que la lógica del método tenga toda la información requerida para completar su tarea. En el capítulo 7 se describe el balanceo horizontal con más detalle.

## CÓMO MEJORAR LOS DIAGRAMAS DE SECUENCIA

Una vez dibujado el diagrama de clases, tal vez sea conveniente regresar al diagrama de secuencia e incluir símbolos especiales para cada uno de los distintos tipos de clases que introdujimos en la sección anterior. En particular, los diagramas de secuencia pueden ser abrumadores si un analista no tiene una metodología sistemática para dibujarlos. Los siguientes pasos son un método útil para mejorar un diagrama de secuencia:

1. Incluya el *actor* del diagrama de caso de uso en el diagrama de secuencia mejorado. Éste será una figura de palitos del diagrama de caso de uso. Puede haber un actor adicional del lado derecho del diagrama, como una compañía de tarjetas de crédito o un banco.
2. Defina una o más *clases de interfaz* para cada actor. Cada actor debe tener su propia clase de interfaz.
3. Cree prototipos de páginas Web para todas las interfaces humanas.
4. Asegúrese de que cada caso de uso tenga una *clase de control*, aunque se pueden crear más durante el diseño detallado. Busque esa clase de control e inclúyala en el diagrama de secuencia.
5. Examine el caso de uso para ver qué *clases de entidad* están presentes. Incluya éstas en el diagrama.
6. Tenga en cuenta que el diagrama de secuencia se puede modificar de nuevo al realizar el diseño detallado, como cuando se crean páginas Web o clases de control adicionales (una para cada formulario Web enviado).
7. Para obtener un mayor grado de reutilización, considere mover los métodos de una clase de control a una clase de entidad.

## Un ejemplo de clase para Web

Las clases también se pueden representar mediante el uso de símbolos especiales para las clases de entidad, de límite (o interfaz) y de control. A estos símbolos se les denomina estereotipos, una extensión para el UML; son símbolos especiales que se pueden utilizar durante el análisis, pero a menudo se utilizan al momento de realizar el diseño orientado a objetos. Estos símbolos permiten al analista la libertad de jugar con el diseño para optimizar el grado de reutilización.

Los distintos tipos de clases se utilizan con frecuencia al trabajar en la fase de diseño de sistemas. La figura 10.15 es un ejemplo que muestra un diagrama de secuencia que representa a un estudiante que ve su información personal y del curso. En el diagrama, **:Interfaz de usuario ver estudiante** es un ejemplo de una clase de interfaz; **:Estudiante**, **:Seccion** y **:Curso** son ejemplos de clases de entidad; **:Controlador interfaz ver estudiante** y **:Calcular promedio calificacion** son clases de control.

El estudiante se muestra a la izquierda como un actor y proporciona un **inicioSesionUsuario** a la clase **:Interfaz de usuario ver estudiante**. Éste es un formulario Web que obtiene el ID y la contraseña del usuario. Cuando el estudiante hace clic en el botón **Enviar**, el formulario Web se pasa a un **:Controlador interfaz ver estudiante**. Esta clase es responsable de la coordinación al enviar los mensajes y recibir la información devuelta por las demás clases.

El **:Controlador interfaz ver estudiante** envía un mensaje **obtenerEstudiante()** a la clase **:Estudiante**, la cual lee una tabla en la base de datos y procede a devolver los **datosEstudiante**.

La **paginaWebEstudiante** se devuelve a la **:Interfaz de usuario ver estudiante**, la cual muestra la información en el navegador Web. Al final de la página hay un **botonSiguiente**, en el cual el estudiante hace clic para ver los cursos. Hacer clic en este botón envía un formulario Web al **:Controlador interfaz ver estudiante**. Este formulario contiene el **numeroEstudiante()** que se envía junto con la **paginaWebEstudiante** y se utiliza para enviar un mensaje a la clase **:Seccion** para que obtenga la calificación de la sección. Si no se enviara el **numeroEstudiante()** en forma automática, el estudiante tendría que introducir su **numeroEstu-**

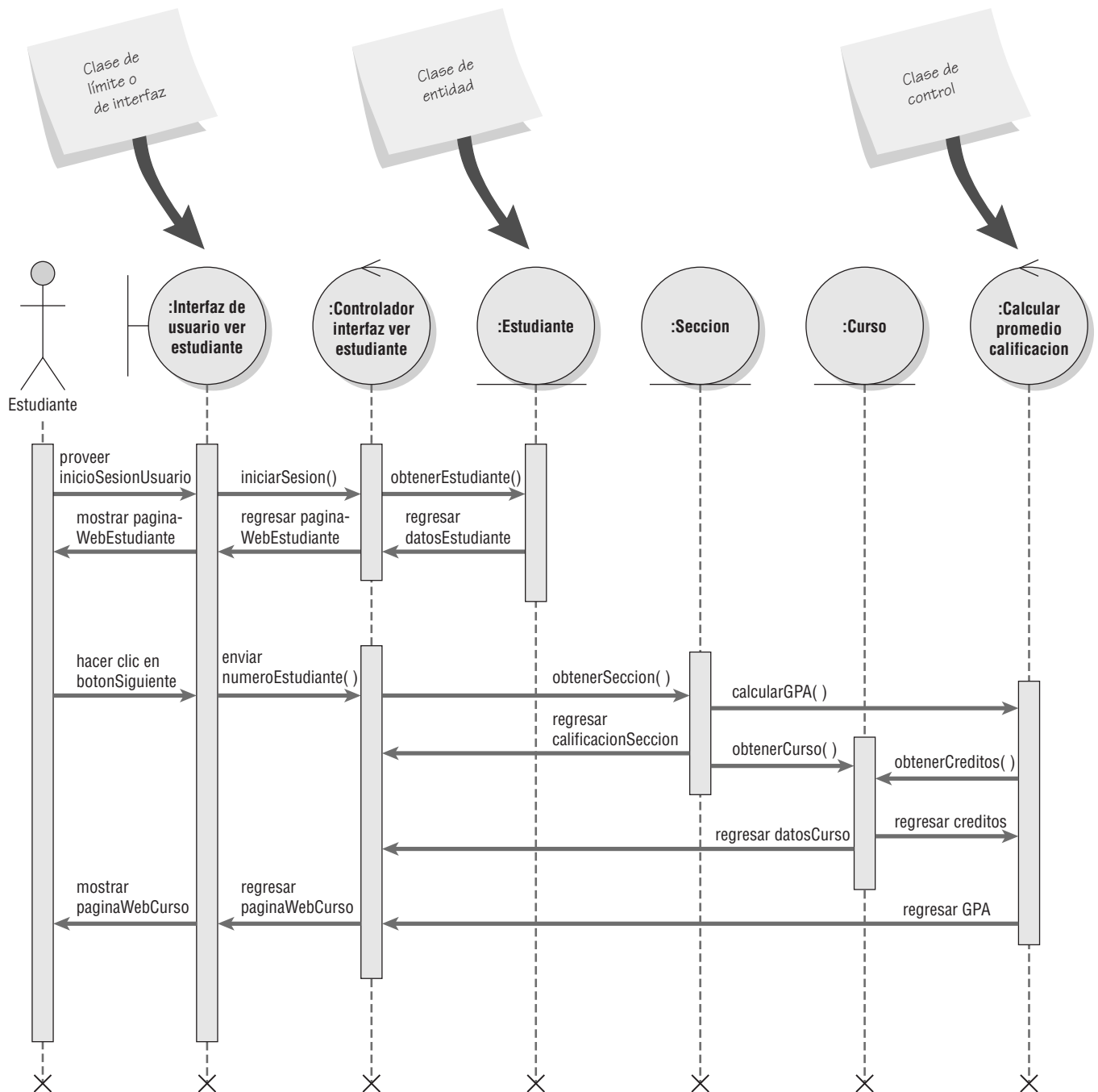


FIGURA 10.15

Un diagrama de secuencia para usar dos páginas Web: una para la información del estudiante y la otra para la información del curso.

**diante()** de nuevo, lo cual implicaría un insatisfactorio tecleo redundante. Cabe mencionar que la clase **:Estudiante** no está involucrada y que el foco de control (la barra vertical conectada a la clase **:Estudiante**) termina antes de que empiece el segundo conjunto de actividades (las flechas horizontales que apuntan a la derecha).

La clase **:Controlador interfaz ver estudiante** envía un mensaje **obtenerSeccion()** a la clase **:Seccion**, la cual devuelve una **calificacionSeccion**. La clase **:Seccion** también envía un mensaje **calcularGPA()** a la clase **:Calcular promedio calificacion**, la cual envía un mensaje de vuelta a la clase **:Curso**. La clase **:Curso** devuelve los **creditos** que permiten a la clase **:Calcular promedio calificacion** determinar el GPA y devolverlo al **:Controlador interfaz ver estudiante**.

El **:Controlador interfaz ver estudiante** repite el envío de mensajes a la clase **:Seccion** hasta que se hayan incluido todas las secciones para el estudiante. En este momento, **:Controlador interfaz ver estudiante** envía la **paginaWebCurso** a la clase **:Interfaz usuario ver estudiante**, que muestra la información en el navegador.

Al utilizar las clases de interfaz de usuario, de control y de entidad, el analista puede explorar el diseño y experimentar con él. El diseño antes mencionado mostraría toda la información personal del estudiante en una página y la información del curso en una segunda página. El analista puede modificar el diseño de manera que la información personal del estudiante y la información del curso aparezcan en una página Web. Estos dos posibles escenarios se revisarían con los usuarios para determinar la mejor opción.

Una de las dificultades para el analista es determinar cómo incluir el **numeroEstudiante** después de hacer clic en el botón **Siguiente**, ya que la clase **:Estudiante** no está disponible en ese momento. Hay tres formas de almacenar y retransmitir los datos desde una página Web:

1. Incluir la información en el URL que se muestra en el área de dirección o ubicación del navegador. En este caso, la línea de la ubicación podría mostrar algo así: `http://www.cpu.edu/estudiante/consultestud.html?numeroEstudiante=12345`

Todo lo que va después del signo de interrogación son datos que los métodos de clase pueden utilizar. Este medio de almacenar los datos es fácil de implementar y se utiliza con frecuencia en los motores de búsqueda.

Hay varias desventajas en cuanto a usar este método, por lo que el analista debe tener cuidado. La primera preocupación es la privacidad; cualquiera puede leer la dirección Web. Si la aplicación involucra información médica, números de tarjeta de crédito, etcétera, ésta no es una buena opción. La mayoría de los navegadores también muestran los datos de direcciones Web anteriores en sesiones posteriores si el usuario introduce los primeros caracteres, por lo que la información se puede ver comprometida, dando pie al robo de identidad. Una segunda desventaja es que por lo general los datos se pierden una vez que el usuario cierra el navegador.

2. Almacenar la información en una cookie: un pequeño archivo almacenado en la computadora cliente (navegador). Las cookies constituyen la única forma de almacenar datos para que persistan más allá de la sesión actual del navegador. Esto permite a la página Web mostrar un mensaje tal como “Bienvenido de vuelta, Robin. Si no es usted Robin, haga clic aquí”. Por lo general las cookies almacenan números de cuenta de claves primarias pero no números de tarjetas de crédito u otro tipo de información privada. Las cookies están limitadas a 20 por dominio (como `www.cpu.edu`) y cada cookie debe ser de 4,000 caracteres o menor.

El analista debe trabajar con otras unidades de negocios para determinar quién necesita usar cookies y debe haber cierto control central sobre los nombres utilizados en las cookies. Si la organización necesita tener más de 20 cookies, una solución común es crear distintos nombres de dominio utilizados por la organización, como `soporte.cpu.edu` o `capacitacion.cpu.edu`.

3. Use campos ocultos en los formularios Web. Por lo general estos campos contienen datos que envía el servidor, son invisibles y no ocupan espacio en la página Web. En el ejemplo sobre ver información del estudiante, la clase **:Controlador interfaz ver estudiante** agregó un campo oculto que contiene el **numeroEstudiante** al formulario **paginaWebEstudiante** junto con el **botonSiguiente**. Cuando el estudiante hace clic en el **botonSiguiente**, se envía el **numeroEstudiante** al servidor y el **:Controlador interfaz ver estudiante** sabe para cuál estudiante debe obtener la información sobre el curso y la calificación. Los datos en formularios ocultos no se guardan de una sesión a otra del navegador, por lo que se mantiene la privacidad.

## Las capas de presentación, negocios y persistencia en los diagramas de secuencia

En el ejemplo anterior mostramos todas las clases en el mismo diagrama. Al escribir código para sistemas es útil analizar los diagramas de secuencia como si tuvieran tres capas:

1. La capa de presentación, que representa lo que ve el usuario. Esta capa contiene las clases de interfaz o de límite.
2. La capa de negocios, que contiene las reglas únicas para esta aplicación. Esta capa contiene las clases de control.
3. La capa de persistencia o acceso a los datos, que describe la forma de obtener y almacenar los datos. Esta capa contiene las clases de entidad.

Lo ideal sería escribir código de programa separado para cada una de estas capas.

Con la introducción de Ajax, las líneas se volvieron borrosas. Ajax, un acrónimo para JavaScript y XML asíncrono, es una colección de técnicas que permiten a las aplicaciones Web recuperar información del servidor



sin alterar la visualización de la página actual. Esto resulta ser una ventaja debido a que no hay que volver a cargar toda la página Web si recibimos datos adicionales del servidor.

Antes de la creación de Ajax, el usuario que visitaba una página Web respondía algunas preguntas mediante la introducción de datos en un formulario basado en Web y después tenía que esperar hasta que se cargara una nueva página. Esto era necesario debido a que el código para validar, obtener los datos y después responder al usuario residía en el servidor. Con la llegada de Ajax, la página Web se actualiza con rapidez debido a que gran parte del proceso de validación y demás lógica de control se incluye ahora en el código de JavaScript del navegador o del lado del cliente. Esto significa que las reglas de negocios se incluyen tanto en la clase de límite como en la clase de control, por lo que no es posible tener tres capas.

## CÓMO MEJORAR LOS DIAGRAMAS DE CLASES

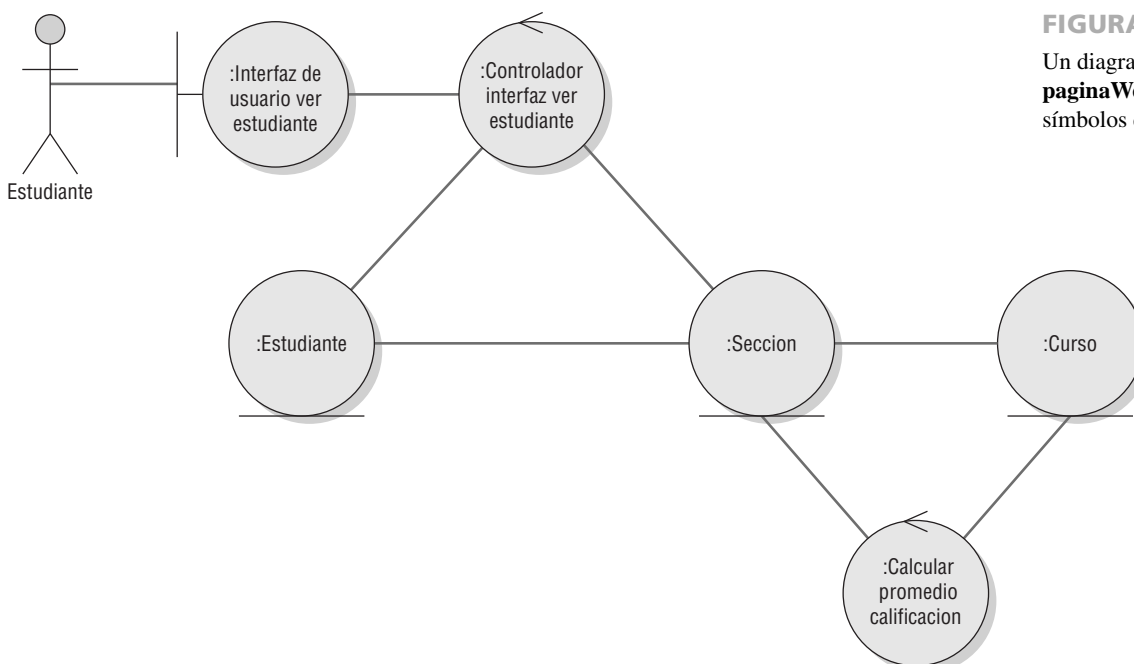
Los símbolos de clases también se pueden usar en los diagramas de clases y de comunicación. La figura 10.16 ilustra el diagrama de clases para un estudiante que ve su información personal y del curso en páginas Web. Cada clase tiene atributos y métodos (los cuales no se muestran en los diagramas que utilizan esta notación).

Si la clase es de tipo interfaz de usuario, los atributos son los controles (o campos) en la pantalla o formulario. Los métodos serían los que trabajan con la pantalla, como enviar o restablecer. También podrían ser JavaScript para una página Web, ya que el código trabaja de manera directa con la página Web.

