PARTE I
Para iniciar

Hora

HORA 1

Introducción al UML

El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

En esta hora se tratarán los siguientes temas:

- Por qué es necesario el UML
- La concepción del UML
- Diagramas del UML
- Para qué tantos diagramas

TÉRMINO NUEVO En el contexto de este libro considere a un *sistema* como una combinación de software y hardware que da una solución a un problema de negocios. El *desarrollo de sistemas* es la creación de un programa para un *cliente*, este último es quien tiene el problema que debe ser resuelto. Un *analista* es el que documenta el problema del cliente y lo comunica a los *desarrolladores*, que son los programadores que generarán el programa que resolverá el problema y lo distribuirán en equipos de computación.

La comunicación de la idea es de suma importancia. Antes del advenimiento del UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o varios programadores y esperar que el producto final cumpliese con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, hay muchas posibilidades de cometer errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente, es decir, probablemente produjo un documento que el cliente no pudo comprender. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, pudieron generar un programa difícil de utilizar y no generar una solución al problema original del cliente.

¿Alguien se preguntará por qué muchos de los sistemas en uso son ineficientes, engorrosos y difíciles de utilizar?

Por qué es necesario el UML

En los principios de la computación, los programadores no realizaban análisis muy profundos sobre el problema por resolver. Si acaso, garabateaban algo en una servilleta. Con frecuencia comenzaban a escribir el programa desde el principio, y el código necesario se escribía conforme se requería. Aunque anteriormente esto agregaba un aura de aventura y atrevimiento al proceso, en la actualidad es inapropiado en los negocios de alto riesgo.

Hoy en día, es necesario contar con un plan bien analizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades (o si cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución en general).

Conforme aumenta la complejidad del mundo, los sistemas informáticos también deberán crecer en complejidad. En ellos se encuentran diversas piezas de hardware y software que se comunican a grandes distancias mediante una red, misma que está vinculada a bases de datos que, a su vez, contienen enormes cantidades de información. Si desea crear sistemas que lo involucren con este nuevo milenio ¿cómo manejará tanta complejidad?

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

Un arquitecto no podría crear una compleja estructura como lo es un edificio de oficinas sin crear primero un anteproyecto detallado; asimismo usted tampoco podría generar un complejo sistema en un edificio de oficinas sin crear un plan de diseño detallado. La idea

es que así como un arquitecto le muestra un anteproyecto a la persona que lo contrató, usted deberá mostrarle su plan de diseño al cliente. Tal plan de diseño debe ser el resultado de un cuidadoso análisis de las necesidades del cliente.

Otra característica del desarrollo de sistemas contemporáneo es reducir el periodo de desarrollo. Cuando los plazos se encuentran muy cerca uno del otro es absolutamente necesario contar con un diseño sólido.

Hay otro aspecto de la vida moderna que demanda un diseño sólido: las adquisiciones corporativas. Cuando una empresa adquiere a otra, la nueva organización debe tener la posibilidad de modificar aspectos importantes de un proyecto de desarrollo que esté en progreso (la herramienta de desarrollo, el lenguaje de codificación, y otras cosas). Un anteproyecto bien diseñado facilitará la conversión. Si el diseño es sólido, un cambio en la implementación procederá sin problemas.

La necesidad de diseños sólidos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica). El UML es esa misma notación.

La concepción del UML

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados recientemente "Los tres amigos", trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre sí y decidieron desarrollar su trabajo en conjunto.

Las horas 2, "Orientación a objetos", y 4, "Uso de relaciones", tratan de la orientación a objetos. Los conceptos de orientación a objetos tienen un papel fundamental en el desarrollo de este libro.

En 1994 Rumbaugh ingresó a Rational Software Corporation, donde ya trabajaba Booch. Jacobson ingresó a Rational un año después; el resto, como dicen, es historia.

Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de administración de objetos) como respuesta a su propuesta para un lenguaje de modelado estándar.

El consorcio aumentó y generó la versión 1.1, misma que se puso nuevamente a consideración del OMG. El grupo adoptó esta versión a finales de 1997. El OMG se encargó de la conservación del UML y produjo otras dos revisiones en 1998. El UML ha llegado a ser el estándar de facto en la industria del software, y su evolución continúa.

Diagramas del UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. En lugar de indicarle a usted cuáles son los elementos y las reglas, veamos directamente los diagramas ya que los utilizará para hacer el análisis del sistema.

Este enfoque es similar a aprender un idioma extranjero mediante el uso del mismo, en lugar de aprender sus reglas gramaticales y la conjugación de sus verbos. Después de un tiempo de hablar otro idioma se le facilitará la conjugación de verbos y la comprensión de las reglas gramaticales.

TÉRMINO NUEVO La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como *modelo*. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación se describirán brevemente los diagramas más comunes del UML y los conceptos que representan. Posteriormente, en la parte I verá cada uno de los diagramas con mayor detenimiento. Recuerde que es posible generar híbridos de estos diagramas y que el UML otorga formas de organizarlos y extenderlos.

Diagrama de clases

Piense en las cosas que le rodean (una idea demasiado amplia, pero ¡inténtelo de cualquier forma!). Es probable que muchas de esas cosas tengan atributos (propiedades) y que realicen determinadas acciones. Podríamos imaginar cada una de esas acciones como un conjunto de tareas.

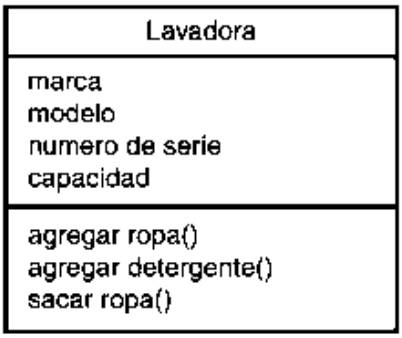
TÉRMINO NUEVO También se encontrará con que las cosas naturalmente se albergan en categorías (automóviles, mobiliario, lavadoras...). A tales categorías las llamaremos clases. Una *clase* es una categoría o grupo de cosas que tienen atributos y acciones similares. He aquí un ejemplo: cualquier cosa dentro de la clase Lavadoras tiene atributos como son la marca, el modelo, el número de serie y la capacidad. Entre las acciones

de las cosas de esta clase se encuentran: "agregar ropa", "agregar detergente", "activarse" y "sacar ropa".

La figura 1.1 le muestra un ejemplo de la notación del UML que captura los atributos y acciones de una lavadora. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

FIGURA 1.1

El símbolo UML de una clase.



¿Qué objetivo tiene pensar en las clases, así como sus atributos y acciones? Para interactuar con nuestro complejo mundo, la mayoría del software moderno simula algún aspecto del mundo. Décadas de experiencia sugieren que es más sencillo desarrollar aplicaciones que simulen algún aspecto del mundo cuando el software representa clases de cosas reales. Los diagramas de clases facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar.

A su vez, los diagramas de clases colaboran en lo referente al análisis. Permiten al analista hablarle a los clientes en su propia terminología, lo cual hace posible que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos.

Diagrama de objetos

TÉRMINO NUEVO

Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones). Su lavadora, por ejemplo, podría tener la marca Laundatorium, el modelo Washmeister, el número de serie GL57774 y una capacidad de 7 Kg.

La figura 1.2 le muestra la forma en que el UML representa a un objeto. Vea que el símbolo es un rectángulo, como en una clase, pero el nombre está subrayado. El nombre de la instancia específica se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

FIGURA 1.2

El símbolo UML del objeto.

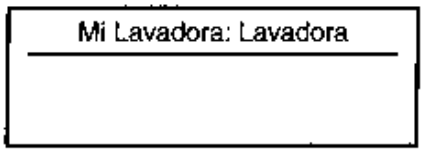


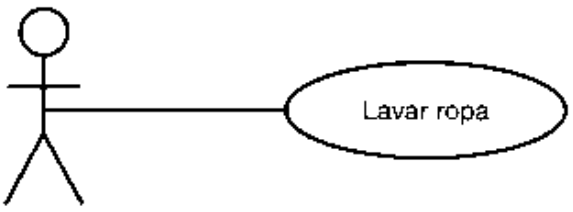
Diagrama de casos de uso

TÉRMINO NUEVO Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, ésta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación).

Posteriormente trataremos este tema con mayor detalle; por ahora, le mostraré un ejemplo sencillo. Usted utiliza una lavadora, obviamente, para lavar su ropa. La figura 1.3 le muestra cómo representaría esto en un diagrama de casos de uso UML.

FIGURA 1.3

Diagrama de casos de uso UML.



TÉRMINO NUEVO A la figura correspondiente al Usuario de la lavadora se le conoce como actor. La elipse representa el caso de uso. Vea que el actor (la entidad que inicia el caso de uso) puede ser una persona u otro sistema.

Diagrama de estados

En cualquier momento, un objeto se encuentra en un estado en particular. Una persona puede ser recién nacida, infante, adolescente, joven o adulta. Un elevador se moverá hacia arriba, estará en estado de reposo o se moverá hacia abajo. Una lavadora podrá estar en la fase de remojo, lavado, enjuague, centrifugado o apagada.

El diagrama de estados UML, que aparece en la figura 1.4, captura esta pequeña realidad. La figura muestra las transiciones de la lavadora de un estado al otro.

El símbolo que está en la parte superior de la figura representa el estado inicial y el de la parte inferior el estado final.

FIGURA 1.4

Diagrama de estados UML.

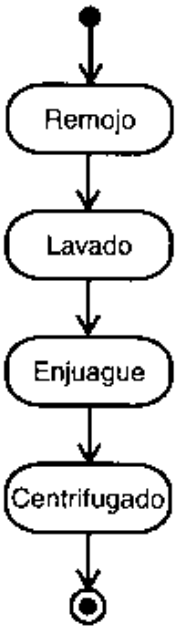


Diagrama de secuencias

Los diagramas de clases y los de objeto representan información estática. No obstante, en un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.

Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran: una manguera de agua (para obtener agua fresca), un tambor (donde se coloca la ropa) y un sistema de drenaje. Por supuesto, estos también son objetos (como verá, un objeto puede estar conformado por otros objetos).

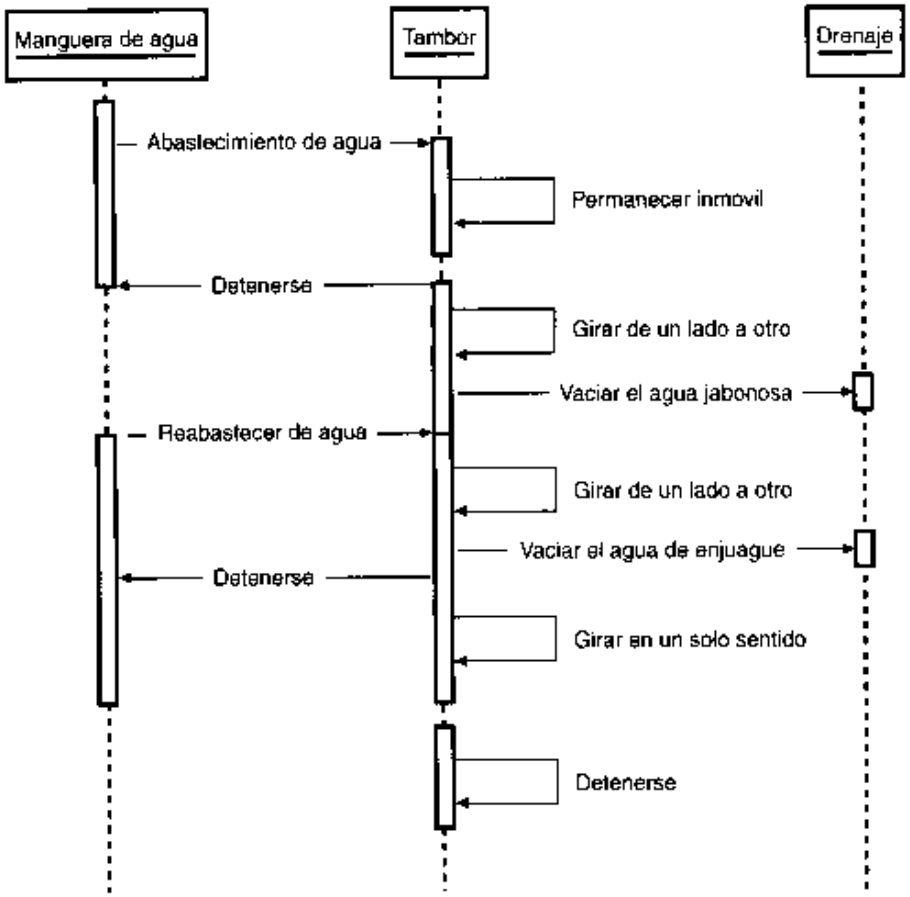
¿Qué sucederá cuando invoque al caso de uso Lavar ropa? Si damos por hecho que completó las operaciones "agregar ropa", "agregar detergente" y "activar", la secuencia sería más o menos así:

1. El agua empezará a llenar el tambor mediante una manguera.
2. El tambor permanecerá inactivo durante cinco minutos.
3. La manguera dejará de abastecer agua.
4. El tambor girará de un lado a otro durante quince minutos.
5. El agua jabonosa saldrá por el drenaje.
6. Comenzará nuevamente el abastecimiento de agua.
7. El tambor continuará girando.

8. El abastecimiento de agua se detendrá.
9. El agua del enjuague saldrá por el drenaje.
10. El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
11. El tambor dejará de girar y el proceso de lavado habrá finalizado.

La figura 1.5 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el abastecimiento de agua, el tambor y el drenaje (representados como rectángulos en la parte superior del diagrama). En este diagrama el tiempo se da de arriba hacia abajo.

FIGURA 1.5
Diagrama de secuencias UML.



Por cierto, volviendo a las ideas acerca de los estados, podríamos caracterizar los pasos 1 y 2 como el estado de remojo, 3 y 4 como el estado de lavado, 5 a 7 como el estado de enjuague y del 8 al 10 como el estado de centrifugado.

Diagrama de actividades

Las actividades que ocurren dentro de un caso de uso o dentro del comportamiento de un objeto se dan, normalmente, en secuencia, como en los once pasos de la sección anterior. La figura 1.6 muestra la forma en que el diagrama de actividades UML representa los pasos del 4 al 6 de tal secuencia.

FIGURA 1.6

Diagrama de actividades UML.

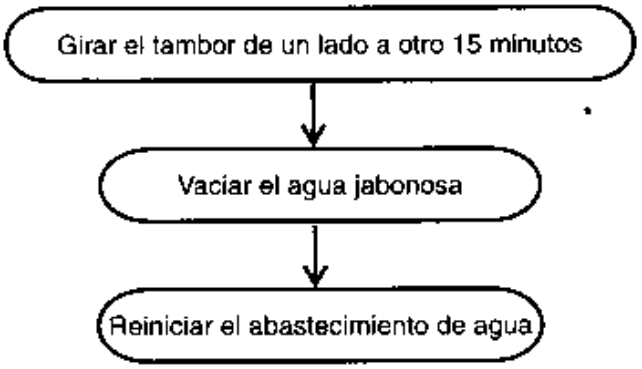


Diagrama de colaboraciones

Los elementos de un sistema trabajan en conjunto para cumplir con los objetivos del sistema, y un lenguaje de modelado deberá contar con una forma de representar esto. El diagrama de colaboraciones UML, diseñado con este fin, se muestra en la figura 1.7. Este ejemplo agrega un cronómetro interno al conjunto de clases que constituyen a una lavadora. Luego de cierto tiempo, el cronómetro detendrá el flujo de agua y el tambor comenzará a girar de un lado a otro.

FIGURA 1.7

Diagrama de colaboraciones UML.

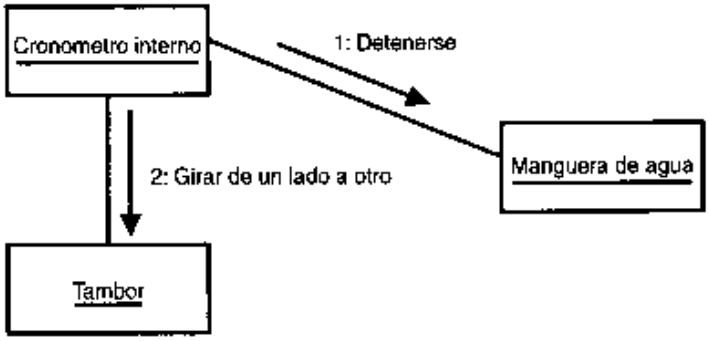


Diagrama de componentes

Este diagrama y el siguiente dejarán el mundo de las lavadoras, dado que están íntimamente ligados con los sistemas informáticos.

El moderno desarrollo de software se realiza mediante componentes, lo que es particularmente importante en los procesos de desarrollo en equipo. Sin extenderme mucho en este punto le mostraré, en la figura 1.8, la manera en que el UML representa un componente de software.

FIGURA 1.8

Diagrama de componentes UML.

