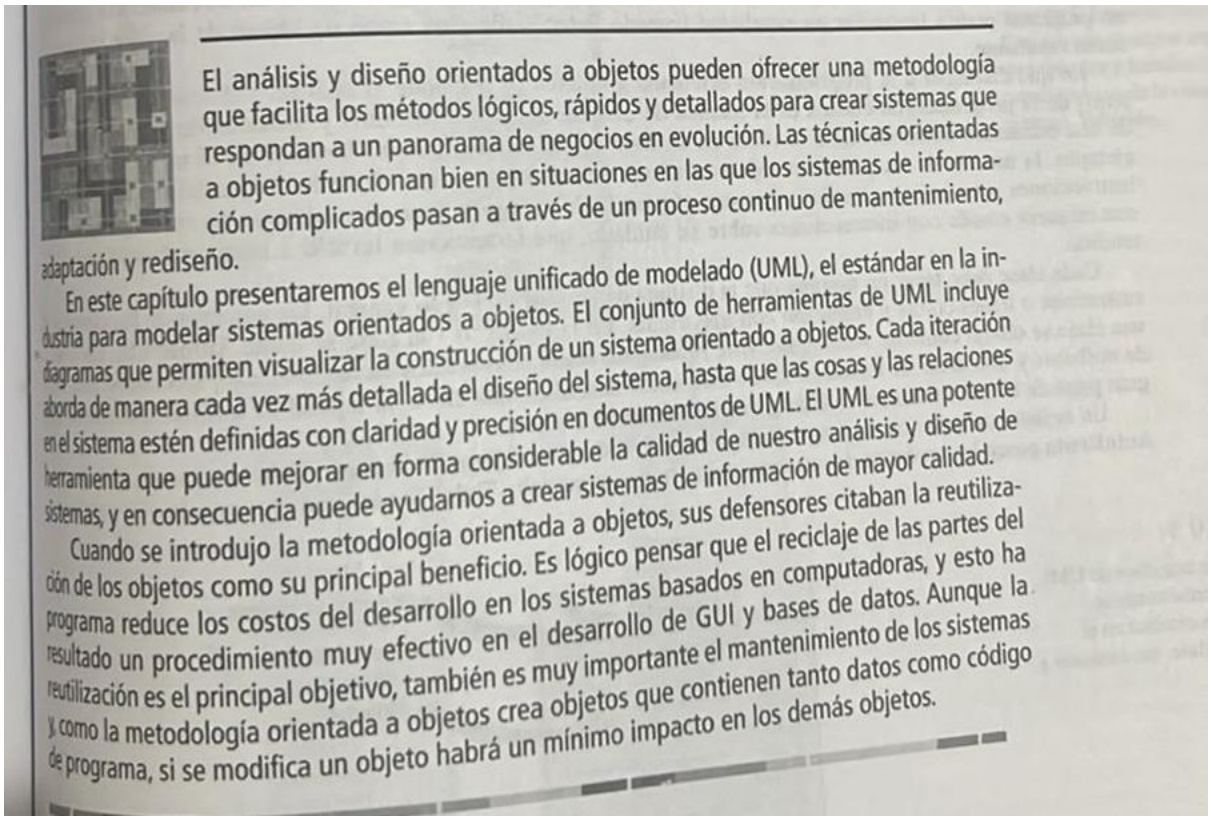
UNIDAD 2: DISEÑO ORIENTADO A OBJETOS

ARTEFACTOS: DIAGRAMA DE CLASES Y DIAGRAMA DE SECUENCIA



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.

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 pelados" 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.

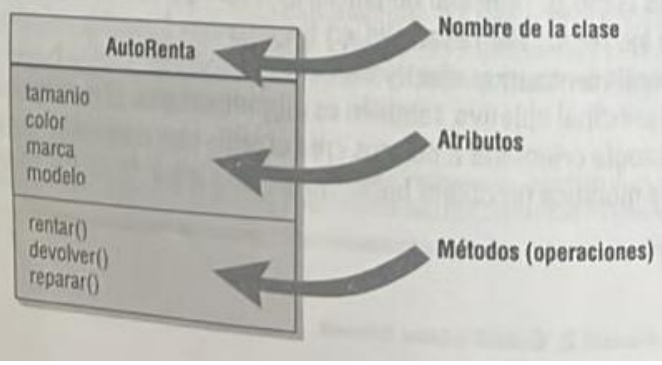
Lo que distingue a la programación orientada a objetos (y por ende al análisis y diseño orientado a objetos) de la programación clásica es la técnica de colocar todos los atributos y métodos de un objeto dentro de una estructura autocontenida, la clase en sí. Éste es un acontecimiento familiar en el 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 tamanio, color, marca y modelo. Todos los automóviles poseen estos atributos

A 10.1

lo de una clase de UML, describe como un lo que consiste en el de la clase, sus atributos y



estas propiedades. [...] por lo general la primera letra va en minúscula. [...] más específico sobre el rango de valores para [...] puede ser azul, blanco

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).

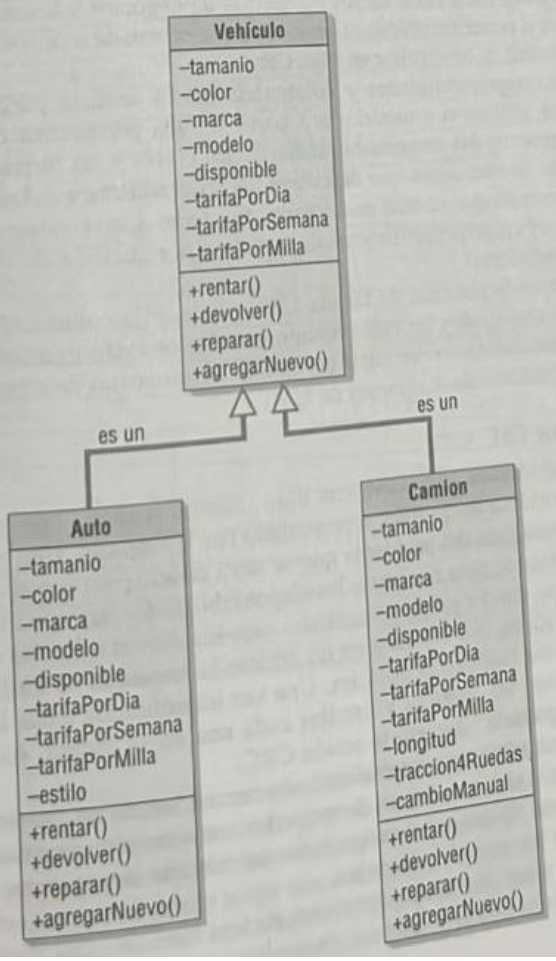


FIGURA 10.2
Un diagrama de clases que muestra la herencia. Auto y Camion son ejemplos específi de vehículos y heredan las características de la clase má general, **Vehículo**.

se pueden utilizar para descri
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 Interfaces Colaboraciones Casos de uso Clases activas Componentes Nodos
	Cosas de comportamiento	Interacciones Máquinas de estado
	Cosas de agrupamiento	Paquetes
	Cosas de anotaciones	Notas
Relaciones	Relaciones estructurales	Dependencias Agregaciones Asociaciones Generalizaciones
	Relaciones de comportamiento	Comunica Incluye Extiende Generaliza
Diagramas	Diagramas estructurales	Diagramas de clases Diagramas de componentes Diagramas de despliegue
	Diagramas de comportamiento	Diagramas de casos de uso Diagramas de secuencia Diagramas de comunicación Diagramas de estados Diagramas de actividad

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.