Si la clase es una clase de control, los atributos serían aquellos que se necesitan para implementar la clase, como las variables que se utilizan sólo en la clase de control. Los métodos serían aquellos utilizados para realizar cálculos, tomar decisiones y enviar mensajes a otras clases.

Si la clase es una clase de entidad, los atributos representan a los atributos almacenados para la entidad y los métodos que trabajan directamente con la entidad, como crear una instancia, modificar, eliminar, obtener o imprimir.

Los sitios Web pueden usar una combinación de muchas clases para cumplir con los objetivos de usuario. Por ejemplo, un sitio Web puede usar JavaScript para realizar una validación previa de los datos y después pasarlos a las clases de control del servidor que realizan una validación detallada, incluyendo la obtención de los datos. A su vez, las clases de control del servidor pueden enviar JavaScript de vuelta a la página Web para aplicar cierto formato. Es muy común que una aplicación Web involucre a muchas clases, algunas de las cuales contienen sólo una línea de código, en un método para poder lograr el objetivo de la reutilización.



**FIGURA 10.16**

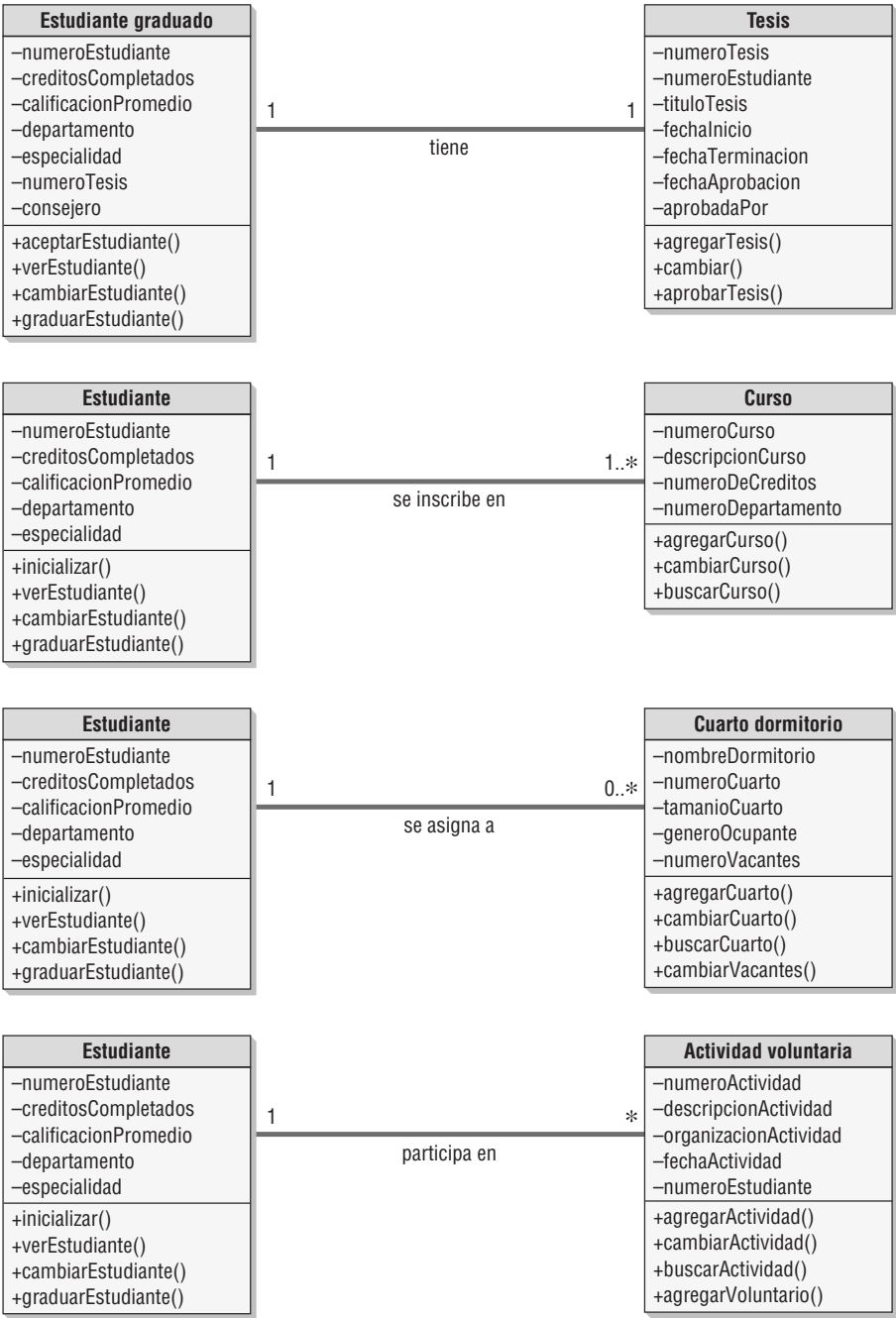
Un diagrama de clases para la **páginaWebEstudiante** que utiliza símbolos de clase especiales.

Relaciones

Otra manera de mejorar los diagramas de clases es mostrar las relaciones. Éstas son conexiones entre las clases, de manera similar a las que se encuentran en un diagrama de entidad-relación. Las relaciones se muestran como líneas que conectan a las clases en un diagrama de clases. Hay dos categorías de relaciones: asociaciones y relaciones entre un todo y sus partes.

**ASOCIACIONES** El tipo más simple de relación es una asociación, o conexión estructural entre clases u objetos. Las asociaciones se muestran como una simple línea en un diagrama de clases. Los puntos finales de la línea se etiquetan con un símbolo que indica la multiplicidad, que es lo mismo que la cardinalidad en un diagrama de entidad-relación. Un cero representa ninguno, un uno representa uno y sólo uno; un asterisco representa muchos. La notación 0..1 representa de cero a uno y la notación 1..\* representa de uno a muchos. En la figura 10.17 se muestran las asociaciones.

**FIGURA 10.17**  
Tipos de asociaciones que pueden ocurrir en los diagramas de clases.



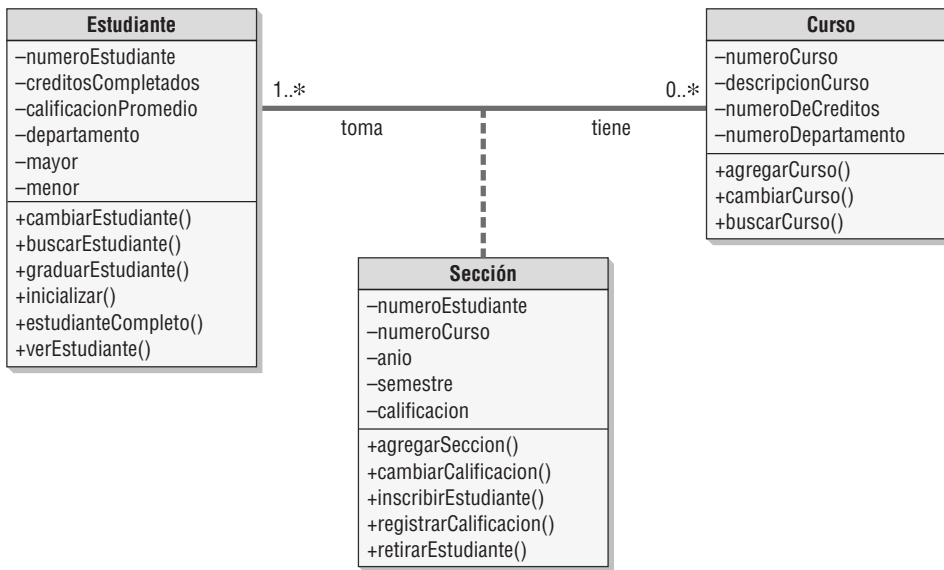


FIGURA 10.18

Un ejemplo de una clase asociativa en la que una sección específica define la relación entre un estudiante y un curso.

Los diagramas de clases no restringen el límite inferior para una asociación. Por ejemplo, una asociación podría ser 5..\*, lo cual indica que debe haber un mínimo de 5 presentes. Lo mismo se aplica para los límites superiores. Por ejemplo, el número de cursos en los que está inscrito actualmente un estudiante podría ser 1..10, lo cual representa de 1 a 10 cursos. También se puede incluir un rango de valores separados por comas, como 2, 3, 4. En el modelo de UML, las asociaciones por lo general se etiquetan con un nombre descriptivo.

Las clases de asociaciones son aquellas que se utilizan para descomponer una asociación de muchos a muchos entre las clases. Son similares a las entidades asociativas en un diagrama de entidad-relación. **Estudiante** y **Curso** tienen una relación de muchos a muchos; para resolver esta relación se agrega una clase de asociación llamada **Seccion** entre las clases de **Estudiante** y **Curso**. La figura 10.18 muestra una clase de asociación llamada **Seccion** con una línea punteada conectada a la línea de la relación de muchos a muchos.

Un objeto en una clase puede tener una relación con otros objetos en la misma clase; a esto se le denomina asociación reflexiva. Un ejemplo sería una tarea que tenga una tarea precedente, o un empleado que supervisa a otro empleado. Esto se muestra como una línea de asociación que conecta a la clase consigo misma, con etiquetas que indican los nombres de los roles, como tarea y tarea precedente.

**RELACIONES ENTRE UN TODO Y SUS PARTES** Estas relaciones se dan cuando una clase representa al objeto como un todo y otras clases representan partes de ese objeto. El todo actúa como un contenedor para las partes. Para mostrar estas relaciones en un diagrama de clases se utiliza una línea con un diamante en un extremo. El diamante está conectado al objeto que es el todo. Las relaciones entre un todo y sus partes (así como la agregación, que veremos más adelante) se muestran en la figura 10.19.

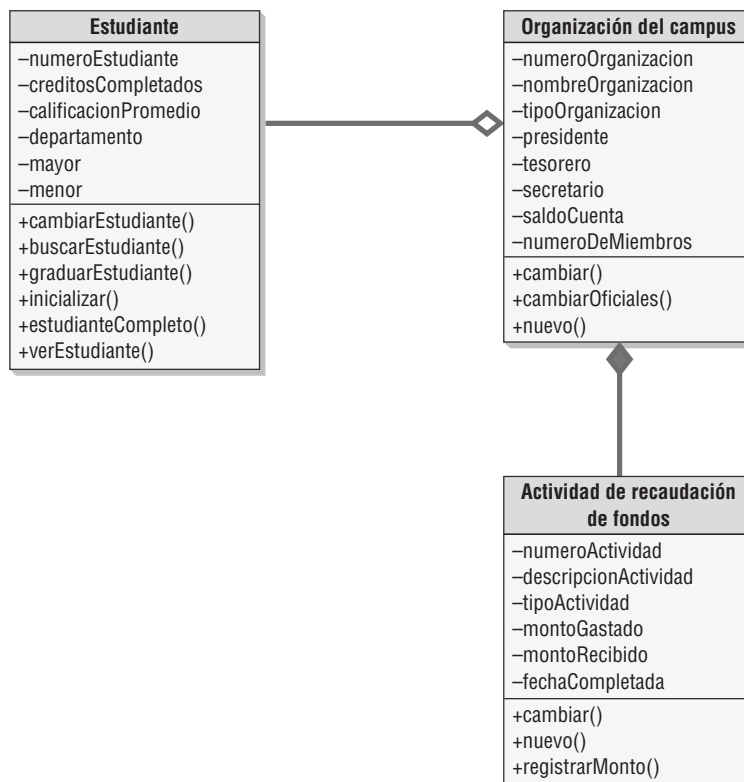
Una relación entre un todo y sus partes puede ser un objeto entidad que tenga partes distintas, como un sistema computarizado que incluye computadora, impresora, pantalla, etc., o un automóvil que tiene un motor, sistema de frenos, transmisión, etc. También es posible usar relaciones entre un todo y sus partes para describir una interfaz de usuario en la que una pantalla de GUI contiene una serie de objetos como listas, cuadros o botones de opción, o tal vez un encabezado, cuerpo y área del pie. Las relaciones entre un todo y sus partes tienen tres categorías: agregación, colección y composición.

**Agregación** Una agregación se describe comúnmente como una relación “tiene un”. La agregación provee el medio para mostrar que todo el objeto está compuesto de la suma de sus partes (otros objetos). En el ejemplo de inscripción de estudiantes, el departamento *tiene un* curso y el curso *es para un* departamento. Ésta es una relación más débil, ya que tal vez el departamento cambie o se elimine y el curso puede seguir existiendo. Un paquete de computadora tal vez ya no esté disponible, pero las impresoras y los demás componentes siguen existiendo. El diamante al final de la línea de la relación no está relleno.

**Colección** Una colección consiste de un todo y sus miembros. Éste puede ser un distrito de votación con votantes o una biblioteca con libros. Los votantes o libros pueden cambiar, pero el todo retiene su identidad. Ésta es una asociación débil.

**FIGURA 10.19**

Un ejemplo de relaciones entre un todo y sus partes y de agregación.



**Composición** La composición es una relación entre un todo y sus partes, donde el todo tiene una responsabilidad por cada parte. Es una relación más fuerte y por lo general se muestra con un diamante relleno. Las palabras clave para la composición son una clase “siempre contiene” a otra clase. Si el todo se elimina, todas las partes se eliminan. Un ejemplo sería una política de seguros con cláusulas adicionales. Si se cancela la póliza también se cancelan las cláusulas adicionales del seguro. En una base de datos se establecería la integridad referencial para eliminar los registros hijos en cascada. En una universidad hay una relación de composición entre un curso y una asignación, así como entre un curso y un examen. Si se elimina el curso se eliminan también las asignaciones y los exámenes.

### Diagramas de generalización/especialización (gen/spec)

Un diagrama de generalización/especialización (gen/spec) se puede considerar un diagrama de clases mejorado. Algunas veces es necesario separar las generalizaciones de las instancias específicas. Como vimos al principio de este capítulo, un oso koala es parte de una clase de marsupiales, que a su vez es parte de una clase de animales. Algunas veces necesitamos diferenciar si un oso koala es un animal o si es un tipo de animal. Además, un oso koala puede ser un animal de peluche. Por lo tanto, a menudo es necesario aclarar estas sutilezas.

**GENERALIZACIÓN** Una generalización describe una relación entre un tipo general de cosa y un tipo más específico de cosa. Este tipo de relación se describe comúnmente como una relación “es un”. Por ejemplo, un auto *es un* vehículo y un camión *es un* vehículo. En este caso, el vehículo es la cosa general, mientras que auto y camión son las cosas más específicas. Las relaciones de generalización se utilizan para modelar la herencia y la especialización de las clases. A una clase general se le conoce algunas veces como superclase, clase base o clase padre; a una clase especializada se le llama subclase, clase derivada o clase hija.

**HERENCIA** Varias clases pueden tener los mismos atributos y/o métodos. Cuando esto ocurre, se crea una clase general que contiene los atributos y métodos comunes. La clase especializada recibe o hereda los atributos y métodos de la clase general. Además, la clase especializada tiene atributos y métodos únicos que sólo se definen en ella. Al crear clases generalizadas y permitir que la clase especializada herede los atributos y métodos ayudamos a fomentar la reutilización, ya que el código se utiliza muchas veces. Esto también ayuda a mantener el código existente del programa. Así, el analista puede definir los atributos y métodos una vez, pero usarlos muchas veces en cada clase heredada.