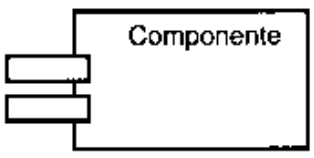
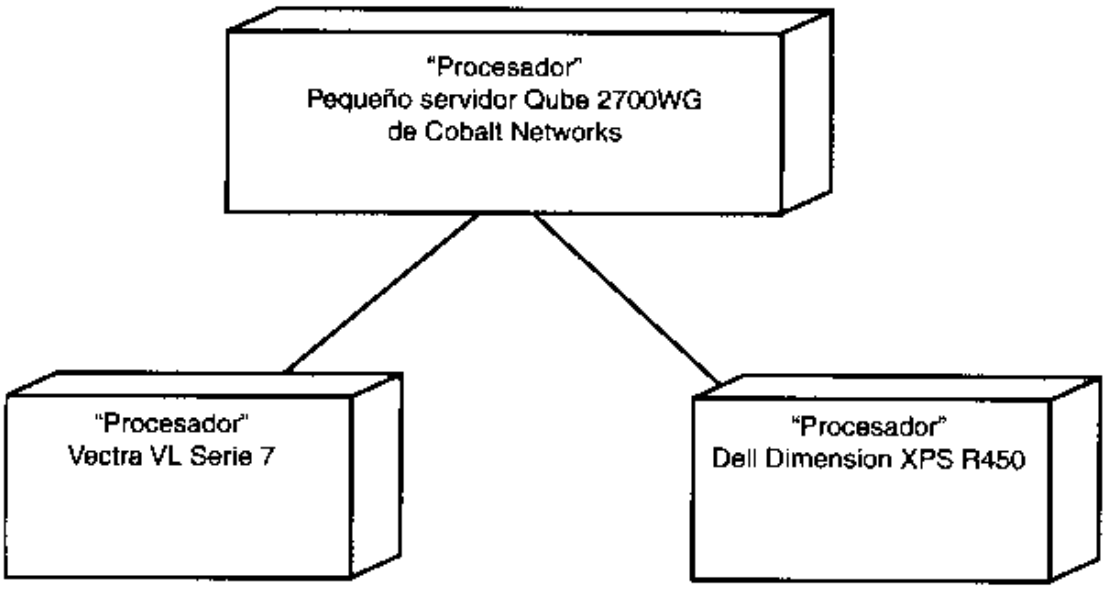


Diagrama de distribución

El diagrama de distribución UML muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora está representada por un cubo y las interacciones entre las computadoras están representadas por líneas que conectan a los cubos. La figura 1.9 presenta un ejemplo.

FIGURA 1.9

Diagrama de distribución UML.



Otras características

Anteriormente, mencioné que el UML proporciona características que le permiten organizar y extender los diagramas.

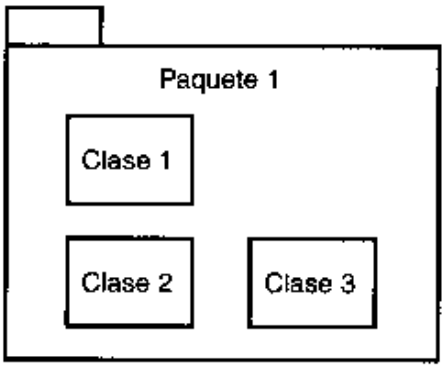
Paquetes

TÉRMINO NUEVO

En algunas ocasiones se encontrará con la necesidad de organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, los agrupará en un *paquete*, que se representará por una carpeta tabular, como se muestra en la figura 1.10.

FIGURA 1.10

El paquete UML le permite agrupar los elementos de un diagrama.



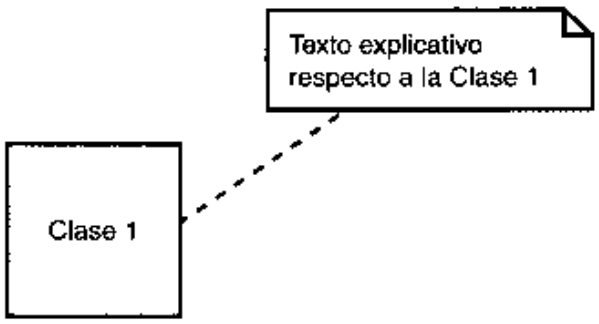
Notas

TÉRMINO NUEVO

Es frecuente que alguna parte del diagrama no presente una clara explicación del porqué está allí o la manera en que trabaja. Cuando éste sea el caso, la *nota* UML será útil. Imagine a una nota como el equivalente gráfico de un papel adhesivo. La nota es un rectángulo con una esquina doblada, y dentro del rectángulo se coloca la explicación. Usted adjunta la nota al elemento del diagrama conectándolos mediante una línea discontinua.

FIGURA 1.11

En cualquier diagrama, podrá agregar comentarios aclaratorios mediante una nota.



Estereotipos

TERMINO NUEVO

El UML otorga varios elementos de utilidad, pero no es un conjunto minucioso de ellos. De vez en cuando diseñará un sistema que requiera algunos elementos hechos a la medida. Los *estereotipos* o clisés le permiten tomar elementos propios del UML y convertirlos en otros. Es como comprar un traje del mostrador y modificarlo para que se ajuste a sus medidas (contrario a confeccionarse uno completamente nuevo). Imagine a un estereotipo como este tipo de alteración. Lo representará como un nombre entre dos pares de paréntesis angulares y después los aplicará correctamente.

TÉRMINO NUEVO

El concepto de una interfaz provee un buen ejemplo. Una interfaz es una clase que realiza operaciones y que no tiene atributos, es un conjunto de acciones que tal vez quiera utilizar una y otra vez en su modelo. En lugar de inventar un nuevo elemento para representar una interfaz, podrá utilizar el símbolo de una clase con «Interfaz» situada justo sobre el nombre de la clase, como se muestra en la figura 1.12.

FIGURA 1.12

Un estereotipo le permite crear nuevos elementos a partir de otros existentes.

Para qué tantos diagramas

Como puede ver, los diagramas del UML le permiten examinar un sistema desde distintos puntos de vista. Es importante recalcar que en un modelo UML no es necesario que aparezcan todos los diagramas. De hecho, la mayoría de los modelos UML contienen un subconjunto de los diagramas que he indicado.

TERMINO NUEVO

¿Por qué es necesario contar con diferentes perspectivas de un sistema? Por lo general, un sistema cuenta con diversas personas implicadas las cuales tienen enfoques particulares en diversos aspectos del sistema. Volvamos al ejemplo de la lavadora. Si diseñara el motor de una lavadora, tendría una perspectiva del sistema; si escribiera las instrucciones de operación, tendría otra perspectiva. Si diseñara la forma general de la lavadora vería al sistema desde una perspectiva totalmente distinta a si tan sólo tratara de lavar su ropa.

El escrupuloso diseño de un sistema involucra todas las posibles perspectivas, y el diagrama UML le da una forma de incorporar una perspectiva en particular. El objetivo es satisfacer a cada persona implicada.

Resumen

El desarrollo de sistemas es una actividad humana. Sin un sistema de notación fácil de comprender, el proceso de desarrollo tiene una gran cantidad de errores.

El UML es un sistema de notación que se ha convertido en estándar en el mundo del desarrollo de sistemas. Es el resultado del trabajo hecho por Grady Booch, James Rumbaugh e Ivar Jacobson. El UML está constituido por un conjunto de diagramas, y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema.

Un modelo UML indica *qué* es lo que supuestamente hará el sistema, mas no *cómo* lo hará.

Preguntas y respuestas

P **He visto que se refiere al Lenguaje Unificado de Modelado como "UML" y como "el UML". ¿Cuál es el correcto?**

R Los creadores del lenguaje prefieren el uso de "el UML".

P **Ha indicado que el UML es adecuado para los analistas. No obstante, el diagrama de distribución no parece ser algo muy útil en la fase de análisis en el desarrollo de un sistema. ¿No sería más apropiado para una fase posterior?**

R En realidad nunca será demasiado pronto para empezar a pensar en la distribución (u otras cuestiones que, tradicionalmente, se dejan para fases posteriores del desarrollo). Aunque es cierto que el analista se interesa por hablar con los clientes y usuarios, en las fases tempranas del proceso el analista debería pensar en los equipos y componentes que constituirían el hardware del sistema. En algunas ocasiones, el cliente dicta esto; en otras, el cliente desea una recomendación del equipo de desarrollo. Ciertamente, un arquitecto de sistemas encontrará útil al diagrama de distribución.

P Ha mencionado que es posible hacer diagramas híbridos. ¿UML, perdón, *el* UML, impone limitaciones respecto a los elementos que podrá combinar en un diagrama?

R No. El UML no establece límites, no obstante, con frecuencia se da el caso de que un diagrama contenga un tipo de elemento. Podrá colocar símbolos de clases en un diagrama de distribución, pero ello no será muy útil.

Taller

Ya se ha iniciado en el UML. Ahora deberá reafirmar su conocimiento de esta gran herramienta al responder algunas preguntas y realizar los ejercicios. Las respuestas aparecerán en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Porqué es necesario contar con diversos diagramas en el modelo de un sistema?
2. ¿Cuáles diagramas le dan una perspectiva estática de un sistema?
3. ¿Cuáles diagramas le dan una perspectiva dinámica de un sistema (esto es, muestran el cambio progresivo)?

Ejercicios

1. Suponga que creará un sistema informático que jugará ajedrez con un usuario. ¿Cuáles diagramas UML serían útiles para diseñar el sistema? ¿Por qué?
2. Para el sistema del ejercicio que ha completado, liste las preguntas que formularía a un usuario potencial y por qué las haría.

HORA 2

Orientación a objetos

Es fundamental que comprenda todo lo relacionado a la orientación a objetos para el proceso que realizará; específicamente, es importante que conozca algunos conceptos sobre la orientación a objetos.

En esta hora se tratarán los siguientes temas:

- Abstracción
- Herencia
- Polimorfismo
- Encapsulamiento o encapsulación
- Envío de mensajes
- Asociaciones
- Agregación

La orientación a objetos ha tomado por asalto y en forma legítima al mundo del software. Como medio para la generación de programas, tiene varias ventajas. Fomenta una metodología basada en componentes para el desarrollo

de software, de manera que primero se genera un sistema mediante un conjunto de objetos, luego podrá ampliar el sistema agregándole funcionalidad a los componentes que ya había generado o agregándole nuevos componentes, y finalmente podrá volver a utilizar los objetos que generó para el sistema cuando cree uno nuevo, con lo cual reducirá sustancialmente el tiempo de desarrollo de un sistema.

La orientación a objetos es tan importante para el diseño de software que el OMG (Grupo de administración de objetos), una corporación no lucrativa que establece las normas para el desarrollo orientado a objetos, predice que los ingresos obtenidos por el software orientado a objetos serán de 3 millardos de dólares en los siguientes tres a cinco años. El UML influye en esto al permitirle generar modelos de objetos fáciles de usar y comprender para que los desarrolladores puedan convertirlos en software.

La orientación a objetos es un paradigma (un paradigma que depende de ciertos principios fundamentales). En esta hora comprenderá dichos principios y verá qué es lo que hace funcionar a los objetos y cómo utilizarlos en el análisis y diseño. En la siguiente hora, empezará a aplicar el UML a tales principios.

Objetos, objetos por doquier

Los objetos concretos y virtuales, están a nuestro alrededor, ellos conforman nuestro mundo. Como indiqué en la hora anterior, el software actual simula al mundo (o un segmento de él), y los programas, por lo general, imitan a los objetos del mundo. Si comprende algunas cuestiones básicas de los objetos, entenderá cómo se deben mostrar éstos en las representaciones de software.

TERMINO NUEVO

Antes que nada, un objeto es la instancia de una clase (o categoría). Usted y yo, por ejemplo, somos instancias de la clase Persona. Un objeto cuenta con una *estructura*, es decir atributos (propiedades) y acciones. Las acciones son todas las *actividades* que el objeto es capaz de realizar. Los atributos y acciones, en conjunto, se conocen como *características* o *rasgos*.

Como objetos de la clase Persona, usted y yo contamos con los siguientes atributos: altura, peso y edad (puede imaginar muchos más). También realizamos las siguientes tareas: comer, dormir, leer, escribir, hablar, trabajar, etcétera. Si tuviéramos que crear un sistema que manejara información acerca de las personas (como una nómina o un sistema para el departamento de recursos humanos), sería muy probable que incorporáramos algunos de sus atributos y acciones en nuestro software.

En el mundo de la orientación a objetos, una clase tiene otro propósito además de la categorización. En realidad es una plantilla para fabricar objetos. Imagínelo como un molde de galletas que produce muchas galletas (algunos alegarían que esto es lo mismo que la categorización, pero evitemos dicho debate).

Regresemos al ejemplo de la lavadora. Si en la clase Lavadora se indica la marca, el modelo, el número de serie y la capacidad (junto con las acciones de agregar ropa, agregar detergente y sacar ropa), tendrá un mecanismo para fabricar nuevas instancias a partir de su clase; es decir, podrá crear nuevos objetos (vea la figura 2.1).

En la hora 3, "Uso de la orientación a objetos", verá que los nombres de las clases, como *lavadora*, se escribirán como Lavadora, y si constara de dos palabras se escribiría como LavadoraIndustrial, y las características como *número de serie* se escribirán como numeroSerie.

Esto es particularmente importante en el desarrollo de software orientado a objetos. Aunque este libro no se enfoca a la programación, le ayudará a comprender la orientación a objetos si sabe que las clases en los programas orientados a objetos pueden crear nuevas instancias.

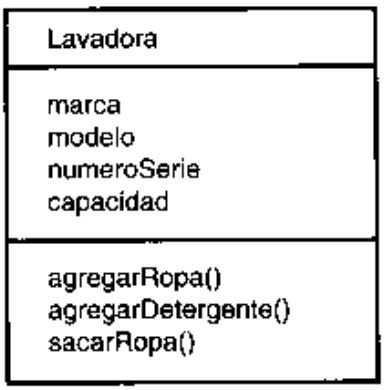
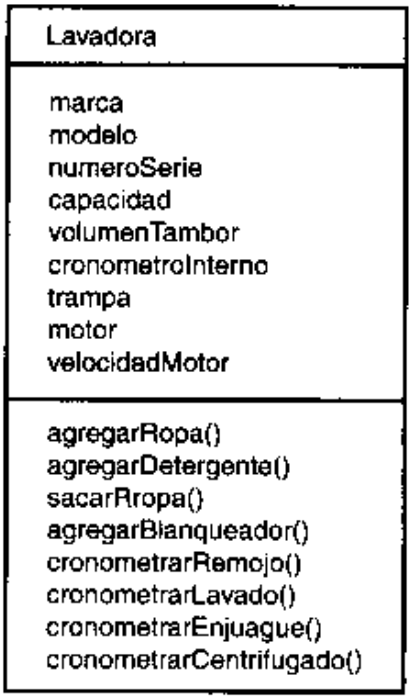


FIGURA 2.1
La clase Lavadora (modelo original) es una plantilla para generar nuevas instancias de Lavadoras.

Es importante que recuerde que el propósito de la orientación a objetos es desarrollar software que refleje particularmente (es decir, que modele) un esquema del mundo. Entre más atributos y acciones tome en cuenta, mayor será la similitud de su modelo con la realidad. En el ejemplo de la lavadora, tendrá un modelo más exacto si incluye los siguientes atributos: volumen del tambor, cronómetro interno, trampa, motor y velocidad del motor. Podría hacerlo más preciso si incluye las acciones de agregar blanqueador, cronometrar el remojo, cronometrar el lavado, cronometrar el enjuague y cronometrar el centrifugado (vea la figura 2.2).

FIGURA 2.2

La adición de atributos y acciones al modelo lo acerca a la realidad.



Algunos conceptos

La orientación a objetos se refiere a algo más que tan sólo atributos y acciones; también considera otros aspectos. Dichos aspectos se conocen como *abstracción*, *herencia*, *polimorfismo* y *encapsulamiento* o *encapsulación*. Otros aspectos importantes de la orientación a objetos son: el *envío de mensajes*, las *asociaciones* y la *agregación*. Examinemos cada uno de estos conceptos.

Abstracción

TÉRMINO NUEVO

La abstracción se refiere a quitar las propiedades y acciones de un objeto para dejar sólo aquellas que sean necesarias. ¿Qué significa esto último?

Diferentes tipos de problemas requieren distintas cantidades de información, aun si estos problemas pertenecen a un área en común. En la segunda fase de la creación de la clase Lavadora, se podrían agregar más atributos y acciones que en la primera fase. ¿Vale la pena?

Valdría la pena si usted pertenece al equipo de desarrollo que generará finalmente la aplicación que simule con exactitud lo que hace una lavadora. Un programa de este tipo (que podría ser muy útil para los ingenieros de diseño que actualmente estén trabajando en el diseño de una lavadora) deberá ser tan completo que permita obtener predicciones exactas respecto a lo que ocurriría cuando se fabrique la lavadora, funcione a toda su capacidad y lave la ropa. De hecho, para este caso podrá quitar el atributo del número de serie, dado que posiblemente no será de mucha ayuda.

Por otra parte, si va a generar un software que haga un seguimiento de las transacciones en una lavandería que cuente con diversas lavadoras, posiblemente no valdrá la pena. En este programa no necesitará todos los atributos detallados y operaciones del párrafo anterior, no obstante, quizá necesite incluir el número de serie de cada objeto Lavadora.

En cualquier caso, con lo que se quedará luego de tomar su decisión respecto a lo que incluirá o desechará, será una abstracción de una lavadora.

Herencia

TERMINO NUEVO

Como ya se mencionó anteriormente, una clase es una categoría de objetos (y en el mundo del software, una plantilla sirve para crear otros objetos). Un objeto es una instancia de una clase. Esta idea tiene una consecuencia importante: como instancia de una clase, un objeto tiene todas las características de la clase de la que proviene. A esto se le conoce como *herencia*. No importa qué atributos y acciones decida usar de la clase Lavadora, cada objeto de la clase heredará dichos atributos y operaciones.

