ANÁLISIS Y DISEÑO DE SISTEMAS ORIENTADO A OBJETOS USANDO EL LENGUAJE UNIFICADO DE MODELACIÓN (UML)

18

Objetivos de aprendizaje

Una vez que haya dominado el material de este capítulo, podrá:

1. **Entender lo que es el análisis y diseño de sistemas orientados a objetos y apreciar su utilidad.**
2. **Comprender los conceptos del lenguaje unificado de modelación (UML), el enfoque estándar para modelar un sistema en el mundo orientado a objetos.**
3. **Aplicar los pasos usados en UML para dividir el sistema en un modelo de casos de uso y después en un modelo de clases.**
4. **Diagramar sistemas con el conjunto de herramientas de UML con el fin de que se puedan describir y diseñar apropiadamente.**
5. **Documentar y comunicar el sistema orientado a objetos recién modelado.**

El desafío de desarrollar nuevos sistemas de información para aplicaciones de comercio electrónico, inalámbricas y portátiles en entornos económicos, legales, sociales y físicos dinámicos requiere nuevas técnicas de análisis y diseño. El análisis y diseño orientado a objetos puede ofrecer un enfoque que habilite los métodos lógicos, rápidos y minuciosos necesarios para crear nuevos sistemas en respuesta al cambiante entorno de un negocio. Las técnicas orientadas a objetos son adecuadas en situaciones en que los sistemas de información complicados requieren de mantenimiento, adaptación y rediseño continuos.

Los sistemas orientados a objetos describen las entidades como objetos. Los objetos son parte de un concepto general denominado clases. El deseo de poner elementos en las clases no es nuevo. La descripción del mundo como se ha hecho con los animales, vegetales y minerales es un ejemplo de clasificación, aunque tiene pocas bases científicas. El enfoque científico incluye clases de animales (como mamíferos) y después divide las clases en subclases (como animales ovíparos y marsupiales).

La idea de las clases es tener un punto de referencia y describir las similitudes o diferencias que un objeto específico posee con respecto a los miembros de su propia clase. Con ello, es más eficaz para alguien decir: "El oso koala es un marsupial (o animal con bolsa) con una cabeza redonda y grande y orejas peludas", que describir un oso koala con todas sus características como mamífero. Es más eficaz describir características, apariencia e incluso la conducta de esta manera. Cuando se oye la palabra *reutilizable* en el mundo orientado a objetos, significa que uno puede ser más eficaz, debido a que no es necesario describir un objeto desde el principio cada vez que se necesite para el desarrollo de software.

Por Julie E. Kendall, Kenneth E. Kendall y Allen Schmidt.

Cuando se introdujo por primera vez el enfoque orientado a objetos, sus defensores mencionaron la reusabilidad de objetos como el principal beneficio de su enfoque. Es evidente que el reciclaje de partes de programas debe reducir los costos de desarrollo en los sistemas computacionales. Esto ya ha demostrado su eficacia en el desarrollo de GUIs y bases de datos. Aunque la reusabilidad es la meta principal, el mantenimiento de sistemas también es muy importante, y al crear objetos que contienen datos y código de programación, un cambio en un objeto tiene un impacto mínimo en otros objetos.

En este capítulo presentamos el lenguaje unificado de modelación (UML, por sus siglas en inglés), el estándar de la industria para modelar sistemas orientados a objetos. El conjunto de herramientas UML incluye diagramas que permiten visualizar la construcción de un sistema orientado a objetos. El UML es una herramienta poderosa que puede mejorar enormemente la calidad del análisis y diseño de sistemas, y contribuir por tanto a crear sistemas de información de alta calidad.

Con el uso iterativo de UML es posible lograr una mayor comprensión entre los equipos de negocios y los de TI en relación con los requerimientos del sistema y los procesos que necesitan realizarse en este último para cumplir dichos requerimientos. En cada iteración el diseño del sistema toma una apariencia más detallada hasta que las cosas y relaciones en el sistema se definen con claridad y precisión en los documentos de UML. Las características más importantes de cada fase se podrían definir inicialmente, y después incorporarse en el proceso de desarrollo. Aunque el proceso es iterativo, es importante que quede tan completo como sea posible desde el principio.

Al terminar el análisis y diseño, se tendría un conjunto preciso y detallado de especificaciones para las clases, procesos y otros artefactos del sistema, lo cual contribuye a evitar el costo de volver a codificar a causa de una pobre planeación inicial. Un artefacto es un término general que se utiliza para describir cualquier pieza de información usada o producida al desarrollar sistemas. Podría ser un diagrama, texto descriptivo, instrucciones de usuario, métodos del código, programas o cualquier otro componente del sistema.

CONCEPTOS ORIENTADOS A OBJETOS

La programación orientada a objetos difiere de la programación por procedimientos tradicional, pues examina los objetos que son parte de un sistema. Cada objeto es una representación en computadora de alguna cosa o evento real. En esta sección se presentan descripciones generales de los principales conceptos orientados a objetos de las clases, la herencia y los objetos. En secciones posteriores de este mismo capítulo se ofrece más información de otros conceptos de UML.

OBJETOS

Los objetos son personas, lugares o cosas que son relevantes para el sistema bajo análisis. Los objetos podrían ser clientes, artículos, pedidos, etc. Los objetos también podrían ser pantallas GUI o áreas de texto en la pantalla.

CLASES

Los objetos se representan y agrupan en clases que son óptimas para reutilizarse y darles mantenimiento. Una clase define el conjunto de atributos y comportamientos compartidos por cada objeto de la clase. Por ejemplo, los registros de los estudiantes en la sección de un curso almacenan información similar para cada estudiante. Se podría decir que los estudiantes constituyen una clase. Los valores podrían ser diferentes para cada estudiante, pero el tipo de información es el mismo. Los programadores deben definir las diversas clases en el programa que escriben. Cuando el programa corre, los objetos se pueden crear a partir de la clase establecida. El término *instanciar* se usa cuando un objeto se crea a partir de una clase. Por ejemplo, un programa podría instanciar a un estudiante llamado Peter Wellington como un objeto de la clase denominada estudiante.

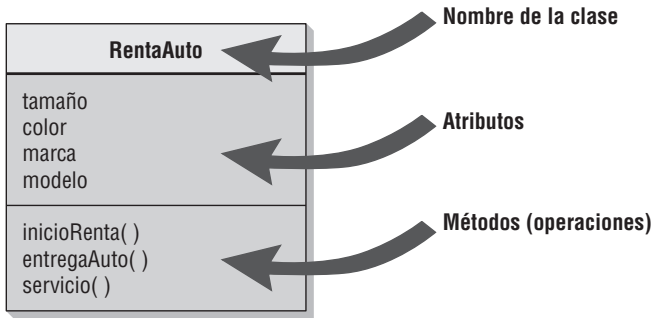


FIGURA 18.1
Ejemplo de una clase de UML. Una clase se describe como un rectángulo que consiste de nombre de la clase, atributos y métodos.

Lo que hace a la programación orientada a objetos, y por consiguiente al análisis y diseño orientado a objetos, diferente de la programación clásica, es la técnica de poner todos los atributos y métodos de un objeto en una estructura independiente, la propia clase. Ésta es una situación común en el mundo físico. Por ejemplo, un paquete con harina para pastel empacado es similar a una clase ya que contiene los ingredientes y las instrucciones para mezclar y hornear el pastel. Un suéter de lana es similar a una clase porque incluye una etiqueta con instrucciones del cuidado que advierten que se debe lavarlo a mano y ponerlo a secar extendido.

Cada clase debe tener un nombre que la distinga de todas las demás. Los nombres de clase normalmente son sustantivos o frases cortas y empiezan con una letra mayúscula. En la figura 18.1 la clase se llama **RentaAuto**. En el UML, una clase se representa como un rectángulo. El 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 es una parte principal de lo que llamamos análisis y diseño orientado a objetos.

Un atributo describe alguna propiedad de todos los objetos de la clase. Observe que la clase **RentaAuto** posee los atributos tamaño, color, marca y modelo. Todos los automóviles poseen estos atributos, pero los atributos de cada automóvil tendrán diferentes valores. Por ejemplo, un automóvil puede ser azul, blanco o de algún otro color. Más adelante demostraremos que es posible ser más específico acerca del rango de valores para estas propiedades. Al especificar atributos, normalmente la primera letra es minúscula.

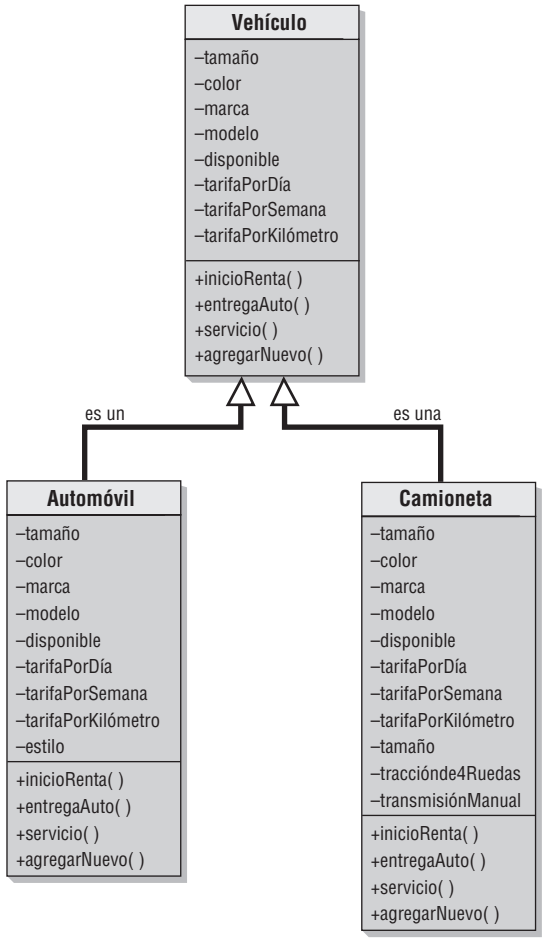
Un método es una acción que se puede solicitar a cualquier objeto de la clase. Los métodos son los procesos que una clase sabe cómo realizar. Los métodos también se llaman operaciones. La clase **RentaAuto** podría tener los siguientes métodos: **inicioRenta( )**, **entregaAuto( )** y **servicio( )**. Al especificar métodos, normalmente la primera letra es minúscula.

HERENCIA

Otro concepto importante de los sistemas orientados a objetos es la herencia. Las clases pueden tener hijos; es decir, una clase se puede crear a partir de otra clase. En el UML, la clase original —o madre— se conoce como clase base. La clase hija se denomina clase derivada. Ésta se puede crear de tal manera que herede todos los atributos y comportamientos de la clase base. Sin embargo, una clase derivada podría tener atributos y comportamientos adicionales. Por ejemplo, podría haber una clase **Vehículo** para una compañía de renta de automóviles que contenga atributos como **tamaño**, **color** y **marca**. Como se muestra en la figura 18.2, las clases derivadas podrían ser **Automóvil** o **Camioneta**.

La herencia reduce el trabajo de la programación usando fácilmente objetos comunes. El programador sólo necesita declarar que la clase **Automóvil** hereda de la clase **Vehículo** y después proporcionar cualesquier detalles adicionales sobre nuevos atributos o comportamientos que sean únicos para un automóvil. Todos los atributos y comportamientos de la clase **Vehículo** son automática e implícitamente parte de la clase **Automóvil** y no requieren ninguna programación adicional. Esto le permite al analista definir una sola vez pero usar muchas veces y es similar a los datos que están en la tercera forma normal, definidos una sola vez en una tabla de la base de datos (como se analizó en el capítulo 13).

FIGURA 18.2
Diagrama de clases que muestra la herencia. Automóvil y Camioneta son ejemplos específicos de vehículos y heredan las características de la clase más general, **Vehículo**.



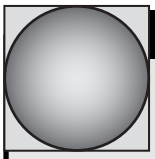
En la figura 18.2, los atributos son precedidos por signos de resta y los métodos por signos de suma. Explicaremos esto con mayor detalle más adelante en este capítulo, pero por ahora tome nota de que los signos de resta significan que estos atributos son privados (no compartidos con otras clases) y estos métodos son públicos (podrían ser invocados por otras clases).

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

TARJETAS CRC Y PENSAMIENTO EN OBJETOS

Ahora que hemos cubierto los conceptos fundamentales del análisis y diseño de sistemas orientados a objetos, necesitamos examinar las formas de crear clases y objetos a partir de los problemas de negocios y sistemas que estamos enfrentando. Una forma de empezar a establecer el enfoque orientado a objetos es comenzar a pensar y hablar de esta nueva forma. Un enfoque conveniente es desarrollar tarjetas CRC.

CRC significa clase, responsabilidades y colaboradores. El analista puede usar estos conceptos cuando empiece a hablar del, o a modelar el, sistema desde una perspectiva orientada a objetos. Las tarjetas CRC se usan para representar las responsabilidades de las clases y sus interacciones. Los analistas crean las tarjetas con base en escenarios que delinean los requerimientos del sistema. Estos escenarios modelan el comportamiento del sistema que se está estudiando. Si se van a utilizar en grupo, las tarjetas CRC se pueden crear manualmente en pequeñas tarjetas de notas o se pueden crear en una computadora.



OPORTUNIDAD DE CONSULTORÍA 18.1

HACIENDO LA BOBINA MÁGICA*

Fred y Ginger, propietarios de la cadena de tiendas FilmMagic (que renta vídeos, DVDs y videojuegos), siempre se han interesado por la nueva tecnología. Debido a que continuamente incorporan nuevos productos para rentar (como DVDs y nuevos juegos para PlayStation II), su negocio ha crecido con éxito en varias ciudades.

Puesto que la casa donde usted vive está cerca de la tienda original de Fred y Ginger, es amigo de ellos desde hace 12 años cuando comenzaron su negocio, rentándoles cintas cuando pasaron de la pantalla grande a los vídeos. Con frecuencia intercambian opiniones acerca de las películas "rompe récord" y las que son "una pérdida de tiempo".

En vista de que les comentó acerca de los enfoques orientados a objetos que ha estado aprendiendo, ellos desean que usted analice su negocio con este enfoque. En la figura 7.15 se encuentra un resumen de las actividades de negocios de FilmMagic. Tome en cuenta también la serie de diagramas de flujo de datos que hay en ese capítulo, con el fin de conceptualizar el problema y empezar la transición al pensamiento orientado a objetos.

Como usted tiene tan buena amistad con Fred y Ginger, y como no le vendrá mal una poca de experiencia práctica en el pensamiento orientado a objetos, se compromete a aplicar sus conocimientos y darles un informe. Una vez que haya vuelto a leer las actividades de negocios de FilmMagic, realice una revisión detallada mediante las siguientes tareas:

- Use la técnica de las tarjetas CRC para listar las clases, responsabilidades y colaboradores.
- Use la técnica del pensamiento orientado a objetos para listar "conocimientos" y atributos correspondientes para los objetos de las clases que haya identificado en la etapa anterior.

Escriba ambos pasos y dé una vuelta por las oficinas centrales de FilmMagic con su informe a la mano. Obviamente, Fred y Ginger esperan una revisión minuciosa.

*Basado en un problema escrito por el doctor Ping Zhang.

Hemos agregado dos columnas a la plantilla original de la tarjeta CRC: la columna Pensamiento del Objeto y la columna Propiedad. Los enunciados del pensamiento del objeto se escriben en español sencillo, y el nombre de la propiedad, o atributo, se introduce en su propio lugar. El propósito de estas columnas es clarificar el pensamiento y ayudar a crear los diagramas de UML.

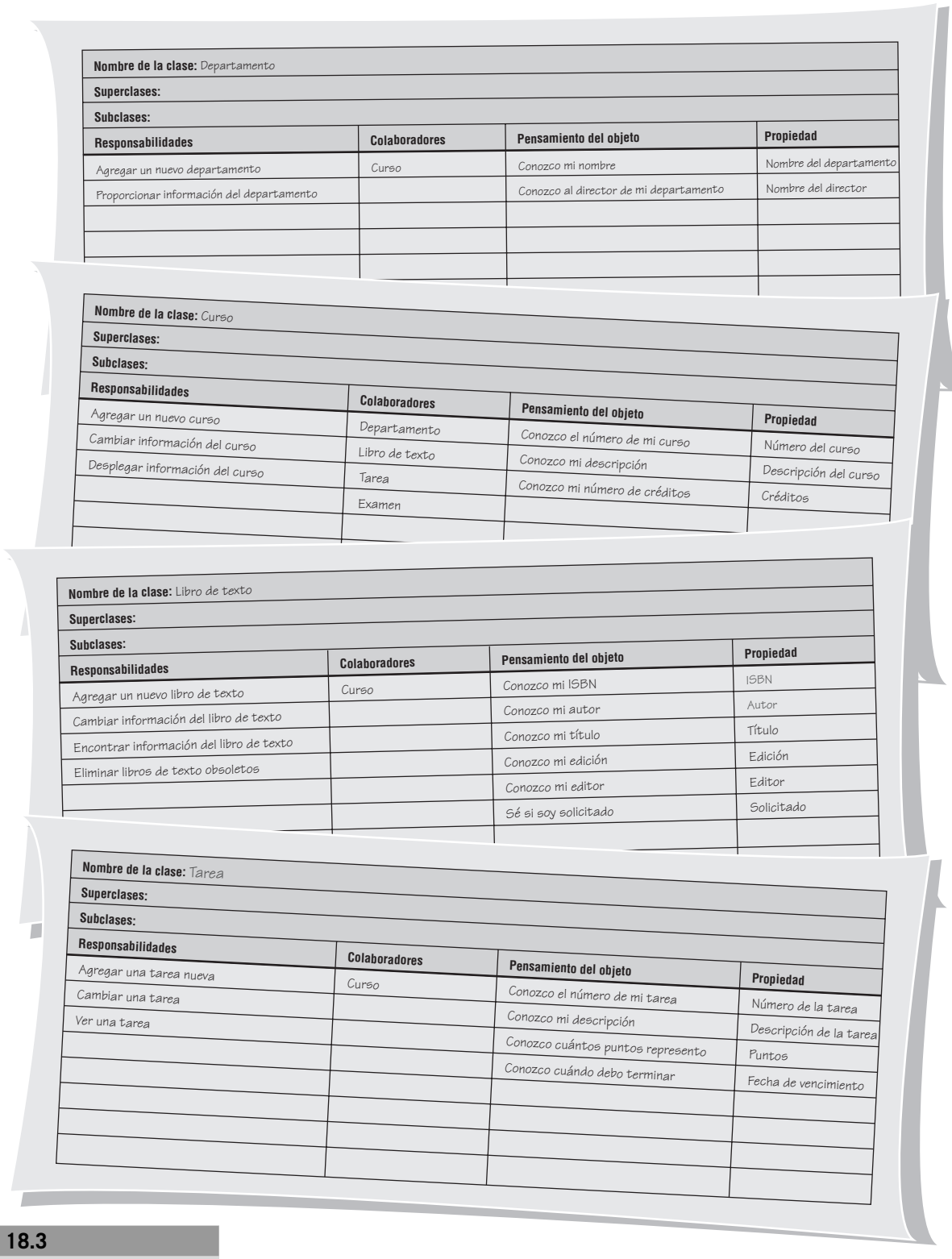
INTERACCIÓN DURANTE UNA SESIÓN DE CRC

Las tarjetas CRC se pueden crear en forma interactiva con un puñado de analistas que pueden trabajar en conjunto para identificar la clase en el dominio del problema que tenga el negocio. Una sugerencia es encontrar todos los sustantivos y verbos en un enunciado del problema que se ha creado para entender el problema. Normalmente los sustantivos indican las clases en el sistema y las responsabilidades pueden identificarse mediante los verbos.

Con su grupo de analistas, realice sesiones de lluvia de ideas para identificar todas las clases. Siga el formato estándar de la lluvia de ideas, consistente en no criticar las respuestas de los participantes, sino en estimular tantas respuestas como sea posible. Una vez que se han identificado todas las clases, los analistas pueden reunirlas, eliminar las ilógicas y escribir cada una en su propia tarjeta. Asigne una clase a cada persona del grupo, quien la "poseerá" durante la sesión de CRC.

A continuación, el grupo crea escenarios, que en realidad son repasos estructurados de las funciones del sistema, tomando la funcionalidad deseada del documento de requerimientos previamente creado. Primero se deben considerar los métodos de sistemas típicos. Las excepciones, tales como la recuperación de errores, se deberán analizar después de que se hayan cubierto los métodos rutinarios.

Conforme el grupo decide qué clase es responsable de una función particular, el analista que posee la clase para la sesión toma esa tarjeta y declara: "Necesito cumplir mi responsabilidad". Cuando una tarjeta se sostiene en el aire, se considera un objeto y puede realizar acciones. Entonces el grupo procede a refinar la responsabilidad en tareas más y más pequeñas, si es posible. Si es conveniente, el objeto puede cumplir estas tareas, o el grupo puede decidir que se pueden cumplir interactuando con otras cosas. Si no existen otras clases apropiadas, será necesario que el grupo las genere.