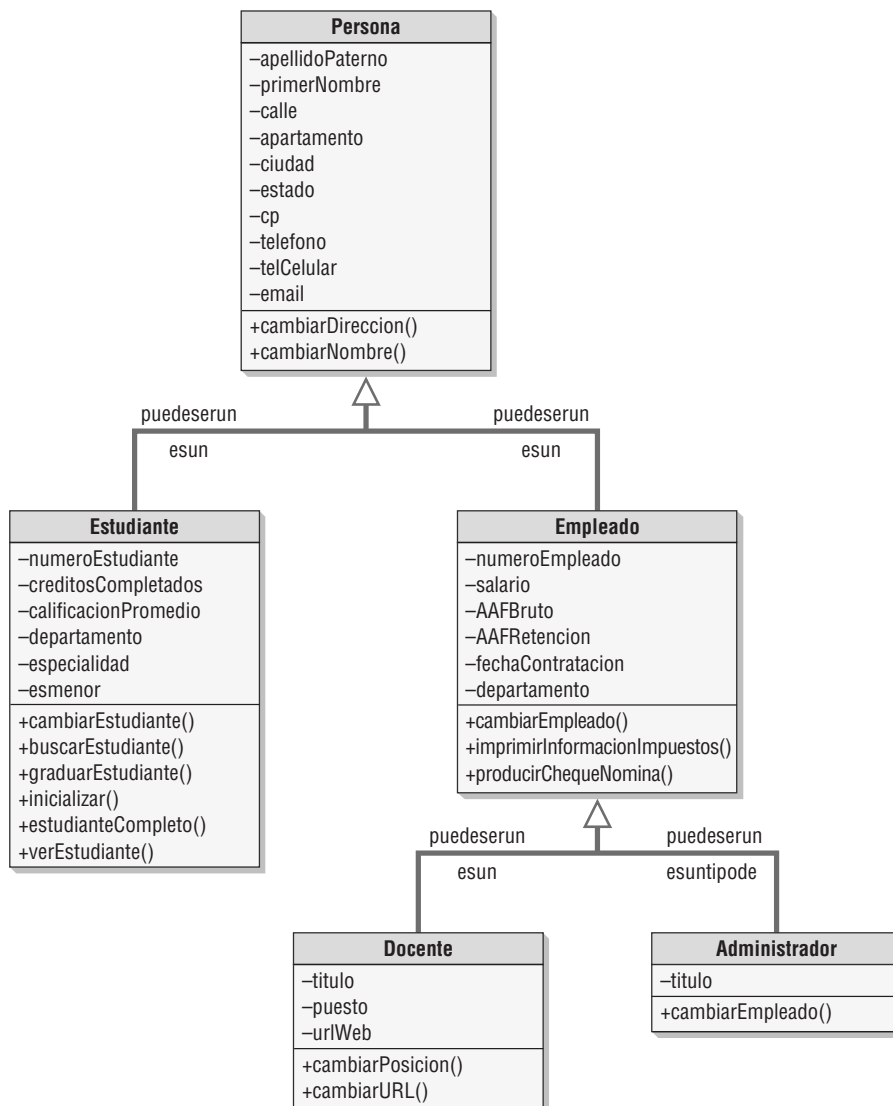
Una de las características especiales de la metodología orientada a objetos es la creación y el mantenimiento de extensas bibliotecas de clases disponibles en muchos lenguajes. Por ejemplo, un programador que utilice Java, .NET o C# tendrá acceso a una enorme cantidad de clases que ya se han desarrollado.

**POLIMORFISMO** El polimorfismo (que significa muchas formas) o redefinición de métodos (no es lo mismo que sobrecarga de métodos) es la capacidad de un programa orientado a objetos de tener varias versiones de un mismo método con el mismo nombre dentro de una relación superclase/subclase. La subclase hereda un método padre pero le puede agregar elementos o modificarlo. La subclase puede cambiar el tipo de los datos, o puede cambiar la forma en que trabaja el método. Por ejemplo, podría haber un cliente que reciba un descuento adicional por volumen, con lo que se usaría un método modificado para calcular el total del pedido. Se dice que el método de la subclase redefine al método de la superclase.

Cuando los atributos o métodos se definen más de una vez, se utiliza el más específico (el más bajo en la jerarquía de clases). El programa compilado recorre la cadena de clases en busca de los métodos.

**CLASES ABSTRACTAS** Las clases abstractas son clases generales y se utilizan cuando se incluye la generalización/especificación (gen/spec) en el diseño. La clase general se convierte en la clase abstracta. La clase abstracta no tiene objetos directos o instancias de clase, y se utiliza sólo en conjunto con clases especializadas. Por lo general las clases abstractas tienen atributos y pueden tener unos cuantos métodos.

La figura 10.20 es un ejemplo de un diagrama de clases de generalización/especificación. La flecha apunta a la clase general, o superclase. A menudo las líneas que conectan dos o más subclases con una superclase se



**FIGURA 10.20**

Un diagrama de generalización/especificación es una forma refinada de un diagrama de clases.

unen mediante una flecha que apunta a la superclase, pero también se podrían mostrar como flechas separadas. Observe que el nivel superior es *Persona*, la clase que representa a cualquier persona. Los atributos describen cualidades que poseen todas las personas en la universidad. Los métodos permiten a la clase cambiar el nombre y la dirección (incluyendo el teléfono y la dirección de correo electrónico). Ésta es una clase abstracta, sin instancias.

Estudiante y Empleado son subclases ya que tienen distintos atributos y métodos. Un empleado no tiene una calificación promedio y un estudiante no tiene un salario. Ésta es una versión simple y no incluye a los empleados que son estudiantes ni a los estudiantes que trabajan para la universidad. Si se agregaran, serían subclases de las clases **Empleado** y **Estudiante**. **Empleado** tiene dos subclases, **Docente** y **Administrador**, ya que hay distintos atributos y métodos para cada una de estas clases especializadas.

Las subclases cuentan con verbos especiales para definirlos. A menudo son palabras seguidas, como *esun* para “es un”, *esuntipode* para “es un tipo de” y *puedeserun* para “puede ser un”. No hay distinción entre “es un” y “es una”; ambas utilizan *esun*.

|                   |  |
|-------------------|--|
| <i>esun</i>       | Docente <i>esun</i> Empleado             |
| <i>esuntipode</i> | Administrador <i>esuntipode</i> Empleado |
| <i>puedeserun</i> | Empleado <i>puedeserun</i> Docente       |

**IDENTIFICAR CLASES ABSTRACTAS** Tal vez pueda identificar las clases abstractas si encuentra varias clases o tablas de base de datos que tengan los mismos elementos, o si varias clases tienen los mismos métodos. Podemos crear una clase general al extraer los atributos y métodos comunes, o podríamos crear una clase especializada para los atributos y métodos únicos. Si usamos un ejemplo bancario como un retiro, un pago sobre un préstamo o la escritura de un cheque, todos tendrán el mismo método: restar dinero del saldo del cliente.

**BUSCAR CLASES** Hay varias formas de determinar las clases. Se pueden descubrir durante las sesiones de entrevista o JAD (que vimos en el capítulo 4), durante las sesiones facilitadas del equipo o a partir de las sesiones de lluvia de ideas. Al analizar documentos y memos también se pueden revelar clases. Una de las formas más sencillas es usar el método CRC que vimos antes en este capítulo. El analista también debe examinar los casos de uso en busca de sustantivos. Cada sustantivo puede generar un candidato a una clase potencial. Se les llama clases candidatas debido a que algunos de los sustantivos pueden ser atributos de una clase.

Cada clase debe existir para un objeto distinto que tenga una definición clara. Pregúntese qué conoce la clase, los atributos; y qué es lo que la clase sabe cómo hacer, los métodos. Identifique las relaciones entre clases y la multiplicidad para cada extremo de la relación. Si la relación es de muchos a muchos, cree una clase de intersección o asociativa, similar a la entidad asociativa en un diagrama de entidad-relación.

**DETERMINAR LOS MÉTODOS DE LAS CLASES** El analista debe determinar los atributos y métodos de las clases. Los atributos son fáciles de identificar, pero los métodos que trabajan con los atributos pueden ser más difíciles de identificar. Algunos de los métodos son estándar y siempre están asociados con una clase, como *new()*, o el método *<<crear>>*, que es una extensión para el UML creada por una persona u organización, a lo cual se le conoce como estereotipo. Los símbolos *<< y >>* no son simplemente pares de símbolos mayor que y menor que, sino que se les conoce como paréntesis angulares.

Otra manera útil de determinar los métodos es examinar una matriz CRUD (vea el capítulo 7). La figura 10.21 muestra una matriz CRUD para los ofrecimientos de cursos. Cada letra requiere un método distinto. Si hay una C para crear, se agrega un método *new()*. Si hay una U para actualizar, se agrega un método *actualizar()* o *cambiar()*. Si hay una D para eliminar, se agrega un método *eliminar()* o *quitar()*. Si hay una R para leer, se agregan métodos para buscar, ver o imprimir. En el ejemplo mostrado, la clase **librotexto** necesitaría un método crear para agregar un libro de texto y un método leer para iniciar una consulta sobre un curso, cambiar un libro de texto o buscar un libro de texto. Si se reemplazara un libro de texto se necesitaría un método actualizar, y si se quitara un libro de texto se requeriría un método eliminar.

**MENSAJES** Para poder lograr un trabajo útil, la mayoría de las clases necesitan comunicarse entre sí. La información se puede enviar mediante un objeto en una clase a un objeto en otra clase a través de un mensaje, en forma similar a una llamada en un lenguaje de programación tradicional. Un mensaje también actúa como comando para indicar a la clase receptora que haga algo. Un mensaje consiste en el nombre del método en la clase receptora, así como los atributos (parámetros o argumentos) que se pasan con el nombre del método. La clase receptora debe tener un método que corresponda al nombre del mensaje.

| Actividad               | Departamento | Curso | Librotexto | Asignacion | Examen |
|-------------------------|--------------|-------|------------|------------|--------|
| Agregar departamento    | C            |       |            |            |        |
| Ver departamento        | R            |       |            |            |        |
| Agregar curso           | R            | C     |            |            |        |
| Cambiar curso           | R            | U     |            |            |        |
| Consulta de curso       | R            | R     | R          | R          | R      |
| Agregar libro de texto  | R            | R     | C          |            |        |
| Cambiar libro de texto  |              | R     | RU         |            |        |
| Buscar libro de texto   |              | R     | R          |            |        |
| Eliminar libro de texto |              | R     | D          |            |        |
| Agregar asignación      |              | R     |            | C          |        |
| Cambiar asignación      |              | R     |            | RU         |        |
| Cambiar asignación      |              | R     |            | R          |        |
| Ver asignación          |              | R     |            |            | R      |
| Agregar examen          |              | R     |            |            | RU     |
| Cambiar examen          |              | R     |            |            | R      |

**FIGURA 10.21**

Se puede usar una matriz CRUD para ayudar a determinar qué métodos se necesitan. Esta matriz CRUD se utiliza para determinar los métodos y las operaciones para los ofrecimientos de cursos.

Como los mensajes se envían de una clase a otra, se pueden considerar como entrada o salida. La primera clase debe proveer los parámetros incluidos con el mensaje y la segunda clase utiliza esos parámetros. Si existe un diagrama de flujo de datos hijo físico para el dominio del problema, puede ser útil para descubrir métodos. El flujo de datos de un proceso primitivo a otro representa el mensaje y hay que examinar los procesos primitivos como candidatos a métodos.

## DIAGRAMAS DE ESTADOS

El diagrama de estados, o de transiciones de estado, es otra herramienta para determinar los métodos de las clases. Se utiliza para examinar los distintos estados que puede tener un objeto.

Se crea un diagrama de estados para una sola clase. Por lo general los objetos se crean, pasan por cambios y se eliminan o quitan.

Los objetos existen en estos diversos estados, que son las condiciones de un objeto en un momento específico. Los valores del atributo de un objeto definen el estado en que se encuentra el objeto y algunas veces hay un atributo tal como Estado del pedido (pendiente, en recolección, empaquetado, enviado, recibido, etcétera) que indica el estado. Un estado tiene un nombre en el que cada palabra empieza con mayúscula. El nombre debe ser único y descriptivo. Un estado también tiene acciones de entrada y de salida: las cosas que el objeto debe hacer cada vez que entra o sale de un estado específico.

Un evento es algo que ocurre en un tiempo y lugar específicos. Los eventos provocan un cambio del estado del objeto y se dice que una transición se “dispara”. Los estados separan eventos, como un pedido que espera a ser llenado, y los eventos separan estados, como un evento Pedido recibido o un evento Pedido completo.

Un evento produce la transición y ocurre cuando se cumple una condición de guardia. Esta condición de guardia es algo que se evalúa como verdadero o falso y puede ser tan simple como “Hacer clic para confirmar el pedido”. También puede ser una condición que ocurra en un método, como un elemento que esté agotado. Las condiciones de guardia se muestran entre corchetes a un lado de la etiqueta del evento.



Además hay eventos diferidos, o eventos que se retienen hasta que un objeto cambia a un estado que pueda aceptarlos. Un usuario que teclea algo cuando un procesador de palabras realiza un respaldo sincronizado es un ejemplo de un evento diferido. Una vez que se completa el respaldo sincronizado, el texto aparece en el documento.

Los eventos se clasifican en tres categorías distintas:

- 1. Señales o mensajes asíncronos, que ocurren cuando el programa que hace la llamada no espera un mensaje de retorno, como una característica que se opera desde un menú.
- 2. Mensajes sincrónicos, que son llamadas a funciones o subrutinas. El objeto que hace la llamada se detiene y espera a que se le regrese el control, junto con un mensaje opcional.
- 3. Eventos temporales, que ocurren en un tiempo predeterminado. Por lo general no involucran a un actor o a un evento externo.

Los objetos materiales tienen persistencia; es decir, existen durante un extenso periodo de tiempo. Los vuelos de aviones, conciertos y eventos deportivos tienen una persistencia más corta (pueden tener estados que cambian en un tiempo más corto). Algunos objetos, conocidos como objetos transitorios, no sobreviven al final de una sesión. Éstos incluyen la memoria principal, los datos de una URL (o ubicación) Web, las páginas Web, pantallas CICS, etcétera. La única forma de guardar los objetos transitorios es almacenar la información sobre ellos, como cuando se guardan los datos Web en una cookie.

Cada vez que un objeto cambia de estado, algunos de los atributos cambian sus valores. Además, cada vez que cambian los atributos de un objeto, debe haber un método para cambiar esos atributos. Cada uno de los métodos necesitaría una pantalla o formulario Web para agregar o modificar los atributos. Éstos se convierten en los objetos de interfaz. Comúnmente la pantalla o formulario Web contiene más controles (o campos) que los atributos que cambian. Por lo general tienen claves primarias, información de identificación (como un nombre o dirección) y otros atributos necesarios para una buena interfaz de usuario. La excepción es un evento temporal, que puede usar tablas de la base de datos o una cola para contener la información.

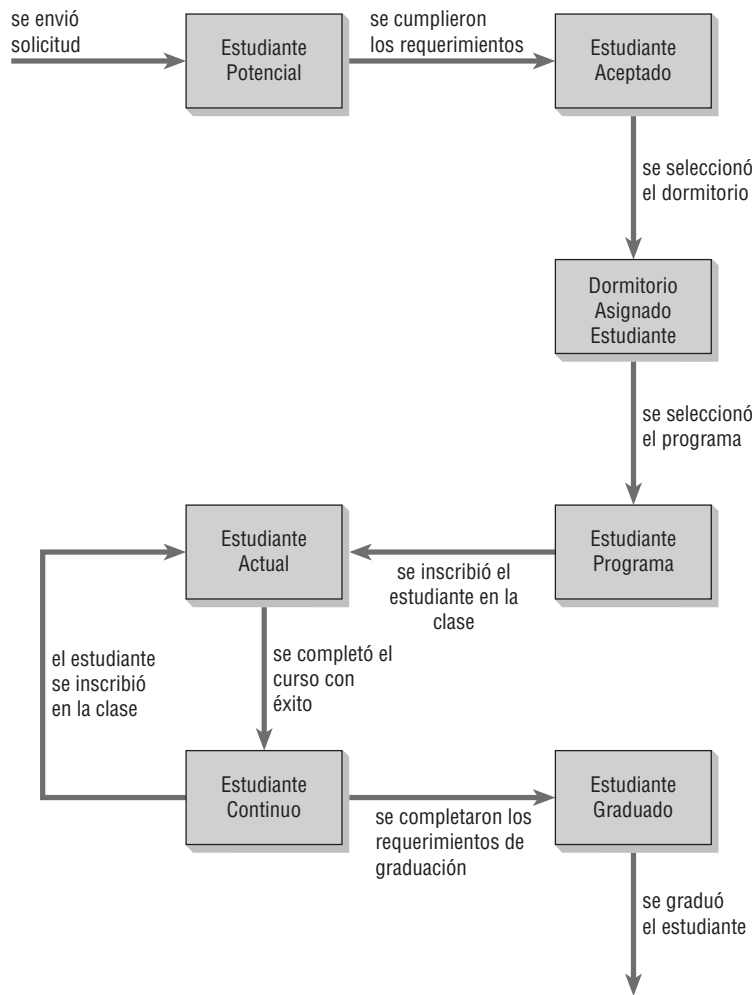
**Ejemplo de una transición de estado**

Veamos el ejemplo de un estudiante que se inscribe en una universidad y los diversos estados por los que pasará. A continuación le mostramos una lista detallada de tres de esos estados:

|                               |  |
|-------------------------------|--|
| <b>Estado:</b>                | Estudiante potencial   |
| <b>Evento:</b>                | Aplicación enviada   |
| <b>Método:</b>                | new()  |
| <b>Atributos modificados:</b> | Numero<br>Nombre<br>Direccion  |
| <b>Interfaz de usuario:</b>   | Formulario Web Solicitud de estudiante                                     |
| <b>Estado:</b>                | Estudiante Aceptado  |
| <b>Evento:</b>                | Requerimientos cumplidos   |
| <b>Método:</b>                | aceptarEstudiante()  |
| <b>Atributos modificados:</b> | Fecha de admisión<br>Estado del estudiante<br>Devolver carta de aceptación |
| <b>Interfaz de usuario:</b>   | Pantalla Aceptar estudiante  |
| <b>Estado:</b>                | Dormitorio Asignado Estudiante   |
| <b>Evento:</b>                | Dormitorio seleccionado  |
| <b>Método:</b>                | asignarDormitorio()  |
| <b>Atributos modificados:</b> | Nombre del dormitorio<br>Cuarto del dormitorio<br>Plan de comidas          |
| <b>Interfaz de usuario:</b>   | Pantalla Asignar dormitorio a estudiante                                   |

Los otros estados son **Estudiante Programa**, **Estudiante Actual**, **Estudiante Continuo** y **Estudiante Graduado**. A cada estado hay que asociar un evento, métodos, atributos modificados y una interfaz de usuario. Podemos usar esta serie de estados para determinar los atributos y métodos que forman parte de la clase.