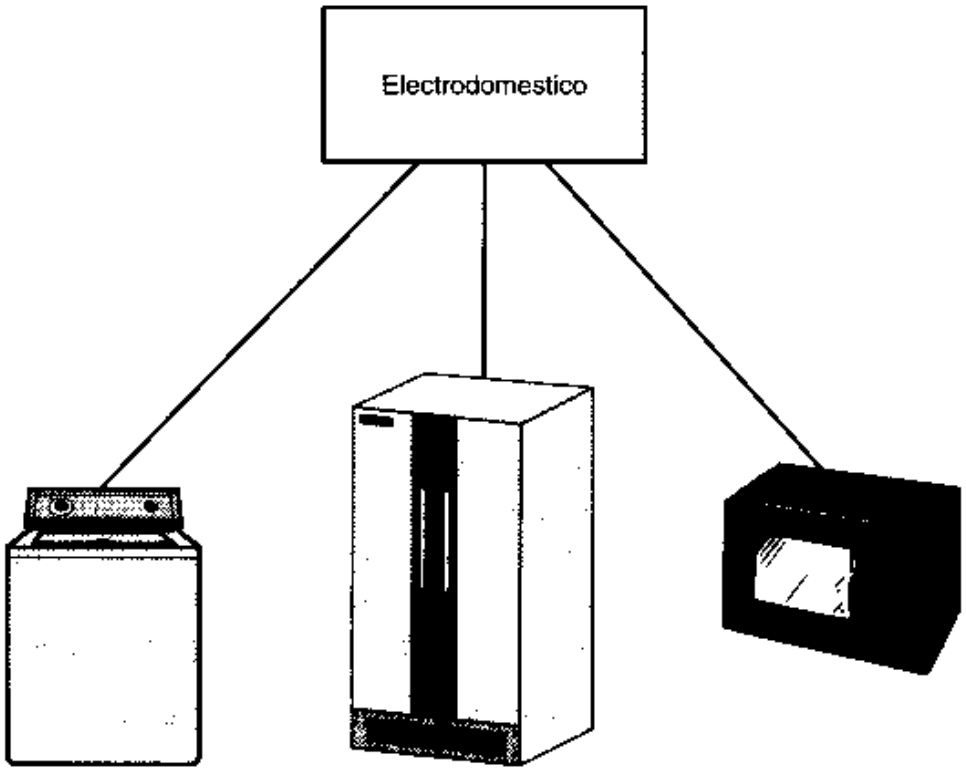
Un objeto no sólo hereda de una clase, sino que una clase también puede heredar de otra. Las lavadoras, refrigeradores, hornos de microondas, tostadores, lavaplatos, radios, licuadoras y planchas son clases y forman parte de una clase más genérica llamada: Electrodomesticos. Un electrodoméstico cuenta con los atributos de interruptor y cable eléctrico, y las operaciones de encendido y apagado. Cada una de las clases Electrodomestico heredará los mismos atributos; por ello, si sabe que algo es un electrodoméstico, de inmediato sabrá que cuenta con los atributos y acciones de la clase Electrodomestico.

TERMINO NUEVO

Otra forma de explicarlo es que la lavadora, refrigerador, horno de microondas y cosas por el estilo son *subclases* de la clase Electrodomestico. Podemos decir que la clase Electrodomestico es una *superclase* de todas las demás. La figura 2.3 le muestra la relación de superclase y subclase.

FIGURA 2.3

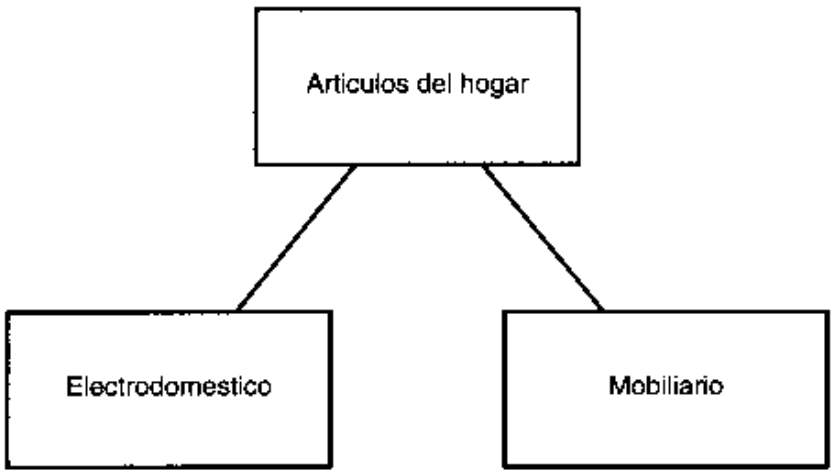
Los electrodomésticos heredan los atributos y acciones de la clase Electrodomestico. Cada electrodoméstico es una subclase de la clase Electrodomestico. La clase Electrodomestico es una superclase de cada subclase.



La herencia no tiene por qué terminar aquí. Por ejemplo, Electrodomestico es una subclase de Articulos del hogar, como le muestra la figura 2.4. Otra de las subclases de Articulos del hogar podría ser Mobiliario, que tendrá sus propias subclases.

FIGURA 2.4

Las superclases también pueden ser subclases, y heredar de otras superclases.



Polimorfismo

TERMINO NUEVO

En ocasiones una operación tiene el mismo nombre en diferentes clases. Por ejemplo, podrá abrir una puerta, una ventana, un periódico, un regalo o una cuenta de banco, en cada uno de estos casos, realizará una operación diferente. En la orientación a objetos, cada clase "sabe" cómo realizar tal operación. Esto es el *polimorfismo* (vea la figura 2.5).

FIGURA 2.5

En el polimorfismo, una operación puede tener el mismo nombre en diversas clases, y funcionar distinto en cada una.

En primera instancia, parecería que este concepto es más importante para los desarrolladores de software que para los modeladores, ya que los desarrolladores tienen que crear el software que implemente tales métodos en los programas de computación, y deben estar conscientes de diferencias importantes entre las operaciones que pudieran tener el mismo nombre. Y podrán generar clases de software que "sepan" lo que se supone que harán.

No obstante, el polimorfismo también es importante para los modeladores ya que les permite hablar con el cliente (quien está familiarizado con la sección del mundo que será modelada) en las propias palabras y terminología del cliente. En ocasiones, las palabras y terminología del cliente nos conducen a palabras de acción (como "abrir") que pueden

tener más de un significado. El polimorfismo permite al modelador mantener tal terminología sin tener que crear palabras artificiales para sustentar una unicidad innecesaria de los términos.

Encapsulamiento

En cierto comercial televisivo, dos personas discuten acerca de todo el dinero que ahorrarían si marcaran un prefijo telefónico en particular antes de hacer una llamada de larga distancia.

Uno de ellos pregunta con incredulidad: "¿Cómo funciona?"

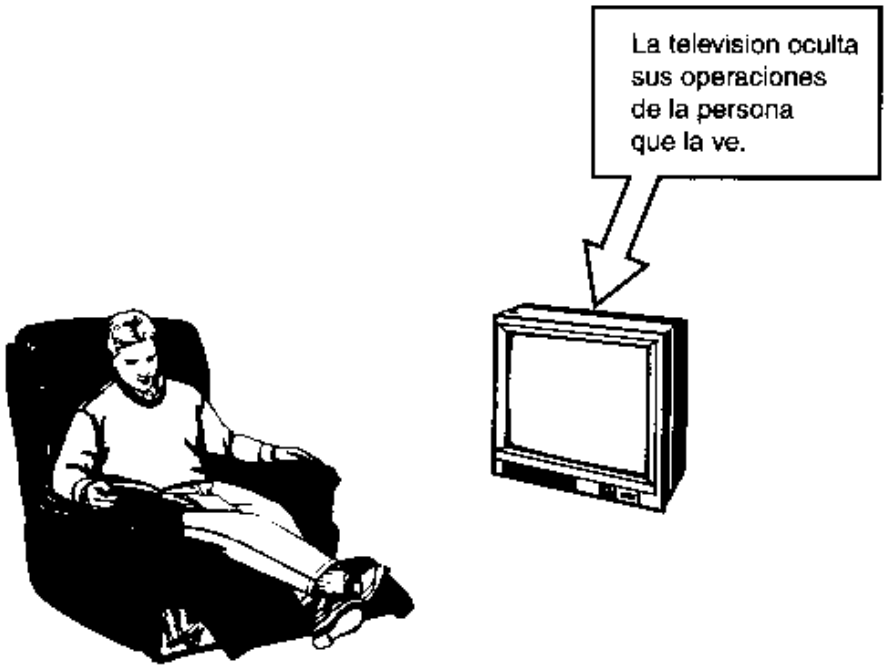
Y el otro responde: "¿Cómo hacen que las rosetas de maíz estallen? ¿A quién le importa?"

TÉRMINO NUEVO

La esencia del *encapsulamiento* (o encapsulación) es que cuando un objeto trae consigo su funcionalidad, esta última se oculta (vea la figura 2.6). Por lo general, la mayoría de la gente que ve la televisión no sabe o no se preocupa de la complejidad electrónica que hay detrás de la pantalla ni de todas las operaciones que tienen que ocurrir para mostrar una imagen en la pantalla. La televisión hace lo que tiene que hacer sin mostrarnos el proceso necesario para ello y, por suerte, la mayoría de los electrodomésticos funcionan así.

FIGURA 2.6

Los objetos encapsulan lo que hacen; es decir, ocultan la funcionalidad interna de sus operaciones, de otros objetos y del mundo exterior.



¿Cuál es la importancia de esto? En el mundo del software, el encapsulamiento permite reducir el potencial de errores que pudieran ocurrir. En un sistema que consta de objetos, éstos dependen unos de otros en diversas formas. Si uno de ellos falla y los especialistas de software tienen que modificarlo de alguna forma, el ocultar sus operaciones de otros objetos significará que tal vez no será necesario modificar los demás objetos.

En el mundo real, también verá la importancia del encapsulamiento en los objetos con los que trabaje. Por ejemplo, el monitor de su computadora, en cierto sentido, oculta sus operaciones de la CPU, es decir, si algo falla en su monitor, lo reparará o lo reemplazará; pero es muy probable que no tenga que reparar o reemplazar la CPU al mismo tiempo que el monitor.

TERMINO NUEVO

Ya que estamos en el tema, existe un concepto relacionado. Un objeto oculta lo que hace a otros objetos y al mundo exterior, por lo cual al encapsulamiento también se le conoce como *ocultamiento de la información*. Pero un objeto tiene que presentar un "rostro" al mundo exterior para poder iniciar sus operaciones. Por ejemplo, la televisión tiene diversos botones y perillas en sí misma o en el control remoto. Una lavadora tiene diversas perillas que le permiten establecer los niveles de temperatura y agua. Los botones y perillas de la televisión y de la lavadora se conocen como *interfaces*.

Envío de mensajes

Mencioné que en un sistema los objetos trabajan en conjunto. Esto se logra mediante el envío de mensajes entre ellos. Un objeto envía a otro un mensaje para realizar una operación, y el objeto receptor ejecutará la operación.

Una televisión y su control remoto pueden ser un ejemplo muy intuitivo del mundo que nos rodea. Cuando desea ver un programa de televisión, busca el control remoto, se sienta en su silla favorita y presiona el botón de encendido. ¿Qué ocurre? El control remoto le envía, literalmente, un mensaje al televisor para que se encienda. El televisor recibe el mensaje, lo identifica como una petición para encenderse y así lo hace. Cuando desea ver otro canal, presiona el botón correspondiente del control remoto, mismo que envía otro mensaje a la televisión (cambiar de canal). El control remoto también puede comunicar otros mensajes como ajustar el volumen, silenciar y activar los subtítulos.

Volvamos a las interfaces. Muchas de las cosas que hace mediante el control remoto, también las podrá hacer si se levanta de la silla, va a la televisión y presiona los botones correspondientes (¡alguna vez lo habrá hecho ya!). La interfaz que la televisión le presenta (el conjunto de botones y perillas) no es, obviamente, la misma que le muestra al control remoto (un receptor de rayos infrarrojos). La figura 2.7 le muestra esto.

Asociaciones

Otro acontecimiento común es que los objetos se relacionan entre sí de alguna forma. Por ejemplo, cuando enciende su televisor, en términos de orientación a objetos, usted se asocia con su televisor.

La asociación "encendido" es en una sola dirección (una vía), esto es, usted enciende la televisión, como se ve en la figura 2.8. No obstante, a menos que vea demasiada televisión, ella no le devolverá el favor. Hay otras asociaciones que son en dos direcciones, como "casamiento".

FIGURA 2.7

Ejemplo de un mensaje enviado de un objeto a otro: el objeto "control remoto" envía un mensaje al objeto "televisión" para encenderse. El objeto "televisión" recibe el mensaje mediante su interfaz, un receptor infrarrojo.

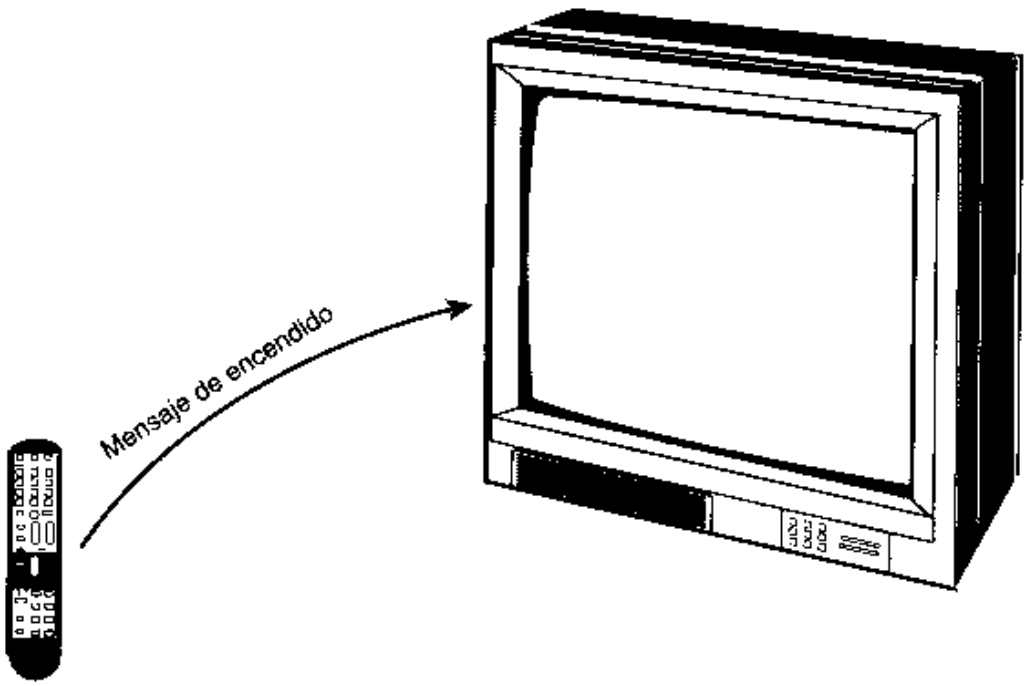
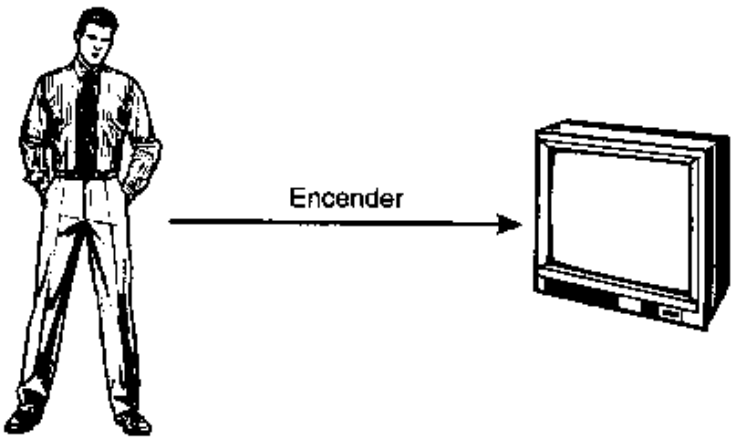


FIGURA 2.8

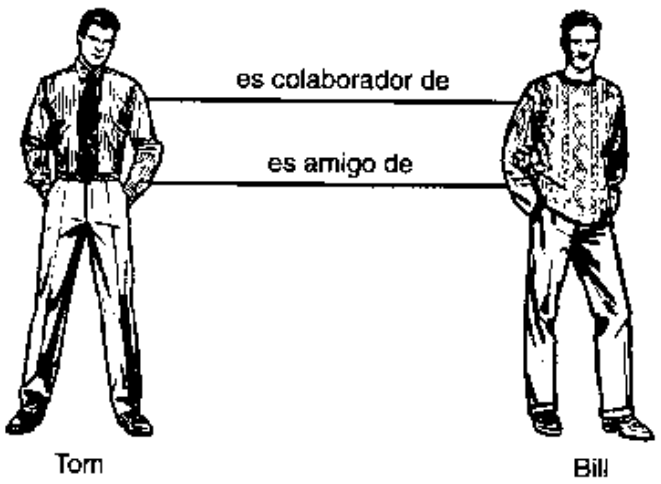
Con frecuencia los objetos se relacionan entre sí de alguna forma. Cuando usted enciende su televisión, tendrá una asociación en una sola dirección con ella.



En ocasiones, un objeto podría asociarse con otro en más de una forma. Si usted y su colaborador son amigos, ello servirá de ejemplo. Usted tendría una asociación "es amigo de", así como "es colaborador de", como se aprecia en la figura 2.9.

FIGURA 2.9

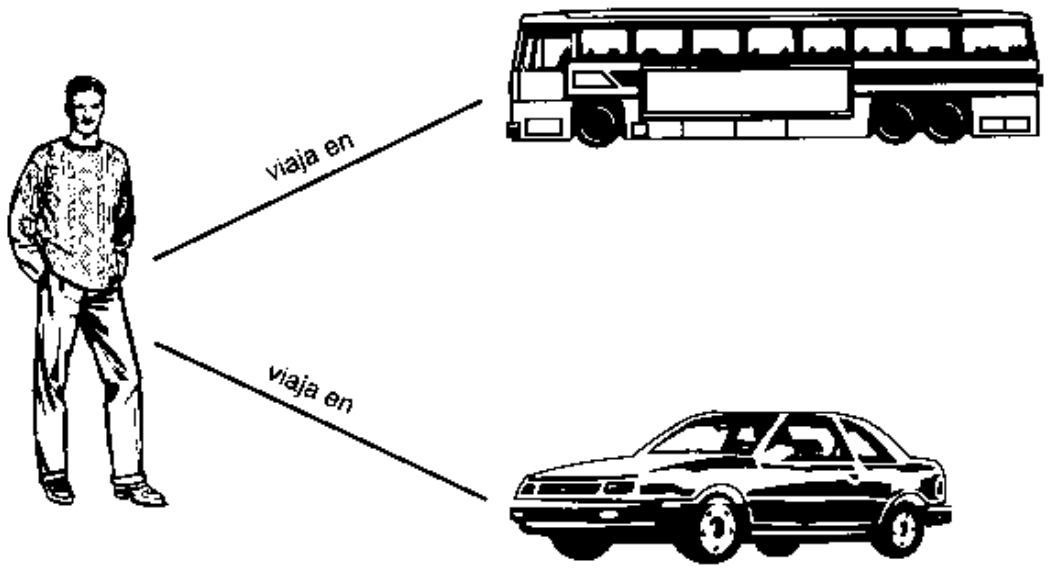
En ocasiones, los objetos pueden asociarse en más de una forma.