Nombre de la clase: Departamento
Superclases:
Subclases:

Responsabilidades	Colaboradores	Pensamiento del objeto	Propiedad
Agregar un nuevo departamento	Curso	Conozco mi nombre	Nombre del departamento
Proporcionar información del departamento		Conozco al director de mi departamento	Nombre del director

Nombre de la clase: Curso
Superclases:
Subclases:

Responsabilidades	Colaboradores	Pensamiento del objeto	Propiedad
Agregar un nuevo curso	Departamento	Conozco el número de mi curso	Número del curso
Cambiar información del curso	Libro de texto	Conozco mi descripción	Descripción del curso
Desplegar información del curso	Tarea	Conozco mi número de créditos	Créditos
	Examen		

Nombre de la clase: Libro de texto
Superclases:
Subclases:

Responsabilidades	Colaboradores	Pensamiento del objeto	Propiedad
Agregar un nuevo libro de texto	Curso	Conozco mi ISBN	ISBN
Cambiar información del libro de texto		Conozco mi autor	Autor
Encontrar información del libro de texto		Conozco mi título	Título
Eliminar libros de texto obsoletos		Conozco mi edición	Edición
		Conozco mi editor	Editor
		Sé si soy solicitado	Solicitado

Nombre de la clase: Tarea
Superclases:
Subclases:

Responsabilidades	Colaboradores	Pensamiento del objeto	Propiedad
Agregar una tarea nueva	Curso	Conozco el número de mi tarea	Número de la tarea
Cambiar una tarea		Conozco mi descripción	Descripción de la tarea
Ver una tarea		Conozco cuántos puntos represento	Puntos
		Conozco cuándo debo terminar	Fecha de vencimiento

FIGURA 18.3
Cuatro tarjetas CRC para las ofertas de cursos muestran la manera en que los analistas completan los detalles para las clases, responsabilidades y colaboradores, así como también para los enunciados de pensamiento del objeto y los nombres de propiedad.

Las cuatro tarjetas CRC descritas en la figura 18.3 muestran cuatro clases para ofertas de cursos. Observe que en una clase denominada **Curso**, el analista de sistemas se refiere a cuatro colaboradores: el departamento, el libro de texto, la tarea del curso y el examen del curso. A continuación estos colaboradores se describen como clases en las otras tarjetas CRC.

Posteriormente, las responsabilidades mencionadas se convertirán en lo que en UML se denomina métodos. Los enunciados del pensamiento del objeto parecen elementales, pero tienen un tono informal para animar a los grupos de analistas a describir tantos enunciados como sea posible durante una sesión de CRC. Como se muestra en el ejemplo, todo el diálogo durante una sesión de CRC se lleva a cabo en primera persona, para que incluso el **libro de texto** hable: "Conozco mi ISBN". "Conozco a mi autor". En consecuencia, estos enunciados se pueden usar para describir los atributos en UML. Estos atributos se pueden llamar por sus nombres de variables, como **edición** y **editor**.

CONCEPTOS Y DIAGRAMAS DEL LENGUAJE UNIFICADO DE MODELACIÓN (UML)

Vale la pena investigar y entender el enfoque de UML por su gran aceptación y uso. UML proporciona 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, similar a la forma en que un conjunto de planos permite a las personas visualizar la construcción de un edificio. Ya sea que usted esté trabajando independientemente o con un equipo grande de desarrollo de sistemas, la documentación que crea con UML proporciona un medio eficaz de comunicación entre el equipo de desarrollo y el equipo de negocios en un proyecto.

Como se ilustra en la figura 18.4, UML consiste de cosas, relaciones y diagramas. Los primeros componentes, o elementos principales, de UML se denominan cosas. Quizá usted prefiera otra palabra, como objeto, pero en UML se denominan cosas. Las cosas estructurales son más comunes. Las cosas estructurales son clases, interfaces, casos de uso y muchos otros elementos que proporcionan una forma de crear modelos. Las cosas estructurales permiten al usuario describir relaciones. Las cosas de comportamiento describen cómo funcionan las cosas. Las interacciones y las máquinas de estado son ejemplos de cosas de comportamiento. Las cosas de agrupamiento se usan para definir límites. Un ejemplo de una cosa de agrupamiento es un paquete. Por último, tenemos las cosas de anotación, para que podamos agregar notas a los diagramas.

Las relaciones son el pegamento que une las cosas. Es útil considerar a las relaciones de dos formas. Las relaciones estructurales se usan para enlazar 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 usan 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. Por ejemplo, los diagramas estructurales se usan para describir las relaciones entre las clases. Incluyen diagramas de clases, diagramas de objetos, diagramas de componentes y diagramas de despliegue. Por otro lado, los diagramas de comportamiento se pueden usar para describir la interacción entre las personas (denominadas actores en UML) y la cosa a la que nos referimos como caso de uso, o cómo usan los actores el sistema. Los diagramas de comportamiento incluyen diagramas de caso de uso, diagramas de secuencias, diagramas de colaboración, diagramas de gráfico de estado y diagramas de actividades.

En el resto de este capítulo analizaremos primero el modelado de casos de uso, la base para todas las técnicas de UML. Después, veremos cómo se emplea un caso de uso para derivar actividades, secuencias y clases —los diagramas de UML que más se utilizan—. Debido a que todos los libros se dedican a la sintaxis y uso de UML (el documento de especificaciones de UML contiene alrededor de 800 páginas), sólo proporcionamos un breve resumen de los aspectos más valiosos y comúnmente usados de UML.

FIGURA 18.4
Vista general de UML y sus componentes: cosas, relaciones y diagramas.

Categoría 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 anotación	Notas
Relaciones	Relaciones estructurales	Dependencias Agregaciones Asociaciones Generalizaciones
	Relaciones de comportamiento	Comunica Incluye Extiende Generaliza
Diagramas	Diagramas estructurales	Diagramas de clase Diagramas de componentes Diagramas de despliegue
	Diagramas de comportamiento	Diagramas de caso de uso Diagramas de secuencias Diagramas de colaboración Diagramas de gráfico de estado Diagramas de actividades

Los seis diagramas de UML que más se utilizan son:

1. Diagrama de caso de uso, que describe cómo se usa el sistema. Los analistas empiezan con un diagrama de caso de uso.
2. Escenario de caso de uso (aunque técnicamente no es un diagrama), es una descripción verbal de las excepciones para el comportamiento principal descrito por el caso de uso principal.
3. Diagrama de actividades, ilustra el flujo general de actividades. Cada caso de uso podría crear un diagrama de actividades.
4. Diagramas de secuencias, muestran la secuencia de actividades y las relaciones de las clases. Cada caso de uso podría crear uno o más diagramas de secuencias. Una alternativa para un diagrama de secuencias es un diagrama de colaboración, el cual contiene la misma información en formato diferente.
5. Diagramas de clases, muestran las clases y las relaciones. Los diagramas de secuencias se usan (junto con las tarjetas CRC) para determinar las clases. Un vástago de un diagrama de clases es un diagrama gen/esp (que significa generalización/especialización).
6. Diagramas de gráfico de estado, muestra las transiciones de estado. Cada clase podría crear un diagrama de gráfico de estado, el cual es útil para determinar los métodos de la clase.

En la figura 18.5 se ilustra cómo se relacionan entre sí estos diagramas. En las siguientes secciones discutiremos cada uno de estos diagramas.

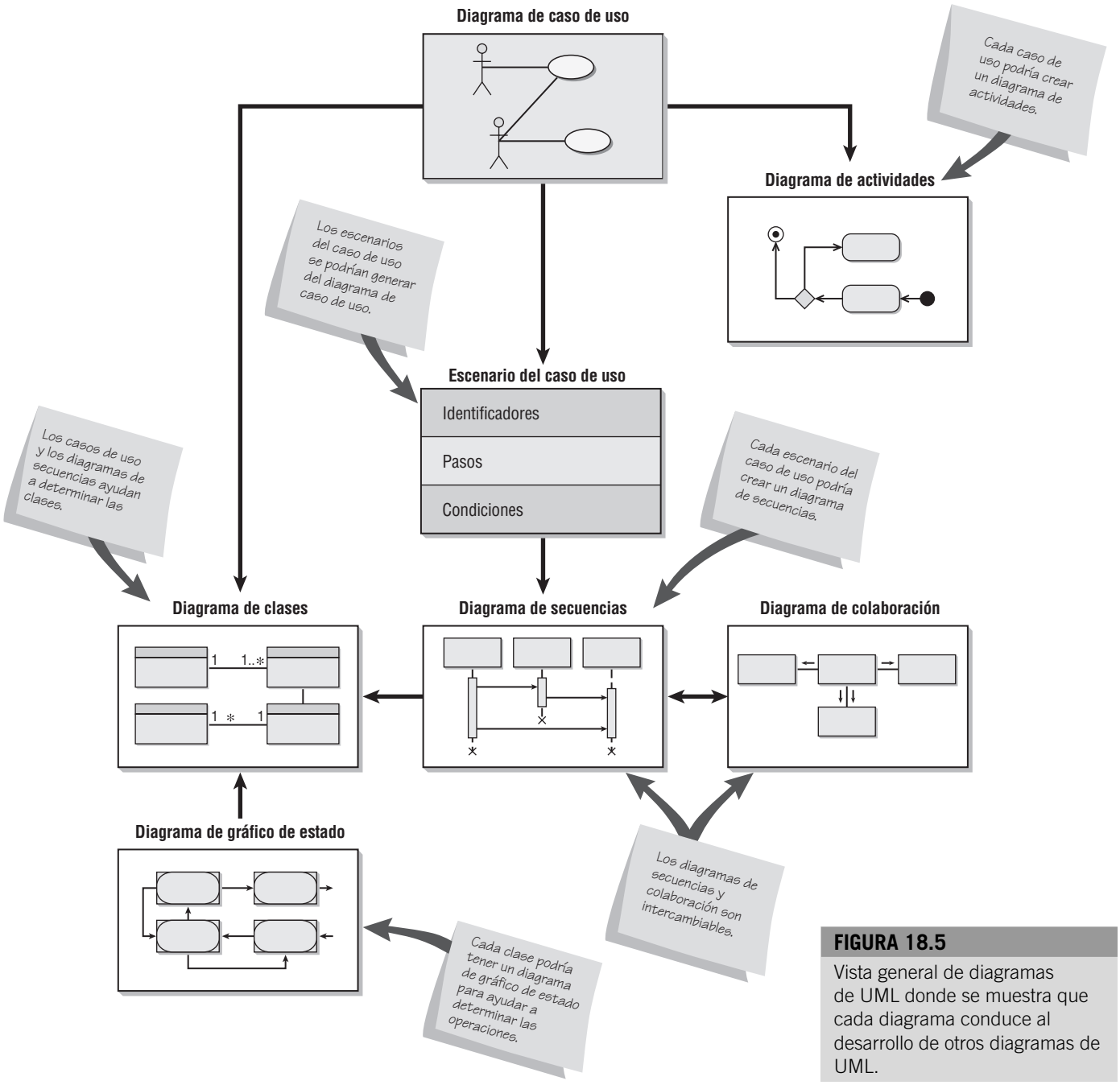


FIGURA 18.5
Vista general de diagramas de UML donde se muestra que cada diagrama conduce al desarrollo de otros diagramas de UML.

MODELADO DE CASOS DE USO

El UML está basado fundamentalmente en una técnica de análisis orientada a objetos conocida como modelado de casos de uso, en la cual la palabra *uso* se pronuncia como sustantivo en lugar de verbo. Un modelo de caso de uso describe lo *que* hace un sistema sin describir *cómo* lo hace; es decir, es un modelo lógico del sistema. (Los modelos lógico o conceptual se introdujeron en el capítulo 7.) El modelo de caso de uso refleja la vista del sistema desde la perspectiva de un usuario fuera del sistema (es decir, los requerimientos del sistema). El UML se puede usar para analizar el modelo de caso de uso y para derivar objetos del sistema y sus interacciones entre sí y con los usuarios del sistema. Usando las técnicas de UML, analiza más a fondo los objetos y sus interacciones para derivar comportamiento del objeto, atributos y relaciones.

Un analista desarrolla casos de uso en colaboración con los expertos del negocio que ayudan a definir los requerimientos del sistema. El modelo de caso de uso proporciona medios eficaces de comunicación entre el equipo del negocio y el equipo de desarrollo. Un modelo de caso de uso divide la funcionalidad del sistema en comportamientos, servicios y respuestas (los casos de uso) que son significativos para los usuarios del sistema.

Desde la perspectiva de un actor (o usuario), un caso de uso debe producir algo que es de valor. Por lo tanto, el analista debe determinar lo que es importante para el usuario y recordar incluirlo en el diagrama de caso de uso. Por ejemplo, ¿una contraseña está introduciendo algo de valor para el usuario? Se podría incluir si el usuario tiene una preocupación sobre la seguridad o si es crítico para el éxito del proyecto.

SÍMBOLOS DEL CASO DE USO

Un diagrama de caso de uso contiene el actor y símbolos de caso de uso, junto con líneas de conexión. Los actores son parecidos a las entidades externas; existen fuera del sistema. El término *actor* se refiere a un papel particular de un usuario del sistema. Por ejemplo, un actor podría ser un empleado, pero también podría ser un cliente en el almacén de la compañía. Aunque quizás es la misma persona en el mundo real, se representa como dos símbolos diferentes en un diagrama de caso de uso, debido a que la persona interactúa con el sistema en diferentes papeles. El actor existe fuera del sistema e interactúa con éste de una forma específica. Un actor puede ser un humano, otro sistema o un dispositivo tal como un teclado, módem o conexión Web. Los actores pueden iniciar una instancia de un caso de uso. Un actor podría interactuar con uno o más casos de uso y viceversa.

Los actores se podrían dividir en dos grupos. Los actores principales proporcionan datos o reciben información del sistema. Los actores secundarios ayudan a mantener el sistema en ejecución o proporcionan ayuda. Éstas son las personas que operan el centro de atención telefónica, los analistas, programadores, etcétera.

Un caso de uso proporciona a los desarrolladores una visión de lo que quieren los usuarios. No contiene detalles técnicos o de implementación. Podemos pensar en un caso de uso como una secuencia de transacciones en un sistema. El modelo de caso de uso se basa en las interacciones y relaciones de casos de uso individuales.

Un caso de uso siempre describe tres cosas: un actor que inicia un evento; el evento que activa un caso de uso, y el caso de uso que desempeña las acciones activadas por el evento. En un caso de uso, un actor que usa el sistema comienza un evento que empieza una serie relacionada de interacciones en el sistema. Los casos de uso se utilizan para documentar una sola transacción o evento. Un evento es una entrada al sistema que pasa en un tiempo y lugar específicos y ocasiona que el sistema haga algo.

Es mejor crear pocos casos de uso en lugar de muchos. Con frecuencia no se incluyen consultas e informes; 20 casos de uso (y no más de 40 o 50) son suficientes para un sistema grande. Los casos de uso también se podrían anidar, si es necesario. Puede incluir un caso de uso en varios diagramas, pero el caso de uso real sólo se define una vez en el depósito o diccionario. Un caso de uso se nombra con un verbo y un sustantivo.

RELACIONES DEL CASO DE USO

Las relaciones activas se denominan como relaciones de comportamiento y se emplean principalmente en los diagramas de caso de uso. Hay cuatro tipos básicos de relaciones de comportamiento: comunica, incluye, extiende y generaliza. Observe que todos estos términos son verbos de acción. La figura 18.6 muestra las flechas y líneas usadas para diagramar cada uno de los cuatro tipos de relaciones de comportamiento. Las cuatro relaciones se describen a continuación.

Comunica La relación de comportamiento comunica se usa para conectar a un actor con un caso de uso. Recuerde que la tarea del caso de uso es dar alguna clase de resultado que es

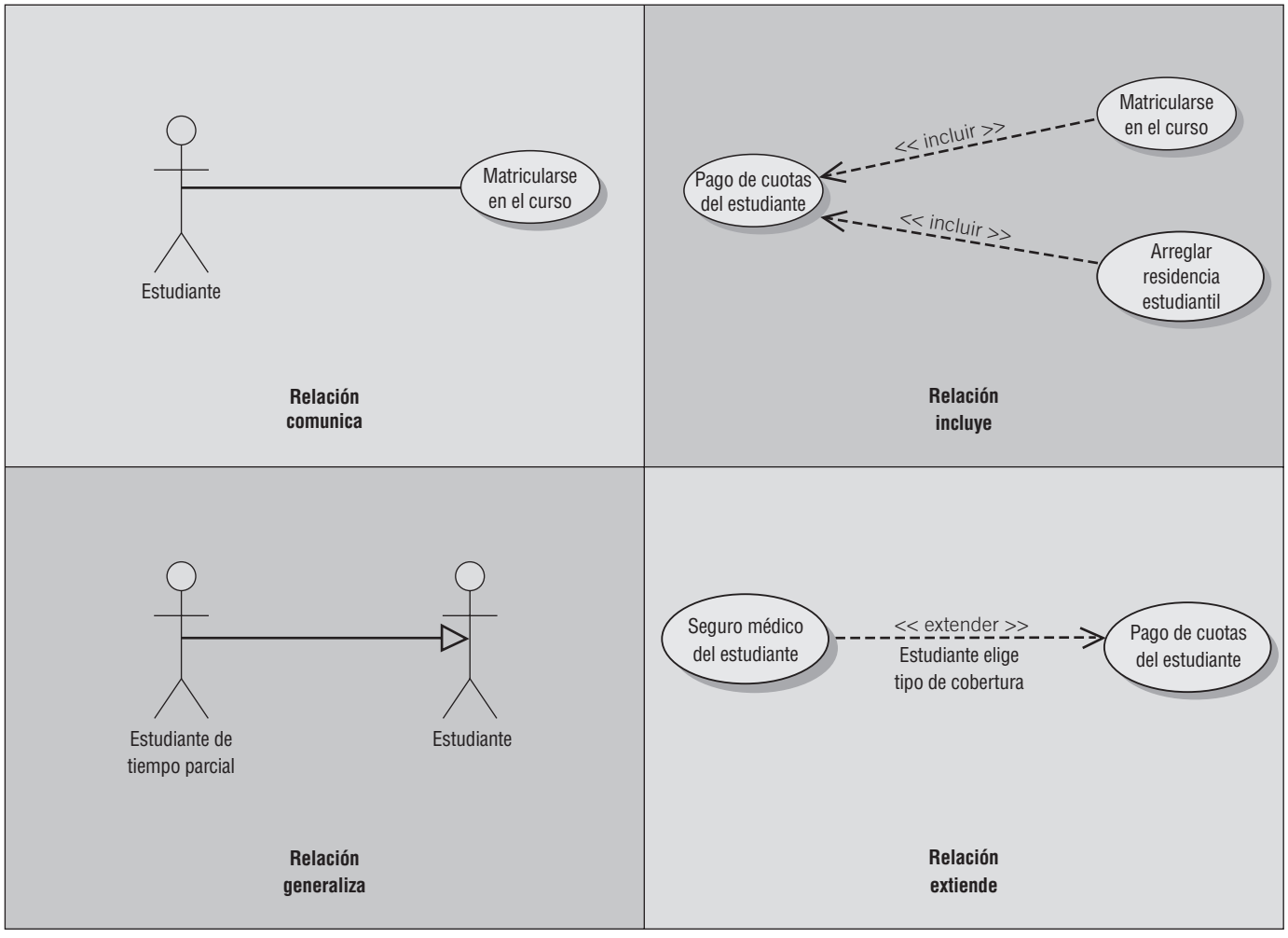
Relación	Símbolo	Significado
Comunica		Un actor se conecta a un caso de uso usando una línea sin puntas de flecha.
Incluye	<<incluir>>	Un caso de uso contiene un comportamiento que es más común que otro caso de uso. La flecha apunta al caso de uso común.
Extiende	<<extender>>	Un caso de uso diferente maneja las excepciones del caso de uso básico. La flecha apunta desde el caso de uso extendido hacia el básico.
Generaliza		Una "cosa" de UML es más general que otra "cosa". La flecha apunta a la "cosa" general.

FIGURA 18.6
Algunos componentes de los diagramas de caso de uso muestran actores, casos de uso y relaciones para un ejemplo de matriculación de un estudiante.

benéfico para el actor en el sistema. Por lo tanto, es importante documentar estas relaciones entre actores y casos de uso. En nuestro ejemplo, un **Estudiante** se comunica con **Matricularse en el curso**. En los diagramas de caso de uso de la figura 18.7 se muestran ejemplos de algunos componentes de un ejemplo de matriculación del estudiante.

Incluye La relación incluye describe la situación en que un caso de uso contiene un comportamiento que es común para más de un caso de uso. Es decir, el caso de uso común se incluye en otros casos de uso. Una flecha punteada que apunta al caso de uso común indica la relación incluye. Un ejemplo sería un caso de uso **Pago de cuotas del estudiante** que se incluye en **Matricularse en el curso** y **Arreglar residencia estudiantil**, debido a que en ambos

FIGURA 18.7
Cuatro tipos de relaciones además de flechas y líneas de comportamiento de UML usadas para representar las relaciones.



casos los estudiantes deben pagar sus cuotas. Esto se podría usar por varios casos de uso. La flecha apunta hacia el caso de uso común.

Extiende La relación extiende describe la situación en la que un caso de uso posee el comportamiento que permite al nuevo caso de uso manejar una variación o excepción del caso de uso básico. Por ejemplo, el caso de uso extendido **Seguro médico del estudiante** extiende el caso de uso básico **Pago de cuotas del estudiante**. La flecha va del caso de uso extendido al básico.

Generaliza La relación generaliza implica que una cosa es más típica que otra. Esta relación podría existir entre dos actores o dos casos de uso. Por ejemplo, **Estudiante de tiempo parcial** generaliza un **Estudiante**. Del mismo modo, algunos empleados universitarios son profesores. La flecha apunta a la cosa general.

DESARROLLO DE DIAGRAMAS DE CASO DE USO

El caso de uso principal (también denominado ruta principal o ruta feliz) consiste de un flujo estándar de eventos en el sistema que describe un comportamiento estándar del sistema. El caso de uso principal representa la realización normal, esperada y exitosa del caso de uso. Las variaciones o excepciones (también denominadas rutas alternativas) también se pueden diagramar y describir.