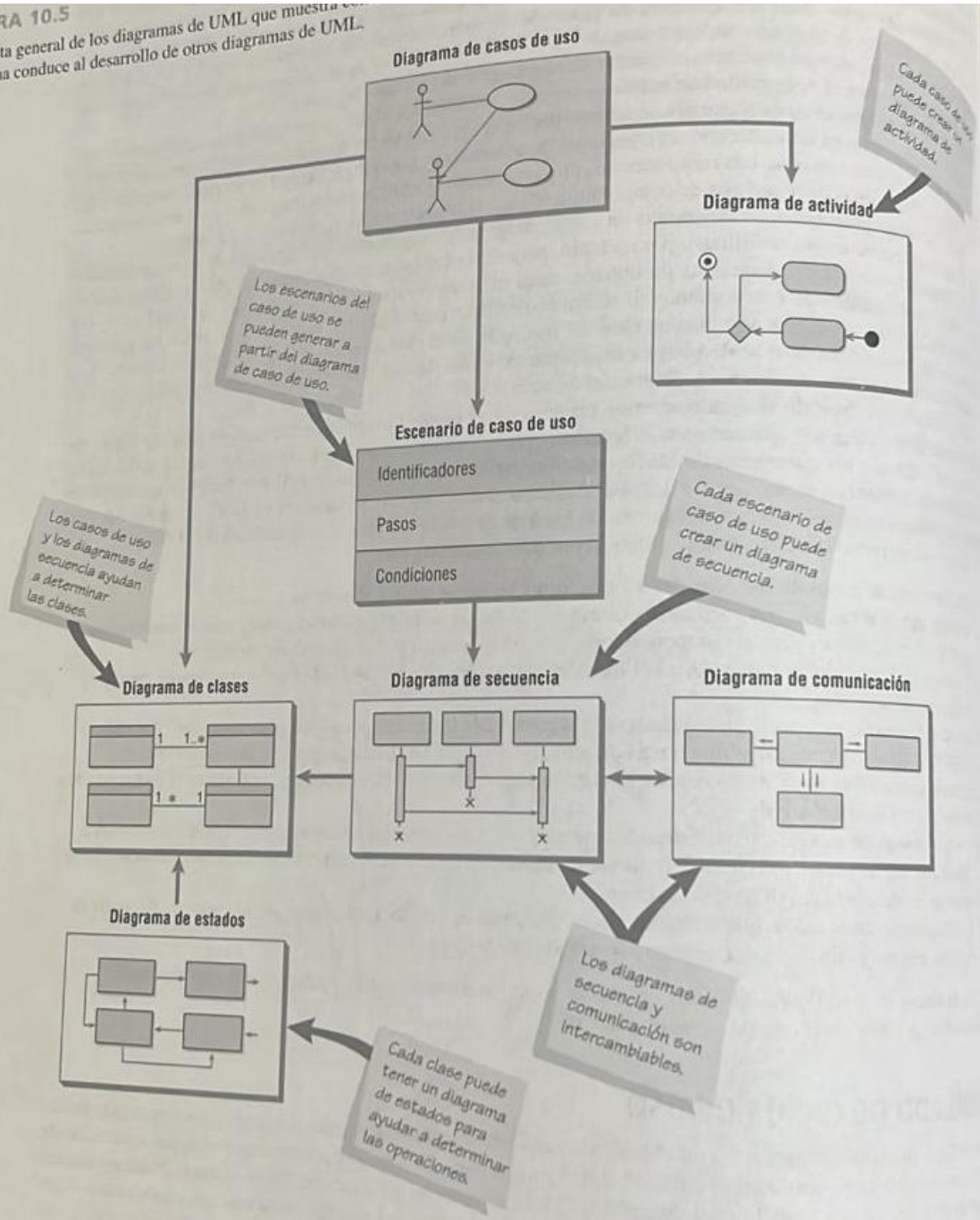
MODELADO DE CASOS DE USO

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
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. Cab
mencionar que se representan sólo las funciones más importantes. El caso de uso **Agregar estudiante** no indi
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 sis
tema y no debiera incluirse en el diagrama de casos de uso, pero como esta información cambia con frecuencia, la ad