Una clase se puede asociar con más de una clase distinta. Una persona puede viajar en automóvil, pero también puede hacerlo en autobús (vea la figura 2.10).

FIGURA 2.10

Una clase puede asociarse con más de una clase distinta.



TÉRMINO NUEVO La *multiplicidad* (o diversificación) es un importante aspecto de las asociaciones entre objetos. Indica la cantidad de objetos de una clase que se relacionan con otro objeto en particular de la clase asociada. Por ejemplo, en un curso escolar, el curso se imparte por un solo instructor, en consecuencia, el curso y el instructor están en una asociación de uno a uno. Sin embargo, en un seminario hay diversos instructores que impartirán el curso a lo largo del semestre, por lo tanto, el curso y el instructor tienen una asociación de uno a muchos.

Podrá encontrar todo tipo de multiplicidades si echa una mirada a su alrededor. Una bicicleta rueda en dos neumáticos (multiplicidad de uno a dos), un triciclo rueda en tres, y un vehículo de 18 ruedas, en 18.

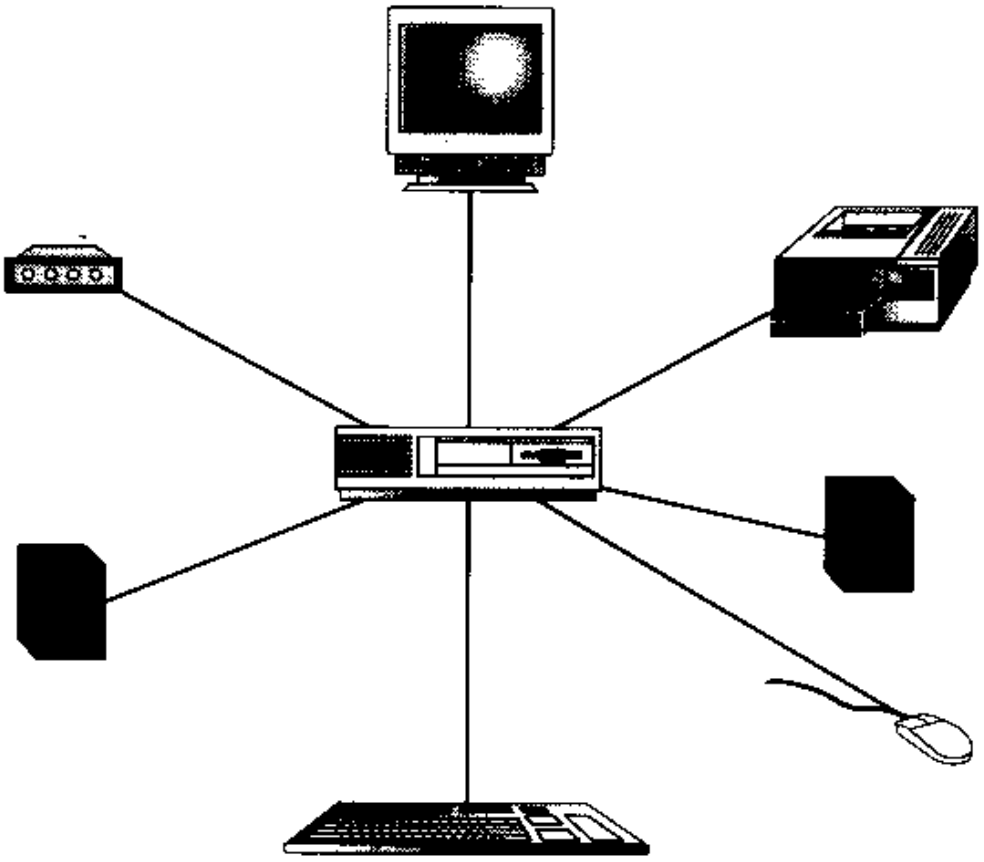
Agregación

Vea su computadora: cuenta con un gabinete, un teclado, un ratón, un monitor, una unidad de CD-ROM, uno o varios discos duros, un módem, una unidad de disquete, una impresora y, posiblemente, bocinas. Dentro del gabinete, junto con las citadas unidades de disco, tiene una CPU, una tarjeta de vídeo, una de sonido y otros elementos sin los que, sin duda, difícilmente podría vivir.

TÉRMINO NUEVO Su computadora es una *agregación* o adición, otro tipo de asociación entre objetos. Como muchas otras cosas que valdrían la pena tener, su equipo está constituido de diversos tipos de componentes (vea la figura 2.11). Tal vez conozca varios ejemplos de agregaciones.

FIGURA 2.11

Una computadora es un ejemplo de agregación: un objeto que se conforma de una combinación de diversos tipos de objetos.



TÉRMINO NUEVO

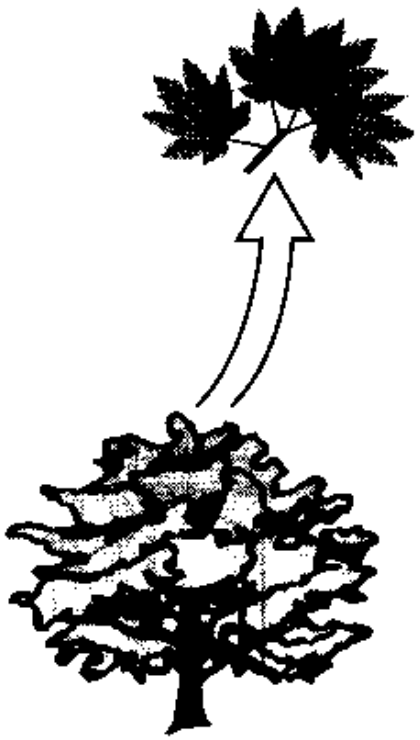
Un tipo de agregación trae consigo una estrecha relación entre un objeto agregado y sus objetos componentes. A esto se le conoce como *composición*. El punto central de la composición es que el componente se considera como tal sólo como parte del objeto compuesto. Por ejemplo: una camisa está compuesta de cuerpo, cuello, mangas, botones, ojales y puños. Suprima la camisa y el cuello será inútil.

En ocasiones, un objeto compuesto no tiene el mismo tiempo de vida que sus propios componentes. Las hojas de un árbol pueden morir antes que el árbol. Si destruye al árbol, también las hojas morirán (vea la figura 2.12).

La agregación y la composición son importantes dado que reflejan casos extremadamente comunes, y ello ayuda a que cree modelos que se asemejen considerablemente a la realidad.

FIGURA 2.12

En una composición, un componente puede morir antes que el objeto compuesto. Si destruye al objeto compuesto, destruirá también a los componentes.



La recompensa

Los objetos y sus asociaciones conforman la columna vertebral de la funcionalidad de los sistemas. Para modelarlos, necesitará comprender lo que son las asociaciones. Si está consciente de los posibles tipos de asociaciones, tendrá una amplia gama de posibilidades cuando hable con los clientes respecto a sus necesidades, obtendrá sus requerimientos y creará los modelos de los sistemas que los ayudarán a cumplir con sus retos de negocios.

TERMINO NUEVO

Lo importante es utilizar los conceptos de la orientación a objetos para ayudarle a comprender el área de conocimiento de su cliente (su *dominio*), y esclarecer sus puntos de vista al cliente en términos que él o ella puedan comprender.

Es allí donde se aplica el UML. En las siguientes tres horas, aprenderá a aplicar el UML para visualizar los conceptos que ha aprendido durante esta hora.

Resumen

La orientación a objetos es un paradigma que depende de algunos principios fundamentales. Un objeto es una instancia de una clase. Una clase es una categoría genérica de objetos que tienen los mismos atributos y acciones. Cuando crea un objeto, el área del problema en que trabaje determinará cuántos de los atributos y acciones debe tomar en cuenta.

La herencia es un aspecto importante de la orientación a objetos, un objeto hereda los atributos y operaciones de su clase. Una clase también puede heredar atributos y acciones de otra.

El polimorfismo es otro aspecto importante, ya que especifica que una acción puede tener el mismo nombre en diferentes clases y cada clase ejecutará tal operación de forma distinta.

Los objetos ocultan su funcionalidad de otros objetos y del mundo exterior. Cada objeto presenta una interfaz para que otros objetos (y personas) puedan aprovechar su funcionalidad.

Los objetos funcionan en conjunto mediante el envío de mensajes entre ellos. Los mensajes son peticiones para realizar operaciones.

Por lo general, los objetos se asocian entre sí y esta asociación puede ser de diversos tipos. Un objeto en una clase puede asociarse con cualquier cantidad de objetos distintos en otra clase.

La agregación es un tipo de asociación. Un objeto agregado consta de un conjunto de objetos que lo componen y una composición es un tipo especial de agregación. En un objeto compuesto, los componentes sólo existen como parte del objeto compuesto.

Preguntas y respuestas

P **Usted dijo que la orientación a objetos ha tomado por asalto al mundo del software. ¿Qué no hay algunas aplicaciones importantes que no están orientadas a objetos?**

R Sí, y se conocen como sistemas "heredados" (programas que en muchos casos son ejecutados para mostrar su época). La orientación a objetos ofrece diversas ventajas, como la reusabilidad y un rápido periodo de desarrollo. Por tales razones, muy probablemente verá las nuevas aplicaciones (y las versiones rediseñadas de varias aplicaciones antiguas) escritas bajo el esquema de la orientación a objetos.

Taller

Para repasar lo que ha aprendido de la orientación a objetos, intente responder a algunas preguntas y realizar los siguientes ejercicios. Las respuestas las encontrará en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Qué es un objeto?
2. ¿Cómo trabajan los objetos en conjunto?
3. ¿Qué establece la multiplicidad?
4. ¿Pueden asociarse dos objetos entre sí en más de una manera?

Ejercicios

Esta es una hora teórica, así que no incluí ejercicios. Verá algunos en las siguientes horas.

HORA 3

Uso de la orientación a objetos

A continuación conjugaremos las características del UML con los conceptos de la orientación a objetos. Aquí reafirmará su conocimiento de la orientación a objetos al tiempo que aprenderá otras cosas del UML.

En esta hora se tratarán los siguientes temas:

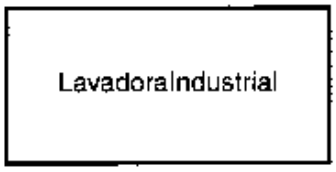
- Concepción de una clase
- Atributos
- Operaciones
- Responsabilidades y restricciones
- Qué es lo que hacen las clases y cómo encontrarlas

Concepción de una clase

Como lo indiqué en la primera hora, en el UML un rectángulo es el símbolo que representa una clase. El nombre de la clase es, por convención, una palabra con la primera letra en mayúscula y normalmente se coloca en la parte superior del rectángulo. Si el nombre de su clase consta de dos palabras, únalas e inicie cada una con mayúscula (como en LavadoraIndustrial en la figura 3.1).

FIGURA 3.1

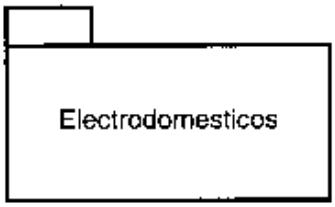
La representación UML de una clase.



Otra estructura del UML, el paquete, puede jugar un papel en el nombre de la clase. Como indiqué en la hora 1, "Introducción al UML", un paquete es la manera en que el UML organiza un diagrama de elementos. Tal vez recuerde que el UML representa un paquete como una carpeta tabular cuyo nombre es una cadena de texto (vea la figura 3.2).

FIGURA 3.2

Un paquete del UML.



TERMINO NUEVO Si la clase "Lavadora" es parte de un paquete llamado "Electrodomesticos", podrá darle el nombre "Electrodomesticos::Lavadora". El par de dos puntos separa al nombre del paquete, que está a la izquierda, del nombre de la clase, que va a la derecha. A este tipo de nombre de clase se le conoce como *nombre de ruta* (vea la figura 3.3).

Posiblemente haya notado que en los nombres se han evitado los caracteres acentuados (como en Electrodomesticos) y la letra eñe. Esto se debe a que en el alfabeto inglés, tales caracteres no están contemplados y no podemos asegurar que el utilizarlos en sus identificadores no le traiga problemas, tanto en el UML como en el lenguaje de programación que piense utilizar para traducir los modelos. Por ello, evitaremos los acentos en todos los diagramas que se presentan a lo largo de este libro, de igual manera, evitaremos el uso de la letra eñe, misma que sustituiremos —en su caso— por "ni" (como en Anio, en lugar de Año).

FIGURA 3.3

Una clase con un nombre de ruta.

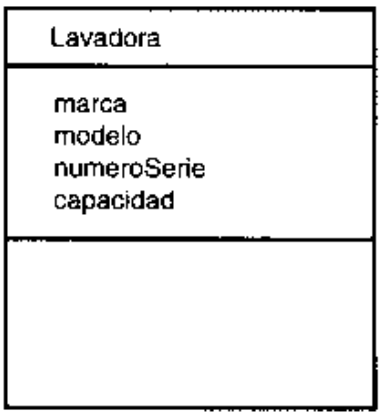
Atributos

TERMINO NUEVO Un *atributo* es una propiedad o característica de una clase y describe un rango de valores que la propiedad podrá contener en los objetos (esto es, instancias) de la clase. Una clase podrá contener varios o ningún atributo. Por convención, si el atributo consta de una sola palabra se escribe en minúsculas; por otro lado, si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primer palabra que comenzará en minúscula. La lista

de nombres de atributos iniciará luego de una línea que la separe del nombre de la clase, como se aprecia en la figura 3.4.

FIGURA 3.4

Una clase y sus atributos.

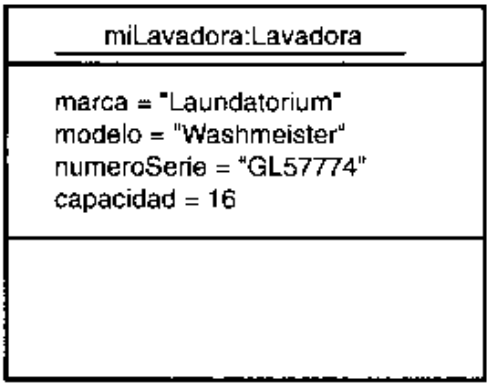


Todo objeto de la clase tiene un valor específico en cada atributo. La figura 3.5 le muestra un ejemplo. Observe que el nombre de un objeto inicia con una letra minúscula, y está precedido de dos puntos que a su vez están precedidos del nombre de la clase, y todo el nombre está subrayado.

El nombre `miLavadora:Lavadora` es una *instancia con nombre*; pero también es posible tener una *instancia anónima*, como `:Lavadora`.

FIGURA 3.5

Un objeto cuenta con un valor específico en cada uno de los atributos que lo componen.

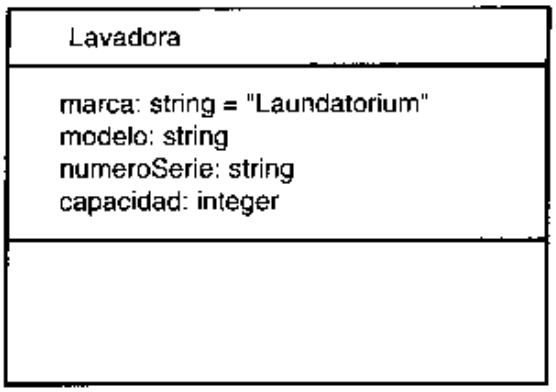


El UML le da la opción de indicar información adicional de los atributos. En el símbolo de la clase, podrá especificar un tipo para cada valor del atributo. Entre los posibles tipos se encuentran cadena (string), número de punto flotante (float), entero (integer) y booleano (boolean), así como otros tipos enumerados. Para indicar un tipo, utilice dos puntos (:) para separar el nombre del atributo de su tipo. También podrá indicar un valor predeterminado para un atributo. La figura 3.6 le muestra las formas de establecer atributos.

Aunque no parece haber restricción en la designación de tipos a las variables, utilizaremos los nombres en inglés para ceñirnos a los tipos que aparecen en los lenguajes de programación.

FIGURA 3.6

Un atributo puede mostrar su tipo así como su valor predeterminado.



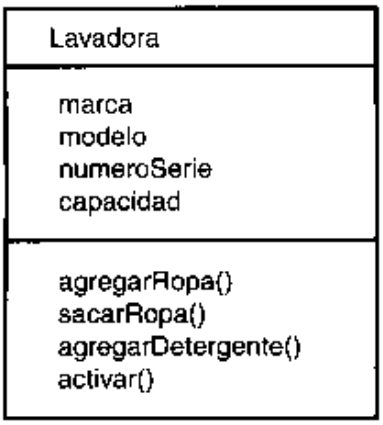
Operaciones

TÉRMINO NUEVO

Una *operación* es algo que la clase puede realizar, o que usted (u otra clase) pueden hacer a una clase. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe en minúsculas si consta de una sola palabra. Si el nombre constara de más de una palabra, únalas e inicie todas con mayúscula exceptuando la primera. La lista de operaciones se inicia debajo de una línea que separa a las operaciones de los atributos, como se muestra en la figura 3.7.

FIGURA 3.7

La lista de operaciones de una clase aparece debajo de una línea que las separa de los atributos de la clase.