Al diagramar un caso de uso, empiece pidiendo a los usuarios que mencionen todo lo que el sistema debe hacer para ellos. Esto se puede hacer con entrevistas, en una sesión de diseño conjunto de aplicaciones (JAD) (como se describió en el capítulo 4) o a través de otras sesiones de equipo facilitadas. Escriba quién está involucrado con cada caso de uso y las responsabilidades o servicios que el caso de uso debe proporcionar a los actores u otros sistemas. En las fases iniciales, ésta podría ser una lista parcial que se extiende en las últimas fases del análisis. Use los siguientes lineamientos:

1. Revise las especificaciones del negocio e identifique los actores en el dominio del problema.
2. Identifique los eventos de alto nivel y desarrolle los casos de uso principales que describen dichos eventos y cómo los inician los actores. Examine cuidadosamente los papeles jugados por los actores para identificar todos los posibles casos de uso principales iniciados por cada actor. No se necesita mostrar los casos de uso con poca o ninguna interacción del usuario.
3. Revise cada caso de uso principal para determinar las posibles variaciones de flujo a través del caso de uso. Con este análisis, establezca las rutas alternativas. Debido a que el flujo de eventos normalmente es diferente en cada caso, busque actividades que podrían tener éxito o fallar. También busque cualesquier ramas en la lógica de caso de uso en que son posibles resultados diferentes.

Si se ha creado un diagrama de flujo de datos de nivel contexto, puede ser un punto de partida para crear un caso de uso. Las entidades externas son actores potenciales. Entonces examine el flujo de datos para determinar si iniciaría un caso de uso o sería producido por uno.

La figura 18.8 es un ejemplo de caso de uso de matriculación del estudiante a una universidad. Observe que sólo se representan las funciones más importantes. El caso de uso **Agregar estudiante** no indica cómo agregar estudiantes, que sería el método de implementación. Los estudiantes se podrían agregar personalmente, usando Web, usando un teléfono de tonos o cualquier combinación de 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 **Matricularse en la clase** y podría ser parte de un sistema para matricular a los estudiantes en un curo en línea.

Pareciera como si el caso de uso **Cambiar información del estudiante** fuera una característica menor del sistema y no se debiera incluir en el diagrama de caso de uso, pero debido a que esta información cambia con frecuencia, la administración tiene un interés sutil en

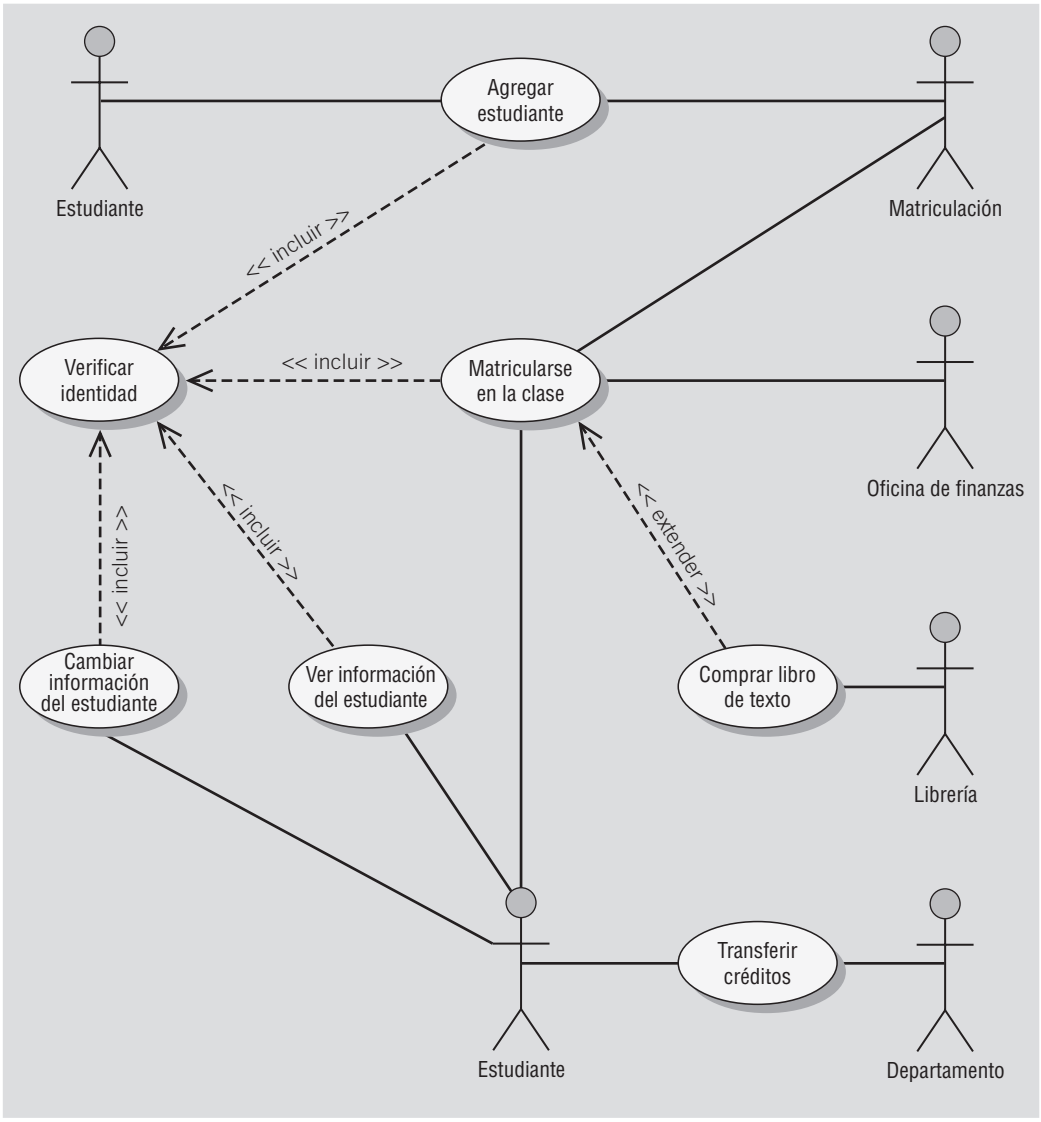


FIGURA 18.8
Ejemplo de caso de uso de matriculación del estudiante.

permitir a los estudiantes cambiar su propia información personal. El hecho de que los administradores juzguen esto como importante no sólo justifica, sino que exige, el caso de uso a ser escrito.

A los estudiantes no se les permitiría cambiar su promedio de calificaciones, cuotas a pagar y otra información. Este caso de uso también incluye el caso de uso **Verificar identidad**, y en esta situación, significa que el estudiante tiene que introducir una clave de usuario y contraseña antes de acceder al sistema. **Ver información del estudiante** permite a los estudiantes ver su información personal, así como también los cursos y calificaciones.

Los diagramas de caso de uso son un buen punto de partida, pero se necesita una descripción más completa de ellos para su documentación. Un caso de uso completo incluirá un diagrama de caso de uso y una serie de descripciones explicadas en la siguiente sección.

DESARROLLO DE ESCENARIOS DE CASO DE USO

Cada caso de uso tiene una descripción. Nos referiremos a la descripción como un escenario de caso de uso. Como se mencionó, el caso de uso principal representa el flujo estándar de eventos en el sistema y las rutas alternativas describen las variaciones para el comportamiento. Los escenarios de caso de uso podrían describir lo que pasa si un artículo comprado está agotado o si una compañía de tarjeta de crédito rechaza la compra solicitada de un cliente.

Nombre del caso de uso:	Cambiar información del estudiante	
Área:	Sistema del estudiante	**ID única:** Estudiante UC 005
Actor(es):	Estudiante	
Descripción:	Permite al estudiante cambiar su propia información, tal como nombre, dirección de la casa, número telefónico, dirección en el campus, teléfono en el campus, teléfono celular y otra información usando un sitio Web seguro.	
Activar evento:	El estudiante usa el sito Web Cambiar información del estudiante, introduce la clave de usuario y contraseña del estudiante y hace clic en el botón **Enviar**.	
Tipo de señal:	☒ Externa ☐ Temporal	
Pasos desempeñados (ruta principal)		**Información para los pasos**
1. El estudiante se conecta a un servidor Web seguro.		Clave de usuario y contraseña del estudiante
2. El registro del estudiante se lee y la contraseña se verifica.		Registro, clave de usuario, contraseña del estudiante
3. Se despliega la información actual personal del estudiante en la página Web Cambiar estudiante.		Registro del estudiante
4. El estudiante introduce los cambios en el formulario Web Cambiar estudiante y hace clic en el botón **Enviar**.		Formulario Web Cambiar estudiante
5. Los cambios se validan en el servidor Web.		Formulario Web Cambiar estudiante
6. Se escribe el registro en el archivo de Registro de cambios del estudiante.		Formulario Web Cambiar estudiante
7. El registro del estudiante se actualiza en el Maestro de estudiantes.		Formulario Web Cambiar estudiante, registro del estudiante
8. La página Web de confirmación se envía al estudiante.		Página de confirmación
Precondiciones:	El estudiante está en la página Web Cambiar información del estudiante.	
Poscondiciones:	El estudiante ha cambiado exitosamente la información personal.	
Suposiciones:	El estudiante tiene un navegador, una clave de usuario y una contraseña válidas.	
Reunir requerimientos:	Permite a los estudiantes cambiar la información personal usando un sitio Web seguro.	
Aspectos sobresalientes:	¿Se debe controlar el número de veces que un estudiante se puede conectar al sistema?	
Prioridad:	Media	
Riesgo:	Media	

FIGURA 18.9
Un escenario de caso de uso se divide en tres secciones: identificación e iniciación, pasos desempeñados y condiciones, suposiciones y preguntas.

No hay ningún formato estándar de escenario de caso de uso, de modo que cada organización se enfrenta con especificar qué estándares se deben incluir. Con frecuencia los casos de uso se documentan con una plantilla de documento de caso de uso predeterminada por la organización, la cual hace los casos de uso fáciles de leer y proporciona información estándar para cada caso de uso en el modelo.

En la figura 18.9 se muestra un ejemplo de escenario de caso de uso. Algunas de las áreas incluidas son opcionales y no se podrían usar en todas las organizaciones. Las tres áreas principales son:

1. Identificadores e iniciadores de caso de uso.
2. Pasos desempeñados.
3. Condiciones, suposiciones y preguntas.

La primera área, identificadores e iniciadores de caso de uso, orientan al lector y contiene el nombre de caso de uso y una ID única; el área de aplicación o sistema que le pertenece a este caso de uso; los actores involucrados en el caso de uso; una breve descripción de lo que logra el caso de uso, y la iniciación (activación) del evento, es decir, lo que ocasionó que empezara el caso de uso, y el tipo de activación, externo o temporal. Los eventos externos son aquellos empezados por un actor. Esto podría ser una persona u otro sistema que pide la información, tal como un sistema de reservación de aerolínea que pide la información del vuelo de un sistema de la aerolínea. Los eventos temporales son aquellos que se activan o se empiezan por tiempo. Los eventos ocurren en un momento específico, tal como enviar un correo electrónico sobre ofertas especiales una vez por semana la tarde del domingo, enviando las facturas en un día específico o generando estadísticas gubernamentales en una fecha específica cada trimestre.

La segunda área del caso de uso incluye los pasos desempeñados y la información requerida para cada uno de los pasos. Estas declaraciones representan el flujo estándar de eventos y los pasos tomados para la realización exitosa del caso de uso. Se desea escribir un caso de uso para la ruta principal y después escribir uno por separado para cada una de las rutas alternativas, en lugar de usar declaraciones IF... THEN...

La tercera área del caso de uso incluye las precondiciones, o la condición del sistema antes de que se pudiera desempeñar el caso de uso; las poscondiciones, o el estado del sistema después de que el caso de uso se ha terminado; cualesquier suposiciones hechas que pudieran afectar el método del caso de uso; cualesquier asuntos excelentes o preguntas que se deben responder antes de la implementación del caso de uso; una declaración opcional de prioridad del caso de uso, y una declaración opcional de riesgo involucrada el crear el caso de uso.

Una vez que desarrolle los escenarios de caso de uso, asegúrese de revisar sus resultados con los expertos de negocios para verificar y refinar los casos de uso si es necesario. Después de finalizar el proceso de verificación y de que todos los expertos de negocios coincidan en que los casos de uso son precisos, puede proceder a utilizar las técnicas de diagramación de UML para completar el análisis y diseño de sistemas.

DIAGRAMAS DE ACTIVIDADES

Los diagramas de actividades muestran las secuencias de actividades de un proceso, incluyendo las actividades secuenciales, las actividades paralelas y las decisiones que se toman. Por lo general, un diagrama de actividades se elabora para un caso de uso y podría reflejar los diferentes escenarios posibles.

En la figura 18.10 se ilustran los símbolos de un diagrama de actividades. Un rectángulo con esquinas redondeadas representa una actividad, ya sea manual, como firmar un documento legal; o automatizada, como un método o un programa.

Una flecha representa un evento. Los eventos representan cosas que ocurren en un tiempo y lugar determinados.

Un diamante representa una decisión (también conocida como rama) o una fusión. Las decisiones tienen una flecha que entra en el diamante y varias que salen de él. Se podría incluir una condición que muestre los valores que puede tomar dicha condición. Las fusiones muestran varios eventos que se combinan para formar otro evento.

Un rectángulo largo y plano representa una barra de sincronización. Esta barra se utiliza para representar actividades paralelas, y podría representar un evento entrando a ella y varios eventos saliendo de la misma, lo que se conoce como bifurcación. Una sincronización en la cual varios eventos se fusionan en uno solo se conoce como unión.

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

Los rectángulos que rodean otros símbolos llamados carriles (*swimlanes*) indican un particionamiento y se utilizan para mostrar cuáles actividades se realizan en qué plataforma, como un navegador, un servidor o un *mainframe*; o para mostrar actividades realizadas por diferentes grupos de usuarios. Los carriles son zonas que pueden describir la lógica y la responsabilidad de una clase.

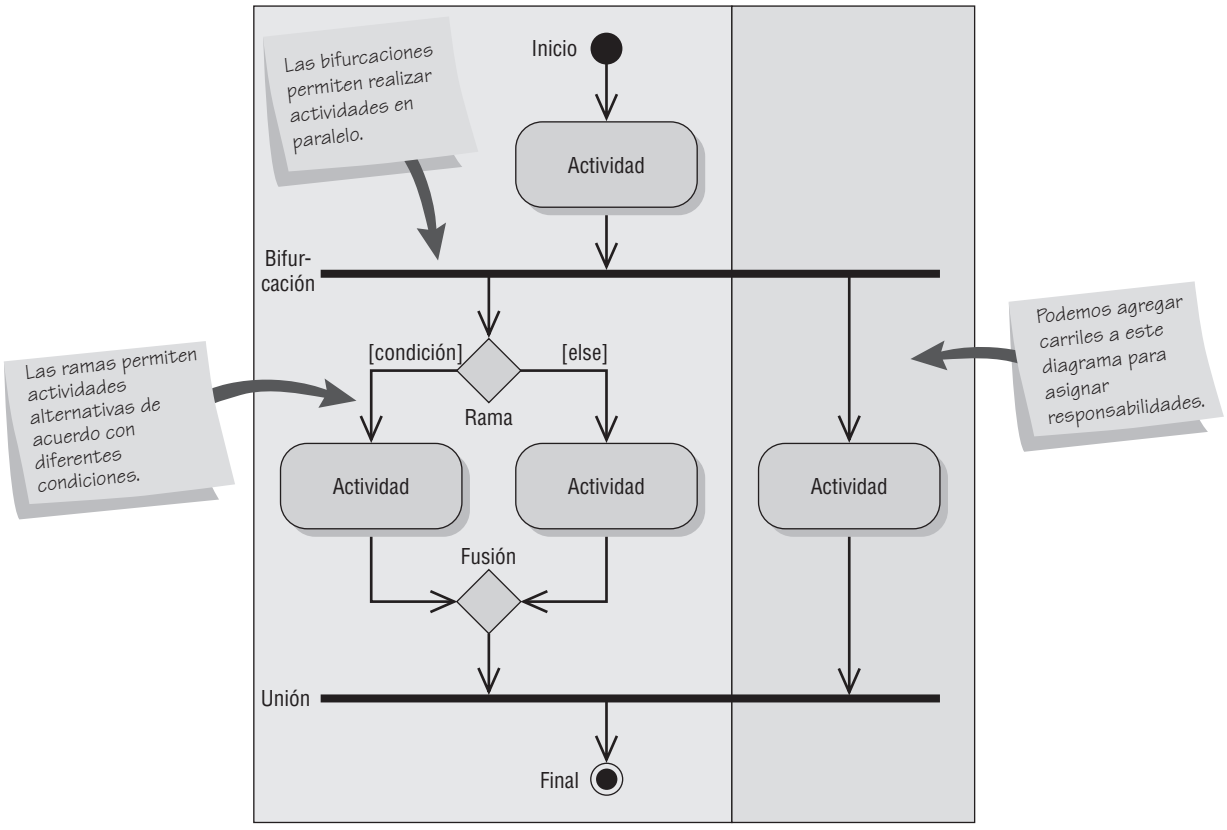


FIGURA 18.10
Para dibujar un diagrama de actividades se usan símbolos especializados.

Usted puede ver un ejemplo de carriles en la figura 18.11, la cual ilustra un diagrama de actividades para el caso de uso **Cambiar Información del Estudiante**. Empieza con el estudiante que inicia sesión en el sistema completando un formulario Web y haciendo clic en el botón **Enviar**. El formulario se transmite al servidor Web, que a su vez pasa los datos al *mainframe*. Éste accede a la base de datos ESTUDIANTES y manda al servidor Web un mensaje "No se encontró" o los datos seleccionados sobre el estudiante.

El diamante debajo del estado **Obtener Registro del Estudiante** indica esta decisión. Si no se localiza el registro del estudiante, el servidor Web despliega un mensaje de error en la página Web. Si se localiza el registro, el servidor Web genera una nueva página Web con los datos actuales del estudiante en un formulario Web. El estudiante podría cancelar el cambio desde los estados **Sistema de Inicio de Sesión** o **Introducir Cambios**, y la actividad se detiene.

Si el estudiante realiza cambios en el formulario Web y hace clic en el botón **Enviar**, los datos modificados se transmiten al servidor y comienza a ejecutarse un programa que valida los datos. Si hay errores, se envía un mensaje de error a la página Web. Si los datos son válidos, el registro del estudiante se actualiza y se escribe un Registro Periódico de Cambios del Estudiante. Después de una actualización válida, se envía una página Web de confirmación al navegador y finaliza la actividad.

CREACIÓN DE DIAGRAMAS DE ACTIVIDADES

Los diagramas de actividades se crean preguntando qué pasa en primer lugar, qué pasa en segundo lugar, y así sucesivamente. Usted debe determinar si las actividades se realizan en secuencia o en paralelo. Si se han creado diagramas de flujo de datos físicos (como se describió en el capítulo 7), se podrían examinar para determinar la secuencia de actividades. Busque lugares donde se tomen decisiones, y pregunte qué ocurre con los resultados de cada una de las decisiones. Los diagramas de actividades se podrían crear examinando todos los escenarios para un caso de uso.

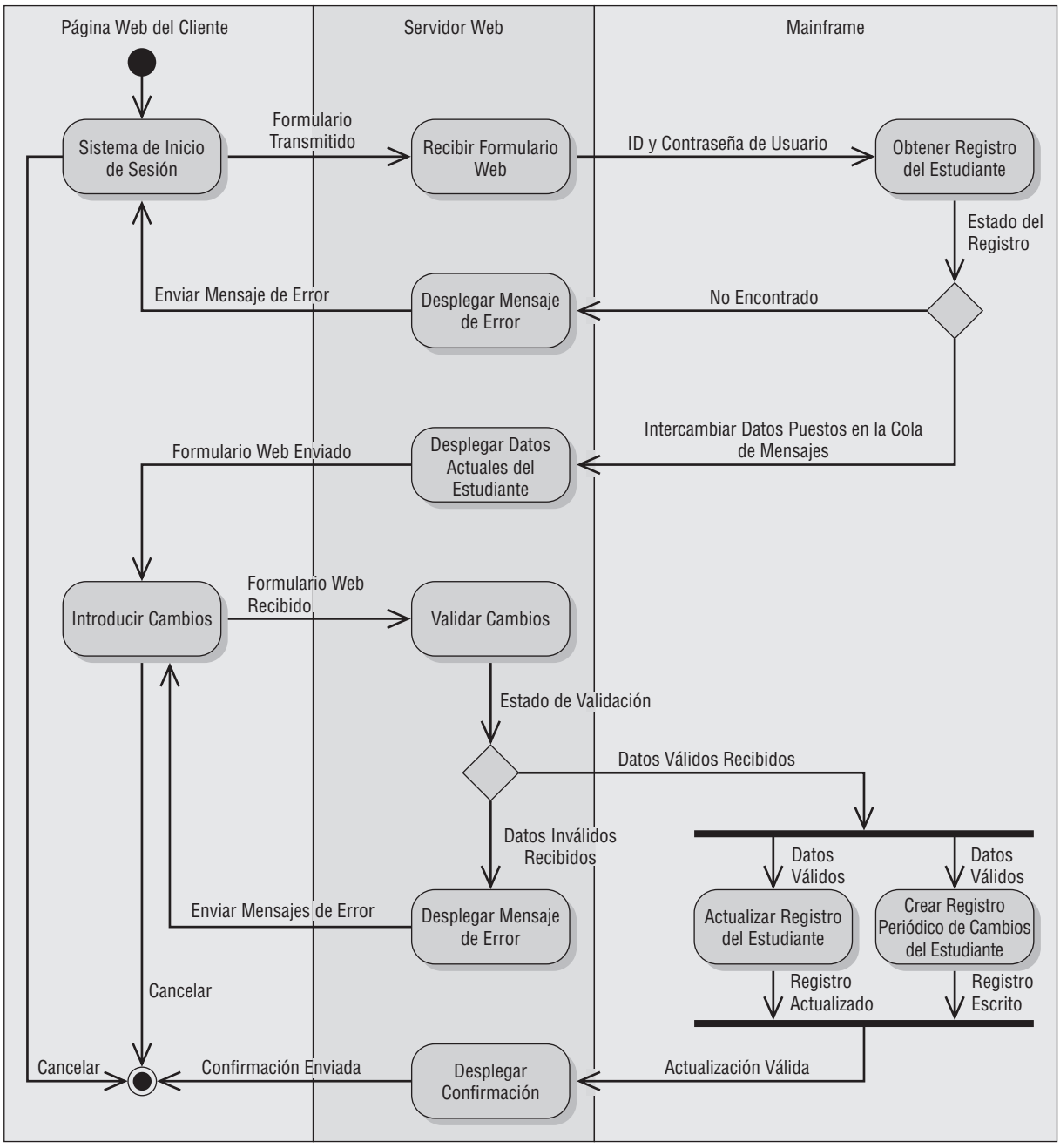


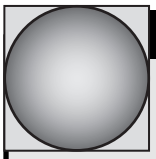
FIGURA 18.11

Este diagrama de actividades muestra tres carriles: Página Web del Cliente, Servidor Web y Mainframe.