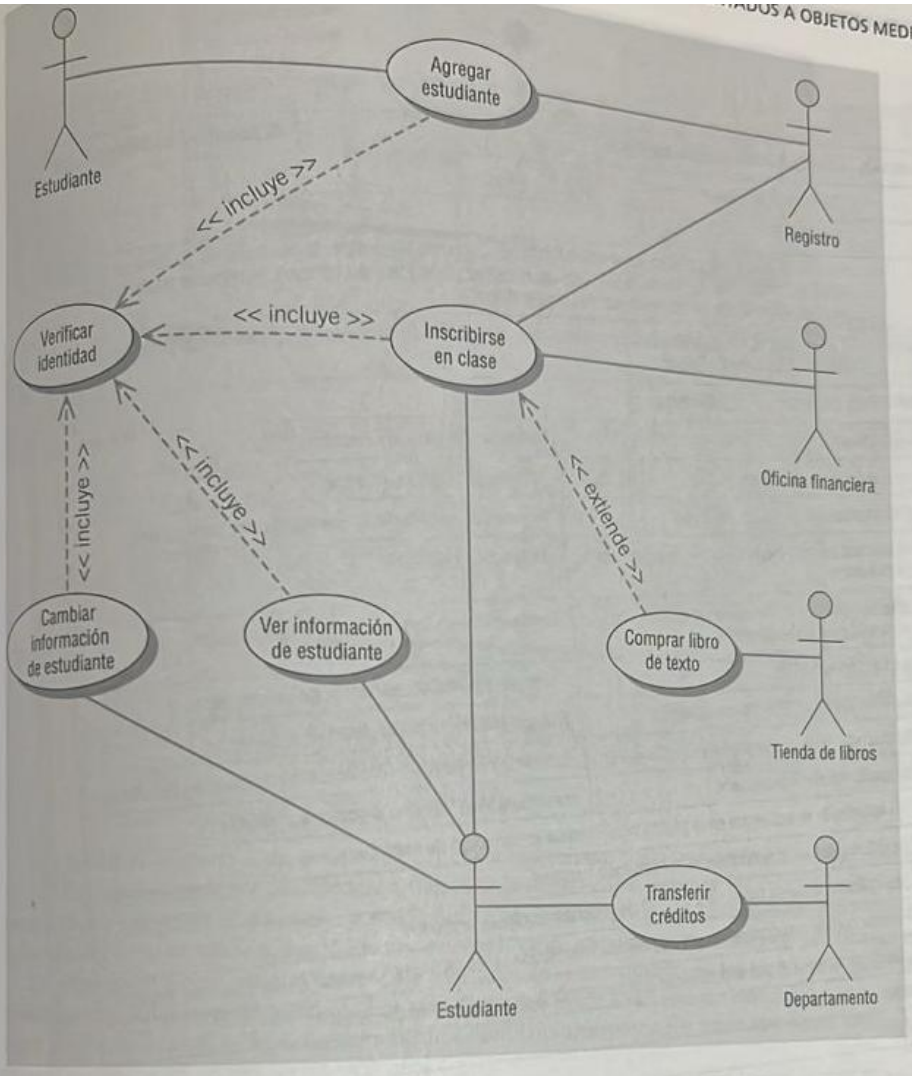


FIGURA 10.6
Ejemplo de casos de uso sobre la inscripción de estudiantes.

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.

Nombre del caso de uso:	Cambiar información de estudiante	**ID única:** Estudiante UC 005
Área:	Sistema de estudiantes	
Actor(es):	Estudiante	
Descripción:	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.	
Evento desencadenador:	El estudiante usa el sitio Web Cambiar información de estudiante, introduce el ID y contraseña de estudiante y hace clic en el botón **Enviar**.	
Tipo de desencadenador:	☒ Externo ☐ Temporal	
Pasos realizados (ruta principal)		**Información para los pasos**
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 **Enviar**.		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
Pre-condiciones:	El estudiante se encuentra en la página Web Cambiar información de estudiante.	
Post-condiciones:	El estudiante cambió con éxito su información personal.	
Suposiciones:	El estudiante tiene un navegador y un ID de usuario y contraseña válidos.	
Requerimientos cumplidos:	Permitir que los estudiantes puedan cambiar su información personal mediante el uso de un sitio Web seguro.	
Cuestiones pendientes:	¿Hay que controlar el número de veces que se permite a un estudiante iniciar sesión?	
Prioridad:	Media	
Riesgo:	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—.

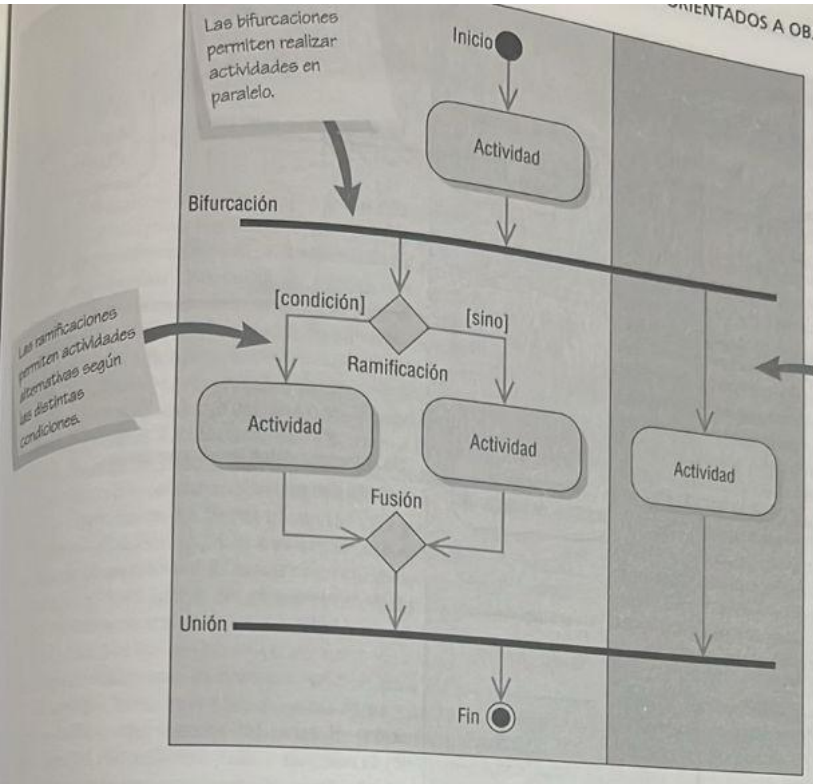


FIGURA 10.8
Se utilizan símbolos especializados para dibujar un diagrama de actividad.

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.