TÉRMINO NUEVO

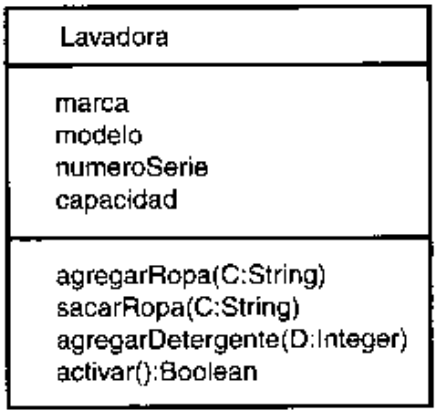
Así como es posible establecer información adicional de los atributos, también lo es en lo concerniente a las operaciones. En los paréntesis que preceden al nombre de la operación podrá mostrar el parámetro con el que funcionará la operación junto con su tipo de dato. La *función*, que es un tipo de operación, devuelve un valor luego que finaliza su trabajo. En una función podrá mostrar el tipo de valor que regresará.

TÉRMINO NUEVO

Estas secciones de información acerca de una operación se conocen como la *firma* de la operación. La figura 3.8 le muestra cómo representar la firma.

FIGURA 3.8

La firma de una operación.

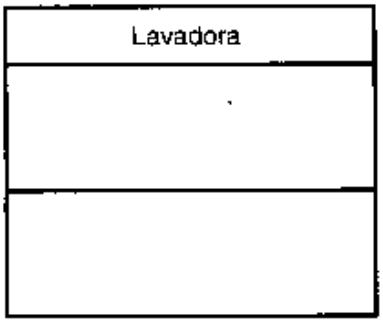


Atributos, operaciones y concepción

Hasta ahora, hemos tratado a las clases como entidades aisladas, y hemos visto todos los atributos y operaciones de una clase. No obstante, en la práctica mostrará más de una clase a la vez; cuando lo haga, no será muy útil que siempre aparezcan todos los atributos y operaciones, ya que el hacerlo le crearía un diagrama muy saturado. En lugar de ello podrá tan sólo mostrar el nombre de la clase y dejar ya sea el área de atributos o el de operaciones (o ambas) vacía, como se muestra en la figura 3.9.

FIGURA 3.9

En la práctica, no siempre mostrará todos los atributos y operaciones de una clase.

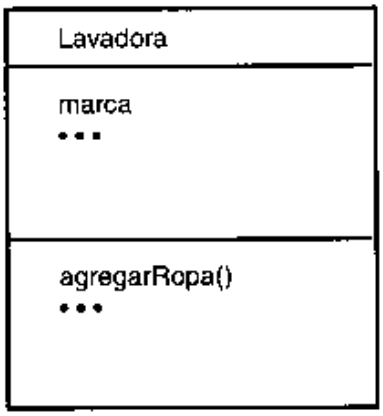


TERMINO NUEVO

En ocasiones será bueno mostrar algunos (pero no todos) de los atributos u operaciones. Para indicar que sólo enseñará algunos de ellos, seguirá la lista de aquellos que mostrará con tres puntos (...), mismos que se conocen como *puntos suspensivos*. A la omisión de ciertos o todos los atributos y operaciones se le conoce como *abreviar* una clase. La figura 3.10 le muestra el uso de los puntos suspensivos.

FIGURA 3.10

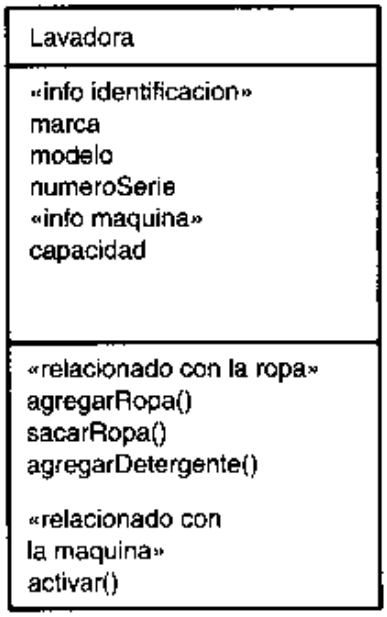
Los puntos suspensivos indican atributos u operaciones que no se encuentran en todo el conjunto.



Si usted tiene una larga lista de atributos u operaciones podrá utilizar un estereotipo para organizarla de forma que sea más comprensible. Un estereotipo es el modo en que el UML le permite extenderlo, es decir, crear nuevos elementos que son específicos de un problema en particular que intente resolver. Como lo mencioné en la hora 1, usted muestra un estereotipo como un nombre bordeado por dos pares de paréntesis angulares. Para una lista de atributos, podrá utilizar un estereotipo como encabezado de un subconjunto de atributos, como en la figura 3.11.

FIGURA 3.11

Podrá usar un estereotipo para organizar una lista de atributos u operaciones.



El estereotipo es una estructura flexible, la cual podrá utilizar de diversos modos. Por ejemplo, podrá utilizar el estereotipo sobre el nombre de una clase en un símbolo de clase para indicar algo respecto al papel de la clase.

Responsabilidades y restricciones

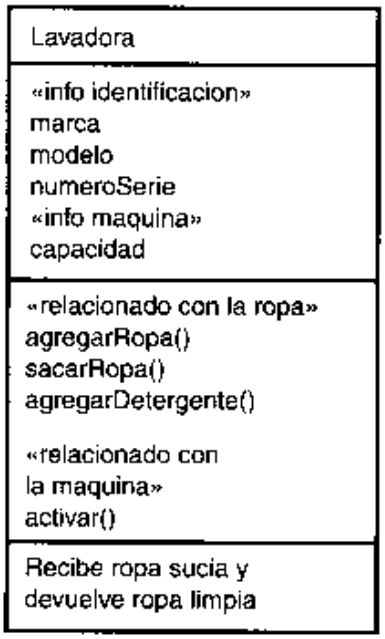
TERMINO NUEVO

El símbolo de clase le permite establecer otro tipo de información de sí misma. En un área bajo la lista de operaciones, podrá mostrar la responsabilidad de la clase. La *responsabilidad* es una descripción de lo que hará la clase, es decir, lo que sus atributos y operaciones intentan realizar en conjunto. Una lavadora, por ejemplo, tiene la responsabilidad de recibir ropa sucia y dar por resultado ropa limpia.

En el símbolo, indicará las responsabilidades en un área inferior a la que contiene las operaciones (vea la figura 3.12).

FIGURA 3.12

En un símbolo de clase, que irá debajo de la lista de operaciones, escribirá las responsabilidades de la clase.



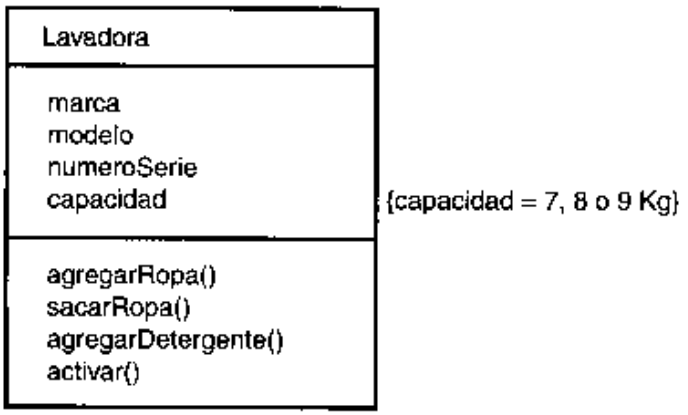
La idea es incluir información suficiente para describir una clase de forma inequívoca.

TÉRMINO NUEVO

Una manera más formal es agregar una *restricción*, un texto libre bordeado por llaves; este texto especifica una o varias reglas que sigue la clase. Por ejemplo, suponga que en la clase Lavadora usted desea establecer que la capacidad de una lavadora será de 7, 8 o 9 Kg (y así, "restringir" el atributo capacidad de la clase Lavadora). Usted escribirá {capacidad = 7, 8 o 9 Kg} junto al símbolo de la clase Lavadora. La figura 3.13 le muestra cómo hacerlo.

FIGURA 3.13

La regla entre llaves restringe al atributo capacidad para contener uno de tres posibles valores.



El UML funciona con otra forma –aún más formal– de agregar restricciones que hacen más explícitas las definiciones. Es todo un lenguaje conocido como *OCL (Lenguaje de restricción de objetos)*. OCL cuenta con su propio conjunto de reglas, términos y operadores, lo que lo convierte en una herramienta avanzada y, en ocasiones, útil.

Notas adjuntas

Por encima y debajo de los atributos, operaciones, responsabilidades y restricciones, puede agregar mayor información a una clase en la figura de notas adjuntas.

Con frecuencia agregará una nota a un atributo u operación. La figura 3.14 le muestra una nota que se refiere a una norma gubernamental que indica dónde encontrar la manera en que se generan los números de serie para los objetos de la clase Lavadora.

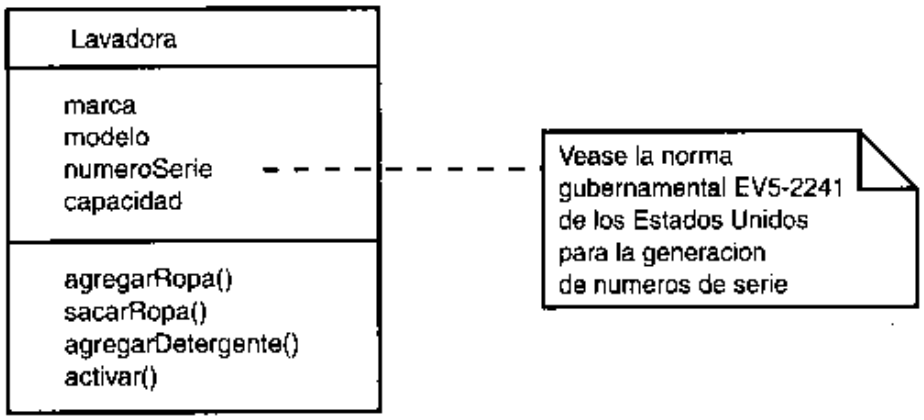


FIGURA 3.14

Una nota adjunta proporciona mayor información respecto a la clase.

Una nota puede contener tanto una imagen como texto.

Qué es lo que hacen las clases y cómo encontrarlas

Las clases son el vocabulario y terminología de un área del conocimiento. Conforme hable con los clientes, analice su área de conocimiento y diseñe sistemas de computación que resuelvan los problemas de dicha área, comprenderá la terminología y modelará los términos como clases en el UML.

En sus conversaciones con los clientes preste atención a los sustantivos que utilizan para describir las entidades de sus negocios; ya que dichos sustantivos se convertirán en las clases de su modelo. También preste atención a los verbos que escuche, dado que constituirán las operaciones de sus clases. Los atributos surgirán como sustantivos relacionados con los nombres de la clase. Una vez que tenga una lista básica de las clases, pregunte a los clientes qué es lo que cada clase hace dentro del negocio. Sus respuestas le indicarán las responsabilidades de la clase.

Suponga que usted es un analista que generará un modelo del juego de baloncesto, y que entrevista a un entrenador para comprender el juego. La conversación podría surgir como sigue:

Analista: "Entrenador, ¿de qué se trata el juego?"

Entrenador: "Consiste en arrojar el balón a través de un aro, conocido como cesto, y hacer una mayor puntuación que el oponente. Cada equipo consta de cinco jugadores: dos defensas, dos delanteros y un central. Cada equipo lleva el balón al cesto del equipo oponente con el objetivo de hacer que el balón sea encestado."

Analista: "¿Cómo se hace para llevar el balón al otro cesto?"

Entrenador: "Mediante pases y dribles. Pero el equipo tendrá que encestar antes de que termine el lapso para tirar."

Analista: "¿El lapso para tirar?"

Entrenador: "Así es, son 24 segundos en el baloncesto profesional, 30 en un juego internacional, y 35 en el colegial para tirar el balón luego de que un equipo toma posesión de él."

Analista: "¿Cómo funciona el puntaje?"

Entrenador: "Cada canasta vale dos puntos, a menos que el tiro haya sido hecho detrás de la línea de los tres puntos. En tal caso, serán tres puntos. Un tiro libre contará como un punto. A propósito, un tiro libre es la penalización que paga un equipo por cometer una infracción. Si un jugador infracciona a un oponente, se detiene el juego y el oponente puede realizar diversos tiros al cesto desde la línea de tiro libre."

Analista: "Háble me más acerca de lo que hace cada jugador."

Entrenador: "Quienes juegan de defensa son, en general, quienes realizan la mayor parte de los dribles y pases. Por lo general tienen menor estatura que los delanteros, y éstos, a su vez, son menos altos que el central (que también se conoce como 'poste'). Se supone que todos los jugadores pueden burlar, pasar, tirar y rebotar. Los delanteros realizan la mayoría de los rebotes y los disparos de mediano alcance, mientras que el central se mantiene cerca del cesto y dispara desde un alcance corto."

Analista: "¿Qué hay de las dimensiones de la cancha? Y ya que estamos en eso ¿cuánto dura el juego?"

Entrenador: "En un juego internacional, la cancha mide 28 metros de longitud y 15 de ancho; el cesto se encuentra a 3.05 m del piso. En un juego profesional, el juego dura 48 minutos, divididos en cuatro cuartos de 12 minutos cada uno. En un juego colegial e internacional, la duración es de 40 minutos, divididos en dos mitades de 20 minutos. Un cronómetro del juego lleva un control del tiempo restante."

La plática podría continuar, pero hagamos una pausa y veamos con qué contamos. Aquí hay varios sustantivos que ha descubierto: balón, cesto, equipo, jugadores, defensas, delanteros, centro (o poste), tiro, lapso para tirar, línea de los tres puntos, tiro libre, infracción, línea de tiro libre, cancha, cronómetro del juego.

Y los verbos: tirar, avanzar, driblar (o burlar), pasar, infraccionar, rebotar. También cuenta con cierta información adicional respecto a algunos de los sustantivos (como las

estaturas relativas de los jugadores de cada posición, las dimensiones de la cancha, la cantidad total de tiempo en un lapso de tiro y la duración de un juego).

Finalmente, su propio sentido común podría entrar en acción para generar ciertos atributos por usted mismo. Usted sabe, por ejemplo, que el balón cuenta con ciertos atributos, como volumen y diámetro.

A partir de esta información, podrá crear un diagrama como el de la figura 3.15. En él se muestran las clases y se proporcionan ciertos atributos, operaciones y restricciones. El diagrama también muestra las responsabilidades. Podría usar este diagrama como fundamento para otras conversaciones con el entrenador para obtener mayor información.

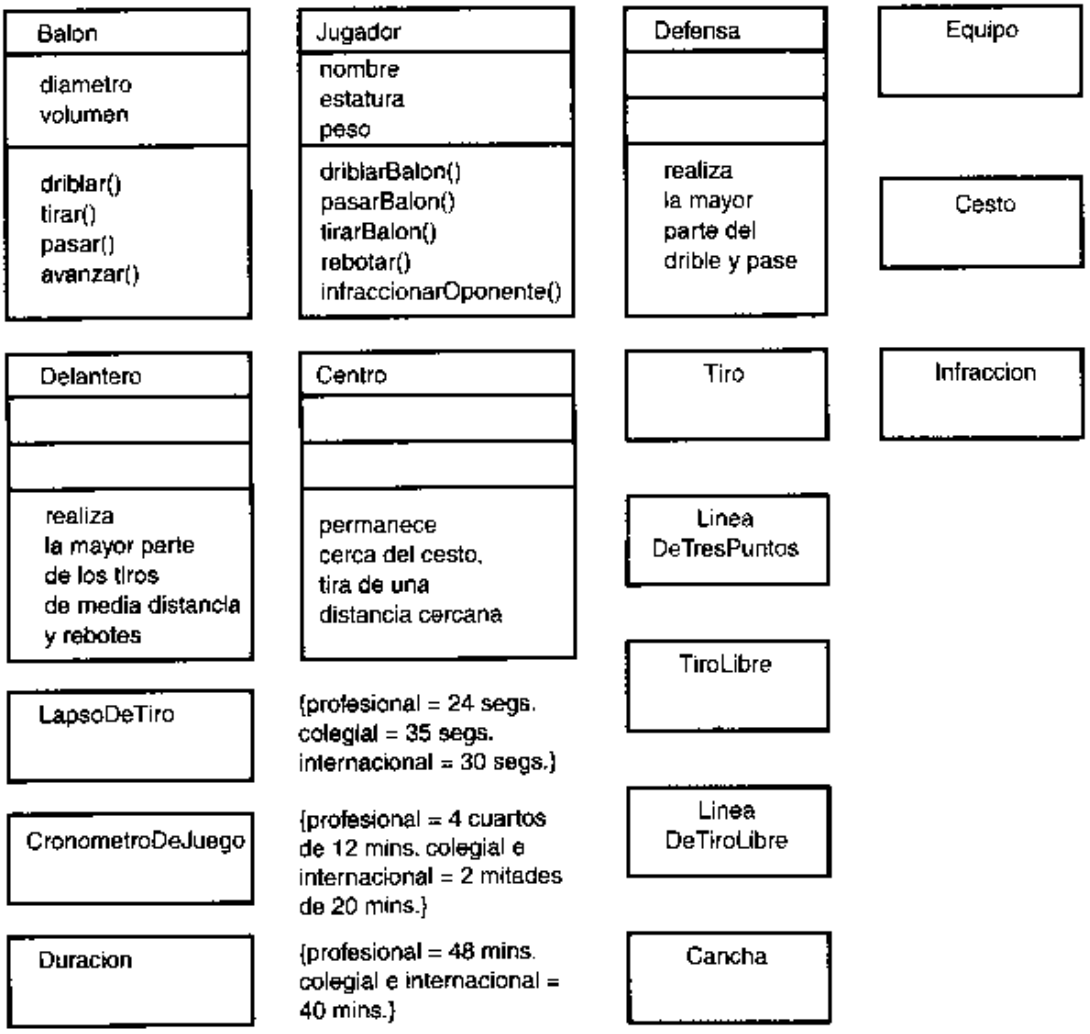


FIGURA 3.15

Un diagrama inicial para modelar el juego de baloncesto.