Cada ruta a través de las diversas decisiones incluidas en el caso de uso es un escenario diferente. En la ruta principal estaría el **Sistema de Inicio de Sesión**, **Recibir Formulario Web**, **Obtener Registro del Estudiante**, **Desplegar Datos Actuales del Estudiante**, **Introducir Cambios**, **Validar Cambios**, **Actualizar Registro del Estudiante**, **Crear Registro Periódico de Cambios del Estudiante** y **Desplegar Confirmación**.

Éste no es el único escenario posible de este caso de uso. Podría ocurrir otros. Una posibilidad podría ser el **Sistema de Inicio de Sesión**, **Recibir Formulario Web**, **Obtener Registro del Estudiante** y **Desplegar Mensaje de Error**. Otro escenario podría ser el Sistema de **Inicio de Sesión**, **Recibir Formulario Web**, **Obtener Registro del Estudiante**, **Desplegar Datos Actuales del Estudiante**, **Introducir Cambios**, **Validar Cambios** y **Desplegar Mensaje de Error**.

Los carriles son útiles para mostrar cómo deben transmitirse o convertirse los datos, como en el caso de la Web al servidor o del servidor al *mainframe*. Por ejemplo, el diagrama de actividades **Cambiar Registro del Estudiante** tiene tres carriles.



OPORTUNIDAD DE CONSULTORÍA 18.2

RECICLAJE DEL ENTORNO DE PROGRAMACIÓN

"Siento como si estuviera escribiendo el mismo código una y otra vez", dice Benito Pérez, un programador que trabaja en un nuevo diseño automatizado del almacén. "Últimamente he escrito muchos programas relacionados con la robótica, que se controlan por sí mismos: carritos automatizados para el correo, robots de vigilancia en edificios, limpiadores de piscinas automáticos, cortadoras de césped automáticas, trenes monorriel y ahora carritos para almacén. Todas son variaciones de un tema."

Lisa Bernoulli, la gerente de proyecto, ha escuchado quejas similares a ésta durante años. Ella contesta: "Vamos, Ben. Estas cosas no tienen mucha relación. ¿Cómo puedes comparar un robot de correo, un almacén automatizado y un tren monorriel? Apuesto que menos de 10 por ciento del código es igual".

"Mira", dice Benito. "Los tres involucran máquinas que tienen que encontrar un punto de partida, siguen una ruta indirecta, se detienen para cargar y descargar, y al final llegan a un punto de detención. Todos tienen que tomar decisiones en las ramas de sus rutas. Los tres tienen que evitar choques con las cosas. Estoy cansado de rediseñar código que ya conozco bastante".

"Hmmm", murmura Lisa mientras examina los requerimientos básicos para el sistema del almacén y recuerda el sistema monorriel en el que ella y Benito trabajaron el año pasado. Los requerimientos tenían que ver con una empresa fabricante de partes electrónicas que deseaba automatizar su almacén y su sistema de desplazamiento de productos. El almacén contiene partes entrantes, trabajos en marcha y productos terminados. El almacén automatizado usa un carrito remolcador robotizado. Este robot es un carrito eléctrico de cuatro ruedas, similar a un carrito de golf sólo que no tiene asientos. Los carritos remolcadores robotizados tienen una superficie de carga de 1.80 $\times$ 1.20 cm, y 90 cm de altura. Estos carritos tienen un dispositivo de radiocomunicaciones que ofrece un enlace de datos en tiempo real a una computadora central ubicada en el almacén. También cuentan con dos sensores: un sensor de trayectoria que detecta un tipo especial de pintura y un sensor de movimiento. Estos carritos se desplazan por rutas pintadas en el piso de la fábrica. Códigos de pintura especiales marcan las bifurcaciones y ramas de la trayectoria, puntos de partida y llegada de los carritos, así como puntos de localización general.

Las instalaciones cuentan con tres plataformas de carga y 10 estaciones de trabajo. Cada estación tiene una terminal de vídeo o una computadora conectada a la computadora central. Cuando se necesitan los productos o están listos para recopilarlos de una estación de trabajo, el operador de la estación informa a la computadora central. A continuación, la computadora central envía los carritos necesarios. Cada estación cuenta con un punto de descarga y uno de carga. Los carritos remolcadores se desplazan a través de la fábrica recogiendo el trabajo en los puntos de carga y dejándolo en los puntos de descarga. El programa que controlará los carritos debe tener una constante interacción con el programa de calendarización de trabajos que contribuye a planificar las tareas de las estaciones de trabajo.

¿Cómo debe Lisa reutilizar el trabajo que Benito Pérez realizó con el monorriel en su tarea actual de crear un objeto para el carrito? Explique su respuesta en dos párrafos.

El carril del lado izquierdo muestra actividades que ocurren en el navegador del cliente. Deben crearse páginas Web para estas actividades. El carril central muestra actividades que pasan en el servidor. Los eventos, como **Formulario Transmitido**, representan datos transmitidos del navegador al servidor, y debe haber programas en el servidor que reciban y procesen los datos del cliente.

El carril del lado derecho representa el *mainframe*. En las organizaciones grandes es común que muchas aplicaciones Web trabajen con un *mainframe*. Gran parte de los datos en las organizaciones grandes se encuentran en las bases de datos del *mainframe* y existe un número considerable de programas para *mainframe*.

Cuando un evento cruza el carril del servidor al *mainframe*, debe haber un mecanismo para transmitir los datos del evento entre las dos plataformas. Los servidores usan un formato diferente al de los *mainframes* para representar los datos (los primeros usan ASCII y los últimos emplean formato EBCDIC). Debe haber middleware que se haga cargo de la conversión entre estos formatos. Las computadoras IBM usan un mqueue (cola de mensajes). La cola de mensajes recibe datos de los programas del servidor, los coloca en un área de espera y llama a un programa del *mainframe*, escrito por lo general en un lenguaje conocido como CICS. Este programa recupera o actualiza los datos y envía los resultados de regreso a la cola de mensajes.

En el diagrama de actividades del ejemplo que se muestra, la decisión debajo del estado **Obtener Registro del Estudiante** se realiza en el *mainframe*. Esto significa que la cola de mensajes recibe un mensaje "No Encontrado" o el registro del estudiante que se localiza en la base de datos. Si el *mainframe* tan sólo colocara el **Estado del Registro Recibido** en la co-

la de mensajes y la decisión se evaluara en el servidor, éste tendría que llamar al *mainframe* otra vez para obtener los datos válidos. Esto retrasaría la respuesta a la persona que espera en el navegador.

Los carriles también ayudan a dividir las tareas en un equipo. Se necesitarían diseñadores Web para las páginas Web desplegadas en el navegador del cliente. Otros miembros trabajarían con lenguajes de programación, como Java, PERL o .NET, en el servidor. Los programadores de CICS escribirían programas para *mainframe* que trabajarían con la cola de mensajes. El analista debe garantizar que los datos requeridos por los diversos miembros del equipo estén disponibles y correctamente definidos. En ocasiones los datos en la cola de mensajes son un documento de XML. Si se trabaja con una organización externa, los datos también podrían ser un documento de XML.

El diagrama de actividades proporciona un mapa de un caso de uso, y permite al analista experimentar con la transferencia de partes del diseño a plataformas diferentes y plantearse la pregunta "¿qué pasaría si?" para una variedad de decisiones. El uso de símbolos únicos y carriles favorece que las personas prefieran este diagrama para comunicarse con otros.

DIAGRAMAS DE SECUENCIAS Y DE COLABORACIÓN

Un diagrama de interacción puede ser un diagrama de secuencias o uno de colaboración, que muestran esencialmente la misma información. Estos diagramas, junto con los diagramas de clases, se utilizan en la realización de un caso de uso.

DIAGRAMAS DE SECUENCIAS

Los diagramas de secuencias pueden ilustrar una sucesión de interacciones entre clases o instancias de objetos en un periodo determinado. Los diagramas de secuencias se utilizan con frecuencia para representar el proceso descrito en los escenarios de caso de uso. En la práctica, los diagramas de secuencias se derivan del análisis de casos de uso y se emplean en el diseño de sistemas para generar las interacciones, relaciones y métodos de los objetos del sistema. Los diagramas de secuencias se utilizan para mostrar el patrón general de las actividades o interacciones en un caso de uso. Cada escenario de caso de uso podría crear un diagrama de secuencias, aunque no siempre se crean diagramas de este tipo para los escenarios menores.

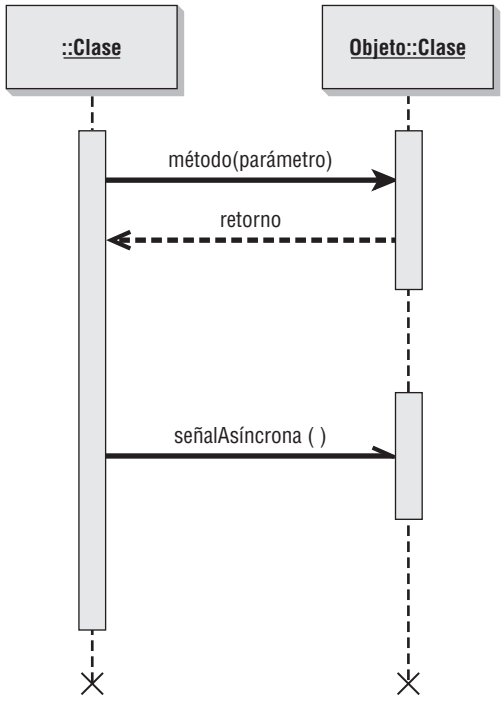
En la figura 18.12 se muestran los símbolos que se utilizan en diagramas de secuencias. Los actores y las clases o instancias de los objetos se muestran en recuadros en la parte superior del diagrama. El objeto del extremo izquierdo es el objeto inicial y podría ser una persona (para la cual se emplea símbolo de actor de caso de uso), una ventana, un cuadro de diálogo u otra interfaz de usuario. Algunas de las interacciones sólo son físicas, como firmar un contrato. Los rectángulos de la parte superior usan indicadores en el nombre para denotar si el rectángulo representa un objeto, una clase, o una clase y un objeto.

nombreDelObjeto:	Un nombre seguido de dos puntos representa un objeto.
:clase	Dos puntos seguidos de un nombre representan una clase.
nombreDelObjeto:clase	Un nombre, seguido de dos puntos y otro nombre, representa un objeto de una clase.

Una línea vertical representa la trayectoria de la vida de la clase o del objeto, que comienza cuando se crea y finaliza cuando se destruye. Una X en el fondo de la trayectoria de la vida indica cuándo se destruye el objeto. Una barra lateral o rectángulo vertical en la trayectoria de la vida muestran el enfoque de control cuando el objeto se encuentra realizando algo.

Las flechas horizontales muestran mensajes o signos que se envían entre las clases. Los mensajes pertenecen a la clase receptora. Hay algunas variaciones en las flechas de mensaje. Las puntas de flecha sólidas representan llamadas síncronas, que son las más comunes. Éstas se usan cuando la clase emisora espera una respuesta de la clase receptora, y el control se devuelve a la clase emisora cuando la clase que recibe el mensaje termina su ejecución.

FIGURA 18.12
Símbolos especializados usados para dibujar un diagrama de secuencias.



Las flechas con media punta (o abiertas) representan llamadas asíncronas, es decir, llamadas que se envían sin esperar a que sean devueltas a la clase que las emite. Un ejemplo podría ser el de usar un menú para ejecutar un programa. Un retorno se muestra como una flecha, a veces con una línea punteada. Los mensajes se etiquetan mediante alguno de los formatos siguientes:

- El nombre del mensaje seguido por paréntesis vacíos: **nombreDelMensaje( )**.
- El nombre del mensaje seguido por parámetros entre paréntesis: **nombreDelMensaje(parámetro1, parámetro2...)**.
- El nombre del mensaje seguido por el tipo del parámetro, nombre del parámetro y cualquier valor predeterminado para el parámetro entre paréntesis: **nombreDelMensaje(tipoDelParámetro:nombreDelParámetro(valorPredeterminado).** Los tipos de parámetro indican el tipo de los datos, como numérico, alfanumérico o de tipo de fecha.
- El mensaje podría ser un estereotipo, como **≪Create≫**, lo cual indica que se crea un nuevo objeto como resultado del mensaje.

En el diagrama de secuencias el tiempo se despliega de arriba abajo; la primera interacción se representa en la parte superior del diagrama, y la última, en la parte inferior. Las flechas de interacción comienzan en la barra del actor o del objeto que inicia la interacción, y terminan apuntando hacia la barra del actor o el objeto que recibe la solicitud de interacción. El actor, la clase o el objeto iniciales se muestran a la izquierda. Éste podría ser el actor que inicia la actividad o podría ser una clase que represente la interfaz de usuario.

La figura 18.13 es un ejemplo simplificado de un diagrama de secuencias para un caso de uso que admite a un estudiante a una universidad. En la parte izquierda se encuentra la clase **interfazDeUsuarioDeNuevoEstudiante** que se utiliza para obtener información del estudiante. El mensaje **inicializar( )** se envía a la clase Estudiante, que crea un nuevo registro del estudiante y devuelve el número de este último. Para simplificar el diagrama, se han omitido los parámetros que se envían a la clase **Estudiante**, pero entre éstos estarían el nombre del estudiante, la dirección, etc. La siguiente actividad es enviar un mensaje **seleccionarDormitorio** a la clase **Dormitorio**. Este mensaje incluiría información relativa a la selección del dormitorio, como un dormitorio del área de salud u otros requerimientos del

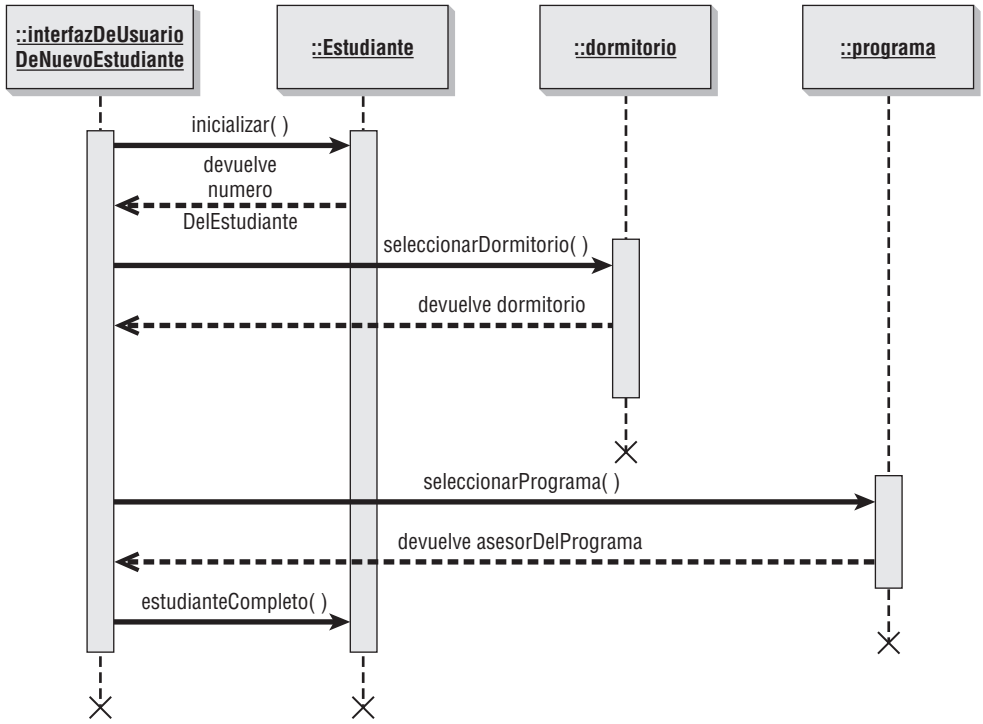


FIGURA 18.13
Diagrama de secuencias para la admisión del estudiante. Los diagramas de secuencias ponen énfasis en la clasificación de los mensajes según el tiempo.

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 otra información del curso de estudio. El nombre del asesor del programa se devuelve a la clase **interfazDeUsuarioDeNuevoEstudiante**. Un mensaje **estudianteCompleto** se envía a la clase **Estudiante** con el dormitorio, el nombre del asesor e información adicional.

Los diagramas de secuencias pueden usarse para traducir el escenario de caso de uso a una herramienta visual para el análisis de sistemas. El diagrama de secuencias inicial utilizado en el análisis de sistemas muestra los actores y clases del sistema y las interacciones que ocurren entre ellos para un proceso específico. Usted puede usar esta versión del diagrama de secuencias para verificar procesos con los expertos del área de negocios que le han ayudado a desarrollar los requerimientos del sistema. Un diagrama de secuencias pone énfasis en la clasificación de los mensajes según el tiempo (secuencia).

Los diagramas de secuencias se refinan durante la fase de diseño del sistema para derivar los métodos e interacciones entre las clases. Los mensajes de una clase se utilizan para identificar las relaciones de la clase. Los actores de los primeros diagramas de secuencias se traducen en interfaces y las interacciones se traducen en métodos de clase. Los métodos de clase que se utilizan para crear instancias de otras clases y para realizar otras funciones internas del sistema surgen en el diseño del sistema al utilizar diagramas de secuencias.

DIAGRAMAS DE COLABORACIÓN

Las colaboraciones describen las interacciones de dos o más cosas en el sistema, las cuales desempeñan en conjunto un comportamiento superior al que puede realizar cualquiera de las cosas por sí sola. Por ejemplo, un automóvil puede dividirse en miles de partes individuales. Las partes se conjuntan para formar los principales subsistemas del vehículo: el motor, la transmisión, el sistema de frenos, etc. Las partes individuales del automóvil se pueden considerar como clases, porque tienen distintos atributos y funciones. Las partes individuales del motor forman una colaboración, porque "colaboran" entre sí para hacer funcionar el motor cuando el conductor pisa el acelerador.

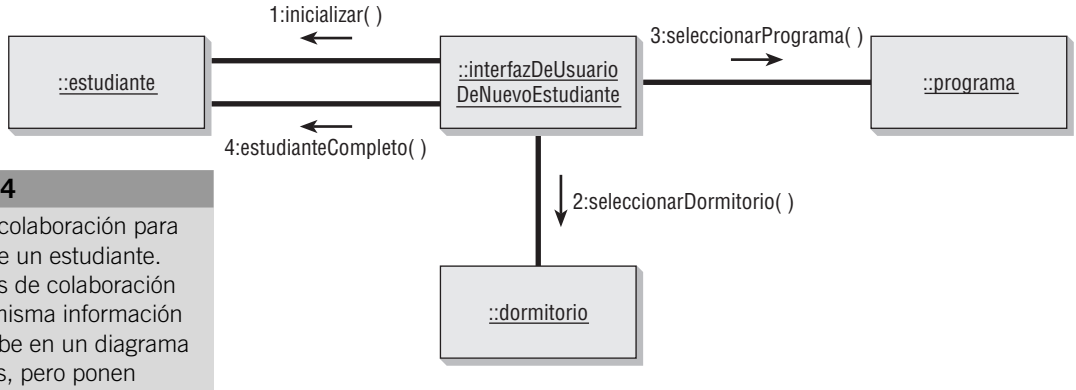


FIGURA 18.14

Diagrama de colaboración para la admisión de un estudiante. Los diagramas de colaboración muestran la misma información que se describe en un diagrama de secuencias, pero ponen énfasis en la organización de los objetos en lugar de en la clasificación según el tiempo.

Los diagramas de colaboración muestran la misma información que un diagrama de secuencias, pero su lectura podría ser más difícil. Para denotar la clasificación en el tiempo, usted debe indicar un número de secuencia y describir el mensaje.

Un diagrama de colaboración pone énfasis en la organización de los objetos, en tanto que un diagrama de secuencias lo pone en la clasificación de los mensajes según el tiempo. Un diagrama de colaboración mostrará una ruta para indicar cómo se enlaza un objeto con otro.

Algunas herramientas CASE, como Rose de IBM, convierten automáticamente un diagrama de secuencias en un diagrama de colaboración o viceversa, con sólo un clic en un botón. En la figura 18.14 se ilustra un diagrama de colaboración para el ejemplo de admisión del estudiante. Cada rectángulo representa 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 ilustran junto a las líneas conectoras. Los mensajes se numeran para mostrar la secuencia de tiempo. Los valores devueltos también se pueden incluir y numerar para indicar en qué momento de la secuencia de tiempo se devuelven.

DIAGRAMAS DE CLASE

Las metodologías orientadas a objetos se enfocan en descubrir clases, atributos, métodos y relaciones entre las clases. Puesto que la programación se realiza al nivel de la clase, la definición de 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 particular. Un diagrama de clases también muestra la naturaleza de las relaciones entre las clases.