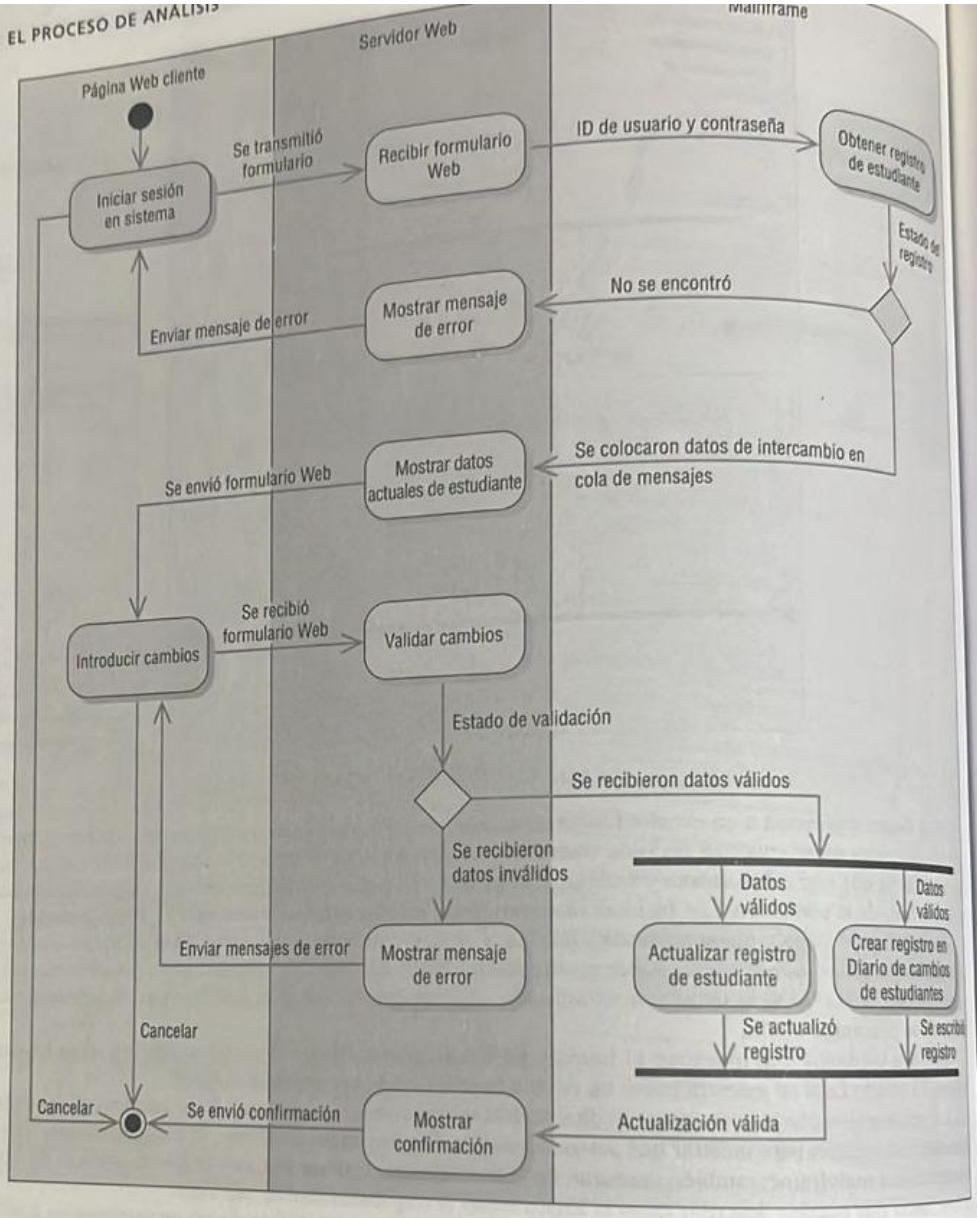


FIGURA 10.9
Este diagrama de actividad muestra tres carriles: Página Web cliente, Servidor Web y Mainframe.

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**.

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.

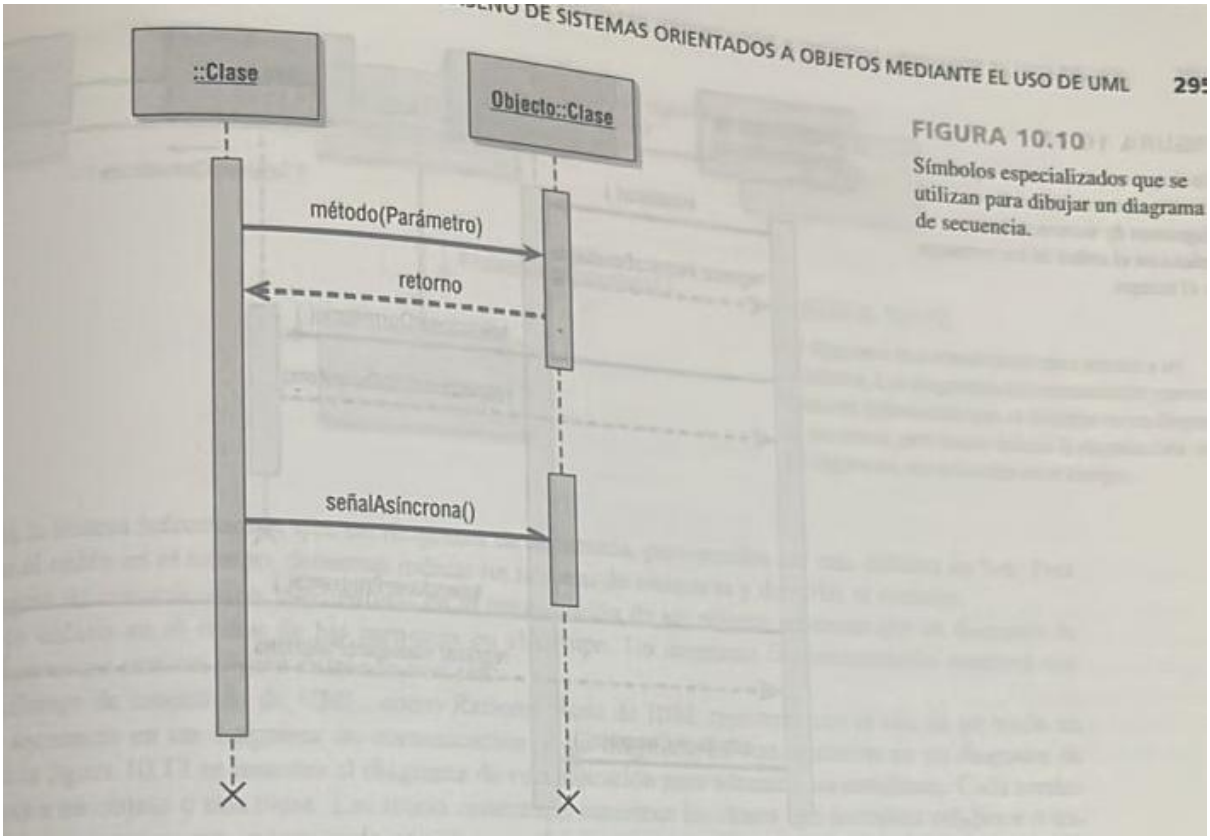


FIGURA 10.10
Símbolos especializados que se utilizan para dibujar un diagrama de secuencia.

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

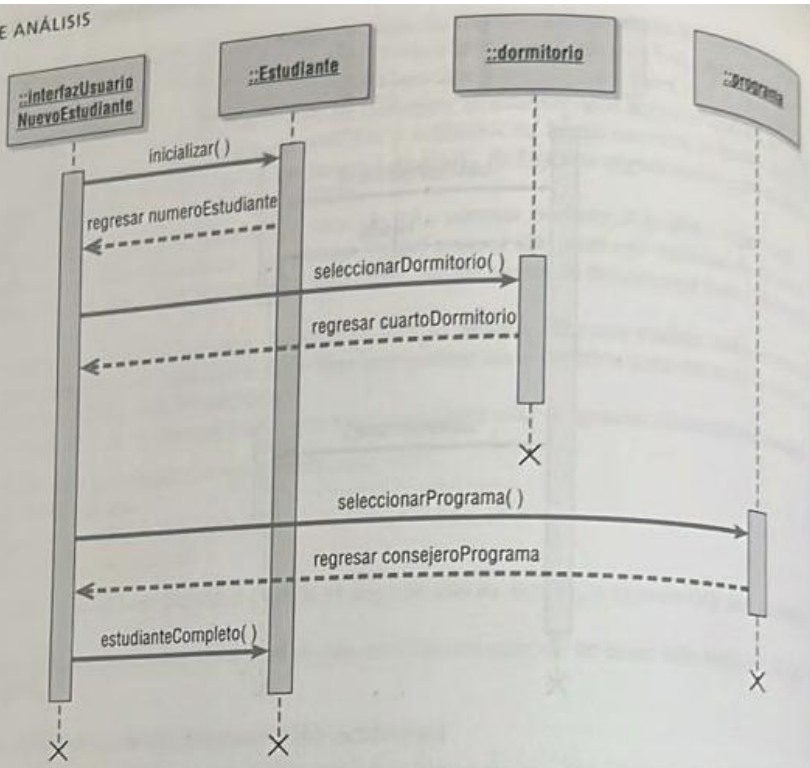


FIGURA 10.11
Un diagrama de secuencia para admitir a un estudiante. Los diagramas de secuencia hacen énfasis en el orden de los mensajes en el tiempo.