Los estados y eventos que desencadenan los cambios se pueden representar en un diagrama de estados (o en un diagrama de transiciones de estado). En la figura 10.22 se muestra el diagrama de estados para **Estudiante**.

**FIGURA 10.22**

Un diagrama de estados donde se muestra cómo un estudiante progresa de estudiante potencial a estudiante graduado.

Los estados se representan mediante rectángulos y los eventos o actividades son las flechas que enlazan los estados y hacen que un estado cambie a otro estado. Los eventos de transición se nombran en pasado, porque ya ocurrieron para crear la transición.

No hay que crear diagramas de estados para todas las clases. Se crean cuando:

1. Una clase tiene un ciclo de vida complejo.
2. Una instancia de una clase puede actualizar sus atributos en varias formas durante el ciclo de vida.
3. Una clase tiene un ciclo de vida operacional.
4. Dos clases dependen una de la otra.
5. El comportamiento actual del objeto depende de lo que ocurrió antes.

Al examinar un diagrama de estados, aproveche la oportunidad para buscar errores y excepciones. Inspeccione el diagrama para detectar si los eventos ocurren en un tiempo inadecuado. Verifique también que se hayan representado todos los eventos y estados. En los diagramas de estados sólo hay que evitar dos problemas: verifique que no entren todas las transiciones en un estado o que salgan todas de éste.

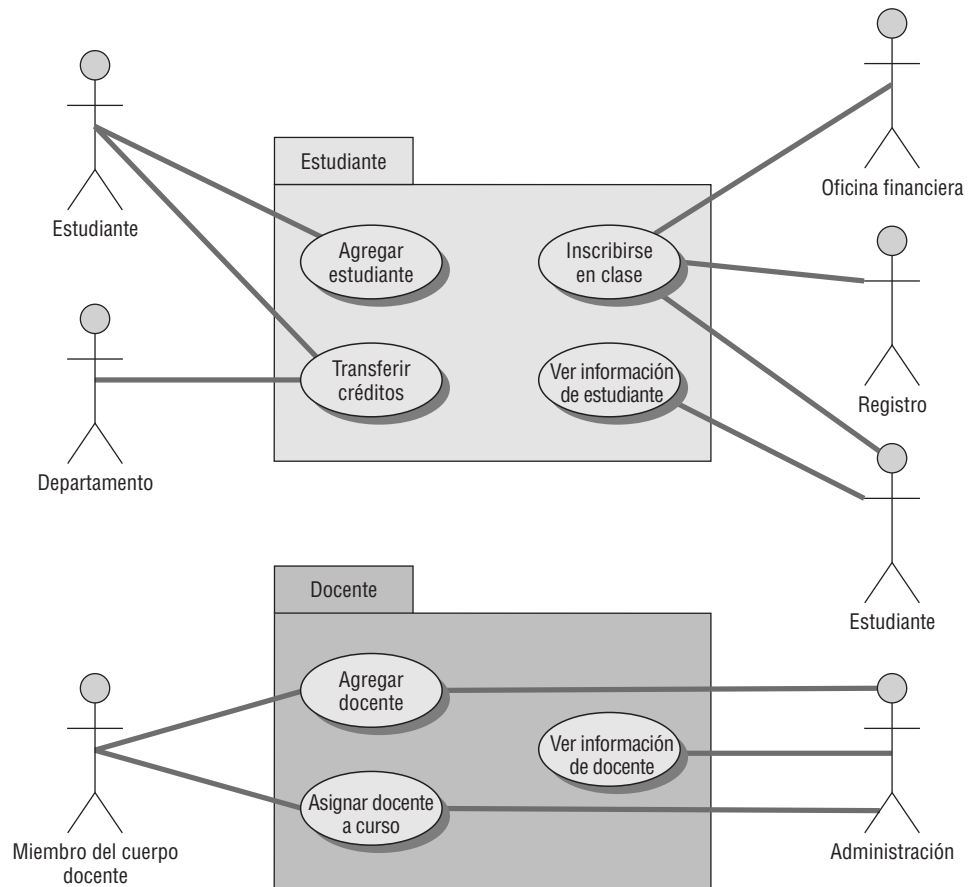
Cada estado debe tener por lo menos una transición de entrada y de salida. Algunos diagramas de estados usan los mismos símbolos de inicio y terminación que los que utiliza un diagrama de actividad: un círculo relleno para representar el inicio y círculos concéntricos con el centro relleno para indicar el fin del diagrama.

## PAQUETES Y OTROS ARTEFACTOS DE UML

Los paquetes son contenedores para otras cosas de UML, como los casos de uso o las clases. Los paquetes pueden mostrar el particionamiento del sistema, para indicar qué clases o casos de uso se agrupan en un subsistema, a lo cual se le denomina paquetes lógicos. También pueden ser paquetes componentes, que contienen componen-

FIGURA 10.23

Los casos de uso se pueden agrupar en paquetes.



tes físicos del sistema, o paquetes de casos de uso que contienen un grupo de casos de uso. Los paquetes usan un símbolo de carpeta con el nombre del paquete en la ficha de la carpeta o centrado en la misma. El empaquetamiento puede ocurrir durante el análisis del sistema o en una etapa posterior de diseño del sistema. Los paquetes también pueden tener relaciones de manera similar a los diagramas de clases, que pueden incluir asociaciones y herencia.

La figura 10.23 es un ejemplo de un diagrama de paquetes de casos de uso. Muestra que cuatro casos de uso, **Agregar estudiante**, **Inscribirse en clase**, **Transferir créditos** y **Ver información de estudiante** son parte del paquete **Estudiante**. Hay tres casos de uso, **Agregar docente**, **Ver información de docente** y **Asignar docente a curso** que forman parte del paquete **Docente**.

A medida que continúe construyendo diagramas le será conveniente utilizar los diagramas de componentes, los diagramas de despliegue y las cosas de anotaciones. Éstos permiten distinta perspectivas sobre el trabajo a realizar.

El diagrama de componentes es similar a un diagrama de clases, sólo que es más como una vista superficial de la arquitectura del sistema. El diagrama de componentes muestra los componentes del sistema, como un archivo de clase, un paquete, las bibliotecas compartidas, una base de datos, etcétera, y la forma en que se relacionan entre sí. Los componentes individuales en un diagrama de componentes se consideran con más detalle dentro de otros diagramas de UML, como los diagramas de clases y los diagramas de casos de uso.

El diagrama de despliegue ilustra la implementación física del sistema, incluyendo el hardware, las relaciones entre el hardware y el sistema en el que se va a desplegar. El diagrama de despliegue puede mostrar los servidores, estaciones de trabajo, impresoras, etcétera.

Las cosas de anotaciones proveen a los desarrolladores más información sobre el sistema. Estas cosas consisten en notas que se pueden unir a cualquier cosa en UML: objetos, comportamientos, relaciones, diagramas o cualquier cosa que requiera descripciones detalladas, suposiciones o cualquier información relevante para el diseño y la funcionalidad del sistema. El éxito del UML depende de la documentación completa y precisa de nuestro modelo del sistema para proveer toda la información que sea posible al equipo de desarrollo. Las notas proveen una fuente de conocimiento y comprensión común sobre su sistema para ayudar a que sus desarrolladores estén coordinados. Las notas se muestran como un símbolo de un papel con una esquina doblada y una línea que las conecta con el área que necesita más detalles.



## OPORTUNIDAD DE CONSULTORÍA 10.3

### Desarrollo de un sistema fino cuyo plazo de entrega se venció hace mucho: uso del análisis orientado a objetos para el Sistema de la biblioteca pública Ruminski\*

Mientras Dewey Dezmal entra a la sala de lectura de techos altos y paneles de madera de la Biblioteca Pública Ruminski, una joven sentada en una larga mesa de roble asoma su cabeza por encima de un monitor, lo ve y al pararse dice: “Bienvenido, soy Peri Otticle, directora de la biblioteca. Entiendo que está usted aquí para ayudarnos a desarrollar nuestro nuevo sistema de información”.

Intimidado aún por la belleza del antiguo edificio de la biblioteca y la yuxtaposición de tanta tecnología y tanta historia, Dewey se presenta como analista de sistemas con una pequeña empresa de consultoría de TI, People and Objects, Inc.

“Es la primera vez que me asignan a este tipo de proyecto, aunque en realidad es interesante para mí, ya que me gradué de la Information Studies School en la Upstate University. Hay especializaciones en ciencias bibliotecarias o TI, por lo que muchos de mis compañeros entraron a trabajar en bibliotecas públicas; yo opté por el grado de TI”.

“Entonces vamos a trabajar bien juntos”, dice Peri. “Vamos a mi oficina para no molestar a los usuarios y poder enseñarte un informe que escribí”.

Al pasar por la bella escalera en espiral, al parecer esculpida en madera, Peri observa que Dewey ve los alrededores y dice: “Tal vez te preguntes sobre la magnificencia del edificio, ya que somos una institución pública. Somos afortunados. Nuestro benefactor es Valerian Ruminski. De hecho, ha donado tanto dinero a tantas bibliotecas que el personal lo llama afectuosamente “Valerian el Bibliotecario”.

Después de pasar varios usuarios, Peri continúa: “Como puedes ver, es un lugar muy concurrido. Y sin importar nuestro antiguo entorno, no estamos detenidos en el pasado”.

Dewey lee el reporte que le entrega Peri. Una sección extensa se titula “Resumen de los principales requerimientos de los usuarios” y la lista en viñetas dice:

- Un usuario de la biblioteca que esté registrado en el sistema puede sacar libros y revistas del sistema.
- El sistema de la biblioteca debe revisar en forma periódica (por lo menos una vez a la semana) si se venció la fecha de entrega de un ejemplar de un libro o publicación especializada que un usuario haya sacado. De ser así se enviará un aviso al usuario.
- Un usuario puede reservar un libro o publicación especializada que se encuentre prestada o que esté en proceso de compra. La reservación se cancelará cuando el usuario

saque el libro o publicación especializada, o por medio de un servicio de cancelación formal.

Al tiempo que deja de ver el informe, Dewey dice a Peri: “Estoy empezando a entender los requerimientos del usuario. Veo muchas similitudes entre mi antigua biblioteca universitaria y ésta. Pero el tema que no vi que cubrieras fue el de cómo decidir qué es lo que la biblioteca debe recolectar y qué es lo que debe desechar”.

Peri sonrre y contesta: “Es una pregunta intuitiva. El personal de la biblioteca maneja la compra de nuevos libros y publicaciones especializadas para la biblioteca. Si algo es popular se compran más de dos copias. Podemos crear, actualizar y eliminar información sobre los títulos y copias de los libros y publicaciones especializadas, usuarios, préstamo de materiales y reservaciones en el sistema”.

Dewey deja de ver su libreta de anotaciones y dice: “Aún estoy un poco confundido. ¿Cuál es la diferencia entre los términos *título* y *ejemplar*?”.

Peri responde: “La biblioteca puede tener varios ejemplares de un título. El título por lo general se refiere al nombre de un libro o publicación especializada. Son los ejemplares de un libro los que se sacan de la biblioteca”.

Con base en la entrevista de Dewey con Peri y la descripción de requerimientos en su informe, así como en su propia experiencia con el uso de los servicios de bibliotecas, use UML para responder a las siguientes preguntas (*Nota:* Es importante que se asegure de que sus soluciones sean lógicas y funcionales; establezca sus suposiciones con claridad siempre que sea necesario):

1. Dibuje un diagrama de casos de uso para representar a los actores y los casos de uso en el sistema.
2. Describa los pasos para cada caso de uso (como hicimos para organizar los casos de uso).
3. Describa escenarios para los pasos. En otras palabras, cree un usuario y escriba un ejemplo del usuario a medida que recorre cada paso.
4. Desarrolle una lista de cosas.
5. Cree diagramas de secuencia para los casos de uso con base en los pasos y escenarios.
6. Complete el diagrama de clases; determine las relaciones entre las clases y defina los atributos y métodos de cada clase. Use la cosa de agrupamiento llamada paquete para simplificar el diagrama de clases.

\* Basado en un problema escrito por el Dr. Wayne Huang.

## PONGA EL UML A TRABAJAR

El UML provee un útil conjunto de herramientas para el análisis y diseño de sistemas. Al igual que con cualquier producto creado con la ayuda de herramientas, el valor de los entregables de UML en un proyecto depende de la experiencia con la que el analista de sistemas maneje las herramientas. En un principio el analista usará el conjunto de herramientas de UML para descomponer los requerimientos del sistema en un modelo de

casos de uso y en un modelo de objetos. El modelo de casos de uso describe los casos de uso y los actores. El modelo de objetos describe los objetos y sus asociaciones, además de las responsabilidades, colaboradores y atributos de los objetos.

1. Defina el modelo de casos de uso.
  - Busque los actores en el dominio del problema; revise los requerimientos del sistema y entreviste algunos expertos de negocios.
  - Identifique los eventos principales iniciados por los actores y desarrolle un conjunto de casos de uso primarios en un nivel muy alto, que describan los eventos desde la perspectiva de cada actor.
  - Desarrolle los diagramas de casos de uso para comprender cómo se relacionan los actores con los casos de uso que definirán el sistema.
  - Refine los casos de uso primarios para desarrollar una descripción detallada de la funcionalidad del sistema para cada caso de uso primario. Provea detalles adicionales al desarrollar los escenarios de casos de uso que documenten los flujos alternos de los casos de uso primarios.
  - Revise los escenarios de casos de uso con los expertos del área de negocios para verificar los procesos e interacciones. Haga las modificaciones necesarias hasta que los expertos del área de negocios estén de acuerdo en que los escenarios de los casos de uso son completos y precisos.
2. Continúe con la creación de diagramas de UML para modelar el sistema durante la fase de análisis de sistemas.
  - Derive los diagramas de actividad a partir de los diagramas de casos de uso.
  - Desarrolle los diagramas de secuencia y de comunicación a partir de los escenarios de los casos de uso.
  - Revise los diagramas de secuencia con los expertos del área de negocios para verificar los procesos y las interacciones. Haga las modificaciones necesarias hasta que los expertos del área de negocios estén de acuerdo en que los diagramas de secuencia son completos y precisos. Con frecuencia, la revisión adicional de los diagramas de secuencia gráficos provee a los expertos del área de negocios la oportunidad de reconsiderar y refinar los procesos con un detalle más atómico que la revisión de los escenarios de los casos de uso.
3. Desarrolle los diagramas de clases.
  - Busque los sustantivos en los casos de uso y haga una lista. Son objetos potenciales. Una vez que identifique los objetos, busque similitudes y diferencias en los objetos debido a sus estados o comportamiento y después cree las clases.
  - Defina las principales relaciones entre las clases. Busque las relaciones “tiene un” y “es un” entre las clases.
  - Examine los diagramas de casos de uso y de secuencias para poder determinar las clases.
  - Empezando con los casos de uso que son los más importantes para el diseño del sistema, cree los diagramas de clases que muestren las clases y relaciones que existen en los casos de uso. Un diagrama de clases puede representar a las clases y relaciones descritas en varios casos de uso relacionados.
4. Dibuje diagramas de estados.
  - Desarrolle diagramas de estados para ciertos diagramas de clase, de manera que pueda proveer un análisis más detallado del sistema en este punto. Use los diagramas de estados como ayuda para comprender los procesos complejos que no se puedan derivar por completo de los diagramas de secuencia.
  - Determine los métodos al examinar los diagramas de estados. Derive los atributos de las clases de los estados (datos) a partir de los casos de uso, expertos de las áreas de negocios y métodos de las clases. Indique si los métodos y atributos de la clase son públicos (acceso externo) o privados (acceso interno para la clase). Los diagramas de estados son en extremo útiles para modificar los diagramas de clases.
5. Empezar el diseño de sistemas; refine los diagramas de UML y utilícelos para derivar las clases y sus atributos y métodos.
  - Revise todos los diagramas de UML existentes para el sistema. Escriba especificaciones de clases para cada clase que incluyan los atributos, métodos y descripciones de la clase. Revise los diagramas de secuencia para identificar otros métodos de clases.
  - Desarrolle especificaciones de métodos que muestren con detalle los requerimientos de entrada y de salida para el método, junto con una descripción detallada del procesamiento interno del método.
  - Cree otro conjunto de diagramas de secuencia (si es necesario) para reflejar los métodos reales de las clases, además las interacciones entre las clases y las interfaces del sistema.
  - Cree diagramas de clases mediante los símbolos de clase especializados para la clase de límite o de interfaz, la clase de entidad y la clase de control.
  - Analice los diagramas de clases para derivar los componentes del sistema; es decir, clases relacionadas en función y lógica que se compilarán y desplegarán en conjunto como un archivo .DLL, .COM, un objeto, un Java Bean, un paquete, etcétera.