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

La figura 18.15 ilustra un diagrama de clases para ofrecimientos de cursos. Observe que el nombre se centra en la parte superior de la clase, por lo general en negritas. El área directamente debajo del nombre muestra los atributos, y los métodos se encuentran en la parte inferior. El diagrama de clases denota los requerimientos de almacenamiento de datos así como los de procesamiento. Más adelante explicaremos el significado de los símbolos de diamante que se aprecian en esta figura.

Por lo general, los atributos (o propiedades) se designan como privados, o disponibles sólo para el objeto. Esto se representa en un diagrama de clases mediante un signo de resta antes del nombre del atributo. Los atributos también pueden designarse como protegidos, lo cual se indica con el símbolo de número (#). Estos atributos están ocultos para todas las

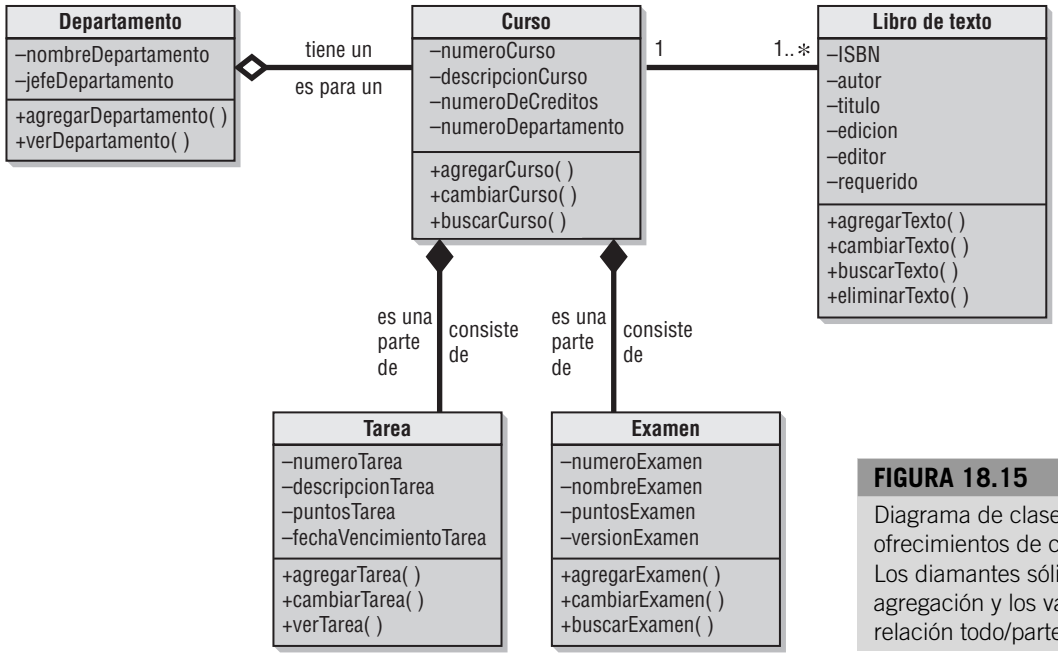


FIGURA 18.15
Diagrama de clases para los ofrecimientos de cursos. Los diamantes sólidos muestran agregación y los vacíos, una relación todo/parte.

clases, excepto para las subclases inmediatas. En circunstancias poco comunes, un atributo es público, lo cual significa que es visible para otros objetos fuera de su clase. Al hacer privados a los atributos sólo están disponibles para los objetos externos a través de los métodos de la clase, una técnica llamada encapsulamiento, u ocultamiento de información.

Un diagrama de clases podría mostrar simplemente el nombre de la clase; o el nombre de la clase y los atributos; o el nombre de la clase, los atributos y los métodos. Mostrar sólo el nombre de la clase es útil cuando el diagrama es muy complejo e incluye muchas clases. Si el diagrama es más sencillo, se podrían incluir atributos y métodos. Cuando se incluyen atributos, hay tres maneras de mostrar su información correspondiente. La más simple es incluir sólo el nombre del atributo, que toma la menor cantidad de espacio.

El tipo de datos (por ejemplo, numérico, alfanumérico, entero o fecha) podría incluirse en el diagrama de clases. Las descripciones más completas podrían incluir un signo de igual (=) después del tipo de datos, seguido por el valor inicial del atributo. La figura 18.16 ilustra los atributos de la clase.

Si el atributo debe adoptar algún valor de entre un número finito, como un tipo de estudiante con valores de C para tiempo completo, M para medio tiempo y N para no inscrito, éstos pueden incluirse entre llaves, separados por comas: **tipoDeEstudiante:char{C,M,N}**.

El ocultamiento de información significa que los métodos de los objetos deben estar disponibles para otras clases, así que con frecuencia los métodos son públicos, lo cual quiere decir que podrían ser invocados desde otras clases. En un diagrama de clases, los mensajes públicos (y cualquier atributo público) se muestran con un signo de suma (+) antes del

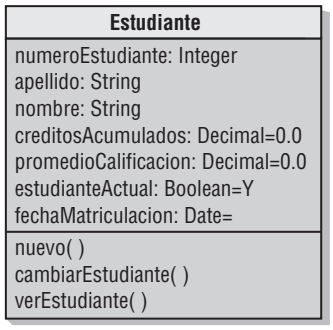


FIGURA 18.16
Una clase **Estudiante** extendida que muestra el tipo de datos y, en algunos casos, su valor inicial o predeterminado.

nombre. Los métodos también tienen paréntesis a continuación del nombre, lo cual indica que se podrían pasar datos como parámetros junto con el mensaje. Los parámetros del mensaje, así como el tipo de datos, se podrían incluir en el diagrama de clases.

Hay dos tipos de métodos: estándar y personalizado. Los métodos estándar son cosas básicas que todas las clases de objetos saben hacer, como crear una nueva instancia del objeto. Los métodos personalizados se diseñan 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 abarca el nombre del método y los parámetros que contiene. El mismo método podría definirse más de una vez en una clase determinada, con la condición de que los parámetros enviados como parte del mensaje sean diferentes; es decir, el mensaje debe tener una firma diferente. Podría tener un número diferente de parámetros, o éstos podrían ser de un tipo diferente, como number (numérico) en un método y string (alfanumérico) en otro método. Un ejemplo de sobrecarga de métodos podría encontrarse en el uso de un signo de suma en muchos lenguajes de programación. Si los atributos a ambos lados del signo de suma son números, los dos números se suman. Si los atributos son cadenas de caracteres, las cadenas se concatenan para formar una cadena larga.

En un ejemplo de depósito bancario, una ficha de depósito podría contener simplemente la cantidad del depósito, en cuyo caso el banco depositaría la cantidad completa, o podría contener la cantidad del depósito y la cantidad de dinero en efectivo que se tendría que devolver. Ambas situaciones usarían un método de cheque de depósito, pero los parámetros (una situación también solicitaría la cantidad de dinero en efectivo que se tendía que devolver) serían diferentes.

TIPOS DE CLASES

Las clases entran en cuatro categorías: de entidad, de interfaz, abstractas y de control. Estas categorías se explican a continuación.

Clases de entidad Las clases de entidad representan elementos de la vida real, como gente, cosas, etc. Las clases de entidad son las entidades representadas en un diagrama entidad-relación. Las herramientas CASE como Visible Analyst le permitirán crear una clase de entidad UML a partir de una entidad de un diagrama de E-R.

El analista necesita determinar qué atributos incluir en las clases. Cada objeto tiene muchos atributos, pero la clase debe incluir sólo aquellos que utiliza la organización. Por ejemplo, al crear una clase de entidad para un estudiante de una universidad, usted necesitaría conocer qué atributos identifican al estudiante, como la dirección de la casa y del campus, así como el promedio de calificaciones, créditos totales, etc. Si usted estuviera dando seguimiento al mismo estudiante para una tienda de ropa en línea, usted tendría que conocer información básica que lo identifique, así como otros atributos descriptivos como medidas o preferencias de color.

Clases de límite, o de interfaz Las clases de límite, o interfaz, ofrecen a los usuarios un medio para trabajar con el sistema. Existen 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 despliegue. También puede ser un teléfono de tonos, un código de barras o algún otro medio que permita a los usuarios interactuar con el sistema. Deben crearse prototipos de las interfaces humanas (como se describió en el capítulo 6), y a menudo se usa un storyboard (una ilustración del argumento o secuencia escena por escena) para modelar la secuencia de interacciones.

Las interfaces del sistema implican el envío o recepción de datos de otros sistemas. Esto podría incluir a las bases de datos de la organización. Si los datos se envían a una organi-

zación externa, a menudo se encuentran en forma de archivos de XML u otras interfaces bien publicadas con mensajes y protocolos definidos de manera clara. Las interfaces externas son las menos estables, porque con frecuencia hay poco o ningún control sobre un socio externo que podría alterar el formato del mensaje o los datos.

XML ayuda a proporcionar estandarización, porque un socio externo podría agregar nuevos elementos al documento XML, pero una corporación que transforme los datos a un formato que pudiera utilizarse para incorporarlos a una base de datos interna podría elegir simplemente ignorar los elementos adicionales sin ningún problema.

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

Clases abstractas Son las clases que no es posible instanciar directamente. Las clases abstractas están vinculadas a clases concretas en una relación generalización/especialización (gen/spec). Por lo general, el nombre de una clase abstracta se denota en letras cursivas.

Clases de control Las clases de control, o activas, se utilizan para controlar el flujo de actividades, y funcionan como coordinadoras al implementar clases. Para lograr clases reutilizables, un diagrama de clases podría incluir muchas clases pequeñas de control. Con frecuencia, las clases de control se derivan durante el diseño del sistema.

A menudo una nueva clase de control se creará sólo con el propósito de hacer reutilizable otra clase. Un ejemplo podría ser el proceso de inicio de sesión. Podría existir una clase de control para la interfaz de usuario de inicio de sesión, que contenga la lógica para verificar la contraseña y la ID de usuario. El problema que surge es que la clase de control de inicio de sesión se diseña para una pantalla de inicio de sesión específica. Al crear una clase de control que sólo maneje esta pantalla de inicio de sesión, los datos se pueden pasar a una clase de control de validación más general que compruebe las contraseñas e IDs de usuario provenientes de muchas otras clases de control que reciban mensajes de interfaces de usuario específicas. Esto incrementa la reusabilidad y aísla los métodos de verificación de inicio de sesión de los métodos que manejan la interfaz de usuario.

UN EJEMPLO DE CLASE PARA LA WEB

También pueden utilizarse símbolos especiales para representar las clases de entidad, límite (o interfaz) y de control. Éstos se denominan estereotipos, una extensión de UML, que son símbolos especiales que podrían utilizarse durante el análisis pero que se emplean a menudo al realizar el diseño orientado a objetos. Estos símbolos dan libertad al analista para experimentar con el diseño y optimizar la reusabilidad.

Los diferentes tipos de clases a menudo se utilizan al trabajar en la fase de diseño de sistemas. La figura 18.17 constituye un ejemplo para ilustrar un diagrama de secuencias que representa a un estudiante que ve su información personal y del curso. En el diagrama, **:Ver Interfaz de Usuario del Estudiante** es un ejemplo de clase de interfaz; **:Estudiante**, **:Seccion** y **:Curso** son ejemplos de clases de entidad, y **:Ver Controlador de Interfaz del Estudiante** y **:Calcular Promedio de Calificaciones** son clases de control.

El estudiante se muestra a la izquierda como un actor y proporciona un **inicioDeSesionDeUsuario** a la clase **:Ver Interfaz de Usuario del Estudiante**. Éste es un formulario Web que obtiene la contraseña e ID de usuario del estudiante. Cuando el estudiante hace clic en el botón Enviar, el formulario Web se pasa a una clase **:Ver Controlador de Interfaz del Estudiante**. Esta clase es responsable de la coordinación del envío de mensajes y de recibir la información devuelta por todas las demás clases.

Las clase **:Ver Controlador de Interfaz del Estudiante** envía un mensaje **obtenerEstudiante( )** a la clase **:Estudiante**, que lee una tabla de la base de datos y procede a devolver los **datosDelEstudiante**.

La **paginaWebDelEstudiante** es devuelta a la clase **:Ver Interfaz de Usuario del Estudiante**, que se encarga de desplegar la información en el navegador Web. En la parte inferior

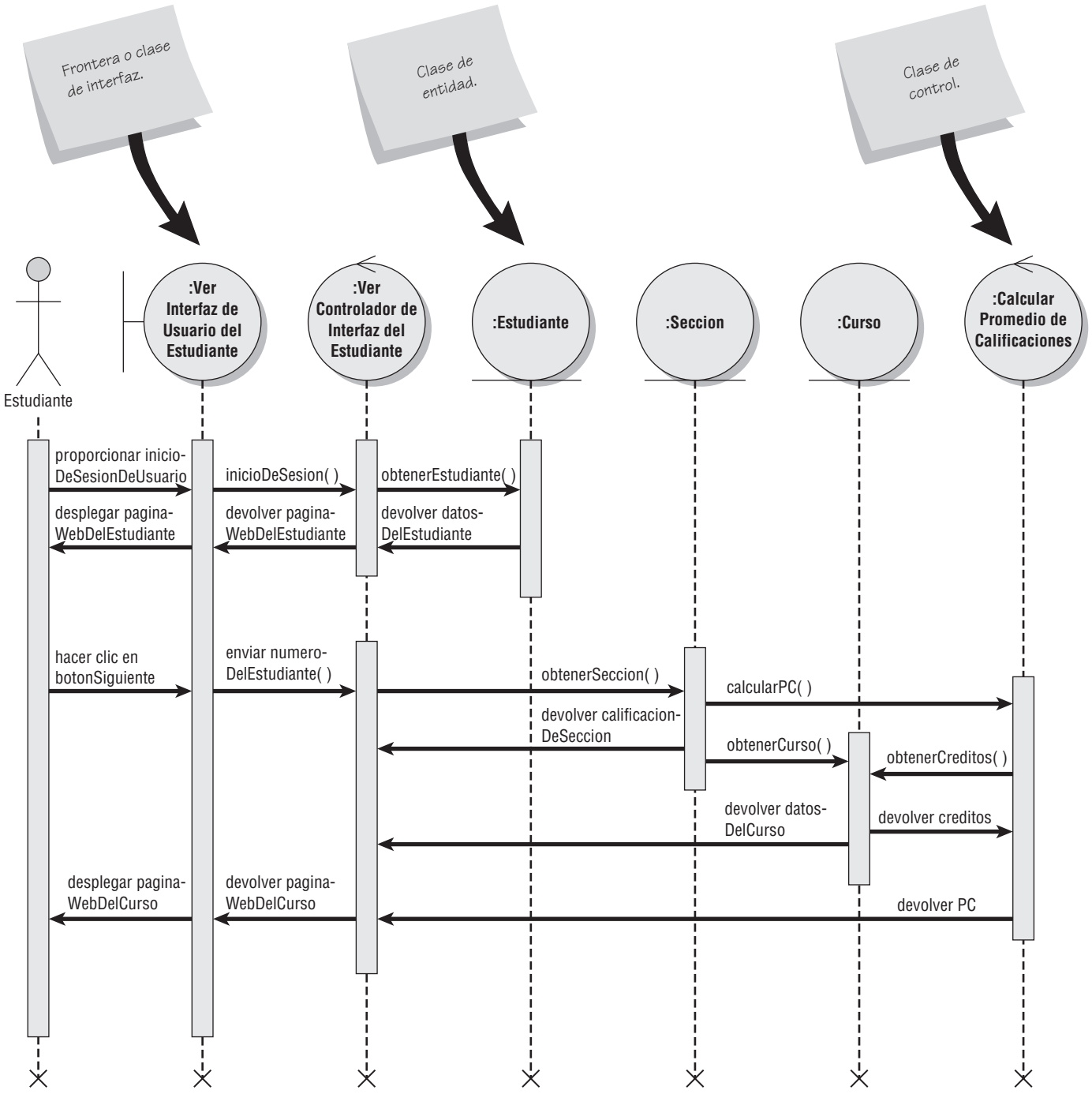


FIGURA 18.17

Diagrama de secuencias para utilizar dos páginas Web: una para información del estudiante y otra para la información de cursos.

de la página se encuentra un **botonSiguiente**, en el cual el estudiante hace clic para ver los cursos. Cuando el usuario hace clic en este botón, envía un formulario Web a la clase **:Ver Controlador de Interfaz del Estudiante**. Este formulario contiene el **numeroDelEstudiante( )**, enviado junto con **paginaWebDelEstudiante**, y se usa para enviar un mensaje a la clase **:Seccion** para obtener la calificación de la sección; es decir, la calificación que obtuvo el estudiante en ese grupo o sección del curso seleccionado. Si el **numeroDelEstudiante( )** no se envía automáticamente, significaría que el estudiante tendría que introducir su **numeroDel Estudiante( )** de nueva cuenta, lo cual denotaría una pobre interfaz de usuario porque implicaría tecleo redundante. Observe que la clase **:Estudiante** no interviene, y que el enfoque del control (la barra vertical que se conecta a la clase **:Estudiante**) finaliza antes de que comience el segundo conjunto de actividades (las flechas horizontales que apuntan ha-

cia la derecha). La clase **:Ver Controlador de Interfaz del Estudiante** envía un mensaje **obtener Seccion( )** a la clase **:Seccion** que devuelve una **calificacionDeSeccion**. La clase **:Seccion** también envía un mensaje **calcularPC( )** a la clase **:Calcular Promedio de Calificaciones**, que devuelve un mensaje a la clase **:Curso**. Esta clase devuelve los créditos, que permiten a la clase **:Calcular Promedio de Calificaciones** determinar el promedio de las calificaciones y devolverlo a la clase **:Ver Controlador de Interfaz del Estudiante**.

La clase **:Ver Controlador de Interfaz del Estudiante** repetiría el envío de mensajes a la clase **:Seccion** hasta que se incluyan todas las secciones del estudiante. En este momento, la clase **:Ver Controlador de Interfaz del Estudiante** enviaría la **paginaWebDelCurso** a la clase **:Ver Interfaz de Usuario del Estudiante**, que desplegaría la información en el navegador.

El uso de las clases de interfaz de usuario, de control y de entidad también permite al analista explorar y experimentar con el diseño. El diseño antes mencionado desplegaría toda la información personal del estudiante en una página y la información del curso en una segunda página. El analista podría modificar el diseño para que la información personal del estudiante y la información del curso aparecieran en una sola página Web. Estos dos escenarios posibles tendrían que revisarse con los usuarios para determinar la mejor alternativa.

Una de las dificultades para el analista es determinar cómo incluir el **numeroDelEstudiante** después de que se haga clic en el botón **Siguiente**, porque la clase **:Estudiante** ya no está disponible. Existen tres maneras para guardar y retransmitir datos de una página Web:

1. Incluir la información en el URL que se despliega en el área de dirección del navegador. En este caso, la línea de localización podría ser parecida a la siguiente:

   http://www.cpu.edu/student/studentinq.html?studentNumber=12345

   Todo lo que se encuentra después del signo de interrogación son datos que podrían ser utilizados por los métodos de clase. Este medio de guardar datos es fácil de implementar y con frecuencia lo utilizan los motores de búsqueda.

   Existen varias desventajas al usar este método, y el analista debe emplearlo con la debida cautela. La primera preocupación es la privacidad —cualquiera puede leer la dirección Web—. Si la aplicación involucra información médica, números de tarjeta de crédito, etc., ésta no es una buena opción. La mayoría de los navegadores también despliegan datos de las direcciones Web anteriores en sesiones subsecuentes si el usuario introduce los primeros caracteres, y de esta manera la información podría comprometerse al propiciar el robo de identidad. Una segunda desventaja es que por lo general los datos se pierden cuando el usuario cierra el navegador.
2. Guardar la información en una *cookie*, un pequeño archivo que se almacena en la computadora (el navegador) del cliente. Las *cookies* constituyen la única manera de guardar datos persistentes, que permanecen aún después de finalizar la sesión actual del navegador. Esto permite a la página Web desplegar un mensaje similar a "Bienvenido, Miguel. Si usted no es Miguel, haga clic aquí". Por lo general, las *cookies* guardan números de cuenta importantes, pero no números de tarjeta de crédito ni otra información privada. Las *cookies s*e limitan a 20 por dominio (como www.cpu.edu) y cada *cookie* debe tener 4,000 caracteres o menos.
3. Utilizar campos de formulario Web ocultos. Estos campos normalmente contienen datos enviados por el servidor, son invisibles y no ocupan espacio en la página Web. En el ejemplo de la vista de información del estudiante, la clase **:Ver Controlador de Interfaz del Estudiante** agregó un campo oculto al formulario **paginaWebDelEstudiante** con el **numeroDelEstudiante** y el **botonSiguiente**. Cuando el estudiante hace clic en el **botonSiguiente**, el **numeroDelEstudiante** se envía al servidor y de esta manera la clase **:Ver Controlador de Interfaz del Estudiante** sabe de qué estudiante debe obtener la información de curso y calificaciones. Los datos de los formularios ocultos no se guardan de una sesión del navegador a otra, por lo que se conserva la privacidad.