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-

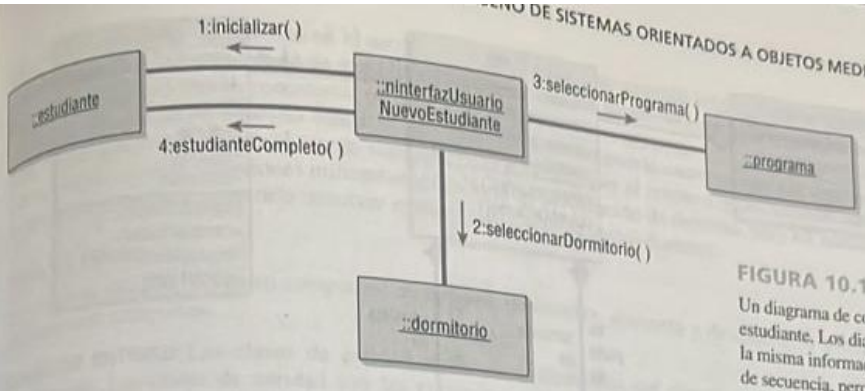


FIGURA 10.12
Un diagrama de comunicación para admitir a un estudiante. Los diagramas de comunicación muestran la misma información que se describe en un diagrama de secuencia, pero hacen énfasis la organización de los objetos en vez del orden en el tiempo.

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 subclases 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.

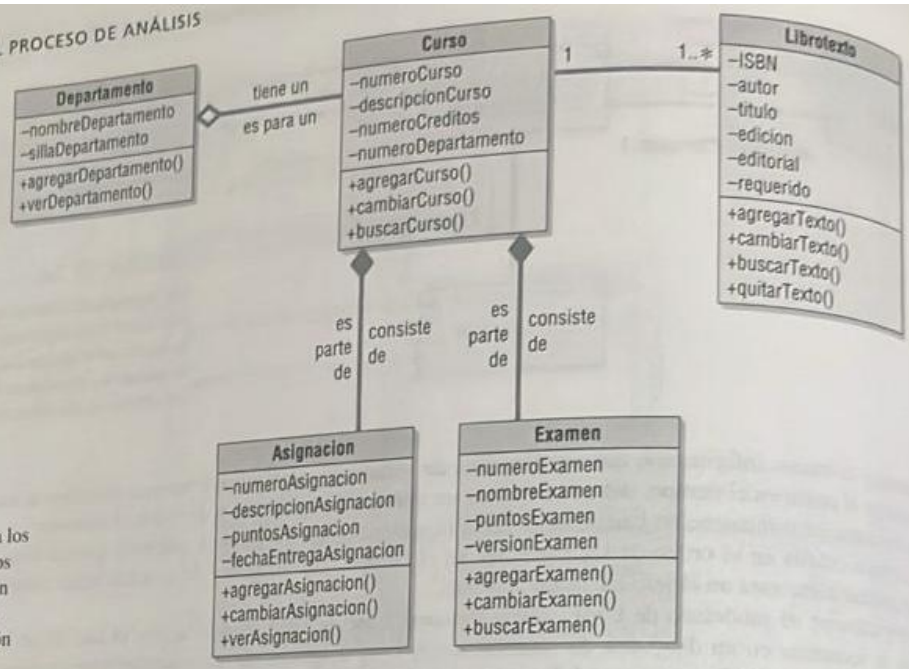


FIGURA 10.13
Un diagrama de clases para los ofrecimientos de cursos. Los diamantes rellenos muestran agregación y los diamantes vacíos muestran una relación entre un todo y sus partes.

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}**.

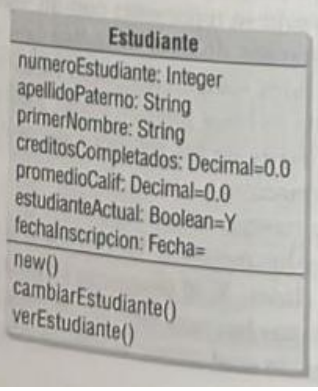
El ocultamiento de información significa que los métodos de los objetos deben estar disponibles para otras clases, por lo que comúnmente los métodos son públicos, lo cual significa que se pueden invocar desde otras clases. En un diagrama de clases, los mensajes públicos (al igual que los atributos públicos) 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.