## OPORTUNIDAD DE CONSULTORÍA 10.4

### C-Shore++

“¡Ellos quieren que el núcleo de la interfaz de usuario de los representantes de servicio al cliente se vuelva a programar por completo!” exclama Bradley Vargo, director de desarrollo de sistemas de información en C-Shore Mutual Funds. “Hace tan sólo ocho meses completamos un proyecto de desarrollo de dos años del Sistema de representantes de servicio al cliente, CSR. Durante todo el proyecto soportamos un desfile de requerimientos cambiantes. Cada mes, los del departamento de marketing inventaban alguna característica nueva y competitiva de servicio al cliente y, en una semana, el grupo de CSR estaba aquí con grandes cambios en las especificaciones del sistema CSR. ¡Pensé que nunca terminaríamos ese proyecto! Ahora parece que tendremos que empezar un nuevo proyecto de reprogramación en un sistema que tiene menos de un año. ¡Pronosticamos una vida de 7 años para este sistema! Creo que ahora inicia la etapa de la eterna reconstrucción”.

Bradley habla con Rachael Ciupek, la analista de sistemas de aplicación en jefe responsable del sistema CSR y con Bridget Ciupek, su hermana y la programadora que escribió la mayor parte de la interfaz de usuario. “Cálmate, Bradley”, dice Rachael. “No es culpa de los niños de marketing o de CSR. La naturaleza de nuestro negocio se ha visto afectada por la competencia a ritmo veloz. Marketing no inventa estos cambios para salir del aburrimiento. A menudo responden a los nuevos servicios para el cliente basados en computadora que ofrece nuestra competencia. Tenemos que permanecer en la punta o por lo menos seguirles el ritmo, ¡de lo contrario estaremos buscando nuevos empleos!”.

“Bradley, Rachael, creo que ustedes saben mejor que nadie que la situación puede ser peor de lo que creen”, interviene Bridget. “De todas formas los programadores han estado realizando pequeñas

modificaciones en la interfaz de usuario durante los últimos ocho meses. Los usuarios de CSR han estado hablando con nosotros directamente para suplicar ayuda. Por lo general sólo quieren un pequeño cambio en una parte aislada del sistema, pero esto ha creado una carga de trabajo adicional debido a que tenemos que volver a certificar todo el sistema. Ustedes saben cómo los efectos de una pequeña modificación se pueden propagar a través de un programa extenso. Hemos facturado el tiempo para el mantenimiento del programa basándonos en el supuesto de que sólo estábamos haciendo pequeños ajustes al sistema completo. Aunque los cambios han sido graduales, en ocho meses hemos vuelto a escribir aproximadamente una cuarta parte del código de la interfaz de usuario de CSR. El trabajo no ha decaído. Es bastante estable aún”.

“Entonces lo que me quieres decir”, dice Bradley, “es que tenemos necesidades del sistema en esta área que han estado cambiando en forma constante mientras tratábamos de escribir las especificaciones, escribir el código del programa y crear un trabajo con soluciones fijas contra un problema fluido. ¿Cómo podemos escribir programas si durarán sólo unos cuantos meses sin que requieran de un costoso mantenimiento?”.

¿Cómo puede Bradley administrar un proceso de desarrollo de sistemas que ya no tenga procesos de negocios fijos o constantes como parte de su conjunto de objetivos? ¿Hay una forma en que Rachael administre los costos de mantenimiento de control y de las especificaciones cuando se pide constantemente a los programadores que modifiquen partes aisladas de un programa extenso? Tenga en cuenta que un objetivo importante es proveer un buen soporte para las necesidades de los usuarios y las estrategias de negocios de la organización.

- Desarrolle diagramas de despliegue para indicar cómo se desplegarán los componentes de su sistema en el entorno de producción.
- 6. Documente el diseño de su sistema en forma detallada. Este paso es imprescindible. Entre más completa sea la información que provea al equipo de desarrollo por medio de la documentación y los diagramas de UML, más rápido será el desarrollo y más sólido será el sistema de producción final.

## LA IMPORTANCIA DE USAR UML PARA EL MODELADO

El UML es una potente herramienta que puede mejorar en forma considerable la calidad de su análisis y diseño de sistemas, y se espera que las prácticas mejoradas se traduzcan en sistemas de mayor calidad.

Al usar el UML en forma iterativa en el análisis y el diseño podemos lograr una mejor comprensión entre el equipo de negocios y el equipo de TI en relación con los requerimientos del sistema y los procesos que deben ocurrir en el sistema para cumplir con esos requerimientos.

La primera iteración del análisis debe realizarse a un nivel muy alto para identificar los objetivos del sistema en general y validar los requerimientos a través del análisis de casos de uso. Identificar los actores y definir el modelo de casos de uso inicial son actividades que forman parte de esta primera iteración. Las iteraciones subsiguientes del análisis refinan más los requerimientos del sistema por medio del desarrollo de escenarios de casos de uso, diagramas de clases, diagramas de secuencia, diagramas de estado, etcétera. Cada iteración requiere una vista cada vez más detallada del diseño del sistema, hasta que las cosas y relaciones en el sistema se definan con claridad y precisión en los documentos de UML.



Cuando esté completo su proceso de análisis y diseño, deberá tener un conjunto preciso y detallado de especificaciones para clases, escenarios, actividades y secuencias en el sistema. En general, podemos relacionar la minuciosidad del análisis y diseño de un sistema con la cantidad de tiempo requerido para desarrollar el sistema y la calidad resultante del producto entregado.

Lo que se pasa por alto con frecuencia en el desarrollo de un nuevo sistema es que entre más progrese un proyecto, más costosas serán las modificaciones en los requerimientos de negocios de un sistema. Es mucho más fácil, rápido y mucho menos costoso modificar el diseño de un sistema mediante el uso de una herramienta CASE, o incluso en papel, durante las fases de análisis y diseño de un proyecto, que hacerlo durante la fase de desarrollo.

Por desgracia algunos empleadores tienen poca visión y creen que sólo cuando un programador o analista está codificando es cuando realmente está trabajando. Algunos empleadores asumen equivocadamente que la productividad del programador se puede juzgar tan sólo con base en la cantidad de código producido, sin reconocer que la creación de diagramas ahorra en última instancia tiempo y dinero que de otra manera se desperdiciaría si se hiciera un prototipo del proyecto sin una planificación apropiada.

En esta situación se adapta muy bien la analogía para la construcción de una casa. Aunque contratamos un constructor para ello, no vamos a querer vivir en una estructura construida sin planeación, una en la que los cuartos y características se agreguen al azar sin relación con la función o con el costo. Queremos que un constructor construya el diseño que acordamos a partir de los planos de construcción, con las especificaciones que revisaron con cuidado todas las partes involucradas. Como un miembro de un equipo de análisis dijo con tanta certeza: “Poner un proyecto en papel antes de codificarlo reducirá su costo a la larga. Es mucho más económico borrar un diagrama que modificar la codificación”.

Cuando cambian los requerimientos de negocios durante la fase de análisis, a veces es necesario volver a dibujar algunos diagramas de UML. Pero si los requerimientos de negocios cambian durante la fase de desarrollo, tal vez se requiera una cantidad considerable de tiempo y dinero para rediseñar, recodificar y reevaluar el sistema. Al confirmar su análisis y diseño en papel (en especial mediante el uso de diagramas de UML) con los usuarios que son expertos en el área de negocios, usted ayudará a asegurar que se cumpla con los requerimientos de negocios correctos cuando el sistema esté completo.

## RESUMEN

Los sistemas orientados a objetos describen las entidades como objetos. Los objetos forman parte de un concepto general conocido como clases, la unidad principal de análisis en el análisis y diseño orientados a objetos. Cuando se introdujo por primera vez la metodología orientada a objetos, los defensores citaron la reutilización de los objetos como el principal beneficio de su metodología. Aunque la reutilización es el principal objetivo, también es muy importante el mantenimiento de los sistemas.

Los analistas pueden usar tarjetas CRC para empezar el proceso de modelado de objetos de una manera informal. Se puede agregar el Pensamiento en objetos a las tarjetas CRC para ayudar al analista a refinar las responsabilidades en tareas cada vez más pequeñas. Se pueden realizar sesiones CRC con un grupo de analistas para determinar las clases y responsabilidades en forma interactiva.

El lenguaje unificado de modelado (UML) provee un conjunto estandarizado de herramientas para documentar el análisis y diseño de un sistema de software. El UML se basa fundamentalmente en una técnica orientada a objetos conocida como modelado de casos. Un modelo de casos de uso describe *qué* hace el sistema sin describir *cómo* lo hace. Un modelo de casos de uso particiona la funcionalidad del sistema en comportamientos (llamados casos de uso) que son importantes para los usuarios del sistema (llamados actores). Se crean distintos escenarios para cada conjunto diferente de condiciones de un caso de uso.

Los principales componentes del UML son cosas, relaciones y diagramas. Los diagramas están relacionados entre sí. Las cosas estructurales son más comunes; incluyen clases, interfaces, casos de uso y muchos otros elementos que proveen la forma de crear modelos. Las cosas estructurales permiten al usuario describir relaciones. Las cosas de comportamiento describen la forma en que trabajan las cosas. Las cosas de agrupamiento se utilizan para definir límites. Las cosas de anotaciones permiten al analista agregar notas a los diagramas.

Las relaciones son el pegamento que mantiene todo unido. Las relaciones estructurales se utilizan para enlazar las cosas en los diagramas estructurales. Las relaciones estructurales incluyen dependencias, agregaciones, asociaciones y generalizaciones. Los diagramas de comportamiento utilizan los cuatro tipos básicos de relaciones de comportamiento: comunica, incluye, extiende y generaliza.

El conjunto de herramientas de UML está compuesto de diagramas de UML. Aquí se incluyen los diagramas de casos de uso, diagramas de actividad, diagramas de secuencia, diagramas de comunicación, diagramas de clases y diagramas de estados. Además de los diagramas, los analistas pueden describir un caso de uso mediante un escenario de casos de uso.

Mediante el uso de UML de manera iterativa en el análisis y el diseño podemos lograr una mejor comprensión entre el equipo de negocios y el equipo de TI en relación con los requerimientos del sistema y los procesos que necesitan ocurrir en el sistema para cumplir con esos requerimientos.





## EXPERIENCIA DE HYPERCASE® 10

“Espero que aún se sienta como si estuviera aprendiendo nuevas cosas sobre MRE. Entiendo que ha estado hablando con algunas de las personas de sistemas: Melissa, Todd, Roger (e incluso con Lewis, nuestro nuevo interno) acerca de usar algunos métodos distintos para crear diagramas y comprendernos mejor. Espero que nos vea como una familia y no sólo como una colección de personas. Sin duda todos nos sentimos como si hubiéramos ‘heredado’ una gran sabiduría de Jimmy Hyatt y del padre de Warren. Estoy completamente de acuerdo en usar su nueva metodología, si nos ayuda a mejorar los informes de nuestros proyectos. Desde luego que Snowden está ansioso de ver su trabajo orientado a objetos. ¿Podría presentarle algo en su escritorio en un par de semanas?”

### Preguntas de HYPERCASE

1. Cree un diagrama de actividad para el caso de uso Progreso del proyecto de informes (Report Project Progress). Consulte las especificaciones del caso de uso en la oficina de Melissa Smith para ver los detalles y un prototipo.
2. Cree un diagrama de actividad para el caso de uso Agregar cliente (Add Client). Consulte las especificaciones del caso de uso en la oficina de Melissa Smith para ver los detalles y un prototipo que encontrará en la oficina de Todd Taylor.
3. Cree un diagrama de secuencia para la ruta principal del caso de uso Progreso del proyecto de informes. Consulte las especificaciones del caso de uso en la oficina de Melissa Smith para ver los detalles y un prototipo.
4. Cree un diagrama de secuencia para la ruta principal del caso de uso Agregar cliente. Consulte las especificaciones del caso de uso en la oficina de Melissa Smith para ver los detalles y un prototipo que encontrará en la oficina de Todd Taylor.
5. Cree un diagrama de estados para la clase Asignación (Assignment). Se crean asignaciones para las tareas, se seleccionan los recursos, se actualizan las horas y se terminan las asignaciones.
6. Cree un diagrama de estados para la clase Tarea (Task). Las tareas se crean, pero no se inician; se planean, algunas veces se retienen, se trabaja en ellas actualmente y se completan.

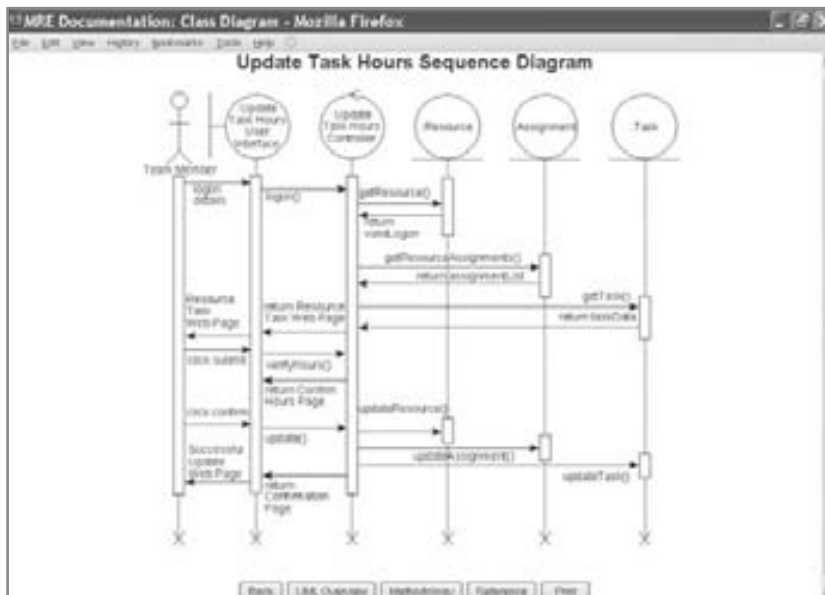


FIGURA 10.HC1

Encontrará los diagramas de secuencia en HyperCase.

## PALABRAS CLAVE Y FRASES

actor  
agregación  
Ajax  
asociación  
barra de sincronización  
bifurcación  
carril

caso de uso primario  
clase  
clase abstracta  
clase de control  
clase de entidad  
clase de límite  
colaboración

cosa de anotación  
 dependencias  
 diagrama de actividad  
 diagrama de casos de uso  
 diagrama de clases  
 diagrama de comunicación  
 diagrama de despliegue  
 diagrama de estados  
 diagrama de secuencia  
 escenario de casos de uso  
 estado  
 estructura todo/parte  
 evento  
 evento temporal  
 fusión  
 generalización/especialización (gen/spec)  
 herencia

lenguaje unificado de modelado (UML)  
 mensaje  
 mensaje asíncrono  
 mensaje sincrónico  
 objeto  
 orientado a objetos  
 paquete  
 polimorfismo  
 proceso unificado  
 ramificación  
 redefinición de métodos  
 relación  
 ruta principal  
 sobrecarga de métodos  
 tarjetas CRC  
 unir

## PREGUNTAS DE REPASO

1. Liste dos razones de utilizar una metodología orientada a objetos para el desarrollo de sistemas.
2. Describa la diferencia entre una clase y un objeto.
3. Explique el concepto de herencia en los sistemas orientados a objetos.
4. ¿Qué significa CRC?
5. Describa lo que agrega el Pensamiento en objetos a la tarjeta CRC.
6. ¿Qué es UML?
7. ¿Cuáles son los tres elementos principales del UML?
8. Haga una lista de lo que incluye el concepto de cosas estructurales.
9. Haga una lista de lo que incluye el concepto de cosas de comportamiento.
10. ¿Cuáles son los dos tipos principales de diagramas en UML?
11. Haga una lista de los diagramas que se incluyen en los diagramas estructurales.
12. Haga una lista de los diagramas que se incluyen en los diagramas de comportamiento.
13. ¿Qué es lo que describe un modelo de casos de uso?
14. ¿Describiría un modelo de casos de uso como un modelo lógico o físico del sistema? Defienda su respuesta en un párrafo.
15. Defina qué es un actor en un diagrama de casos de uso.
16. ¿Cuáles son las tres cosas que un caso de uso siempre debe describir?
17. ¿Qué es lo que describe un diagrama de actividad?
18. Escriba un párrafo que describa el uso de los carriles en los diagramas de actividad.
19. ¿Qué se puede describir en un diagrama de secuencia o de comunicación?
20. ¿Por qué definir clases es una tarea tan importante del análisis orientado a objetos?
21. ¿Qué se puede mostrar en un diagrama de clases?
22. Defina la sobrecarga de métodos.
23. Mencione las cuatro categorías en las que se clasifican las clases.
24. ¿Cuáles son los pasos para crear un diagrama de secuencia?
25. ¿Cuáles son las dos categorías de relaciones entre clases?
26. ¿Para qué se utilizan los diagramas de generalización/especialización (gen/spec)?
27. ¿Cuál es otro término para el polimorfismo?
28. ¿Qué se describe mediante un diagrama de estados?
29. ¿Qué es un paquete en la metodología del UML?
30. ¿Por qué es importante usar el UML para el modelado?