Los símbolos de clase también se podrían usar en diagramas de clases y de colaboración. La figura 18.18 ilustra el diagrama de clases para un estudiante que ve información

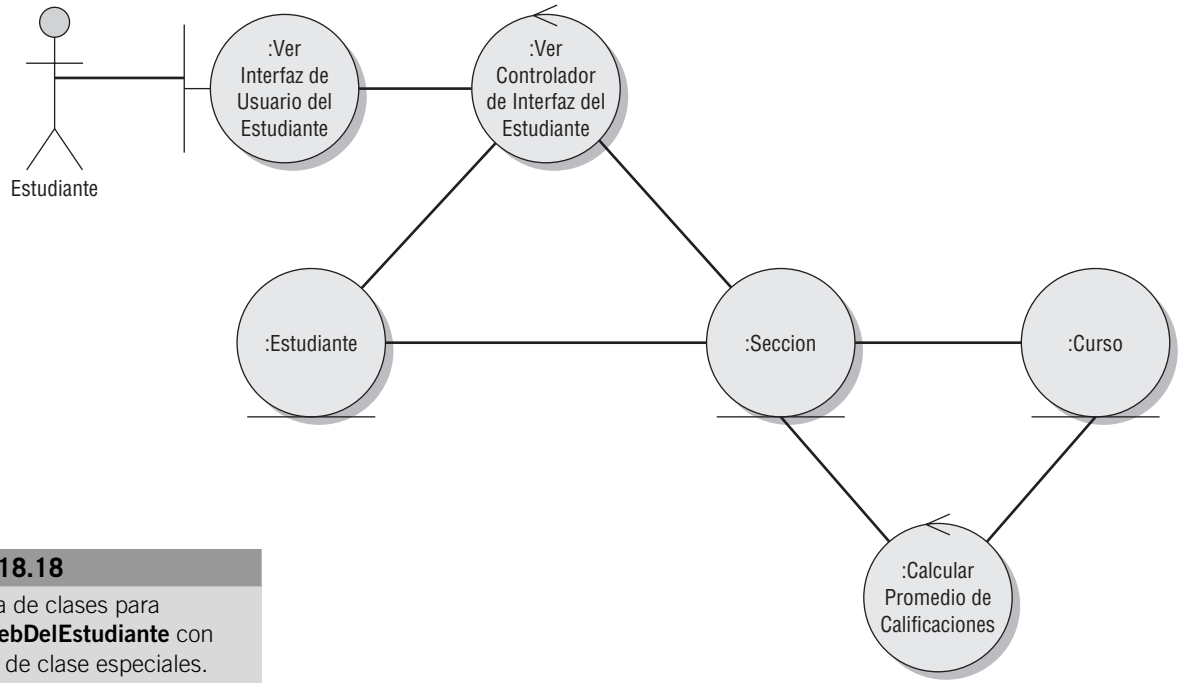


FIGURA 18.18
Diagrama de clases para **paginaWebDelEstudiante** con símbolos de clase especiales.

personal y del curso en páginas Web. Cada clase tiene atributos y métodos (los cuales no se muestran en diagramas que usan esta notación).

Si la clase es del tipo de interfaz de usuario, los atributos son los controles (o campos) de la pantalla o el formulario. Los métodos podrían ser aquellos que interactúan con la pantalla, como enviar o restablecer. También podrían ser JavaScript para una página Web, porque el código trabaja directamente con la página Web.

Si la clase es de control, los atributos podrían ser aquellos necesarios para implementar la clase, como las variables que sólo se utilicen en ella. Los métodos podrían ser aquellos utilizados para realizar cálculos, tomar decisiones y enviar mensajes a otras clases.

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

Los sitios Web podrían usar una combinación de muchas clases diferentes para satisfacer los objetivos del usuario. Por ejemplo, un sitio Web podría usar JavaScript para prevalidar los datos, y pasarlos a continuación a las clases de control del servidor, que realizan la validación completa junto con la obtención de datos. Las clases de control del servidor, a su vez, podrían devolver JavaScript a la página Web para realizar algún formato. No es raro que una aplicación Web incluya muchas clases, algunas de ellas con sólo una línea de código en un método, para conseguir el objetivo de la reusabilidad.

RELACIONES

Las relaciones son conexiones entre las clases, similares a aquellas que se encuentran en un diagrama de entidad-relación. Estas relaciones se muestran como líneas que conectan las clases en un diagrama de clases. Hay dos categorías de relaciones: asociaciones y relaciones todo/parte.

Asociaciones El tipo más simple de relación es una asociación, o una conexión estructural entre clases u objetos. Las asociaciones se muestran como una línea simple 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 re-

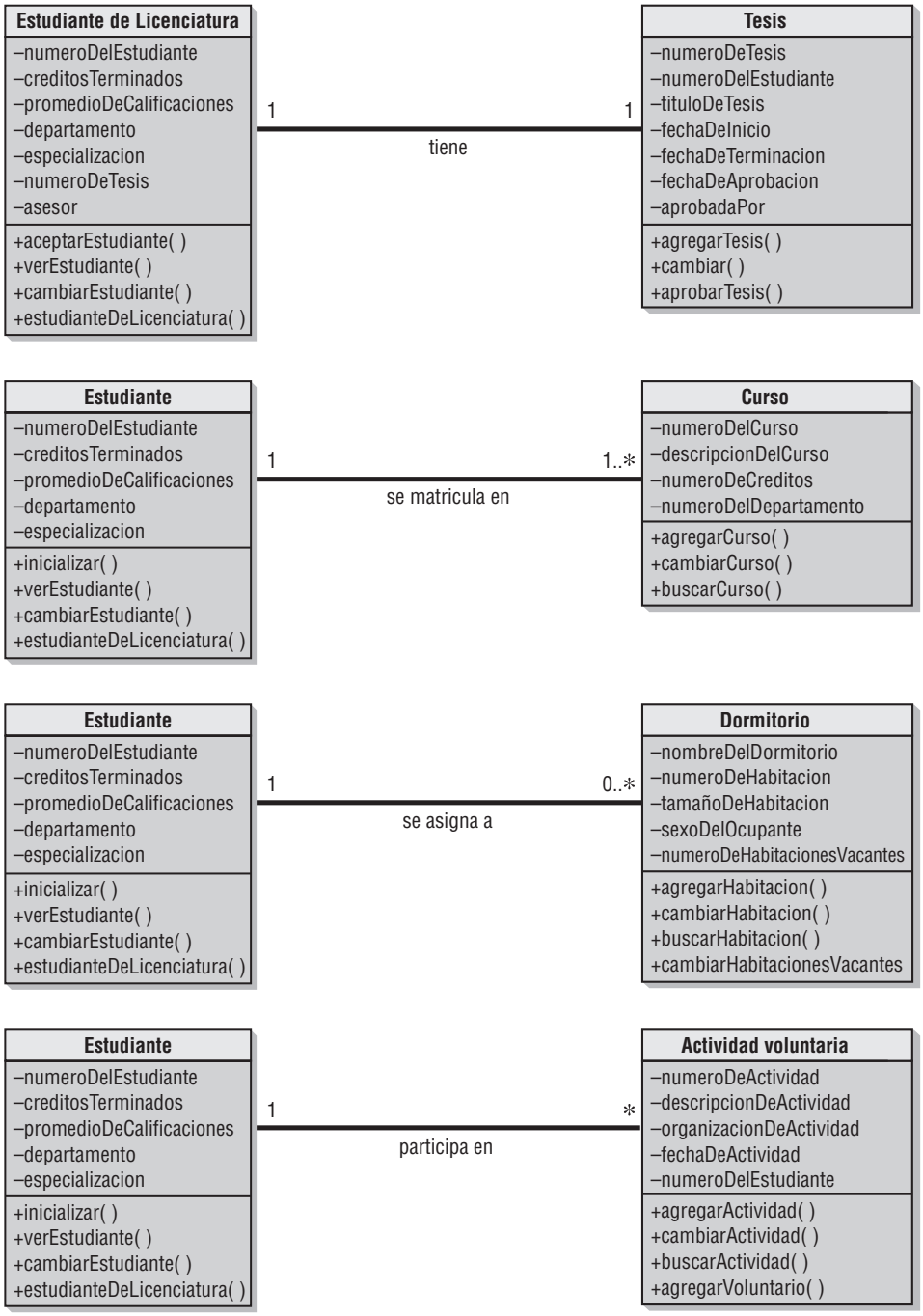


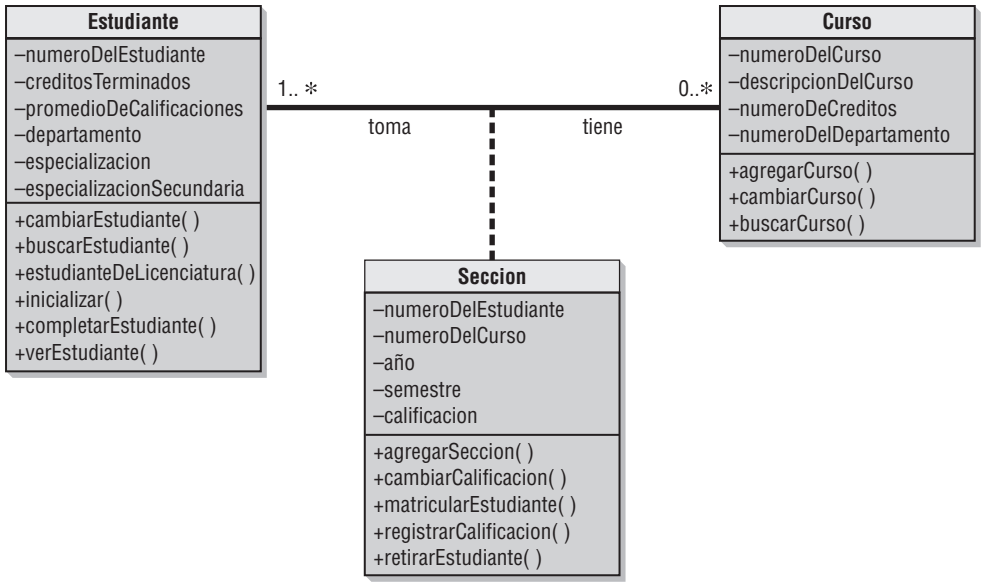
FIGURA 18.19
Tipos de asociaciones que se podrían dar en un diagrama de clases.

presenta ninguno, un uno representa uno y sólo uno, y un asterisco representa muchos. La notación 0..1 representa de cero a uno, y la notación 1..* representa de uno a muchos. Las asociaciones se ilustran en la figura 18.19.

Los diagramas de clases no restringen el límite inferior de una asociación. Por ejemplo, una asociación podría ser 5..*, lo cual indicaría que debe estar presente un mínimo de cinco. Lo mismo se aplica a los límites superiores. Por ejemplo, el número de cursos en que se matricule actualmente un estudiante podría ser 1..10, lo cual representaría de uno a 10 cursos. También 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 asociación son aquellas que se usan para dividir una asociación muchos a muchos entre clases. Éstas son similares a las entidades asociativas en un diagrama entidad-

FIGURA 18.20
Ejemplo de una clase asociativa en la cual una sección particular define la relación entre un estudiante y un curso.



relación. **Estudiante** y **Curso** tienen una relación muchos a muchos, que se resuelve agregando una clase de asociación llamada **Seccion** entre las clases **Estudiante** y **Curso**. La figura 18.20 ilustra una clase de asociación llamada **Seccion**, mostrada con una línea punteada conectada a la línea de la relación muchos a muchos.

Un objeto de una clase podría tener una relación con otros objetos de la misma clase, lo que se conoce como asociación reflexiva. Un ejemplo sería una tarea que tiene una tarea precedente, o un empleado que supervisa a otro empleado. Esto se muestra como una línea de asociación que conecta la clase a sí misma, con etiquetas que indican el nombre del papel, como tarea y tarea precedente.

Relaciones todo/parte Estas relaciones surgen cuando una clase representa al objeto total y otras clases representan partes del mismo. El todo actúa como contenedor de las partes. Estas relaciones se muestran en un diagrama de clases mediante una línea con un diamante en un extremo. El diamante se conecta al objeto total. Las relaciones todo/parte (así como la agregación, que se explica debajo) se muestran en la figura 18.21.

Una relación todo/parte podría ser un objeto entidad que tiene partes distintas, como un sistema de cómputo que incluye computadora, copiadora, monitor, etc., o un automóvil que tiene motor, sistema de frenos, transmisión, etc. Las relaciones todo/parte también se pueden usar para describir una interfaz de usuario, en la cual una pantalla de GUI contiene una serie de objetos como listas, cuadros o botones de opción, o tal vez un área de encabezado, cuerpo y pie. Las relaciones todo/parte tienen varias categorías: agregación, colección y composición.

Agregación. A menudo, una agregación se describe como una relación "tiene un". La agregación proporciona un medio para mostrar que el objeto total se compone de la suma de sus partes (otros objetos). En el ejemplo de matriculación del estudiante, el departamento *tiene un* curso y el curso *es para un* departamento. Ésta es una relación más débil, porque un departamento podría cambiarse o eliminarse y el curso todavía existiría. Un paquete de computadora podría no estar disponible, pero las impresoras y otros componentes todavía existen. El diamante al final de la línea de la relación no aparece sólido.

Colección. Una colección consta de un todo y sus miembros. Éste podría ser un distrito electoral con votantes o una biblioteca con libros. Los votantes o libros podrían cambiar, pero el todo conserva su identidad. Ésta es una asociación débil.

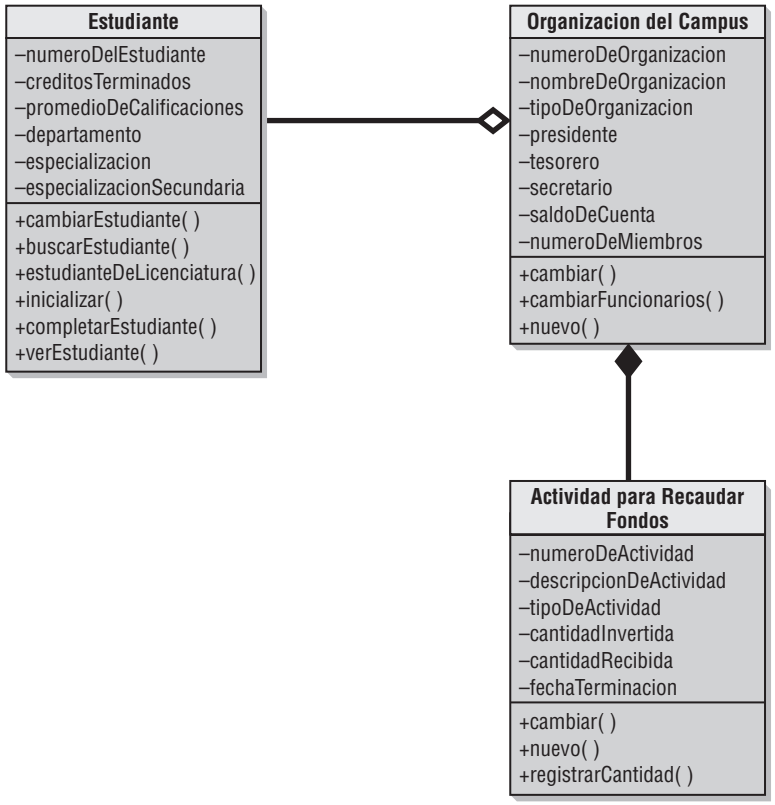


FIGURA 18.21
Ejemplo de relaciones todo/parte y de agregación.

Composición. La composición, una relación todo/parte en la cual el todo tiene una responsabilidad por la parte, es una relación aún más fuerte, y normalmente se muestra con un diamante sólido. Las palabras clave para la composición son que una clase "siempre contiene" a otra clase. Si el todo se elimina, todas las partes se eliminan. Un ejemplo sería una póliza de seguro con cláusulas adicionales. Si la póliza se cancela, las cláusulas adicionales también se cancelan. En una base de datos, se podría establecer integridad referencial para eliminar los registros hijos en cascada. En una universidad hay una relación de composición entre un curso y una tarea así como entre un curso y un examen. Si el curso se elimina, las tareas y exámenes también se eliminan.

DIAGRAMAS DE GENERALIZACIÓN/ESPECIALIZACIÓN

Un diagrama de generalización/especialización (gen/esp) entra en la categoría de diagrama de clases. En ocasiones es necesario separar las generalizaciones de las instancias específicas. Como mencionamos 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. A veces necesitamos distinguir si un oso koala es un animal o un tipo de animal. Además, un oso koala puede ser un animal de peluche. En consecuencia, a menudo requerimos clarificar 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 a menudo como una relación "es un". Por ejemplo, un automóvil *es un* vehículo y un camión *es un* vehículo. En este caso, el vehículo es la cosa general, en tanto que el automóvil y el camión son las cosas más específicas. Las relaciones de generalización se utilizan para modelar la herencia de clases y la especialización. Una clase general a veces se conoce como superclase, clase base o clase madre; una clase especializada se denomina subclase, clase derivada o clase hija.

Herencia Varias clases podrían 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 hereda o recibe los atributos y métodos de la clase general. Además, la clase especializada tiene atributos y métodos que son únicos y sólo están definidos en la clase especializada. La creación de clases generalizadas y el hecho de permitir que la clase especializada herede sus atributos y métodos fomenta la reutilización, porque el código se usa muchas veces. También ayuda a dar mantenimiento al código de los programas existentes. Esto da al analista la posibilidad de definir atributos y métodos una sola vez y usarlos muchas veces, en cada clase heredada.

Una de las características especiales del enfoque orientado a objetos es la creación y mantenimiento de grandes bibliotecas de clases que están disponibles en diversos lenguajes. Así, por ejemplo, un programador que usa Java, .NET o C# tendrá acceso a una considerable cantidad de clases que ya se han desarrollado.

Polimorfismo El polimorfismo (que significa muchas formas), o redefinición de métodos (que es diferente a la sobrecarga de métodos), es la capacidad de un programa orientado a objetos para tener varias versiones del mismo método con el mismo nombre dentro de una relación superclase/subclase. La subclase hereda un método de su clase madre pero podría agregarle comportamiento o modificarlo. La subclase podría cambiar el tipo de datos, o cambiar la forma en que trabaja el método. Por ejemplo, un cliente podría recibir un descuento adicional por volumen, y el método para calcular el total de un pedido se modifica. Se dice que el método de la subclase redefine (o sobrepone) 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 sube por la cadena de clases en busca de los métodos.

Clases abstractas Las clases abstractas son clases generales y se utilizan cuando se incluye gen/esp en el diseño. La clase general se convierte en la clase abstracta. La clase abstracta no tiene objetos directos o instancias de clase, y sólo se usa con clases especializadas. Por lo general, las clases abstractas tienen atributos y podrían incluir algunos métodos.

La figura 18.22 es un ejemplo de un diagrama de clases de gen/esp. La flecha apunta hacia la clase general, o superclase. A menudo las líneas que conectan dos o más subclases a una superclase se unen con una flecha que apunta hacia la superclase, aunque también se podrían mostrar como flechas separadas. Observe que el nivel superior es **Persona**, y representa a cualquier persona. Los atributos describen cualidades que todas las personas de una universidad poseen. 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, porque tienen atributos y métodos diferentes. Un empleado no tiene un promedio de calificaciones y un estudiante no tiene un sueldo. Ésta es una versión simple, y no incluye a empleados que sean estudiantes ni a estudiantes que trabajen para la universidad. Si se agregaran, serían subclases de las clases **Empleado** y **Estudiante**. **Empleado** tiene dos subclases, **Academico** y **Administrador**, porque hay atributos y métodos diferentes para cada una de estas clases especializadas.

Las subclases se definen mediante verbos especiales. Éstos son a menudo palabras seguidas, como *esun* para "es un", *esuntipode* para "es un tipo de" y *puedeserun* para "puede ser un".

esun	Profesor *esun* Empleado
esuntipode	Administrador *esuntipode* Empleado
puedeserun	Empleado *puedeserun* Profesor

Cómo identificar clases abstractas Usted podría identificar clases abstractas verificando si varias clases o tablas de la base de datos tienen los mismos elementos, o si varias clases tie-

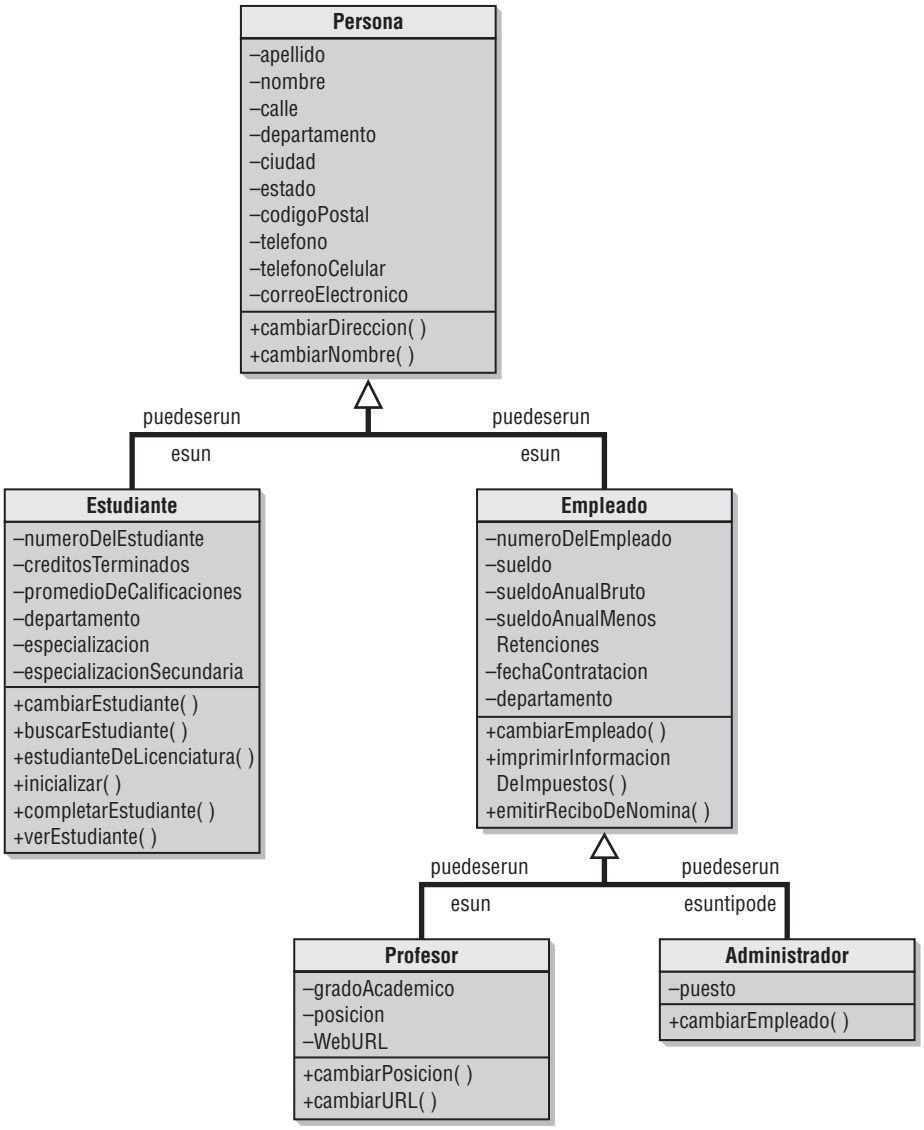


FIGURA 18.22
Un diagrama de gen/esp es un tipo mejorado de diagrama de clases.

nen los mismos métodos. Usted puede crear una clase general extrayendo los atributos y métodos comunes, o podría crear una clase especializada para los atributos y métodos únicos. En un ejemplo bancario, un retiro, un pago de un préstamo o un cheque escrito, todos tienen el mismo método: sustraen dinero del saldo del cliente.