Resumen

Un rectángulo es, en el UML, la representación simbólica de una clase. El nombre, atributos, operaciones y responsabilidades de la clase se colocan en áreas delimitadas dentro del rectángulo. Puede utilizar un estereotipo para organizar las listas de atributos y operaciones y además abreviar una clase al mostrar sólo un subconjunto de sus atributos y operaciones. Esto hace un diagrama de clases menos complejo.

Podrá mostrar el tipo de un atributo, su valor inicial y enseñar los valores con que funciona una operación, así como sus tipos. En una operación, esta información se conoce como firma.

Para reducir la ambigüedad en la descripción de una clase agregue restricciones. El UML también le permite indicar mayor información respecto a una clase mediante notas adjuntas al rectángulo que la representa.

Las clases representan el vocabulario de un área del conocimiento. Las conversaciones con el cliente o un experto en el área dejarán entrever los sustantivos que se convertirán en clases en un modelo, y los verbos se transformarán en operaciones. Podrá utilizar un diagrama de clases como una forma de estimular al cliente a que diga más respecto a su área y que ponga en evidencia cierta información adicional.

Preguntas y respuestas

P **Usted mencionó el uso del "sentido común" para generar el diagrama de clases del baloncesto. Ello suena bien en tal instancia pero, ¿qué ocurre cuando tengo que analizar un área desconocida para mí (donde el sentido común no será de mucha ayuda)?**

R Por lo general, contará con cierto apoyo en un área desconocida para usted. Antes de que se reúna con un cliente o con un experto en el campo, intente convertirse en un "subexperto". Prepárese para la reunión y lea cuanta documentación relacionada tenga a la mano. Pregunte a sus entrevistados respecto a documentos o manuales que hayan escrito. Cuando haya terminado de leer, tendrá cierto conocimiento básico y podrá realizar las preguntas indicadas.

P **¿En qué momento tendría que mostrar la firma de una operación?**

R Tal vez, luego de la fase de análisis de un proceso de desarrollo, conforme se adentre en el diseño. La firma es una sección de información que los desarrolladores podrían encontrar muy útil.

Taller

Para repasar lo que ha aprendido respecto a la orientación a objetos, intente responder a las siguientes preguntas. Las respuestas las encontrará en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cómo representa una clase en el UML?
2. ¿Qué información puede mostrar en un símbolo de clase?
3. ¿Qué es una restricción?
4. ¿Para qué adjuntaría una nota a un símbolo de clase?

Ejercicios

1. He aquí una breve (e incompleta) descripción del balompié:

   Un equipo de balompié (o fútbol soccer) consiste en 11 jugadores de campo (1 portero y el resto, jugadores de cancha que, en ocasiones, se organizan en cuatro defensas, tres centrales y tres delanteros). Los jugadores pueden usar cualquier parte de su cuerpo (excepto las manos) para introducir el balón a la portería del equipo contrario. La única excepción a esta regla la tiene el portero, quien puede utilizar también las manos para jugar el balón, pero sólo dentro del área de meta. El campo de juego es un rectángulo de una longitud máxima de 120 m y mínima de 90 m; y con una anchura no mayor de 90 m, ni menor de 45. Para partidos internacionales, la longitud será de 110 m como máximo y 100 como mínimo; y una anchura no superior a 75 m ni inferior a 64. En cualquier caso, deberá ser mayor la longitud que la anchura. El campo de juego se dividirá en dos mitades transversales de igual tamaño. El centro del campo será marcado con un punto visible, alrededor del cual se trazará una circunferencia de 9.15 m de radio. La meta del juego es pasar el balón a los delanteros, quienes están mejor preparados para patear el balón a la portería. El portero (o arquero) es la última línea de defensa que intentará bloquear, con cualquier parte de su cuerpo, los tiros de sus opositores. Cada vez que evita un gol, es decir, que el balón entre a la portería, habrá salvado su meta. Cada gol equivale a un punto. Un juego dura 90 minutos, divididos en dos periodos de 45 minutos cada uno.

   Válgase de la anterior información para crear un diagrama como el de la figura 3.15. Si usted conoce más del balompié de lo que he descrito, agregue tal información a su diagrama.

2. Si usted conoce más del baloncesto de lo que hay en la figura 3.15, agregue la información a tal diagrama.

HORA 4

Uso de relaciones

En la hora anterior creamos un conjunto de clases que representaban el vocabulario del baloncesto. Aunque ello le da las bases para una mayor exploración de lo que es el baloncesto, tal vez haya sentido que algo le falta.

Ese "algo" es un sentido en el que las clases se relacionan entre sí. Si observa el modelo (vea la figura 3.15), verá que no se indica la manera en que un jugador se relaciona con un balón, ni cómo los jugadores conforman un equipo, ni la forma en que procede el juego. Es como si hubiera construido una lista de elementos, en lugar de una representación de un área del conocimiento. Es importante saber cómo se conectan las clases entre sí.

Ahora trazaremos las conexiones entre las clases y completaremos la representación.

En esta hora se tratarán los siguientes temas:

- Asociaciones
- Multiplicidad
- Asociaciones calificadas

- Asociaciones reflexivas
- Herencia y generalización
- Dependencias

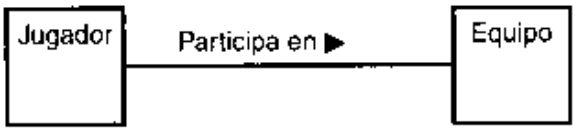
Asociaciones

TÉRMINO NUEVO

Cuando las clases se conectan entre sí de forma conceptual, esta conexión se conoce como *asociación*. El modelo inicial de baloncesto le dará algunos ejemplos. Examinemos uno de ellos: la asociación entre un jugador y un equipo. Podrá caracterizar tal asociación con la frase: "un jugador participa en un equipo". Visualizará la asociación como una línea que conectará a ambas clases, con el nombre de la asociación ("participa en") justo sobre la línea. Es útil indicar la dirección de la relación, y lo hará con un triángulo relleno que apunte en la dirección apropiada. La figura 4.1 le muestra cómo visualizar la asociación "Participa en" entre el jugador y el equipo.

FIGURA 4.1

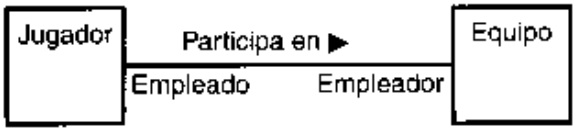
Una asociación entre un jugador y un equipo.



Cuando una clase se asocia con otra, cada una de ellas juega un papel dentro de tal asociación. Puede representar estos papeles en el diagrama escribiéndolos cerca de la línea que se encuentra junto a la clase que juega el papel correspondiente. En la asociación entre un jugador y un equipo, si el equipo es profesional, éste es un empleador y el jugador es un empleado. La figura 4.2 le muestra cómo representar dichos papeles.

FIGURA 4.2

Por lo general, en una asociación cada clase juega un papel. Puede representar tales papeles en el diagrama.



La asociación puede funcionar en dirección inversa: un equipo emplea a jugadores. Podrá mostrar ambas asociaciones en el mismo diagrama con un triángulo relleno que indique la dirección de cada asociación, como en la figura 4.3.

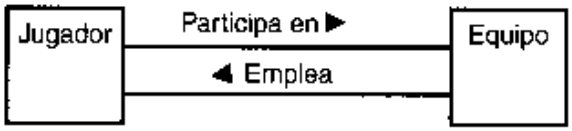


FIGURA 4.3

Pueden aparecer dos asociaciones entre clases en el mismo diagrama.

Las asociaciones podrían ser más complejas que tan sólo una clase conectada a otra. Varias clases se pueden conectar a una. Si toma en cuenta los defensas, delanteros y central, así como sus asociaciones con la clase Equipo, tendrá el diagrama de la figura 4.4.

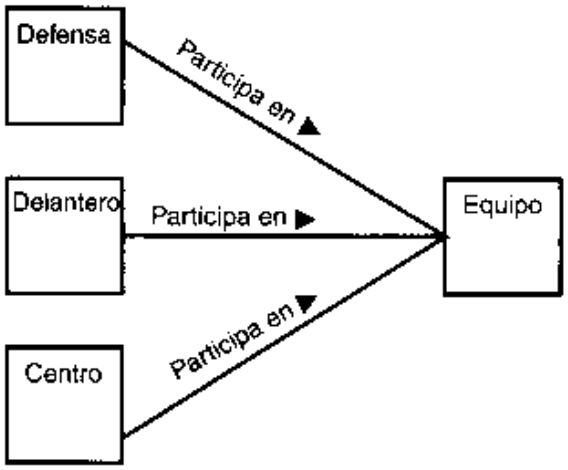


FIGURA 4.4

Pueden asociarse diversas clases con una en particular.

Restricciones en las asociaciones

En ocasiones una asociación entre dos clases debe seguir cierta regla. Ésta se indica al establecer una restricción junto a la línea de asociación. Por ejemplo: un Cajero atiende a un Cliente, pero cada Cliente es atendido en el orden en que se encuentre en la formación. Puede capturar este modelo colocando la palabra *ordenado* entre llaves (para indicar la restricción) junto a la clase Cliente, como se ve en la figura 4.5.

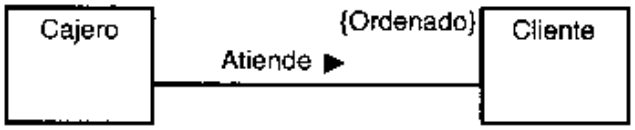


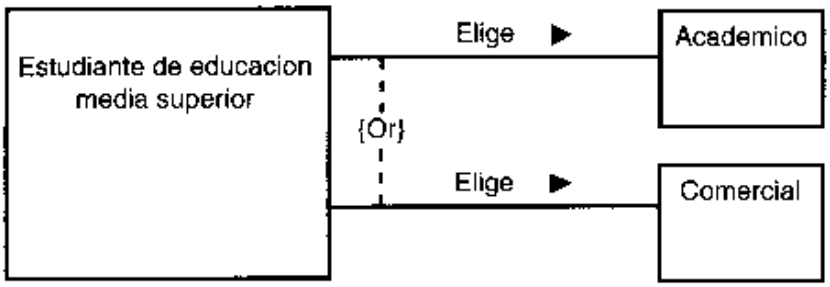
FIGURA 4.5

Puede establecer una restricción en una asociación. En este caso, la asociación Atiende está restringida para que el Cajero atienda al Cliente en turno.

Otro tipo de restricción es la relación O (distinguida como {Or}) en una línea discontinua que conecte a dos líneas de asociación. La figura 4.6 modela a un estudiante de educación media superior que elegirá entre un curso académico o uno comercial.

FIGURA 4.6

La relación O entre dos asociaciones en una restricción.



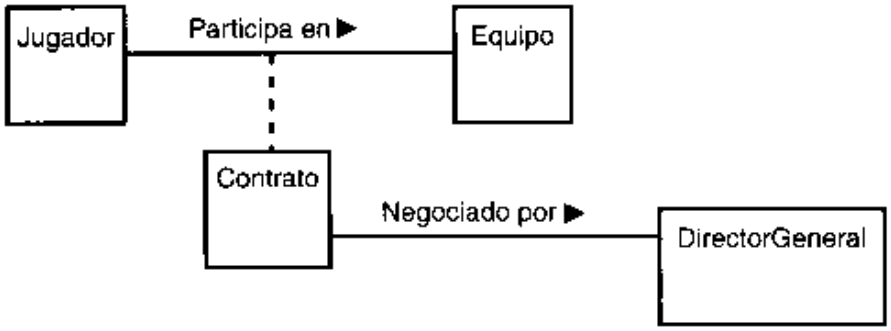
Clases de asociación

TÉRMINO NUEVO

Una asociación, al igual que una clase, puede contener atributos y operaciones. De hecho, cuando éste sea el caso, usted tendrá una *clase de asociación*. Puede concebir a una clase de asociación de la misma forma en que lo haría con una clase estándar, y utilizará una línea discontinua para conectarla a la línea de asociación. Una clase de asociación puede tener asociaciones con otras clases. La figura 4.7 le muestra una clase de asociación para la asociación "Participa en" entre un jugador y un equipo. La clase de asociación, Contrato, se asocia con la clase DirectorGeneral.

FIGURA 4.7

Una clase de asociación modela los atributos y operaciones de una asociación. Se conecta a una asociación mediante una línea discontinua, y puede asociarse a otra clase.

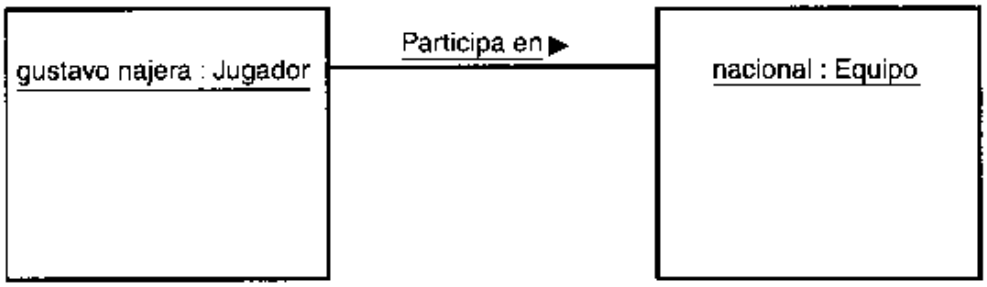


Vínculos

Así como un objeto es una instancia de una clase, una asociación también cuenta con instancias. Si podemos imaginar a un jugador específico que juega para un equipo específico, la relación "Participa en" se conocerá como *vínculo*, y usted lo representará como una línea que conecta a dos objetos. Tal como tuvo que subrayar el nombre de un objeto, deberá subrayar el nombre de un vínculo, como en la figura 4.8.

FIGURA 4.8

Un vínculo es la instancia de una asociación. Conecta a los objetos en lugar de las clases. Deberá subrayar el nombre del vínculo, como se hace en el nombre de un objeto.



Multiplicidad

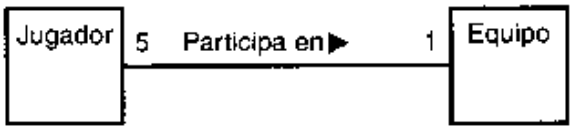
La asociación trazada entre Jugador y Equipo sugiere que las dos clases tienen una relación de uno a uno. No obstante, el sentido común nos indica que éste no es el caso. Un equipo de baloncesto cuenta con cinco jugadores (sin contar a los sustitutos). La asociación Tiene (Has) debe participar en este recuento. En la otra dirección, un jugador puede participar sólo en un equipo, y la asociación "Participa en" debe responder de esto.

Término Nuevo

Tales especificaciones son ejemplos de la *multiplicidad*: la cantidad de objetos de una clase que se relacionan con un objeto de la clase asociada. Para representar los números en el diagrama, los colocará sobre la línea de asociación junto a la clase correspondiente, como se denota en la figura 4.9.

Figura 4.9

La multiplicidad señala la cantidad de objetos de una clase que pueden relacionarse con un objeto de una clase asociada.

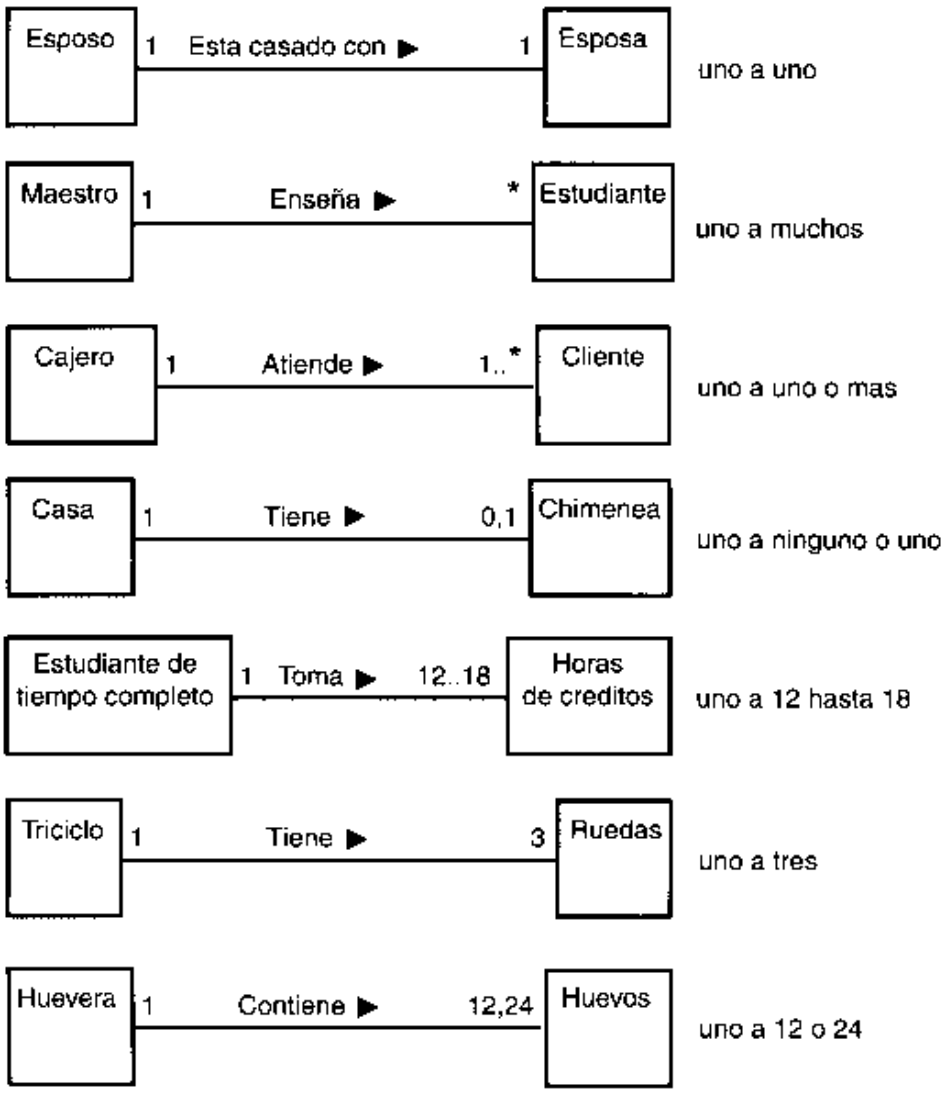


La multiplicidad de este ejemplo no es la única que existe. Hay varios tipos de multiplicidades (una multiplicidad de multiplicidades, por decirlo así). Una clase puede relacionarse con otra en un esquema de uno a uno, uno a muchos, uno a uno o más, uno a ninguno o uno, uno a un intervalo definido (por ejemplo: uno a cinco hasta diez), uno a exactamente *n* (como en este ejemplo), o uno a un conjunto de opciones (por ejemplo, uno a nueve o diez). El UML utiliza un asterisco (*) para representar *más* y para representar *muchos*. En un contexto O se representa por dos puntos, como en "1..*" ("uno o más"). En otro contexto, O se representa por una coma, como en "5, 10" ("5 o 10"). La figura 4.10 le muestra cómo concebir las posibles multiplicidades.