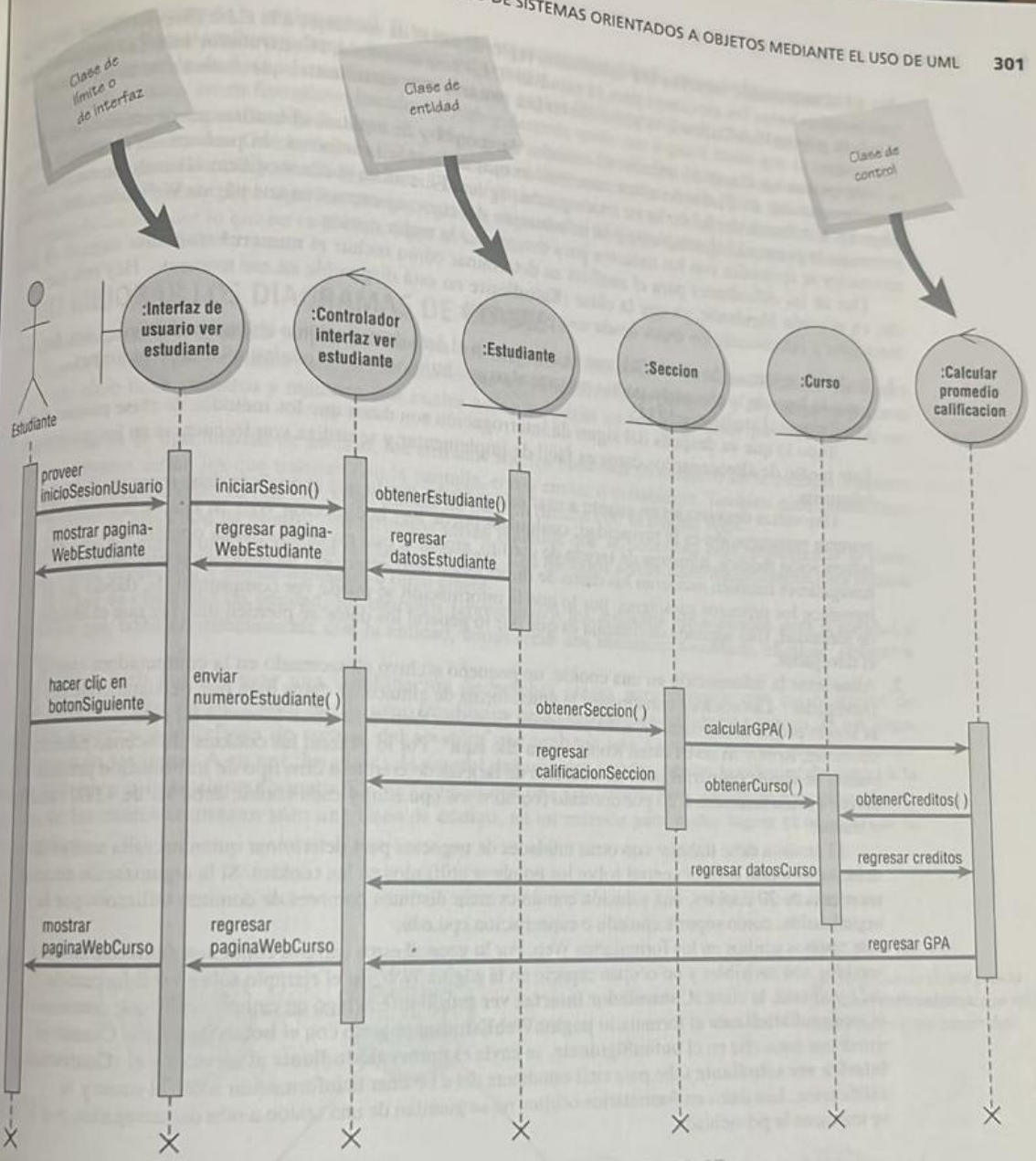
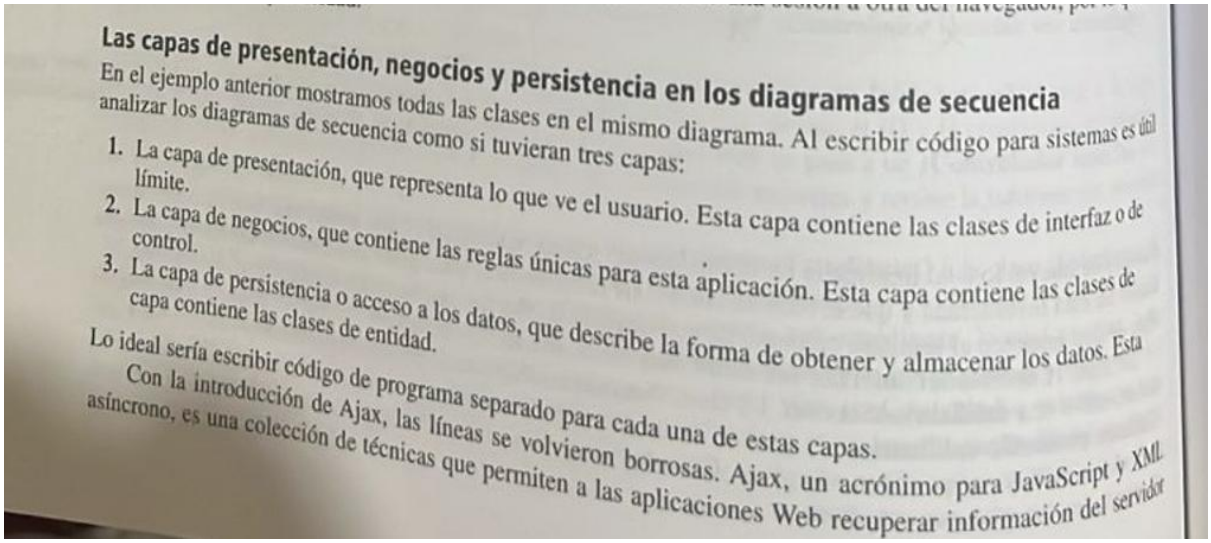


FIGURA 10.15
Un diagrama de secuencia para usar dos páginas Web: una para la información del estudiante y la otra para la información del curso.

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**.



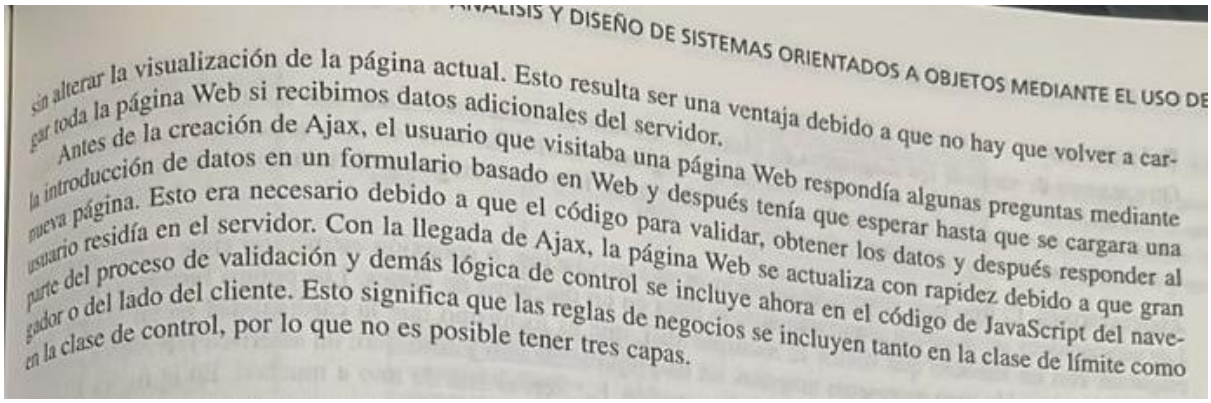
Las capas de presentación, negocios y persistencia en los diagramas de secuencia

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.

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.