Cómo encontrar clases Existen varias maneras para determinar clases. Se podrían descubrir durante entrevistas o sesiones de JAD (que se describieron en el capítulo 4), durante sesiones de equipo dirigidas o en sesiones de lluvia de ideas. El análisis de documentos y memorandos también podría revelar clases. Una de las maneras más fáciles es usar el método de CRC ya descrito en este capítulo. El analista también debe examinar los casos de uso en busca de sustantivos. Cada nombre podría conducir a una clase candidata o potencial. Se les llama clases candidatas porque algunos de los nombres podrían ser atributos de una clase.

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

Cómo determinar los métodos de la clase El analista debe determinar los atributos y los métodos de la clase. Los atributos son fáciles de identificar, pero es más difícil identificar los métodos que trabajan con los atributos. Algunos de los métodos son comunes, y siempre se asocian con una clase, como nuevo( ), o el método ≪create≫, que es una extensión de UML creada por una persona u organización, conocida como estereotipo. Los símbolos ≪ ≫ no son simplemente pares de símbolos mayor que y menor que, sino que se denominan comillas angulares.

Otra manera útil de determinar los métodos es examinar una matriz CLAE (vea el capítulo 7). La figura 18.23 ilustra una matriz CLAE para los ofrecimientos de cursos. Cada inicial requiere un método diferente. Si hay una C para crear, agregue un método nuevo( ). Si hay una A para la actualización, agregue un método actualizar( ) o cambiar( ). Si hay una E para eliminar, agregue un método eliminar( ) o borrar( ). Si hay una L para leer, agregue métodos para buscar, ver o imprimir. En el ejemplo mostrado, la clase **libro de texto** necesitaría un método crear para agregar un libro de texto, y un método leer para iniciar una consulta del curso, cambiar un libro de texto o encontrar un libro de texto. Si se reemplazara un libro de texto, se necesitaría un método de actualización, y si se eliminara un libro de texto, se requeriría un método para eliminar.

Mensajes Para realizar trabajo útil, la mayoría de las clases necesita comunicarse con las demás. Un objeto de una clase necesita enviar información a un objeto de otra clase a través de un mensaje, de manera similar a como se realizan las llamadas en un lenguaje de programación tradicional. Un mensaje también actúa como un comando, que le indica a la clase receptora que realice alguna tarea. Un mensaje consiste del nombre del método de 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 con el nombre del mensaje.

FIGURA 18.23
Una matriz CLAE puede ayudar a determinar los métodos necesarios. Esta matriz CLAE sirve para determinar los métodos y operaciones para los ofrecimientos de cursos.

Actividad	Departamento	Curso	Libro de Texto	Tarea	Examen
Agregar Departamento	C				
Ver Departamento	L				
Agregar Curso	L	C			
Cambiar Curso	L	A			
Consultar Curso	L	L	L	L	L
Agregar Libro de Texto	L	L	C		
Cambiar Libro de Texto		L	LA		
Buscar Libro de Texto		L	L		
Eliminar Libro de Texto		L	E		
Agregar Tarea		L		C	
Cambiar Tarea		L		LA	
Ver Tarea		L		L	
Agregar Examen		L			L
Cambiar Examen		L			LA
Ver Examen		L			L

Dado que los mensajes se envían de una clase a otra, es posible considerarlos como entrada o salida. La primera clase debe proporcionar los parámetros incluidos en el mensaje y la segunda clase los utiliza. Si existe un diagrama de flujo de datos físico hijo para el dominio del problema, podría ayudar a descubrir los métodos. El flujo de datos de un proceso primitivo a otro representa el mensaje, y los procesos primitivos deben examinarse como métodos candidatos.

DIAGRAMAS DE ESTADOS

El diagrama de estados, o de transición de estados, es otra manera de determinar los métodos de una clase. Se usa para examinar los diferentes estados que podría tener un objeto.

Un diagrama de estados se crea para una sola clase. Por lo general, los objetos se crean, sufren cambios y se eliminan.

Los objetos existen en cualquiera de estos estados, que son las condiciones de un objeto en un momento específico. Los valores de los atributos de un objeto definen el estado en que se encuentra el objeto, y en ocasiones existe un atributo, como Estado del Pedido (pendiente, surtido, empaquetado, enviado, recibido, etc.) que indica el estado. Un estado tiene un nombre con cada palabra iniciando con mayúscula. El nombre debe ser único y significativo para los usuarios. Un estado también tiene acciones de entrada y salida, las cosas que el objeto debe hacer cada vez que entra o sale de un estado determinado.

Un evento es algo que ocurre en un momento y lugar específicos. Los eventos causan un cambio en el estado del objeto, y se dice que se "dispara" una transición. Los estados separan eventos, como en el caso de un pedido que espera ser surtido, y los eventos separan estados, como en el caso de un evento Pedido Recibido o Pedido Completo.

Un evento causa la transición, y ocurre cuando se cumple una condición. Una condición es algo que da como resultado verdadero o falso, y puede ser tan sencilla como "Haga clic para confirmar el pedido". También puede ser una condición que ocurra en un método, como un artículo que esté agotado. Las condiciones se muestran entre corchetes junto a la etiqueta del evento.

También hay eventos diferidos, o eventos que sólo se realizan hasta que un objeto cambia a un estado que puede aceptarlos. Un usuario que teclea algo cuando un procesador de texto está realizando una copia de seguridad es un ejemplo de un evento diferido. Después de que termina la copia de seguridad, el texto aparece en el documento.

Los eventos se clasifican en tres categorías diferentes:

1. Señales o mensajes asíncronos, que ocurren cuando el programa que realiza la llamada no espera un mensaje de respuesta, como en el caso de una característica ejecutada de un menú.
2. Mensajes síncronos, que son llamadas a funciones o subrutinas. El objeto que llama se detiene y espera a que el control regrese a él, junto con un mensaje opcional.
3. Eventos temporales, que ocurren en un momento predeterminado. Por lo general, estos eventos no involucran un actor o un evento externo.

Los objetos materiales tienen persistencia; es decir, existen durante un largo periodo. Los vuelos de avión, los conciertos y los eventos deportivos tienen una persistencia más corta (podrían tener estados que cambian en un periodo más breve). Algunos objetos, conocidos como objetos temporales, no sobreviven el fin de una sesión. Éstos incluyen la memoria principal, datos de un URL (o localización) en la Web, páginas Web, pantallas CICS, etc. La única manera de guardar objetos temporales es almacenar información relativa a ellos, como al guardar datos de la 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 cambiarlos. Cada uno de los métodos necesitaría una pantalla o formulario Web para agregar o cambiar los atributos. Éstos se convierten en los objetos de la interfaz. Con frecuencia, la pantalla o formulario Web tendría más controles (o campos) que simplemente los atributos

que cambian. Por lo general, tendrían claves principales, información de identificación (como un nombre o dirección) y otros atributos necesarios para una buena interfaz de usuario. La excepción es un evento temporal, el cual podría usar tablas de la base de datos o una cola que contenga la información.

EJEMPLO DE UNA TRANSICIÓN DE ESTADO

Considere a un estudiante que se matricula en una universidad y los diversos estados por los que tendría que atravesar. Tres de los estados se listan en detalle a continuación:

Estado:	Estudiante Potencial
Evento:	Solicitud Enviada
Método:	nuevo( )
Atributos modificados:	Número
	Nombre
	Dirección
Interfaz de usuario:	Formulario Web de Solicitud del Estudiante
Estado:	Estudiante Aceptado
Evento:	Requisitos Cumplidos
Método:	aceptarEstudiante( )
Atributos modificados:	Fecha de Admisión
	Estado del Estudiante
	Devolver Carta de Aceptación
Interfaz de usuario:	Pantalla para Aceptar al Estudiante
Estado:	Dormitorio Asignado al Estudiante
Evento:	Dormitorio Seleccionado
Método:	asignarDormitorio( )
Atributos modificados:	Nombre de Dormitorio
	Dormitorio
	Plan de Comidas
Interfaz de usuario:	Pantalla para Asignar Dormitorio al Estudiante

Los otros estados son **Estudiante del Programa**, **Estudiante Actual**, **Estudiante Permanente** y **Estudiante de Licenciatura**. Cada estado tendría un evento, métodos, atributos modificados y una interfaz de usuario asociada. Esta serie de estados se puede usar para determinar los atributos y métodos que conforman la clase.

Los estados y eventos que activan los cambios se podrían representar en un diagrama de estados (o un diagrama de transición de estados). En la figura 18.24 se ilustra el diagrama de estados para el **Estudiante**. Los estados se representan mediante rectángulos, y los eventos o actividades son las flechas que unen los estados y causan que un estado cambie a otro estado. Los eventos de transición se nombran en pasado, porque ya ocurrieron para crear la transición.

No se crean diagramas de estados para todas las clases. Estos diagramas se crean cuando:

1. Una clase tiene un ciclo de vida complejo.
2. Una instancia de una clase podría actualizar sus atributos de varias maneras a través de su ciclo de vida.
3. Una clase tiene un ciclo de vida operacional.
4. Dos clases dependen entre sí.
5. El comportamiento actual del objeto depende de lo que haya ocurrido antes.

Cuando examine un diagrama de estados, aproveche la oportunidad para buscar errores y excepciones. Inspeccione el diagrama para ver si los eventos ocurren en un momento equivocado. También revise si todos los eventos y estados se han representado. Los diagra-

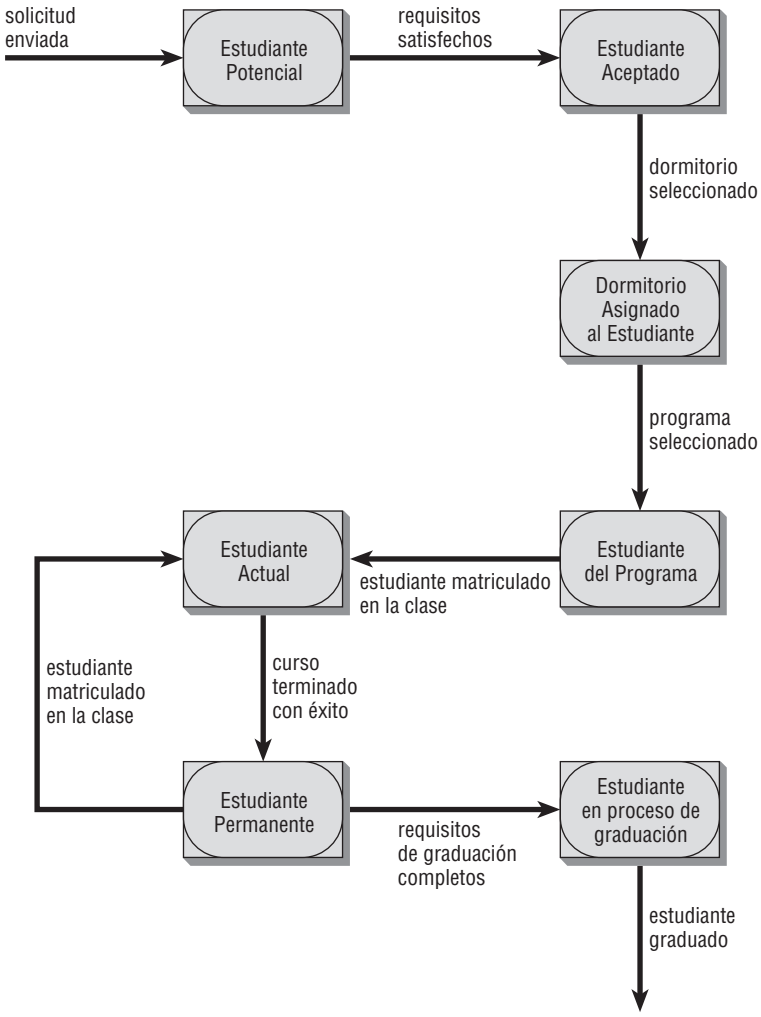


FIGURA 18.24
Diagrama de estados que muestra la manera en que un estudiante progresa de ser un estudiante potencial a un estudiante de licenciatura.

mas de estados sólo tienen que evitar dos problemas. Asegúrese de que un estado no tenga todas las transiciones dirigiéndose hacia el estado o todas sus transacciones saliendo del mismo.

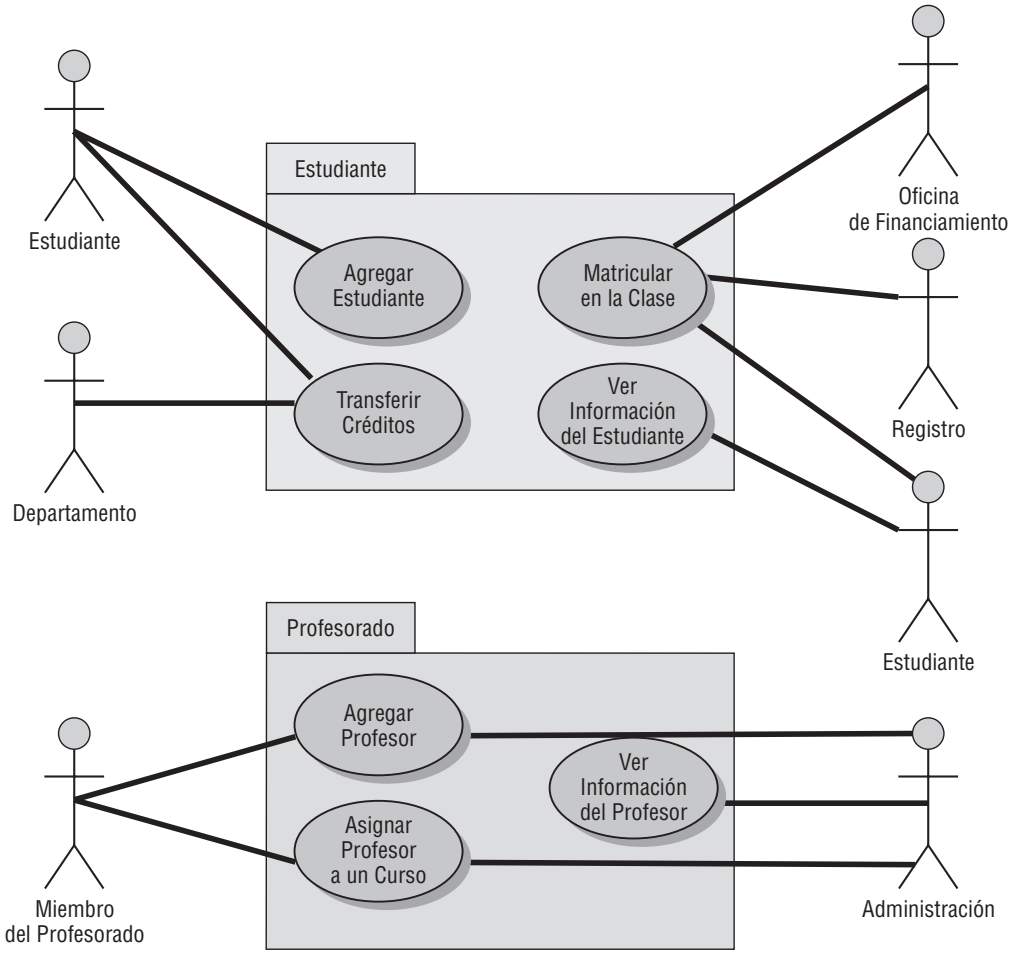
Cada estado debe tener por lo menos una transición que entre y salga de él. Algunos diagramas de estados utilizan los mismos símbolos de inicio y terminación que los diagramas de actividades: un círculo sólido representa el inicio y círculos concéntricos con el centro sólido indican el final del diagrama.

PAQUETES Y OTROS ARTEFACTOS DE UML

Los paquetes son los contenedores para otros elementos de UML, como los casos de uso o las clases. Los paquetes pueden mostrar el particionamiento del sistema, indicando cuáles clases o casos de uso se agrupan en un subsistema, y se conocen como paquetes lógicos. También pueden ser paquetes de componentes (los cuales contienen los componentes 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 pestaña de la carpeta o centrado en esta última. La creación de los paquetes se puede realizar durante el análisis de sistemas, o más tarde en la etapa de diseño del sistema. Los paquetes también podrían tener relaciones, de manera similar a los diagramas de clases, que podrían incluir asociaciones y herencia.

La figura 18.25 constituye un ejemplo de un diagrama de paquete de casos de uso. Muestra que cuatro casos de uso, **Agregar Estudiante**, **Matricularse en la Clase**, **Transferir Créditos** y **Ver Información del Estudiante**, forman parte del paquete **Estudiante**. Hay tres casos de uso, **Agregar Profesor**, **Ver Información del Profesor** y **Asignar Profesor al Curso** que son parte del paquete **Profesor**.

FIGURA 18.25
Los casos de uso pueden agruparse en paquetes.

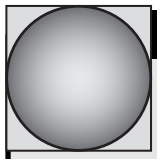


Conforme usted continúe construyendo diagramas, necesitará utilizar diagramas de componentes, diagramas de despliegue y elementos o cosas de anotación. Éstos permiten perspectivas diferentes en el trabajo que se realiza.

El diagrama de componentes es similar a un diagrama de clases, pero da una visión más general de la arquitectura del sistema. El diagrama de componentes muestra los componentes del sistema, como un archivo de clases, un paquete, las bibliotecas compartidas, una base de datos, etc., y cómo se relacionan entre sí. Los componentes individuales de un diagrama de componentes se consideran en 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 que se despliega. El diagrama de despliegue puede mostrar servidores, estaciones de trabajo, impresoras, etcétera.

Los elementos o cosas de anotación dan más información sobre el sistema a los desarrolladores. Consisten en notas que pueden adjuntarse a cualquier elemento de UML: objetos, comportamientos, relaciones, diagramas, o cualquier cosa que requiera descripciones detalladas, suposiciones o información relativa al diseño y funcionamiento del sistema. El éxito de UML depende de que la documentación del modelo del sistema sea suficientemente completa y precisa para proporcionar la información necesaria al equipo de desarrollo. Las notas proporcionan una fuente de conocimiento y comprensión común sobre el sistema, que es útil para coordinar a los desarrolladores. Las notas se muestran como un símbolo de papel con una esquina doblada y una línea que las conecta con el área que requiere atención.



OPORTUNIDAD DE CONSULTORÍA 18.3

DESARROLLO DE UN SISTEMA QUE SE RETRASÓ MUCHO TIEMPO: USO DEL ANÁLISIS ORIENTADO A OBJETOS PARA EL SISTEMA DE LA BIBLIOTECA PÚBLICA RUMINSKI*

Cuando Dewey Dezmal entra al salón de lectura de la biblioteca pública Ruminski, de techo alto y paneles de madera, una mujer joven, sentada ante una larga mesa de roble, se asoma desde atrás de un monitor, ve a Dewey, se levanta y dice: "Bienvenido. Yo soy Peri Otticle, directora de la biblioteca. Tengo entendido que usted está aquí para ayudarnos a desarrollar nuestro nuevo sistema de información".

Aún impresionado con la belleza del viejo edificio de la biblioteca y el marcado contraste de la alta tecnología en medio de tanta historia, Dewey se presentó a sí mismo como un analista de sistemas con una pequeña empresa de consultoría, People and Objects, Inc.

"Es la primera vez que me asignan un proyecto de este tipo, aunque en realidad es interesante para mí, porque tengo una licenciatura en la Escuela de Estudios de Información de la Upstate University. Ahí se puede especializar en biblioteconomía o en TI, así que muchos de mis compañeros tuvieron que trabajar en bibliotecas públicas. Yo me incliné por la licenciatura en TI."

"Entonces, deberemos acoplarnos bien a trabajar", dice Peri. "Vamos a mi oficina para no distraer a nadie, y ahí te hablaré de un informe que escribí."

Cuando pasan por la hermosa escalera de caracol, que parece esculpida en madera, Peri sugiere a Dewey que observe a su alrededor y dice: "Tal vez te asombre la majestuosidad del edificio porque somos una institución pública. Somos afortunados. Nuestro benefactor es Valerian Ruminski. De hecho, él ha donado tanto dinero a tantas bibliotecas que el personal lo llama afectuosamente 'Valerian el Bibliotecario' ".

Mientras pasan junto a varios lectores, Peri continúa: "Como puedes ver, es un lugar muy ocupado. Y, sin tomar en cuenta nuestro histórico entorno, no nos quedamos en el pasado".