Cuando la clase A tiene una multiplicidad de uno a ninguno o uno con la clase B, la clase B se dice que es opcional para la clase A.

FIGURA 4.10

Posibles multiplicidades y cómo representarlas en el UML.



Asociaciones calificadas

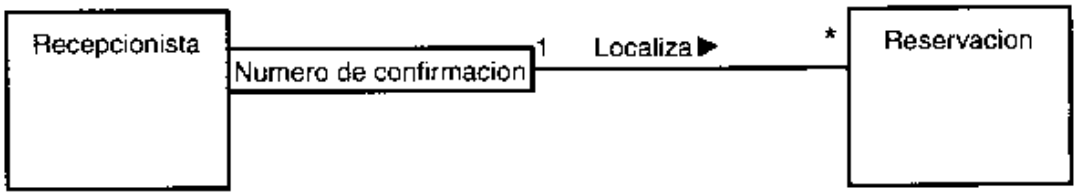
Cuando la multiplicidad de una asociación es de uno a muchos, con frecuencia se presenta un reto muy particular: la búsqueda. Cuando un objeto de una clase tiene que seleccionar un objeto particular de otro tipo para cumplir con un papel en la asociación, la primera clase deberá atenerse a un atributo en particular para localizar al objeto adecuado. Normalmente, dicho atributo es un identificador que puede ser un número de identidad. Por ejemplo, cuando usted realiza una reservación en un hotel, el hotel le asigna un número de confirmación. Si usted quiere hacer preguntas respecto a la reservación, deberá proporcionar el número de confirmación.

TÉRMINO NUEVO

En el UML la información de identidad se conoce como *calificador*. Su símbolo es un pequeño rectángulo adjunto a la clase que hará la búsqueda. La figura 4.11 muestra la representación. La idea es reducir, con eficiencia, la multiplicidad de uno a muchos a una multiplicidad de uno a uno.

FIGURA 4.11

Un calificador en una asociación resuelve el problema de la búsqueda.

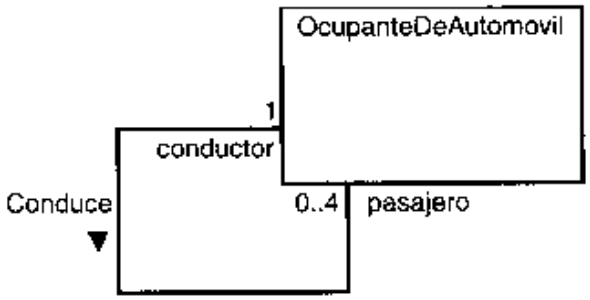


Asociaciones reflexivas

En ocasiones, una clase es una asociación consigo misma. Esto puede ocurrir cuando una clase tiene objetos que pueden jugar diversos papeles. Un OcupanteDeAutomovil puede ser un Conductor o un Pasajero. En el papel del conductor, el OcupanteDeAutomovil puede llevar ninguno o más OcupanteDeAutomovil, quienes jugarán el papel de pasajeros. Esto lo representará mediante el trazado de una línea de asociación a partir del rectángulo de la clase hacia el mismo rectángulo de la clase, y en la línea de asociación indicará los papeles, nombre de la asociación, dirección de la asociación y multiplicidad como ya lo hizo antes. La figura 4.12 le presenta este ejemplo.

FIGURA 4.12

En una asociación reflexiva, trazará la línea de la clase hacia sí misma y podrá incluir los papeles, nombre de la asociación y su dirección, así como su multiplicidad.



Herencia y generalización

Uno de los sellos distintivos de la orientación a objetos es que captura uno de los mayores aspectos del sentido común en cuanto a la vida diaria: si usted conoce algo de una categoría de cosas, automáticamente sabrá algunas cosas que podrá transferir a otras categorías. Si usted sabe que algo es un electrodoméstico, ya sabrá que contará con un interruptor, una marca y un número de serie. Si sabe que algo es un animal dará por hecho que come, duerme, tiene una forma de nacer, de trasladarse de un lugar a otro y algunos otros atributos (y operaciones) que podría listar si pensara en ello por algunos instantes.

TÉRMINO NUEVO

La orientación a objetos se refiere a esto como *herencia*. El UML también lo denomina *generalización*. Una clase (la clase secundaria o subclase) puede heredar los atributos y operaciones de otra (la clase principal o superclase). La clase principal (o madre) es más genérica que la secundaria (o hija).

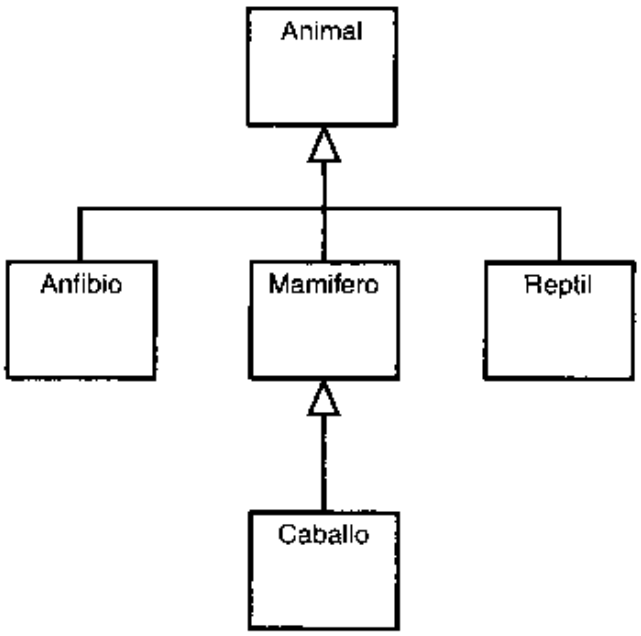
En la generalización, una clase secundaria (hija) es sustituible por una clase principal (madre). Es decir, donde quiera que se haga referencia a la clase madre, también se hace referencia a la clase hija. Sin embargo, en el caso contrario no es aplicable.

La jerarquía de la herencia no tiene que finalizar en dos niveles: una clase secundaria puede ser principal para otra clase secundaria. Un Mamífero es una clase secundaria de Animal, y Caballo es una clase secundaria de Mamífero.

En el UML representará la herencia con una línea que conecte a la clase principal con la secundaria. En la parte de la línea que se conecta con la clase principal, colocará un triángulo sin rellenar que apunte a la clase principal. Este tipo de conexión se interpreta con la frase *es un tipo de*. Un Mamífero *es un tipo de* Animal, y un Caballo *es un tipo de* Mamífero. La figura 4.13 le muestra esta particular jerarquía de la herencia, junto con otras clases. Observe la apariencia del triángulo y las líneas cuando varias clases secundarias son herencia de una clase principal. Al disponer el diagrama de este modo, trae por resultado un diagrama más ordenado en lugar de mostrar todas las líneas y triángulos, aunque el UML no le prohíbe colocarlos todos en la imagen. También vea que no colocó los atributos y operaciones heredadas en los rectángulos de las subclases, dado que ya los había representado en la superclase.

FIGURA 4.13

Una jerarquía de herencia en el reino animal.



Cuando modele la herencia, tenga la seguridad de que la clase secundaria satisfaga la relación *es un tipo de* con la clase principal. Si no se cumple tal relación, tal vez una asociación de otro tipo podría ser más adecuada.

Con frecuencia las clases secundarias agregan otras operaciones y atributos a los que han heredado. Por ejemplo: un Mamífero tiene pelo y da leche, dos atributos que no se encuentran en la clase Animal.

TERMINO NUEVO

Una clase puede no provenir de una clase principal, en cuyo caso será una *clase base* o *clase raíz*. Una clase podría no tener clases secundarias, en cuyo caso será una clase final o clase hoja. Si una clase tiene exactamente una clase principal, tendrá una *herencia simple*. Si proviene de varias clases principales, tendrá una *herencia múltiple*.

Descubrimiento de la herencia

En el proceso de plática con un cliente, un analista descubrirá la herencia de varias formas. Es posible que las clases candidatas que aparezcan incluyan tanto clases principales como clases secundarias. El analista deberá darse cuenta que los atributos y operaciones de una clase son generales y que se aplicarán a, quizá, varias clases (mismas que agregarán sus propios atributos y operaciones).

El ejemplo del baloncesto de la hora 3, "Uso de la orientación a objetos", tiene las clases Jugador, Defensa, Delantero y Central. El Jugador tiene atributos como nombre, estatura, peso, velocidadAlCorrer y saltoVertical. Tiene operaciones como driblar(), pasar(), rebotar() y tirar(). Las clases Defensa, Delantero y Centro heredarán tales atributos y operaciones, y agregarán los suyos. La clase Defensa podría tener las operaciones correrAlFrente() y quitarBalon(). El Central podría tener retacarBalon(). De acuerdo con los comentarios del entrenador respecto a las estaturas de los jugadores, el analista tal vez quisiera colocar restricciones en las estaturas para cada posición.

Otra posibilidad es que el analista note que dos o más clases tienen ciertos atributos y operaciones en común. El modelo del baloncesto tiene un CronometroDeJuego (que controla el tiempo que resta en un periodo de juego) y un LapsoDeTiro (que controla el tiempo restante desde el instante que un equipo tomó posesión del balón, hasta que intente encestar). Si nos damos cuenta de que ambos controlan el tiempo, el analista podría formular una clase Reloj con una operación controlarTiempo() que podrían heredar tanto CronometroDeJuego como LapsoDeTiro.

Dado que LapsoDeTiro controla 24 segundos (profesional) o 35 segundos (colegial) y el CronometroDeJuego controla 12 minutos (profesional) o 20 minutos (colegial), controlarTiempo() será polimórfico.

Clases abstractas

En el modelo del baloncesto, el par de clases que mencioné —Jugador y Reloj— son útiles puesto que funcionan como clases principales para clases secundarias importantes. Las clases secundarias son importantes en el modelo dado que finalmente usted querrá

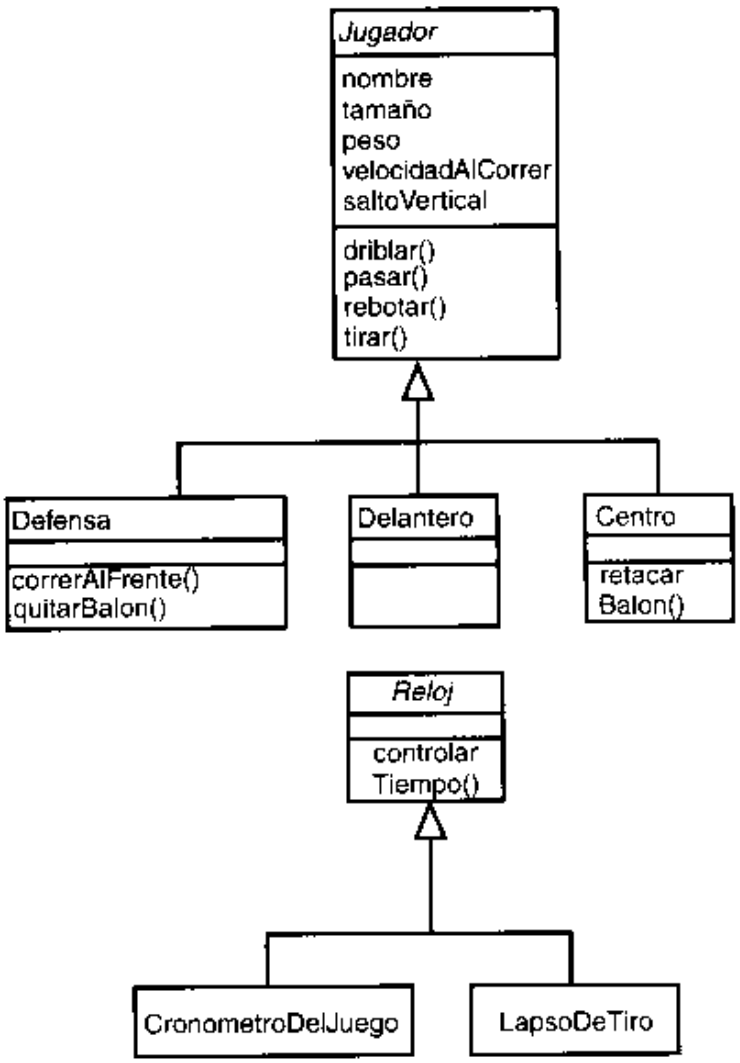
tener instancias de tales clases. Para desarrollar el modelo, necesitará instancias de Defensa, Delantero, Centro, CronometroDeJuego y LapsoDeTiro.

No obstante, Jugador y Reloj no proporcionan ninguna instancia al modelo. Un objeto de la clase Jugador no serviría a ningún propósito, así como tampoco uno de la clase Reloj.

TÉRMINO NUEVO Las clases como Jugador y Reloj —que no proveen objetos— se dice que son *abstractas*. Una clase abstracta se distingue por tener su nombre en cursivas. La figura 4.14 muestra las dos clases abstractas y sus clases secundarias.

FIGURA 4.14

Dos jerarquías de herencia con clases abstractas en el modelo de baloncesto.



Dependencias

TÉRMINO NUEVO En otro tipo de relación, una clase utiliza a otra. A esto se le llama dependencia. El uso más común de una dependencia es mostrar que la firma de la operación de una clase utiliza a otra clase.

Suponga que diseñará un sistema que muestra formularios corporativos en pantalla para que los empleados los llenen. El empleado utiliza un menú para seleccionar el formulario por llenar. En su diseño, tiene una clase Sistema y una clase Formulario. Entre sus

muchas operaciones, la clase Sistema tiene mostrarFormulario(f:Form). El formulario que el sistema desplegará, dependerá, obviamente, del que elija el usuario. La notación del UML para ello es una línea discontinua con una punta de flecha en forma de triángulo sin relleno que apunta a la clase de la que depende, como muestra la figura 4.15.

FIGURA 4.15

Una flecha representada por una línea discontinua con una punta de flecha en forma de triángulo sin relleno simboliza una dependencia.

Resumen

Sin las relaciones, un modelo de clases sería poco menos que una lista de cosas que representarían un vocabulario. Las relaciones le muestran cómo se conectan los términos del vocabulario entre sí para dar una idea de la sección del mundo que se modela. La asociación es la conexión conceptual fundamental entre clases. Cada clase en una asociación juega un papel, y la multiplicidad especifica cuántos objetos de una clase se relacionan con un objeto de la clase asociada. Hay muchos tipos de multiplicidad. Una asociación se representa como una línea entre los rectángulos de clases con los papeles y multiplicidades en cada extremo. Al igual que una clase, una asociación puede contener atributos y operaciones.

Una clase puede heredar atributos y operaciones de otra clase. La clase heredada es secundaria de la clase principal que es de la que se hereda. Descubrirá la herencia cuando encuentre clases en su modelo inicial que tengan atributos y operaciones en común. Las clases abstractas sólo se proyectan como bases de herencia y no proporcionan objetos por sí mismas. La herencia se representa como una línea entre la clase principal y la secundaria, con un triángulo sin rellenar que se adjunta (y apunta a) la clase principal.

En una dependencia, una clase utiliza a otra. El uso más común de una dependencia es mostrar que una firma en la operación de una clase utiliza a otra clase. Una dependencia se proyecta como una línea discontinua que reúne a las dos clases en la dependencia, con una punta de flecha en forma de triángulo sin relleno que adjunta (y apunta a) la clase de la que se depende.

Preguntas y respuestas

P ¿En alguna ocasión se le puede poner nombre a una relación de herencia, como se hace en una asociación?

R El UML no le impide que adjudique un nombre a una relación de herencia, pero por lo general esto no es necesario.

Taller

El cuestionario y los ejercicios se han diseñado para reafirmar su conocimiento del UML en el área de las relaciones. Cada pregunta y ejercicio requiere que usted piense en la simbología del modelado que ha aprendido y la aplique a una situación. Las respuestas se encuentran en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionarios

1. ¿Cómo representaría la multiplicidad?
2. ¿Cómo descubrirá la herencia?
3. ¿Qué es una clase abstracta?
4. ¿Cuál es el efecto de un calificador?

Ejercicios

1. Tome como base el modelo del baloncesto de la Hora 3, y agregue vínculos que expresen las relaciones que ha visto en esta hora. Si conoce el juego del baloncesto, siéntase con libertad de agregar los vínculos que representen su conocimiento.
2. De acuerdo con un viejo adagio: "Un abogado que se defiende a sí mismo, tiene por cliente a un tonto." Cree un modelo que refleje esta pieza de sabiduría.

HORA 5

Agregación, composición, interfaces y realización

Continuaremos con las relaciones entre clases y comprenderá nuevos conceptos respecto a las clases y sus diagramas.