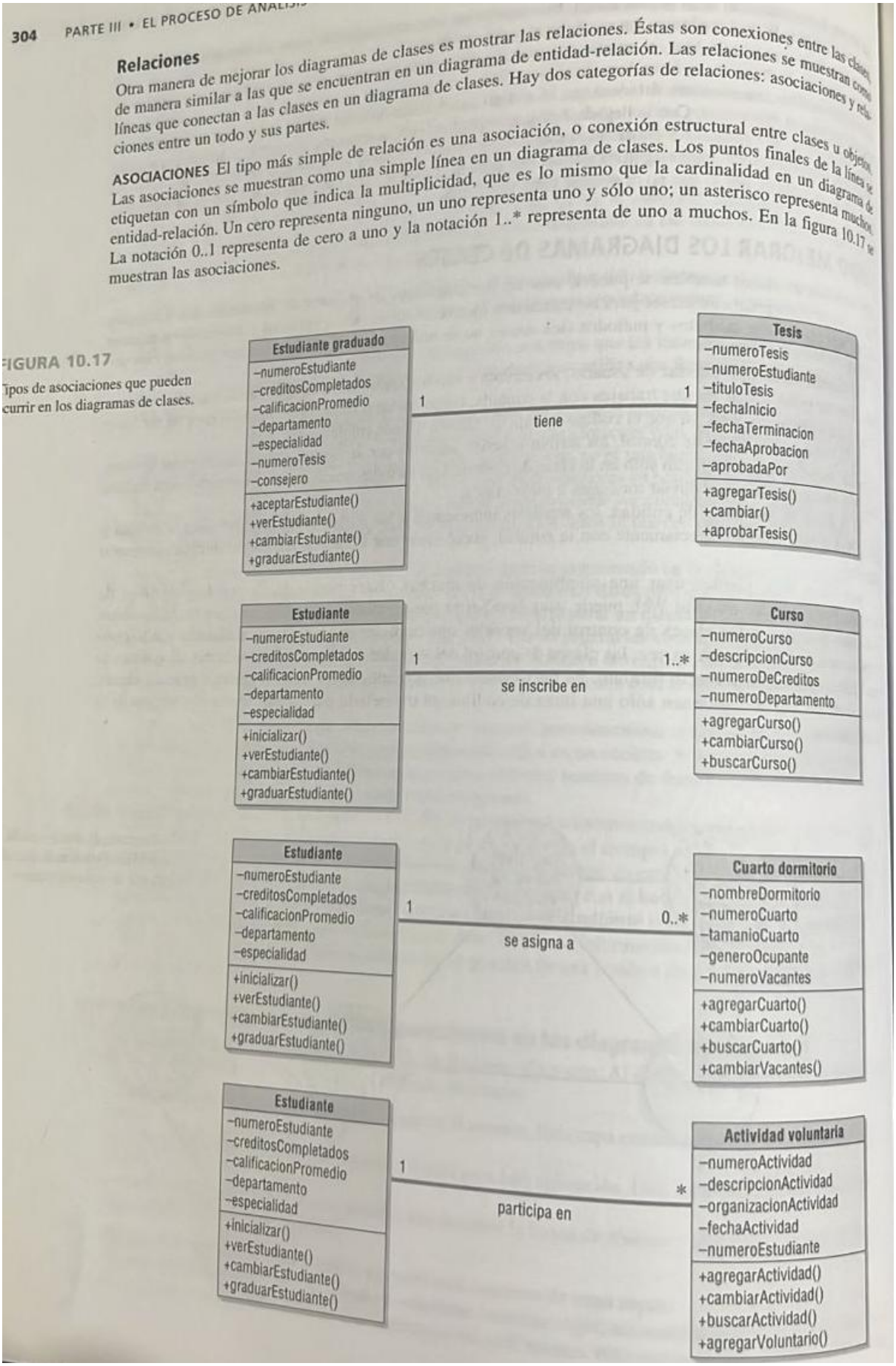


FIGURA 10.17
Tipos de asociaciones que pueden ocurrir en los diagramas de clases.

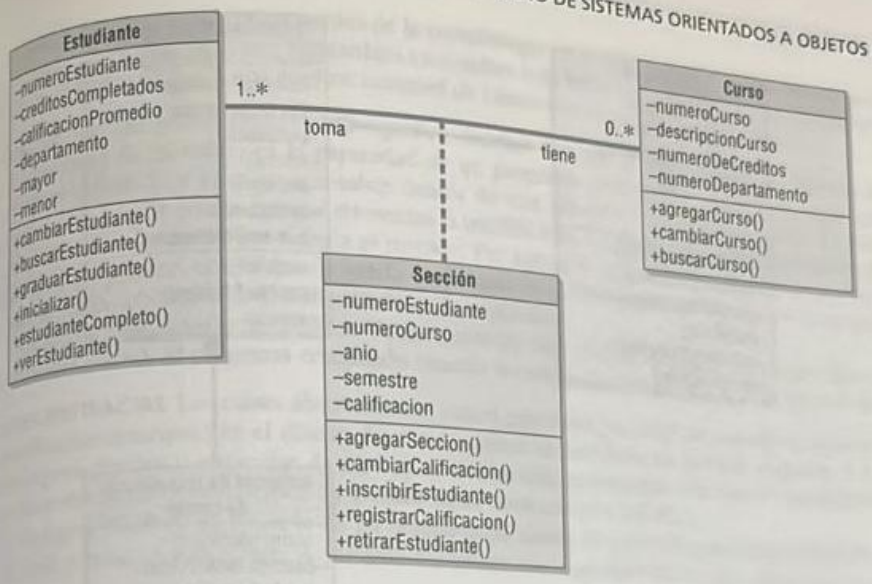


FIGURA 10.18
Un ejemplo de una clase asociativa en la que una sección específica define la relación entre un estudiante y un curso.

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

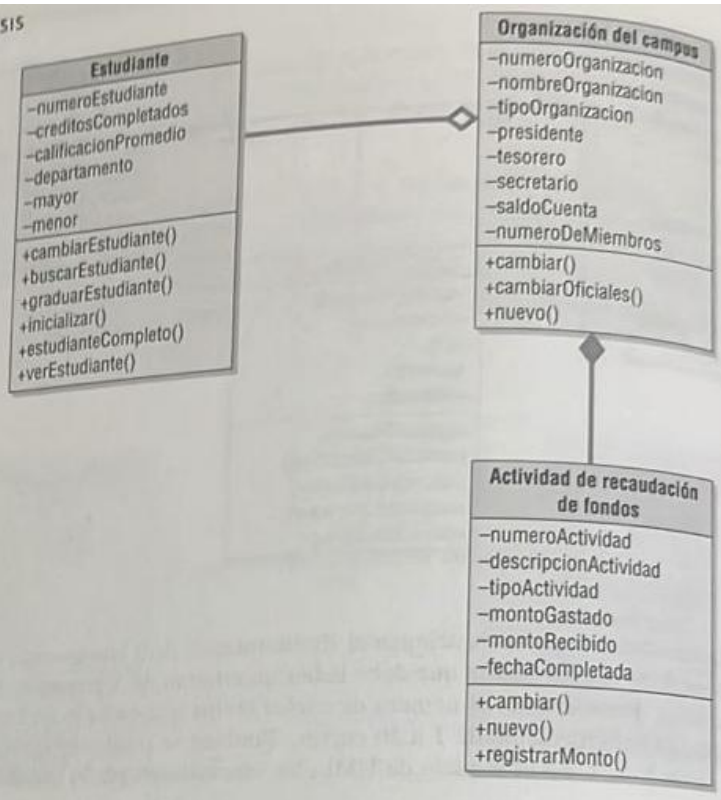


FIGURA 10.19
Un ejemplo de relaciones entre un todo y sus partes y de agregación.