Dewey lee el informe que Peri le entregó. Una sección grande se titula "Resumen de las principales demandas de los lectores", y la lista comienza así:

- Un lector registrado en el sistema de la biblioteca puede pedir libros y revistas prestados al sistema.
- El sistema de la biblioteca debe verificar periódicamente (por lo menos una vez por semana) si se ha vencido la fecha de devolución de algún libro o revista pedidos por un lector. En ese caso se enviará un aviso al lector.
- Un lector puede reservar un libro o una revista que se hayan prestado o que estén en el proceso de compra por parte de la biblioteca. La reservación debe cancelarse cuando el lector reciba en préstamo el libro o revista o a través de un servicio de cancelación formal.

Al tiempo que Dewey levanta la vista del informe, le dice a Peri: "Estoy empezando a entender los requerimientos del lector (o usuario). Veo muchas similitudes entre la vieja biblioteca de mi universidad y la de ustedes. Sin embargo, un elemento que no vi aquí es cómo deciden lo que la biblioteca debe coleccionar y de lo que debe deshacerse".

Peri se ríe entre dientes y contesta: "¡Qué pregunta tan inteligente! El personal de la biblioteca se encarga de la compra de nuevos libros y revistas para la biblioteca. Si algo es popular, se compran más de dos copias. Nosotros podemos crear, actualizar y eliminar información relacionada con los títulos y copias de libros y revistas, lectores, préstamo de materiales y reservaciones en el sistema".

Dewey levanta la vista de su bloc de notas y dice: "Aún estoy un poco desconcertado. ¿Cuál es la diferencia entre los términos *título* y *copia*?"

Peri responde: "La biblioteca puede tener varias copias de un título. Por lo general, el título se refiere al nombre de un libro o revista. Las copias de un título son las que se prestan fuera de la biblioteca".

Basado en la entrevista de Dewey con Peri y en la descripción de requerimientos que esta última asentó en su informe, así como en su propia experiencia en el uso de los servicios de una biblioteca, use el UML para contestar las preguntas siguientes. (*Nota:* es importante asegurarse de que sus soluciones sean lógicas y funcionales. Indique sus suposiciones claramente siempre que sea necesario.)

1. Dibuje un diagrama de casos de uso para representar actores y casos de uso del sistema.
2. Describa los pasos para cada caso de uso (como lo hicimos para organizar los casos de uso).
3. Describa escenarios para los pasos. Es decir, cree un lector y escriba un ejemplo de éste realizando cada paso.
4. Desarrolle una lista de cosas.
5. Cree diagramas de secuencias para casos de uso con base en los pasos y escenarios.
6. Complete el diagrama de clases determinando las relaciones entre las clases y definiendo los atributos y métodos de cada clase. Utilice la cosa de agrupamiento conocida como paquete para simplificar el diagrama de clases.

* Basado en un problema escrito por el doctor Wayne Huang.

UML EN LA PRÁCTICA

El lenguaje de modelado unificado ofrece un conjunto de herramientas útil para el análisis y diseño de sistemas. Como en el caso de cualquier producto creado con ayuda de herramientas, el valor de los productos de UML en un proyecto depende de la pericia con que el

analista de sistemas maneje las herramientas. El analista usará en principio el conjunto de herramientas de UML para dividir los requerimientos del sistema en un modelo de casos de uso y un modelo de objetos. El modelo de casos de uso describe a los casos de uso y los actores. El modelo de objetos describe los objetos, así como sus asociaciones, responsabilidades, colaboradores y atributos.

1. Defina el modelo de casos de uso.
   - Busque los actores en el dominio del problema revisando los requerimientos del sistema y entrevistando a algunos expertos de negocios.
   - Identifique los eventos principales iniciados por los actores y desarrolle un conjunto de casos de uso primarios a un nivel muy alto que describa los eventos desde la perspectiva de cada actor.
   - Desarrolle los diagramas de casos de uso para aclarar la manera en que los actores se relacionan 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. Proporcione detalles adicionales desarrollando escenarios de caso de uso que documenten los flujos alternos de los casos de uso primarios.
   - Revise los escenarios de caso de uso 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 coincidan en que los escenarios de caso de uso están completos y exactos.
2. Continúe la elaboración de diagramas de UML para modelar el sistema durante la fase de análisis.
   - Derive diagramas de actividades a partir de los diagramas de casos de uso.
   - Desarrolle diagramas de secuencias y de colaboración a partir de los escenarios de caso de uso.
   - Revise los diagramas de secuencias 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 coincidan en que los diagramas de secuencias están completos y exactos. Esta revisión adicional de los diagramas de secuencias gráficos a menudo proporciona a los expertos del área de negocios una oportunidad para reconsiderar y refinar los procesos con mayor detalle que en la revisión a los escenarios de caso de uso.
3. Desarrolle los diagramas de clases.
   - Busque sustantivos en los casos de uso y haga una lista. Los sustantivos son objetos potenciales. Una vez que los identifique, busque similitudes y diferencias en el estado o el comportamiento de los objetos, y a continuación elabore las clases.
   - Defina las principales relaciones entre las clases. Busque relaciones "tiene un" y "es un" entre las clases.
   - Examine los diagramas de casos de uso y de secuencias con el fin de determinar las clases.
   - Empezando con los casos de uso más importantes para el diseño del sistema, cree diagramas de clases que muestren las clases y relaciones que existen en los casos de uso. Un diagrama de clases podría representar las clases y relaciones descritas en diversos casos de uso relacionados.
4. Dibuje diagramas de estados.
   - Desarrolle diagramas de estados para algunos diagramas de clases con el propósito de proporcionar un análisis más profundo del sistema en este punto. Utilice diagramas de estados para comprender procesos complejos que no pueden derivarse totalmente a partir de los diagramas de secuencias.
   - Determine métodos examinando los diagramas de estados. Derive atributos de la clase (datos) de un estado a partir de casos de uso, expertos del área de negocios y métodos de la clase. Indique si los métodos y atributos de la clase son públicos (accesibles externamente) o privados (interior a la clase). Los diagramas de estados son sumamente útiles para modificar diagramas de clases.

5. Comience el diseño de sistemas refinando los diagramas de UML, y utilícelos para derivar clases y sus atributos y métodos.
   - Revise todos los diagramas de UML que haya en el sistema. Escriba especificaciones para cada clase, incluyendo sus atributos, métodos y descripciones. Revise los diagramas de secuencias para identificar otros métodos de clase.
   - Desarrolle especificaciones que detallen los requerimientos de entrada y salida de los métodos, junto con una descripción detallada del procesamiento interno del método.
   - Elabore otro conjunto de diagramas de secuencias (si es necesario) para reflejar los métodos de la clase actual y la manera en que interactúan entre sí y con las interfaces del sistema.
   - Genere diagramas de clases utilizando los símbolos de clase especializados para las clases de límite o de interfaz, las clases de entidad y las clases de control.
   - Analice los diagramas de clases para derivar los componentes del sistema; es decir, clases que tengan relación funcional y lógica y que se compilarán y desplegarán juntas como una biblioteca (.DLL), un objeto .COM, un bean de Java, un paquete, etcétera.
   - Desarrolle diagramas de despliegue para indicar la manera en que se desplegarán los componentes de su sistema en el ambiente de producción.
6. Documente en detalle el diseño de su sistema. Este paso es crucial. Entre más completa sea la información que proporcione al equipo de desarrollo a través de la documentación y diagramas de UML, más rápido será el desarrollo y más sólido será el sistema final.

LA IMPORTANCIA DE USAR UML PARA EL MODELADO

El UML es una herramienta poderosa que puede mejorar en gran medida la calidad del análisis y diseño de su sistema, y puede esperarse que las prácticas mejoradas se traduzcan en sistemas de mayor calidad.

Al utilizar el UML de manera iterativa en el análisis y el diseño, usted puede conseguir que los equipos de negocios y de TI comprendan mucho mejor los requerimientos del sistema y los procesos que se tienen que realizar en el sistema para cumplir tales requerimientos.

La primera iteración de análisis debe darse en un nivel muy alto para identificar los objetivos generales del sistema y validar los requerimientos a través del análisis de los casos de uso. La identificación de los actores y la definición del modelo de caso de uso inicial son parte de esta primera iteración. Las iteraciones de análisis subsecuentes refinan aún más los requerimientos del sistema a través del desarrollo de escenarios de caso de uso, diagramas de clases, diagramas de secuencias, diagramas de estados, etc. En cada iteración se realiza una revisión más detallada del diseño del sistema hasta que las cosas y las relaciones del sistema se encuentran definidas de una manera clara y precisa en los documentos de UML.

Cuando su análisis y diseño estén terminados, usted debe tener un conjunto de especificaciones preciso y detallado de las clases, escenarios, actividades y secuencias del sistema. En general, usted puede determinar la minuciosidad del análisis y el diseño de un sistema según la cantidad de tiempo requerido para desarrollarlo y la calidad resultante del producto entregado.

Al desarrollar un sistema, a menudo se ignora el hecho de que entre más progrese un proyecto, más costosos serán los cambios a los requerimientos de negocios del sistema. Cualquier cambio al diseño de un sistema con una herramienta CASE, o incluso en papel, durante las fases de análisis y diseño de un proyecto es más sencillo, más rápido y mucho menos costoso que hacerlo durante la fase de desarrollo del proyecto.

Desafortunadamente algunos empresarios tienen poca visión y creen que un programador o analista sólo trabaja cuando está codificando. Algunos empresarios suponen erró-

OPORTUNIDAD DE CONSULTORÍA 18.4

C-SHORE++

"¡Ellos quieren reprogramar radicalmente el núcleo de la interfaz de usuario del sistema de servicio al cliente (CSR)!", dice Bradley Vargo, el director de desarrollo de Sistemas de Información de C-Shore Mutual Funds. "Hace tan sólo ocho meses que terminamos un proyecto de desarrollo del Sistema de CSR que duró dos años. Durante todo el proyecto tuvimos que soportar constantes cambios de requerimientos. Todos los meses, la gente del Departamento de Marketing inventaba alguna nueva característica competitiva de servicio al cliente, y a la semana, el grupo de CSR tenía que realizar una cantidad enorme de cambios a la especificación del Sistema de CSR. ¡Creí que nunca terminaríamos ese proyecto! Ahora parece que tendremos que empezar un nuevo proyecto de reprogramación de un sistema que tiene menos de un año de vida. ¡Nosotros habíamos previsto que este sistema tendría una vida útil de siete años! Ahora pienso que podría estar en una reconstrucción eterna."

Bradley está conversando con Rachael Ciupek, la analista de sistemas de aplicaciones responsable del Sistema de CSR, y Bridget Ciupek, su hermana y la programadora que escribió la mayor parte de la interfaz de usuario. "Tranquilízate, Bradley", dice Rachael. "No es culpa de los muchachos de Marketing o de CSR. La naturaleza de nuestro negocio ha sufrido los efectos del ritmo acelerado de la competencia. Marketing no inventa estos cambios sólo por fastidiar. A menudo ellos tienen que responder a los nuevos sistemas computarizados de servicio al cliente que ofrece nuestra competencia. Nosotros tenemos que ir adelante de ellos, o por lo menos mantenernos al paso, ¡o todos tendremos que buscar nuevos empleos!"

"Bradley, Rachael, creo que es mejor que sepan que la situación podría ser peor de lo que ustedes piensan", interviene Bridget. "De cualquier modo, en realidad los programadores han estado haciendo pequeños cambios a la interfaz de usuario del Sistema CSR durante los últimos ocho meses. Los usuarios de CSR nos han estado llamando directamente implorando ayuda. Por lo general, lo único que piden es un pequeño cambio a una parte aislada del sistema, pero esto ha requerido mucho trabajo porque tenemos que volver a comprobar todo el sistema. Ustedes saben que los efectos de un pequeño cambio pueden tener repercusiones importantes en un programa grande. Hemos facturado el tiempo de mantenimiento del programa como si sólo estuviéramos poniendo a punto el sistema terminado. Aunque los cambios han sido graduales, en ocho meses hemos vuelto a escribir alrededor de una cuarta parte del código de la interfaz de usuario del Sistema de CSR. El trabajo no ha disminuido. Aún es bastante considerable."

"Lo que me estás diciendo", responde Bradley, "es que tenemos necesidades del sistema en esta área que han estado cambiando constantemente mientras nosotros tratamos de escribir especificaciones, código del programa y una solución fija para lo que es un problema fluido. ¿Cómo podemos permitirnos el lujo de escribir programas si sólo durarán unos meses sin necesitar un mantenimiento costoso?".

¿Cómo puede Bradley manejar un proceso de desarrollo de sistemas que ya no incluye entre sus metas procesos de negocios fijos o constantes? ¿Hay una forma en que Rachael pueda manejar una especificación y controlar los costos de mantenimiento cuando a los programadores les piden constantemente que refinen partes aisladas de un programa grande? Tenga presente que un objetivo importante es ofrecer apoyo oportuno a las necesidades de los usuarios y a las estrategias de negocios de la organización.

neamente que la productividad del programador únicamente puede medirse por la cantidad de código que produce, sin reconocer que la elaboración de diagramas ahorra tiempo y dinero que podrían desperdiciarse si se generaran prototipos de un proyecto sin una planificación adecuada.

En esta situación es muy apropiada una analogía para construir una casa. Aunque contrate a un constructor para que construya su casa, no le agradaría vivir en una estructura construida sin planear, a la que se le agregaran habitaciones y características al azar sin tomar en cuenta la función o el costo. Usted desea que el constructor edifique su diseño con base en los planos que contienen las especificaciones que han sido cuidadosamente revisadas por todos los involucrados. Como miembro de un equipo de analistas estrechamente supervisado: "A la larga, escribir un proyecto en papel antes de codificar dará como resultado un menor costo. Es mucho más barato borrar un diagrama que cambiar código".

Cuando los requerimientos de negocios cambian durante la fase de análisis, tal vez sea necesario volver a dibujar algunos diagramas de UML. Sin embargo, si cambian durante la fase de desarrollo, tal vez se necesite una cantidad sustancial de tiempo y dinero para rediseñar, codificar y probar nuevamente el sistema. Al confirmar en papel su análisis y diseño (sobre todo mediante diagramas de UML) con los usuarios expertos del área de negocios, usted contribuye a garantizar que se cumplirán los requerimientos de negocios correctos cuando el sistema esté terminado.

RESUMEN

Los sistemas orientados a objetos describen las entidades como objetos. Éstos son parte de un concepto general llamado clases, la unidad principal de análisis en el análisis y diseño orientado a objetos. Cuando se introdujo por primera vez el enfoque orientado a objetos, sus partidarios citaron la reusabilidad de los objetos como su principal beneficio. Aunque la reusabilidad es la meta principal, el mantenimiento de los sistemas también es muy importante.

Los analistas pueden usar tarjetas CRC para empezar el proceso de modelado de objetos de una manera informal. Es posible agregar pensamiento del objeto a las tarjetas CRC para ayudar al analista a refinar responsabilidades en tareas más y más pequeñas. Las sesiones de CRC se pueden realizar con un grupo de analistas para determinar las clases y las responsabilidades de manera interactiva.

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

Los principales componentes de UML son cosas, relaciones y diagramas. Los diagramas se relacionan entre sí. Las cosas estructurales son las más comunes; entre ellas se encuentran las clases, las interfaces, los casos de uso y muchos otros elementos que proporcionan una manera de crear modelos. Las cosas estructurales permiten al usuario describir relaciones. Las cosas relativas al comportamiento describen cómo trabajan las cosas. Las cosas agrupadas se usan para definir límites. Las cosas de anotación permiten al analista agregar notas a los diagramas.

Las relaciones constituyen el pegamento que une las cosas. 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 usan 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. Entre éstos se incluyen diagramas de caso de uso, de actividades, de secuencias, de colaboración, de clases y de estado. Además de los diagramas, los analistas pueden describir un caso de uso mediante un escenario de caso de uso.

Al usar el UML de manera iterativa en el análisis y el diseño, usted puede lograr que los equipos de negocios y de tecnología de la información comprendan mucho mejor los requerimientos del sistema y los procesos que se tienen que realizar en este último para satisfacer los requerimientos.

PALABRAS Y FRASES CLAVE

actor
agregación
asociación
barra de sincronización
bifurcación
carril
caso de uso primario
clase
clase abstracta
clase de control
clase de entidad
clase límite
colaboración
comunica
cosa de anotación
dependencias

diagrama de actividades
diagrama de casos de uso
diagrama de clases
diagrama de colaboración
diagrama de despliegue
diagrama de estados
diagrama de secuencias
escenario de caso de uso
estado
estructura todo/parte
evento
evento temporal
extiende
fusión
generaliza
generalización/especialización (gen/esp)
herencia
incluye
lenguaje de modelado unificado (UML)
mensaje
mensaje asíncrono
mensaje síncrono
objeto
orientado a objetos
paquete
polimorfismo
proceso unificado
rama
redefinición de métodos
relación
ruta feliz
ruta principal
sobrecarga de métodos
tarjetas CRC
unión

PREGUNTAS DE REPASO

1. Mencione dos razones para adoptar un enfoque orientado 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 aporta el Pensamiento del Objeto a la tarjeta CRC.
6. ¿Qué es el UML?
7. ¿Cuáles son los tres elementos principales de UML?
8. Mencione qué incluye el concepto de cosas estructurales.
9. Mencione qué incluye el concepto de cosas del comportamiento.
10. ¿Cuáles son los dos tipos principales de diagramas en UML?
11. Mencione qué diagramas se incluyen en los diagramas estructurales.
12. Mencione qué diagramas se incluyen en los diagramas del comportamiento.
13. ¿Qué describe un modelo de caso de uso?
14. ¿Usted describiría a un modelo de caso de uso como modelo lógico o físico del sistema? Argumente su respuesta en un párrafo.
15. Defina lo que es un actor en un diagrama de caso de uso.
16. ¿Cuáles son las tres cosas que siempre debe describir un caso de uso?
17. ¿En los diagramas de casos de uso, qué tipo de relaciones constituyen comunica, incluye, extiende y generaliza?
18. ¿Cuáles son dos nombres adicionales para el caso de uso primario?
19. ¿Cuáles son las tres áreas principales que se proporcionan en un caso de uso primario?
20. ¿Qué describe un diagrama de actividades?
21. Describa en un párrafo cuál es la utilidad de los carriles en los diagramas de actividades.
22. ¿Qué puede ilustrarse en un diagrama de secuencias o de colaboración?
23. ¿Por qué la definición de clases es una tarea importante en el análisis orientado a objetos?
24. ¿Qué puede mostrarse en un diagrama de clases?
25. Defina la sobrecarga de métodos.
26. Mencione las cuatro categorías en las que se dividen las clases.
27. ¿Cuáles son las dos categorías de relaciones entre las clases?
28. ¿Para qué se usan los diagramas de gen/esp?
29. ¿De qué otra manera se conoce el polimorfismo?
30. ¿Qué describe un diagrama de estados?
31. ¿Qué es un paquete en el enfoque de UML?
32. ¿Por qué es importante utilizar el UML para modelar?

PROBLEMAS

1. Elabore una serie de tarjetas CRC para la División de Catálogos de World's Trend. Una vez que se realiza un pedido, el personal dedicado al cumplimiento de pedidos se hace cargo de él y revisa la disponibilidad, surte el pedido y calcula el total del mismo. Utilice cinco tarjetas CRC, una para cada una de las siguientes clases: pedido, cumplimiento del pedido, inventario, producto y cliente. Complete la sección de clases, responsabilidades y colaboradores.
2. Termine las tarjetas CRC del problema 1 creando enunciados de pensamiento del objeto y nombres de las propiedades para cada una de las cinco clases.
3. Dibuje un diagrama de casos de uso para la División de Catálogos de Wolrd's Trend.
4. Para el problema de FilmMagic de la Oportunidad de consultoría 18.1, dibuje un diagrama de clases en UML.
5. Para el problema de FilmMagic de la Oportunidad de consultoría 18.1, dibuje un diagrama de estados para (a) **Cliente** y (b) **Vídeo**.
6. Dibuje cuatro ejemplos que muestren cuatro tipos de relaciones de comportamiento para el concesionario de automóviles BMW Joel Porter's. ¿Qué tipo de relación se requiere cuando un cliente debe solicitar financiamiento? ¿Hay actividades comunes cuando una persona arrienda o compra un automóvil? ¿Qué tipo de relación se da entre un empleado que se desempeña como gerente o uno que es vendedor?
7. Dibuje un diagrama de colaboración para un estudiante que toma un curso de un maestro que es miembro de la facultad.
8. El condado de Coleman tiene un conmutador telefónico que maneja las interconexiones de las llamadas entre quienes las realizan y quienes las reciben. Dados estos tres actores, dibuje un diagrama de secuencias sencillo para hacer una llamada telefónica.
9. 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.
10. Use el UML para dibujar ejemplos de las cuatro relaciones estructurales para la Clínica Kirt.
11. Escriba un escenario de caso de uso para un paciente que ve a un médico de la Clínica Kirt.

BIBLIOGRAFÍA SELECCIONADA

Beck, K. y W. Cunningham, "Laboratory for Teaching Object-Oriented Thinking", OOPSLA'89, como se citó en D. Butler, CRC Card Session Tutorial. Disponible en: <www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/tutorial.html>. Última visita, 6 de febrero de 2001.

Bellin, D. y S. Suchman Simone, *The CRC Card Book*, Reading, MA: Addison-Wesley Longman, 1997.

Booch, G., I. Jacobson y J. Rumbaugh, *The Unified Modeling Language User Guide*, 2a. ed., Boston: Addison-Wesley Publishing Co., 1999.

Cockburn, A., *Writing Effective Use Cases*. Boston: Addison-Wesley Publishing Co., 2001.

Fowler, M. y K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 2a. ed., Boston: Addison-Wesley Publishing Co., 2000.

Unified Modeling Language Notation Guide ad/97-01-09. Santa Clara, CA: Rational Software Corporation, 1997.