## PROBLEMAS

1. Cree una serie de tarjetas CRC para la División de catálogos de World's Trend. Una vez colocado un pedido, el equipo de abastecimiento de pedidos se hace cargo y revisa la disponibilidad, abastece el pedido y calcula el monto total del mismo. Use cinco tarjetas CRC, una para cada una de las siguientes clases: pedido, abastecimiento de pedido, inventario, producto y cliente. Complete la sección sobre clases, responsabilidades y colaboradores.
2. Termine las tarjetas CRC del problema 1; cree enunciados de Pensamiento en objetos y nombres de propiedades para cada una de las cinco clases.

3. Dibuje un diagrama de casos de uso para la División de catálogos de World's Trend.
4. Dibuje cuatro imágenes que muestren ejemplos de cuatro tipos de relaciones de comportamiento para la agencia automotriz BMW de Joel Porter. ¿Qué tipo de relación hay implicada cuando un cliente debe hacer arreglos de financiamiento? ¿Hay actividades comunes involucradas cuando una persona arrenda o compra un automóvil? ¿Qué tipo de relación existe entre un empleado que es un gerente o uno que es un vendedor?
5. Dibuje un diagrama de comunicación para un estudiante que toma un curso de un maestro que forma parte del cuerpo docente.
6. El condado Coleman tiene una central telefónica que se hace cargo de las llamadas entre los que llaman y los que reciben la llamada. Dados estos tres actores, dibuje un diagrama de secuencia simple para realizar una simple llamada telefónica.
7. Usted está listo para empezar el modelado de UML para la clínica Kirt. Dibuje un diagrama de clases que incluya a un médico, un paciente, una cita y la factura de un paciente. No involucre a la compañía de seguros.
8. Use UML para dibujar ejemplos de las cuatro relaciones estructurales para la clínica Kint.
9. Escriba un escenario de caso de uso de ejemplo para un paciente que ve a un médico en la clínica Kint.
10. El Woody's Supermarket, una pequeña cadena de tiendas de abarrotes, está construyendo un sitio Web para permitir a los clientes realizar pedidos de abarrotes y otros artículos. El cliente coloca un pedido Web, se actualiza el archivo maestro de clientes y se crea un registro de pedido. El pedido se imprime en una tienda local y los empleados de la tienda recogen los artículos de los estantes. Los clientes reciben una notificación vía correo electrónico de que su pedido está listo. Cuando recogen el pedido, se agregan los artículos congelados, productos helados y demás artículos relacionados. Dibuje un diagrama de actividad que muestre cómo el cliente coloca su pedido mediante el sitio Web, cómo se verifica el pedido, cómo se confirma, como se envían los detalles relacionados a la tienda local y cómo se envía un mensaje de correo electrónico al cliente.
11. Sludge's Auto (consulte el capítulo 12) es un centro de reciclaje de auto partes que utiliza Ajax en sus sitios Web para que sus clientes naveguen en busca de partes. Ajax permite al sitio Web obtener datos del servidor mientras el usuario permanece en la página Web original. El cliente necesita conocer la marca, modelo y año de un auto así como la pieza. Si la pieza está en existencia, aparecen la descripción, condición de la pieza, precio y costo de envío, junto con la cantidad disponible para cada condición de la pieza y una imagen de la misma. Dibuje un diagrama de secuencia que utilice clases de límite, control y entidad para la Consulta de auto partes de Sludge's Auto.
12. Musixscore.com es un servicio en línea que provee partituras musicales a sus clientes. En la página Web "explorar música" los clientes seleccionan un género de música de una lista desplegable. La página Web utiliza Ajax para obtener una lista de artistas, músicos o grupos que coincidan con el género, lo cual se muestra en un formato de lista desplegable. Al hacer una selección de la lista desplegable del artista, la página Web utiliza Ajax para mostrar una tercera lista desplegable con todos los CD o demás obras del artista. Cuando se selecciona un CD, la página Web utiliza Ajax para obtener todas las canciones del CD en una cuarta lista desplegable. El espectador puede hacer selecciones múltiples. Al hacer clic en la imagen **Agregar al carrito de compras** las canciones se agregan al carrito de compras. El espectador puede cambiar cualquiera de las listas desplegables para seleccionar partituras musicales adicionales y se repite el proceso.
  - a. Escriba una descripción para el caso de uso Explorar partitura musical que represente esta actividad.
  - b. Dibuje un diagrama de secuencia que utilice clases de límite, control y entidad para la página Web de Musixscore.
  - c. Escriba una lista de los mensajes, nombres y parámetros, junto con los tipos de datos que se pasarían a las clases y los valores (con los tipos de datos) que se incluyen con el mensaje de retorno. Haga todas las suposiciones necesarias sobre los datos.
  - d. Cree un diagrama de clases para las clases de entidad utilizadas en el diagrama de secuencia.

## BIBLIOGRAFÍA SELECCIONADA

- Beck, K. y W. Cunningham. "Laboratory for Teaching Object-Oriented Thinking", OOPSLA'89, según lo citado en D. Butler, CRC Card Session Tutorial. [www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc\\_b/tutorial.html](http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/tutorial.html). Último acceso en 29 de julio de 2009.
- Bellin, D. y S. Suchman Simone. *The CRC Card Book*. Indianápolis: Addison-Wesley Professional, 1997.
- Booch, G., I. Jacobson y J. Rumbaugh. *The Unified Modeling Language User Guide*, 2ª. Edición. Indianápolis: Addison-Wesley Professional, 2005.
- Cockburn, A. *Writing Effective Use Cases*. Boston: Addison-Wesley Publishing Co., 2001.
- Dobing, B. y J. Parsons. "How UML Is Used". *Communications of the ACM*, Vol. 49, Núm. 5, mayo de 2006, pp. 109-113.
- Fowler, M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3ª. Edición. Indianápolis: Addison-Wesley Professional, 2003.
- Kulak, D. y E. Guiney. *Use Cases: Requirements in Context*, 2ª. Edición. Indianápolis: Addison-Wesley Professional, 2004.
- Miles, R. y K. Hamilton. *Learning UML 2.0*. Indianápolis: O'Reilly Media, Inc., 2006.
- Sahraoudi, A. E. K. y T. Blum. "Using Object-Oriented Methods in a System Lifecycle Process Model". *ACM SIGSOFT Software Engineering Notes*, Vol. 28, Núm. 2 (marzo de 2003).

## EPISODIO 10

## CASO DE LA CPU

ALLEN SCHMIDT, JULIE E. KENDALL Y KENNETH E. KENDALL

## Objetos con clase

“Los prototipos y diagramas que hemos creado realmente nos ayudaron a entender el proyecto”, sonríe Anna, mirando desde su computadora. “Tengo una buena sensación sobre lo que estamos haciendo”.

“Igual yo”, contesta Chip. “Pero necesitamos trabajar en las páginas Web que serán utilizadas por un amplia variedad de docentes y miembros del personal”.

“¿Por dónde quieres empezar?”, pregunta Anna. “¿Crees que deberíamos trabajar en el diagrama de clases?”.

Chip se reclina en su silla y piensa en voz alta. “No, necesitamos realizar otro modelado para determinar las clases, atributos y métodos primero. Tenemos un diagrama E-R preliminar como punto inicial para las clases de entidad. Deberíamos modelar el comportamiento de varios prototipos. Al examinar un prototipo, deberíamos hacer preguntas sobre cada campo. Si la información se almacena en una tabla de base de datos, deberíamos obtenerla de alguna forma”.

“Sí, por lo general la incluiríamos en una lista desplegable”, reflexiona Anna.

“Bueno, no necesariamente”, responde Chip. “Si hay demasiadas entradas o si una lista depende de una acción previa, deberíamos usar Ajax para recuperar la información y actualizar la página Web, creando listas desplegables o vínculos”.

“Los programas de creación para agregar nueva información serán los que requieran más tecleo, pero los programas para cambiar, eliminar y consultar dependen de los datos almacenados”, agrega Anna. “¿Por dónde deberíamos empezar?”.

“He estado pensando mucho en el prototipo que creamos para usar la Web para actualizar la imagen de todo el software que se almacena en una máquina del laboratorio de computación”, dice Chip, inclinándose hacia delante en su silla. “Y con sala de laboratorio me refiero a cualquier salón de clases o laboratorio que tenga el mismo conjunto de software en cada computadora. La imagen de software se utiliza para refrescar las computadoras dañadas o para reemplazar el software en una máquina infectada por un virus o cualquier otro problema”.

“Bueno, demos un vistazo a ese prototipo ACTUALIZAR IMAGEN DE LABORATORIO”, dice Anna.

Chip muestra el prototipo (consulte el caso de la CPU del capítulo 6). “Necesitamos conocer el número de cuarto y de campus para buscar la imagen de software”, dice Chip mientras apunta al primer par de campos en la página Web. “Estos números se almacenan en tablas de la base de datos”.

“Sí, pero se almacenan con el software de programación administrativa en la mainframe”, interviene Anna. “Podríamos copiarlos a los servidores Web, pero si cambian los edificios o los cuartos tendremos datos inconsistentes y un sistema poco fiable”.

“Ahh, veo lo que quieres decir”, dice Chip. “¡Bien pensado! Tenemos que obtener estos datos de la mainframe”.

“Esto requiere el diagrama de actividad ACTUALIZAR IMAGEN DE LABORATORIO (que se muestra en la figura E10.1)”, comenta Anna. “¿Por qué no creo el diagrama y tu revisas que esté correcto?”.

“Suenan bien”, comenta Chip.

Anna empieza a dibujar el diagrama. “Como sólo ciertos miembros del personal tendrán la autorización para actualizar la imagen del laboratorio, creo que debería empezar con ellos iniciando sesión en el sistema. Si es un inicio de sesión válido, se enviará una solicitud al estado de la mainframe llamada OBTENER SALAS EDIFICIO CAMPUS para obtener los edificios del campus y las salas dentro de éstos. Esta lista se envía de vuelta al programa ENVIAR LISTA EDIFICIO CAMPUS en el servidor Web, en donde se convierte en un documento de XML y se envía a la página Web SELECCIONAR EDIFICIO DE CAMPUS”.

Chip pregunta: “¿Vas a poner todas las salas en la lista para todos los edificios? Eso sería un extenso documento de XML y podría tardar un tiempo en cargarse en el navegador”.

“La otra opción sería incluir sólo los edificios y después solicitar las salas para un edificio de campus seleccionado”, responde Anna pensativamente. “Eso también sería aceptable, pero alentaría el proceso de seleccionar salas ya que el navegador tendría que esperar que las salas se actualizaran en la lista desplegable. Si se enviara todo el documento de XML a la vez, la lista de salas se actualizaría rápidamente a partir de los elementos del documento a nivel del navegador. Sería muy rápido. Así el usuario seleccionaría un edificio del campus y sólo aparecerían las salas para ese edificio”.

“Tengo una idea”, exclama Chip. “Cuando el programa obtiene las salas de los edificios, ¿hay alguna forma de seleccionar sólo las salas que son laboratorios de computadoras?”.

“Es una excelente idea”, dice Anna. “El documento de XML sería más pequeño y se cargaría más rápido en el navegador. Tal vez tengamos que solicitar una modificación de la tabla de la base de datos para tener un código para los laboratorios de computadora”.

“Voy a ver eso”, responde Chip. “Entonces, ¿qué ocurre después de que obtenemos una sala de laboratorio?”.

Anna piensa por un momento, “Necesitamos una tabla de imágenes del laboratorio en el servidor Web que contenga el número de sala y el software incluido en ella. Esto incluye el software estándar, como el sistema operativo, procesamiento de palabras y software para escanear virus, ya que pueden cambiar de una sala a otra. Agregaré un estado para OBTENER SOFTWARE DE COMPUTADORA PARA SALA en el carril Servidor Web”.

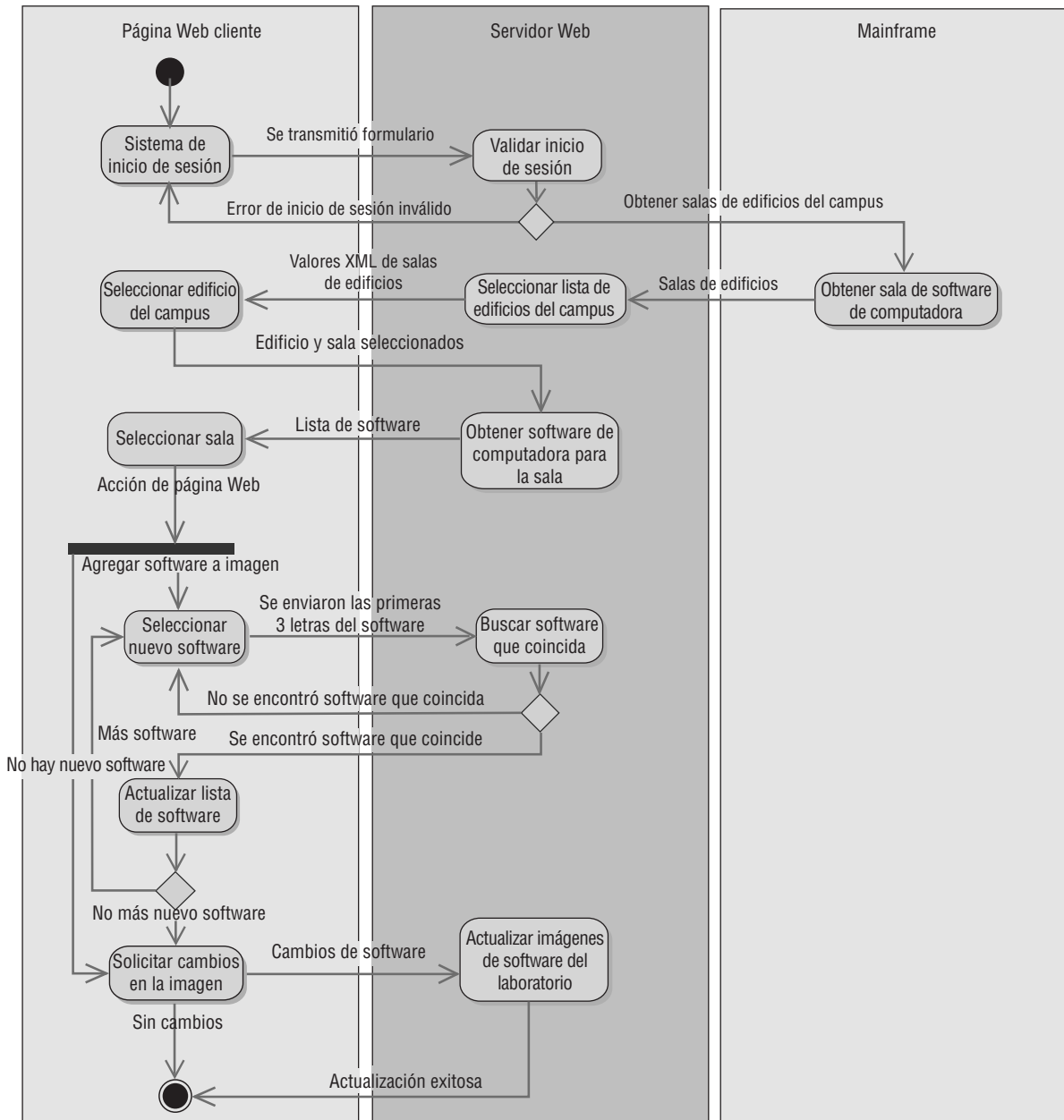


FIGURA E10.1

Diagrama de actividad ACTUALIZAR IMAGEN DE LABORATORIO

Chip pregunta: “¿Incluimos una lista desplegable para el software?”.

“No, eso dificultaría quitar el software o cambiar la versión”, responde Anna pensativamente. “Creo que deberíamos mostrar el software en una lista en la página Web y permitirles actualizar el número de versión”.

“El problema es que tal vez tengamos una lista extensa de software y haya que eliminar parte del software también!”, comenta Chip. “¿Por qué no usamos un modelo de correo electrónico, con una casilla de verificación en frente de cada título de software y limitamos el número de títulos de software por página? Podríamos incluir un botón de siguiente página para mostrar el siguiente grupo de títulos de software”.