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

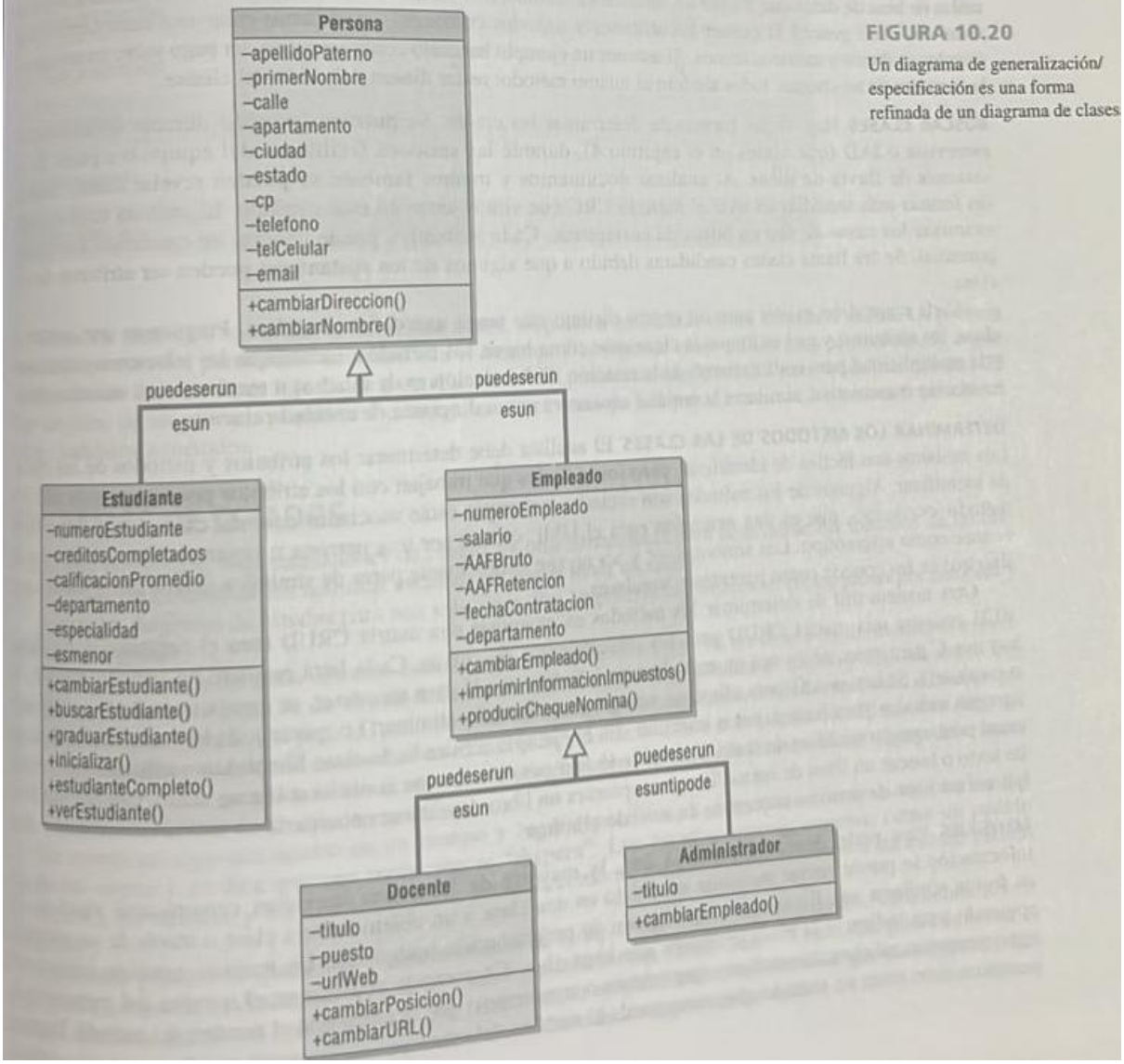


FIGURA 10.20
Un diagrama de generalización/especificación es una forma refinada de un diagrama de clases

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, no 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 dos 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 dos tintos atributos y métodos para cada una de estas clases especializadas.

Las subclases cuentan con verbos especiales para definirlas. 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*.

esun	Docente *esun* Empleado
esuntipode	Administrador *esuntipode* Empleado
puedeserun	Empleado *puedeserun* 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.

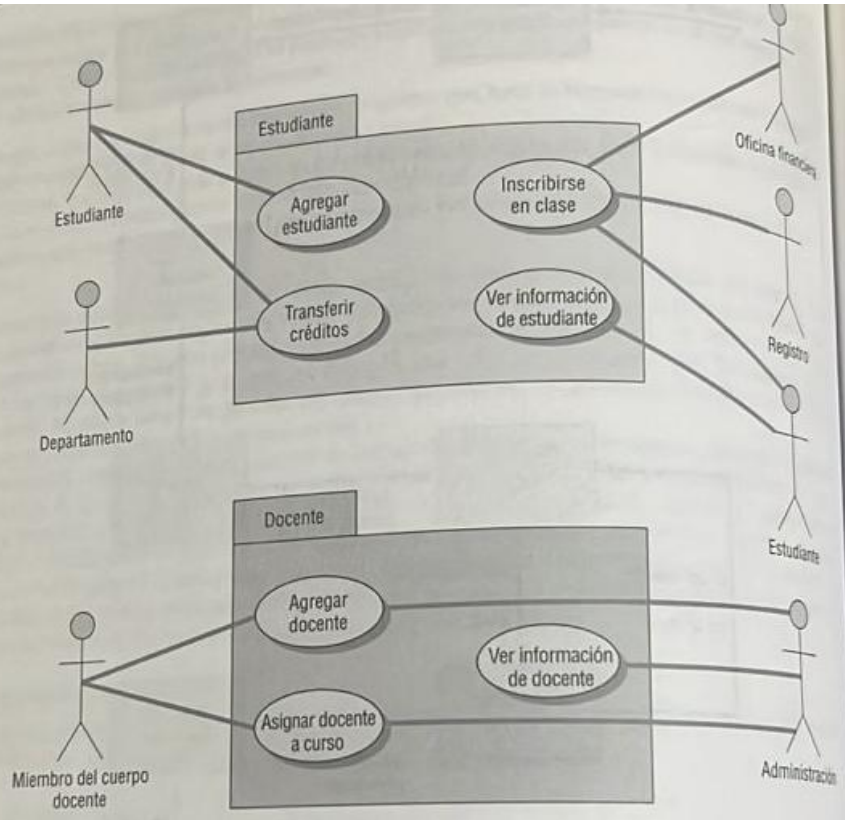
Se crea un diagrama de estados para una sola clase. Por lo general los objetos se crean, pasan por cambios y se eliminan o quitan.

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.

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.