“¿Cómo sabe el programa de eliminación qué paquete de software eliminar?”, pregunta Anna.

“Necesitamos incluir el número de software como campo oculto que se transmita al servidor Web”, responde Chip. “¿Qué piensas sobre agregar software a la imagen del laboratorio?”.

Anna piensa por unos minutos. “¿Por qué no usamos Ajax de nuevo para hacer que introduzcan las primeras letras del título de software en un campo de texto de entrada? Podríamos enviar la solicitud a la tabla Software en el servidor Web y devolver

una lista de los títulos que coincidan. La página Web crearía un bloque rectangular flotante en la parte superior de la página Web con una lista de los vínculos a los títulos de software y los campos ocultos almacenados con el vínculo. El usuario haría clic en un título para agregarlo a la imagen. Después podría agregar otro título de software si fuera necesario. Ésta es la actividad SELECCIONAR NUEVO SOFTWARE que envía las primeras tres letras al estado BUSCAR SOFTWARE QUE COINCIDA del servidor Web. ACTUALIZAR LISTA DE SOFTWARE crea el bloque flotante de títulos”.

“¡Excelente idea!”, exclama Chip.

“Al terminar de agregar software, revisar el software que se va a quitar y cambiar los números de las versiones, hacen clic en el botón enviar y la tabla de la base de datos de imágenes del laboratorio se actualiza con los cambios”, dice Anna. “El estado SOLICITAR CAMBIOS EN IMAGEN envía los cambios al estado ACTUALIZAR IMAGEN SOFTWARE LABORATORIO”.

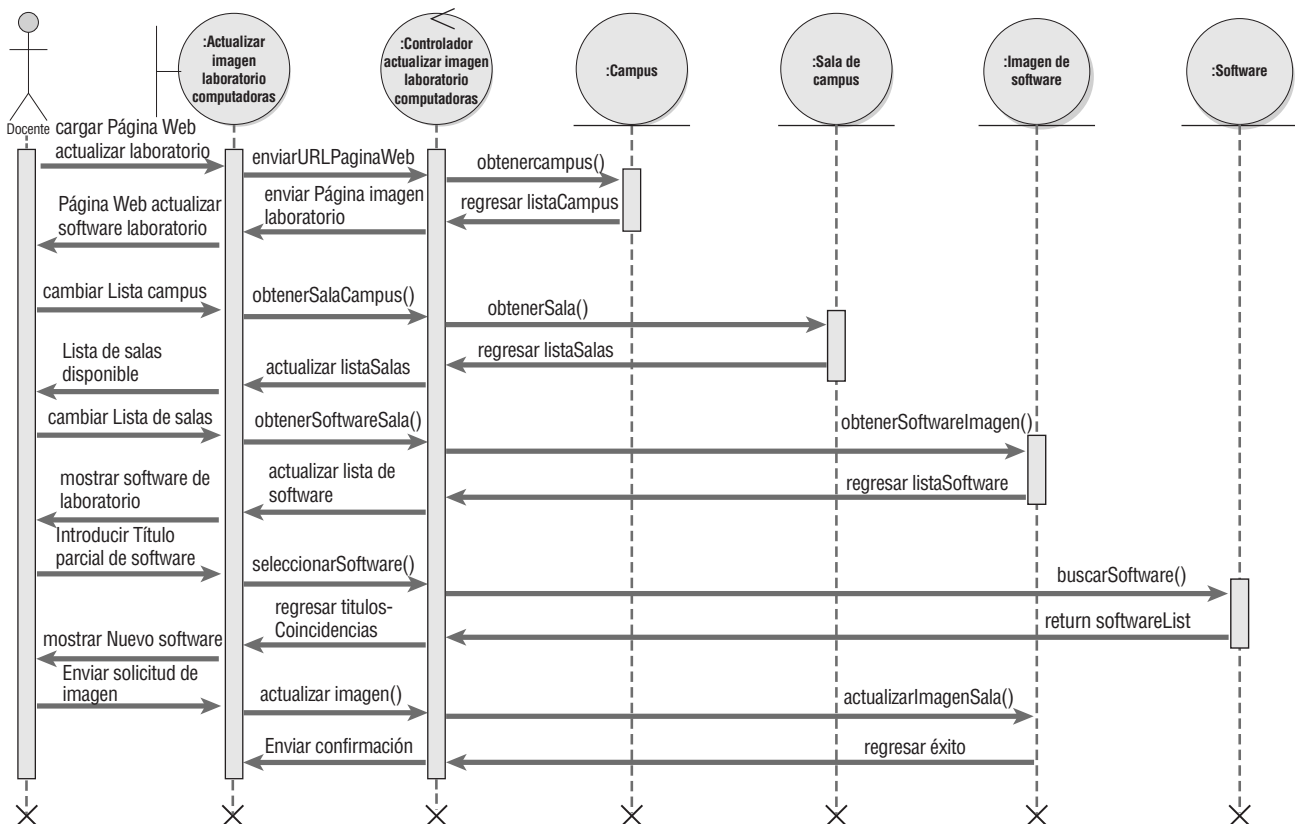
“Esto es divertido”, sonríe Chip. “Voy a trabajar en un diagrama de secuencia para el prototipo ACTUALIZAR IMAGEN LABORATORIO (que se muestra en la figura E10.2). Chip empieza por enviar la solicitud para la página Web Actualizar laboratorio. El servidor envía el mensaje obtenerCampus() a la clase CAMPUS, la cual devuelve la listaCampus incluyendo un códigoCampus y una descripciónCampus. El controlador envía un mensaje a la clase SALA DE CAMPUS para obtener salas que contengan software de laboratorio, las cuales se devuelven al CONTROLADOR ACTUALIZAR IMAGEN LABORATORIO COMPUTADORAS. La clase controladora crea el documento de XML listaSalas y lo envía al navegador Web, el cual crea la lista de selección Edificio de campus. Al cambiar la lista de Edificios del campus, el navegador Web utiliza el mismo documento de XML para cambiar la lista desplegable Número de sala, de manera que sólo incluya las salas de laboratorio en el edificio del campus seleccionado.

Cuando se modifica la lista desplegable Número de sala, se envía una solicitud obtenerSoftwareSala() a la clase controladora, que a su vez envía un mensaje obtenerSoftwareImagen() a la clase IMAGEN SOFTWARE. Mediante el uso del número de sala se obtiene el software y se devuelve en la listaSoftware. LA clase controladora utiliza la listaSoftware para crear el XML que se envía a la clase de interfaz ACTUALIZAR IMAGEN LABORATORIO COMPUTADORAS, que actualiza la página Web con los títulos de software.

Cuando se introduce un título parcial de software, se envía una solicitud seleccionarSoftware() al CONTROLADOR ACTUALIZAR IMAGEN LABORATORIO COMPUTADORAS, el cual envía un mensaje buscarSoftware() a la clase de

**FIGURA E10.2**

Diagrama de secuencia ACTUALIZAR IMAGEN DE LABORATORIO.





entidad SOFTWARE. Se busca el software que coincida y se envía la listaSoftware que contiene el número de software, descripción y versión a la clase controladora. Esto da formato al documento de XML, que se envía a la clase de interfaz ACTUALIZAR IMAGEN LABORATORIO COMPUTADORAS. El navegador Web da formato al bloque flotante de títulos de software. Al seleccionar un título, el navegador lo agrega a la lista de títulos de software y se quita el bloque flotante.

Al hacer clic en el botón Enviar solicitud de imagen en la página Web, se envía la solicitud actualizarImagen() al controlador, el cual envía un mensaje actualizarImagenSala() a la clase de entidad IMAGEN DE SOFTWARE. La tabla de la base de datos se actualiza y se devuelve el mensaje de éxito al controlador, que a su vez envía una página Web de confirmación.

“El software parece un poco complicado, ya que se reemplazan distintas versiones y paquetes de software por otras diferentes”, comenta Anna. “Creo que es buena idea dibujar un diagrama de estados para el software. Esto nos dará una idea de los métodos y atributos de software, además de la interfaz que necesitaremos para cambiar esos atributos”.

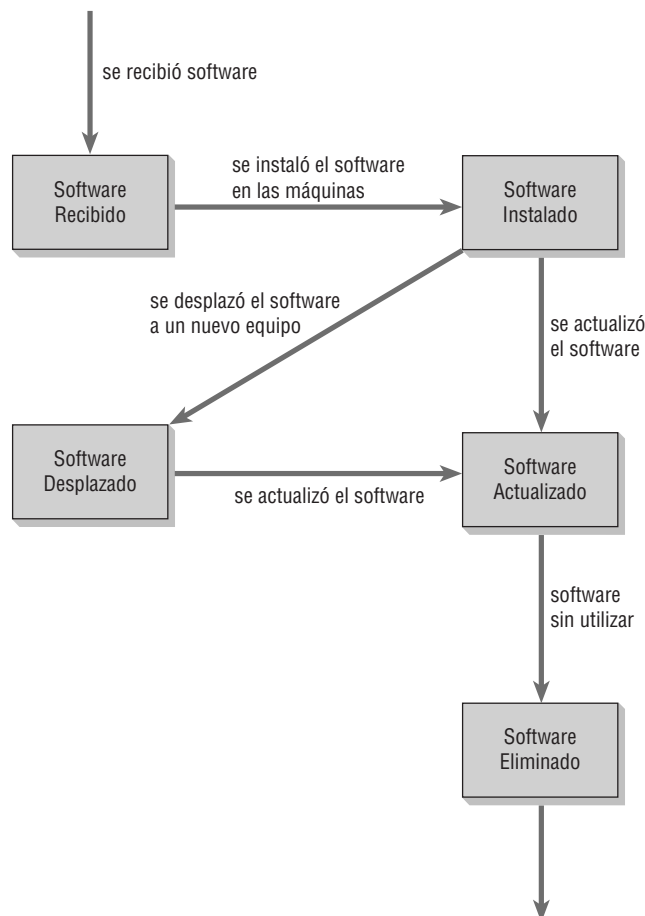
Anna empieza a trabajar en un diagrama de estados SOFTWARE. Al recibir por primera vez el software, se introduce en el sistema mediante el formulario AGREGAR SOFTWARE de Microsoft Access y se cambian los valores de los atributos iniciales. Hay que agregar todo el software antes de poder instalarlo en las máquinas, por lo que se difiere el evento SOFTWARE INSTALADO EN LAS MÁQUINAS hasta que se haya agregado.

Una vez instalado el software en cualquier cantidad de máquinas, existe en el estado SOFTWARE INSTALADO durante mucho tiempo. Se actualiza la tabla relacional HARDWARE-SOFTWARE para reflejar el estado actual. De vez en cuando se reemplaza una máquina y el software se mueve a una máquina distinta. Se vuelve a actualizar la tabla HARDWARE-SOFTWARE para reflejar la nueva ubicación. Cuando hay una nueva versión del software disponible, se actualiza con un formulario CAMBIAR SOFTWARE de Microsoft Access. O también se puede eliminar el software del sistema mediante el formulario ELIMINAR SOFTWARE de Microsoft Access. En la figura E10.3 se muestra el diagrama de estados SOFTWARE completo.

Chip y Anna trabajan en varios diagramas de actividad, secuencia y estados. Después de haber completado varios diagramas, Chip comenta: “Creo que tenemos suficiente información para crear un diagrama de clases”.

Anna concuerda, “Sí, vamos a asignar las relaciones”.

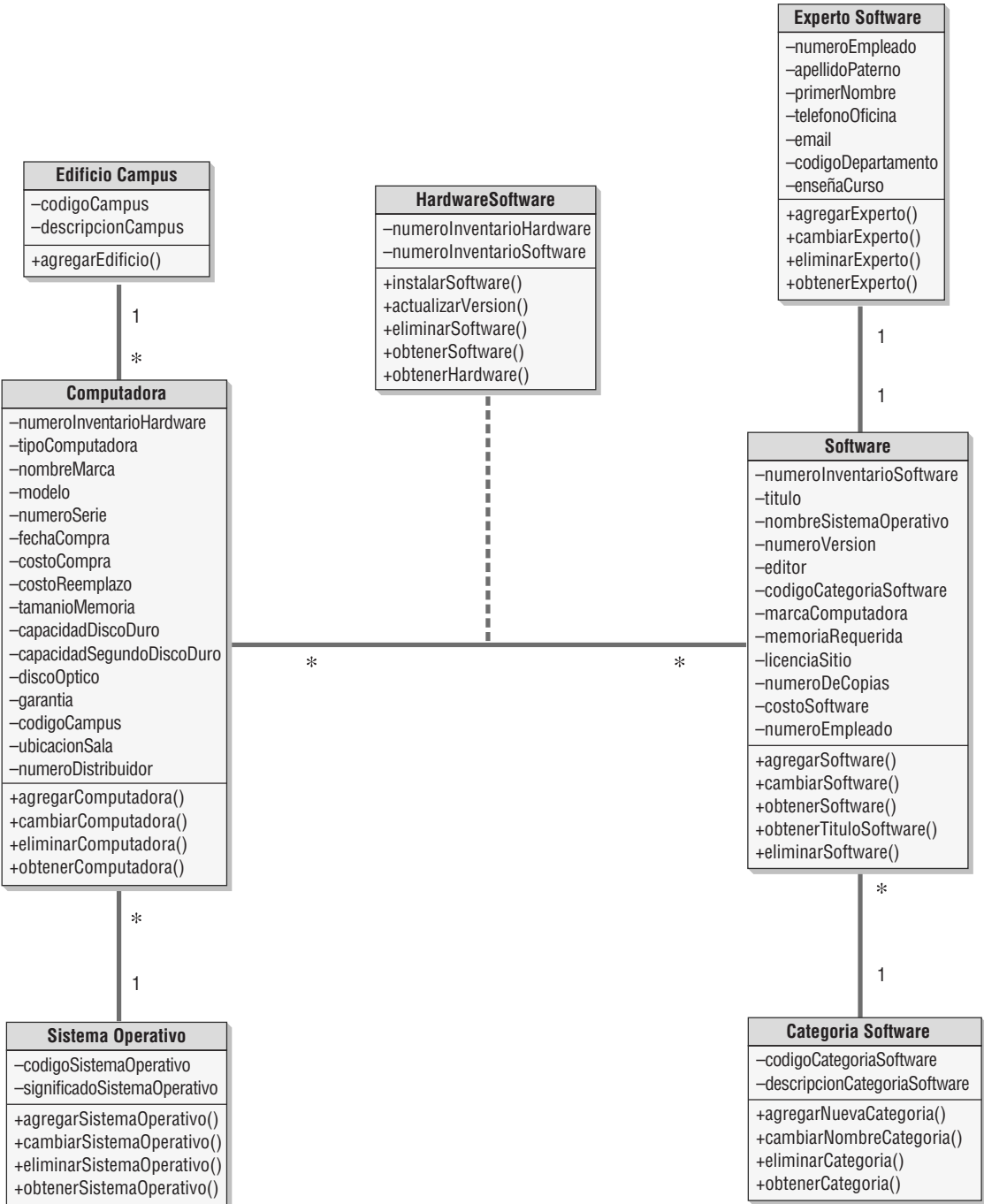
En la figura 10.4 se muestra el diagrama de clases SISTEMA COMPUTARIZADO. Cada clase tiene atributos privados y métodos públicos en Microsoft Access para actualizar los atributos. Las principales clases son Computadora y Software, con una clase asociativa HardwareSoftware que las conecta. Esta clase se utiliza para implementar la relación de muchos a muchos entre el hardware y el software. Cada paquete de software pertenece a una Categoría de software y también tiene un Experto de software a quien se puede llamar para obtener soporte. Cada computadora tiene uno o más sistemas operativos y se localiza en un edificio del campus.



**FIGURA E10.3**





Diagrama de estados SOFTWARE.





**FIGURA E10.4**  
Diagrama de clases SISTEMA COMPUTARIZADO.

**Ejercicios**

-  **E-1.** Use Microsoft Visio o Visible Analyst para ver el diagrama de actividad ACTUALIZAR IMAGEN LABORATORIO (UPDATE LAB IMAGE).
-  **E-2.** Use Microsoft Visio o Visible Analyst para ver el diagrama de secuencia ACTUALIZAR IMAGEN LABORATORIO (UPDATE LAB IMAGE).
-  **E-3.** Use Microsoft Visio o Visible Analyst para ver el diagrama de estados SOFTWARE.
-  **E-4.** Use Microsoft Visio o Visible Analyst para ver el diagrama de clases SISTEMA COMPUTARIZADO (COMPUTER SYSTEM).

Los siguientes ejercicios se pueden realizar con Microsoft Visio o con Visible Analyst. Hay que usar un rectángulo para un símbolo de clase al dibujar diagramas de secuencia con Microsoft Visio o Visible Analyst (los símbolos de estereotipo de clase no están disponibles). Coloque una etiqueta de texto arriba de cada rectángulo para identificar el tipo de clase: interfaz, control o entidad.

**E-5.** Modifique e imprima el diagrama de actividad REGISTRAR PARA CAPACITACIÓN (REGISTER FOR TRAINING). Consulte la figura E12.3 en el Episodio del caso de la CPU del capítulo 12 para ver el prototipo de esta página Web. Agregue los siguientes símbolos de estado y conexiones de eventos:

- El estado CREAR DATOS XML EMPLEADO (CREATE XML EMPLOYEE DATA) en el carril SERVIDOR WEB (WEB SERVER) debajo del estado OBTENER INFORMACIÓN EMPLEADO (GET EMPLOYEE INFORMATION). Conéctelo con una flecha de evento que venga de LEER REGISTRO EMPLEADO (READ EMPLOYEE RECORD). Asigne al evento la etiqueta ENVIAR DATOS EMPLEADO (SEND EMPLOYEE DATA).
- Agregue el estado PÁGINA WEB INFORMACIÓN EMPLEADO (EMPLOYEE INFORMATION WEB PAGE) en el carril PÁGINA WEB CLIENTE (CLIENT WEB PAGE) a la izquierda del estado CREAR DATOS XML EMPLEADO (CREATE XML EMPLOYEE DATA). Conecte los dos estados con una flecha de evento hacia el estado PÁGINA WEB INFORMACIÓN EMPLEADO (EMPLOYEE INFORMATION WEB PAGE) etiquetada como ENVIAR DOCUMENTO XML EMPLEADO (SEND EMPLOYEE XML DOCUMENT).
- Agregue un estado debajo del estado CREAR DATOS XML EMPLEADO (CREATE XML EMPLOYEE DATA) que se llame BUSCAR CLASE CAPACITACIÓN SOFTWARE (FIND SOFTWARE TRAINING CLASS). Conéctelo con una flecha de evento que provenga del estado PÁGINA WEB INFORMACIÓN EMPLEADO (EMPLOYEE INFORMATION WEB PAGE) y etiquétela como NIVEL DE SOFTWARE Y CAPACITACIÓN SELECCIONADO (SELECTED SOFTWARE AND TRAINING LEVEL).
- Incluya un símbolo de diamante de decisión debajo del estado BUSCAR CLASE CAPACITACIÓN SOFTWARE (FIND SOFTWARE TRAINING CLASS). Conéctelo con una flecha de evento que provenga del estado BUSCAR CLASE CAPACITACIÓN SOFTWARE (FIND SOFTWARE TRAINING CLASS). Un evento debe fluir a la izquierda hacia el estado PÁGINA WEB INFORMACIÓN EMPLEADO (EMPLOYEE INFORMATION WEB PAGE), etiquetado como NO SE ENCONTRÓ LA CLASE (CLASS NOT FOUND).
- Agregue un estado SELECCIONAR CLASE SOFTWARE (CHOOSE SOFTWARE CLASS) debajo del estado PÁGINA WEB INFORMACIÓN EMPLEADO (EMPLOYEE INFORMATION WEB PAGE), y un tanto debajo del diamante de decisión. Conecte la parte inferior del diamante de decisión con una flecha de evento que vaya hacia el estado SELECCIONAR CLASE SOFTWARE (CHOOSE SOFTWARE CLASS). Etiquétela como CLASES CAPACITACIÓN SOFTWARE (SOFTWARE TRAINING CLASSES).
- Agregue un estado debajo del diamante de decisión y un tanto debajo del estado SELECCIONAR CLASE SOFTWARE (CHOOSE SOFTWARE CLASS). Etiquételo como ACTUALIZAR PARTICIPANTE CLASE (UPDATE CLASS PARTICIPANT).
- Conecte el estado SELECCIONAR CLASE SOFTWARE (CHOOSE SOFTWARE CLASS) con una flecha de evento que apunte al estado ACTUALIZAR PARTICIPANTE CLASE (UPDATE CLASS PARTICIPANT). Etiquétela como ENVIAR SOLICITUD INSCRIBIR CLASE (SEND ENROLL CLASS REQUEST).
- Agregue un símbolo de círculo de salida en la parte inferior del carril Página Web cliente. Conecte el estado SELECCIONAR CLASE SOFTWARE (CHOOSE SOFTWARE CLASS) con una flecha de evento que apunte al círculo de salida y etiquétela como CANCELAR (CANCEL).
- Conecte el estado ACTUALIZAR PARTICIPANTE CLASE (UPDATE CLASS PARTICIPANT) con una flecha de evento que apunte al círculo de salida y etiquétela como ACTUALIZACIÓN EXITOSA (SUCCESSFUL UPDATE).

**E-6.** Cree e imprima el diagrama de actividad CALENDARIO CAPACITACIÓN (TRAINING CALENDAR). El prototipo para esta página Web se ilustra en la figura E11.4, que encontrará en el Episodio del caso de la CPU del capítulo 11. Agregue un círculo de inicio en la esquina superior izquierda del diagrama y agregue los siguientes carriles, símbolos de estado y conexiones de eventos:

- Agregue un carril a la izquierda etiquetado como PÁGINA WEB CLIENTE (CLIENT WEB PAGE) y uno a la derecha llamado SERVIDOR WEB (WEB SERVER).
- Agregue un círculo de inicio en la parte superior del carril PÁGINA WEB CLIENTE (CLIENT WEB PAGE) y debajo de éste un estado etiquetado como SOLICITAR PÁGINA WEB CALENDARIO CAPACITACIÓN (REQUEST TRAINING CALENDAR WEB PAGE). Conecte el círculo de inicio al estado con una flecha de evento.
- Agregue un estado en el carril SERVIDOR WEB, a la derecha del estado SOLICITAR PÁGINA WEB CALENDARIO CAPACITACIÓN (REQUEST TRAINING CALENDAR WEB PAGE). Etiquételo como OBTENER CLASE CAPACITACIÓN (GET TRAINING CLASS).
- Conecte el estado izquierdo al derecho con una flecha de evento etiquetada como SE TRANSMITIÓ FORMULARIO (FORM TRANSMITTED).
- Coloque un estado debajo del estado OBTENER CLASE CAPACITACIÓN (GET TRAINING CLASS). Etiquételo como OBTENER CLASE CAPACITACIÓN (GET TRAINING CLASS). Conecte los dos estados con una flecha de evento hacia abajo etiquetada como ENVIAR NÚMERO CURSO (SEND COURSE NUMBER).
- Coloque un estado en el carril PÁGINA WEB CLIENTE (CLIENT WEB PAGE) a la izquierda del estado OBTENER CLASE CAPACITACIÓN (GET TRAINING CLASS). Etiquételo como PANTALLA CURSO CALENDARIO CAPACITACIÓN (TRAINING CALENDAR COURSE DISPLAY). Conecte los dos estados con una flecha de evento que apunte hacia la izquierda y etiquétela como ENVIAR VALORES XML CLASE CAPACITACIÓN (SEND TRAINING CLASS XML VALUES).

- g. Coloque un círculo de salida en la parte inferior del carril PÁGINA WEB CLIENTE (CLIENT WEB PAGE). Conecte el estado PANTALLA CURSO CALENDARIO CAPACITACIÓN (TRAINING CALENDAR COURSE DISPLAY) con una flecha de evento que apunte a la derecha y arriba por el lado derecho del carril SERVIDOR WEB (WEB SERVER), hacia el estado PANTALLA CURSO CALENDARIO CAPACITACIÓN (TRAINING CALENDAR COURSE DISPLAY). Etiquétela como CAMBIAR FECHA O CAMBIAR ORDEN (DATE CHANGE OR SORT CHANGE).

**E-7.** Modifique e imprima el diagrama de secuencia REGISTRARSE PARA CAPACITACIÓN (REGISTER FOR TRAINING). Agregue dos nuevas clases de entidad del lado derecho del diagrama y extienda la línea de vida hacia abajo, hasta la parte inferior del diagrama. Las clases son Empleado y Clase. Agregue los siguientes mensajes del CONTROLADOR REGISTRARSE PARA CLASE (REGISTER FOR CLASS CONTROLLER) y agregue los rectángulos de foco de control en donde los mensajes interactúan con la línea de vida de la clase:

- a. obtenerEmpleado() [getEmployee()] del controlador a EMPLEADO (EMPLOYEE).
- b. regresar datosEmpleado (return employeeData) de la clase EMPLEADO (EMPLOYEE) al controlador.
- c. buscarClaseSoftware() [findSoftwareClass()] del controlador a la clase de entidad CLASE (CLASS).
- d. regresar listaClaseSoftware (return softwareClassList) de la clase de entidad CLASE (CLASS) al controlador.
- e. actualizarParticipanteClase() [updateClassParticipant()] del controlador a la clase de entidad CLASE (CLASS).
- f. regresar éxito (return success) de la clase de entidad CLASS al controlador.

**E-8.** Cree e imprima el diagrama de secuencia CALENDARIO CAPACITACIÓN (TRAINING CALENDAR). Agregue el actor Docente (Faculty) en la esquina superior izquierda del diagrama y después las siguientes clases, de izquierda a derecha, a lo largo de la parte superior del diagrama:

- a. La clase de interfaz Mostrar clases capacitación (Display Training Classes).
- b. La clase de control Mostrar clases capacitación (Display Training Classes).
- c. La clase de entidad Clase (Class).
- d. La clase de entidad Curso (Course).

Agregue los siguientes mensajes entre las clases o entre el actor y la clase:

- a. Cargar página Web Calendario capacitación (Load Training Calendar Web page) de Docente (Faculty) a la clase de interfaz Mostrar clases capacitación (Display Training Classes).
- b. enviarURLPaginaWeb (sendWebPageURL) de Mostrar clases capacitación (Display Training Classes) a Controlador Mostrar clases capacitación (Display Training Classes Controller).
- c. obtenerClase() [getClass()] del controlador a la clase de entidad Clase (Class).
- d. regresar listaClases (return classList) de la clase de entidad Clase (Class) al controlador.
- e. obtenerDescripcionCurso() [getCourseDescription()] del controlador a la clase de entidad Curso (Course).
- f. regresar descripcionCurso (return courseDescription) de la clase de entidad Curso (Course) al controlador.
- g. actualizar listaCursos (update courseList) de la clase controladora a la clase de interfaz Mostrar clases capacitación (Display Training Classes).
- h. Página Web Software laboratorio (Lab Software Web Page) de la clase interfaz Mostrar clases capacitación (Display Training Classes) al actor.
- i. Cambiar mes/año (Change Month/Year) del actor a la clase de interfaz Mostrar clases capacitación (Display Training Classes).
- j. Una auto-transición en la clase de interfaz Mostrar clases capacitación (Display Training Classes) (use JavaScript para actualizar el calendario).
- k. Nuevo Calendario (New Calendar) de la clase de interfaz Mostrar clases capacitación (Display Training Classes) al actor.
- l. Cambiar fecha (Change Date) del actor a la clase de interfaz Mostrar clases capacitación (Display Training Classes).
- m. obtenerNuevaClase() [getNewClass()] de Mostrar clases capacitación (Display Training Classes) al controlador.
- n. Repita los pasos c a g.
- o. Lista cursos disponible (Course List Available) de la clase de interfaz Mostrar clases capacitación (Display Training Classes) al actor.

**E-9.** Modifique e imprima el diagrama de estados Capacitación (Training). Agregue dos estados después de CLASE CAPACITACIÓN CANCELADA (CANCELLED TRAINING CLASS) del lado izquierdo del diagrama. Éstos son CLASE CAPACITACIÓN ACTIVA (ACTIVE TRAINING CLASS) y, debajo de éste, CLASE CAPACITACIÓN COMPLETADA (COMPLETED TRAINING CLASS). Agregue una clase debajo de CLASE CAPACITACIÓN PROGRAMADA (SCHEDULED TRAINING CLASS) llamada INSCRITO CLASE CAPACITACIÓN (ENROLLED TRAINING CLASS). Agregue las siguientes transiciones:

- a. SE INSCRIBIERON PARTICIPANTES (PARTICIPANTS ENROLLED) del estado CLASE CAPACITACIÓN PROGRAMADA (SCHEDULED TRAINING CLASS) al estado INSCRITO CLASE CAPACITACIÓN (ENROLLED TRAINING CLASS).
- b. CLASE EN SESIÓN (IN SESSION CLASS) del estado INSCRITO CLASE CAPACITACIÓN (ENROLLED TRAINING CLASS) al estado CLASE CAPACITACIÓN ACTIVA (ACTIVE TRAINING CLASS).
- c. FINALIZÓ SESIÓN CAPACITACIÓN (TRAINING SESSION ENDED) del estado CLASE CAPACITACIÓN ACTIVA (ACTIVE TRAINING CLASS) al estado CLASE CAPACITACIÓN COMPLETADA (COMPLETED TRAINING CLASS).
- d. Una flecha de finalización del estado CLASE CAPACITACIÓN COMPLETADA (COMPLETED TRAINING CLASS) a un área en blanco a la derecha.

**E-10.** Cree e imprima el diagrama de estados COMPUTADORA. Hay dos columnas de estados. En la columna izquierda incluya los siguientes estados de arriba hacia abajo: NUEVA COMPUTADORA (NEW COMPUTER), CLEANING COMPUTER (LIMPIANDO COMPUTADORA) y RECYCLED COMPUTER (COMPUTADORA RECICLADA). En

la columna derecha incluya los siguientes estados de arriba hacia abajo: COMPUTADORA INSTALADA (INSTALLED COMPUTER), COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER) y COMPUTADORA RETENIDA REPARACIÓN (REPAIR HELD COMPUTER). Agregue las siguientes transiciones:


- a. Empiece con SE RECIBIÓ COMPUTADORA (COMPUTER RECEIVED) que pase de un punto en el espacio arriba del rectángulo de estado hacia el estado NUEVA COMPUTADORA (NEW COMPUTER).
- b. COMPUTADORA INSTALADA (INSTALLED COMPUTER) del estado NUEVA COMPUTADORA (NEW COMPUTER) al estado COMPUTADORA INSTALADA (INSTALLED COMPUTER).
- c. SE INSTALÓ SOFTWARE (SOFTWARE INSTALLED) del estado COMPUTADORA INSTALADA (INSTALLED COMPUTER) al estado COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER).
- d. SE PROGRAMÓ MANTENIMIENTO (MAINTENANCE SCHEDULED) del estado COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER) al estado LIMPIANDO COMPUTADORA (CLEANING COMPUTER).
- e. SE COMPLETÓ MANTENIMIENTO del estado LIMPIANDO COMPUTADORA (CLEANING COMPUTER) al estado COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER).
- f. SE REPORTÓ PROBLEMA (PROBLEM REPORTED) del estado COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER) al estado COMPUTADORA RETENIDA EN REPARACIÓN (REPAIR HELD COMPUTER).
- g. SE COMPLETÓ REPARACIÓN (REPAIR COMPLETED) del estado COMPUTADORA RETENIDA EN REPARACIÓN (REPAIR HELD COMPUTER) al estado COMPUTADORA RECICLADA (RECYCLED COMPUTER).
- h. ACTUALIZAR COMPUTADORA IDENTIFICADA del estado COMPUTADORA FUNCIONAL (FUNCTIONAL COMPUTER) al estado COMPUTADORA RECICLADA (RECYCLED COMPUTER).
- i. SE IDENTIFICÓ REPARACIÓN IRREALIZABLE del estado COMPUTADORA RETENIDA EN REPARACIÓN (REPAIR HELD COMPUTER) al estado COMPUTADORA RECICLADA (RECYCLED COMPUTER).
- j. Una flecha de finalización del estado COMPUTADORA RECICLADA (RECYCLED COMPUTER) a un área en blanco debajo del estado.

**E-11.** Modifique e imprima el diagrama de clase COMPUTADORA (COMPUTER). Cada computadora puede tener uno o más sistemas operativos instalados. Mueva la clase Sistema operativo (Operating System) a la derecha de su ubicación actual y agregue una nueva clase llamada Sistema operativo de computadora (Computer Operating System) debajo de la clase Computadora (Computer). Cambie la línea conectora de Computadora a Sistema operativo para conectar la clase Sistema operativo a la clase Sistema operativo de computadora. Agregue una nueva relación entre la clase Computadora (el lado de uno) a la clase Sistema operativo de computadora (el lado de muchos). Agregue los siguientes atributos a la clase Sistema operativo de computadora:

NumeroInventarioHardware  
codigoSistemaOperativo

Agregue los siguientes métodos a la clase Sistema operativo de computadora:

agregarSistemaOperativoComputadora()  
eliminarSistemaOperativoComputadora()

Los ejercicios en los que se antepone un icono  indican material de valor agregado disponible en el sitio Web [www.pearsonhighered.com/kendall](http://www.pearsonhighered.com/kendall). Los estudiantes pueden descargar un archivo de muestra de Microsoft Visio, Visible Analyst, Microsoft Project o Microsoft Access que pueden usar para completar los ejercicios.