En esta hora se tratarán los siguientes temas:

- Agregaciones
- Composiciones
- Contextos
- Interfaces y realizaciones
- Visibilidad

Ya ha visto lo concerniente a asociación, multiplicidad y herencia y está casi listo para crear diagramas de clases significativos. Conforme explore otros tipos de relaciones y detalles relacionados con las clases comprenderá las piezas finales del rompecabezas. La meta final es crear una idea estática de un sistema, con todas las conexiones entre las clases que lo conforman.

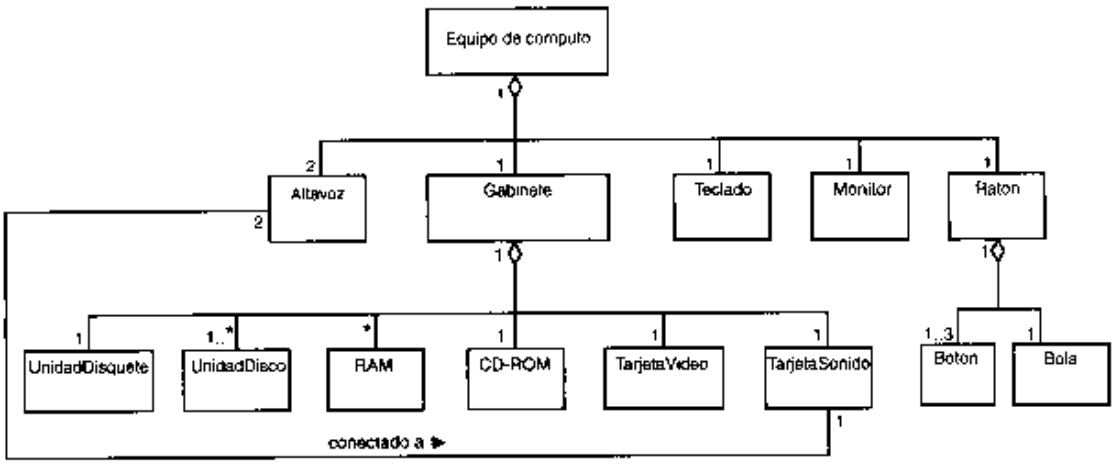
Agregaciones

TÉRMINO NUEVO En ocasiones una clase consta de otras clases. Éste es un tipo especial de relación conocida como *agregación* o *acumulación*. Los componentes y la clase que constituyen son una asociación que conforma un todo. En la hora 2, "Orientación a objetos", mencioné que su computadora es un conjunto de elementos que consta de gabinete, teclado, ratón, monitor, unidad de CD-ROM, una o varias unidades de disco duro, módem, unidad de disquete, impresora y, posiblemente, altavoces. Además de las unidades de disco, el gabinete contiene la memoria RAM, una tarjeta de vídeo y una tarjeta de sonido (tal vez algunos otros elementos).

Puede representar una agregación como una jerarquía dentro de la clase completa (por ejemplo el sistema computacional) en la parte superior, y los componentes por debajo de ella. Una línea conectará el todo con un componente mediante un rombo sin relleno que se colocará en la línea más cercana al todo. La figura 5.1 le muestra el sistema de cómputo como una agregación.

FIGURA 5.1

Una asociación por agregación se representa por una línea entre el componente y el todo con un rombo sin relleno que conforma al todo.



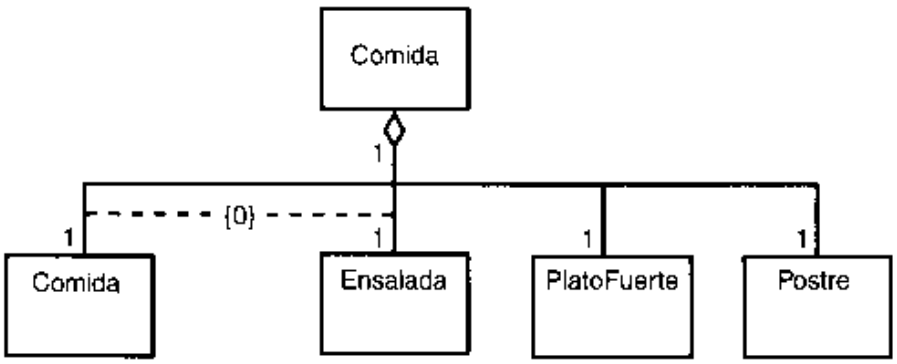
Aunque este ejemplo le muestra cada componente correspondiente a un todo, en una agregación éste no será necesariamente el caso. Por ejemplo: en un sistema casero de entretenimiento, un control remoto podría ser un componente de una televisión, aunque también podría ser un componente de una reproductora de casetes de vídeo.

Restricciones en las agregaciones

En ocasiones el conjunto de componentes posibles en una agregación se establece dentro de una relación O. En ciertos restaurantes, una comida consta de sopa o ensalada, el plato fuerte y el postre. Para modelar esto, utilizaría una restricción: la palabra O dentro de llaves con una línea discontinua que conecte las dos líneas que conforman al todo, como lo muestra la figura 5.2.

FIGURA 5.2

Puede establecer una restricción a una agregación para mostrar que un componente u otro es parte del todo.

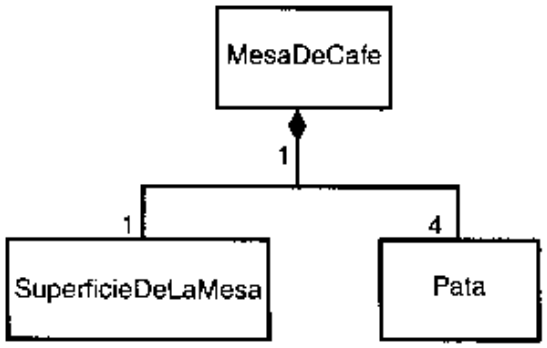


Composiciones

Una composición es un tipo muy representativo de una agregación. Cada componente dentro de una composición puede pertenecer tan sólo a un todo. Los componentes de una mesa de café (la superficie de la mesa y las patas) establecen una composición. El símbolo de una composición es el mismo que el de una agregación, excepto que el rombo está relleno (vea la figura 5.3).

FIGURA 5.3

En una composición, cada componente pertenece solamente a un todo. Un rombo relleno representa esta relación.



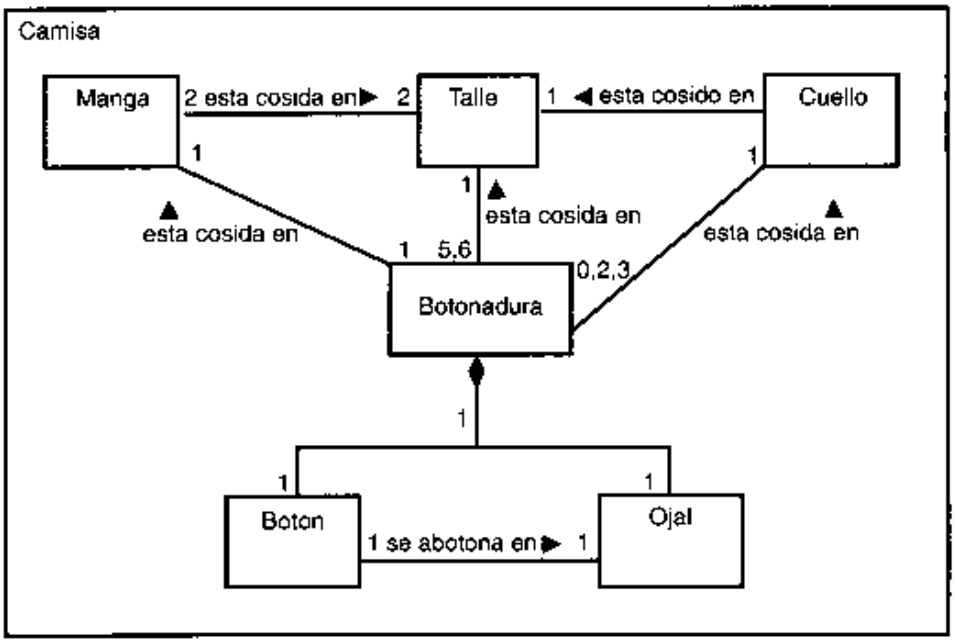
Contextos

Cuando modele un sistema podrían producirse, con frecuencia, agrupamientos de clases, como agregaciones o composiciones. En tal caso, deberá enfocar su atención en un agrupamiento o en otro, y el diagrama de contexto le proporciona la característica de modelaje que requiere para tal fin. Las composiciones figuran en gran medida dentro de los diagramas de contexto. Un diagrama de contexto es como un mapa detallado de alguna sección de un mapa de mayores dimensiones. Pueden ser necesarias varias secciones para capturar toda la información detallada.

He aquí un ejemplo: suponga que está creando un modelo de una camisa y la forma en que se podría combinar con algún atuendo y un guardarropa. Un tipo de diagrama de contexto (vea la figura 5.4) le mostrará la camisa como un gran rectángulo de clase, con un diagrama anidado en el interior, el cual le muestra cómo los componentes de la camisa están relacionados entre sí. Éste es un diagrama de contexto de composición (dado que la sola camisa reúne a cada componente se le denomina *de composición*).

FIGURA 5.4

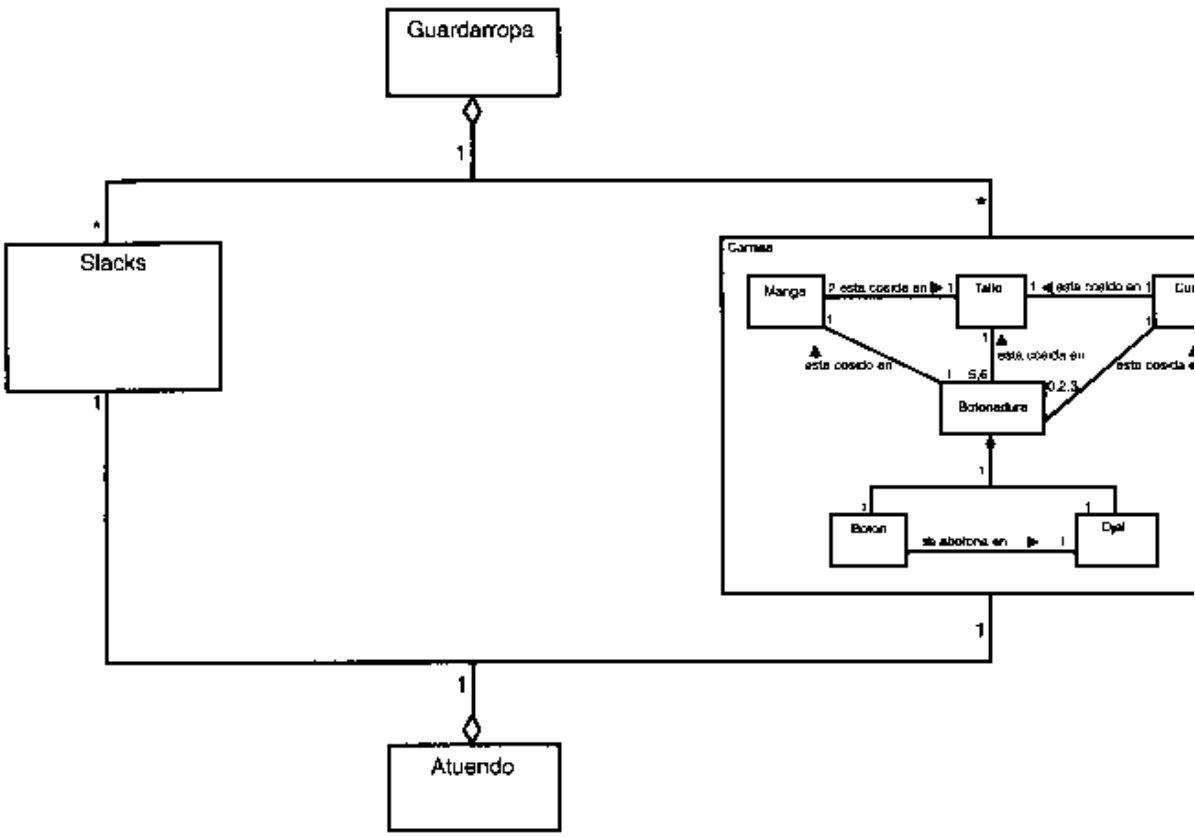
Un diagrama de contexto de composición le muestra los componentes de una clase como un diagrama anidado dentro de un enorme rectángulo de clase.



El diagrama de contexto de composición enfoca la atención en la camisa y sus componentes. Para mostrar la camisa en el contexto del guardarropa y de algún atuendo, tendrá que ampliar su ámbito. Un diagrama de contexto del sistema lo hará por usted. Podrá mostrar la forma en que la clase Camisa se conecta con las clases Guardarropa y Atuendo, como se ve en la figura 5.5.

FIGURA 5.5

Un diagrama de contexto del sistema le muestra los componentes de una clase y la forma en que la clase se relaciona con las otras que hay en el sistema.



Podrá ver de cerca alguna otra clase y presentar sus detalles en algún otro diagrama de contexto.

Interfaces y realizaciones

Una vez que haya creado varias clases, tal vez se dé cuenta que no pertenecen a una clase principal, pero en su comportamiento debe incluir algunas de las mismas operaciones con las mismas firmas de la primera clase. Podría codificar las operaciones en una de las clases y reutilizarlas en otras. Una segunda posibilidad es que desarrolle una serie de operaciones para las clases en un sistema, y reutilizarlas para las clases de otro sistema.

TERMINO NUEVO

De cualquier manera, deseará contar con algún medio para capturar el conjunto reutilizable de operaciones. La interfaz es la estructura del UML que le permite hacerlo. Una *interfaz* es un conjunto de operaciones que especifica cierto aspecto de la funcionalidad de una clase, y es un conjunto de operaciones que una clase presenta a otras.

Con un ejemplo podríamos aclarar lo anterior. El teclado que usted utiliza para comunicarse con su equipo es una interfaz reutilizable. Su operación basada en la opresión de teclas ha provenido de la máquina de escribir. La disposición de las teclas es casi la misma que en una máquina de escribir, pero el punto principal es que la operación por opresión de teclas ha sido cedida de un sistema a otro. Otras operaciones (Mayús, Bloq Mayús y Tab) también se integraron a partir de la máquina de escribir.

Por supuesto, el teclado de una computadora incluye diversas operaciones que no encontrará en una máquina de escribir: Control, Alt, RePág, AvPág y otras. Así pues, la interfaz puede establecer un subconjunto de las operaciones de una clase y no necesariamente todas ellas.

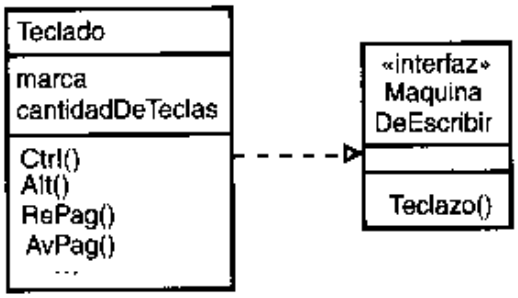
Puede modelar una interfaz del mismo modo en que modelaría una clase, con un símbolo rectangular. La diferencia será que, como un conjunto de operaciones, una interfaz no tiene atributos. Recordará que puede omitir los atributos de la representación de una clase. ¿Entonces, cómo distinguiría entre una interfaz y una clase que no muestra sus atributos? Una forma es utilizar la estructura "estereotipo" y especificar la palabra «interfaz» sobre el nombre de la interfaz en el rectángulo. Otra es colocar la letra "I" al principio del nombre de una interfaz.

TERMINO NUEVO

En cierto sentido, es como si el teclado de la computadora garantizara que esta parte de su funcionalidad "haría las veces" del teclado de una máquina de escribir. Bajo este esquema, la relación entre una clase y una interfaz se conoce como *realización*. Esta relación está modelada como una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que adjunte y apunte a la interfaz. La figura 5.6 le muestra cómo se lleva a cabo esto.

FIGURA 5.6

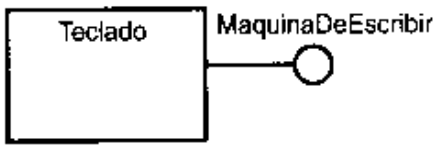
Una interfaz es un conjunto de operaciones que realiza una clase. Esta última se relaciona con una interfaz mediante la realización, misma que se indica por una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que apunte a la interfaz.



Otra forma (omitida) de representar una clase y su interfaz es con un pequeño círculo que se conecte mediante una línea a la clase, como se ve en la figura 5.7.

FIGURA 5.7

La forma omitida de representar una clase que realice una interfaz.



Una clase puede realizar más de una interfaz, y una interfaz puede ser realizada por más de una clase.

Visibilidad

TERMINO NUEVO

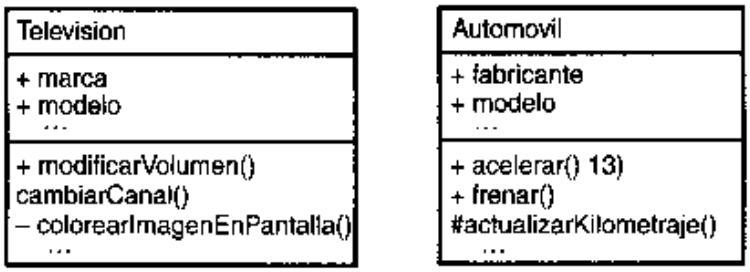
El concepto de visibilidad está muy relacionado con las interfaces y la realización. La *visibilidad* se aplica a atributos u operaciones, y establece la proporción en que otras clases podrán utilizar los atributos y operaciones de una clase dada (o en operaciones de una interfaz). Existen tres niveles de visibilidad: Nivel *público*, en el cual la funcionalidad se extiende a otras clases. En el nivel *protegido* la funcionalidad se otorga sólo a las clases que se heredan de la clase original. En el nivel *privado* sólo la clase original puede utilizar el atributo u operación. En una televisión, modificarVolumen() y cambiarCanal() son operaciones públicas, en tanto que dibujarImagenEnPantalla() es privada. En un automóvil, acelerar() y frenar() son operaciones públicas, pero actualizarKilometraje() o actualizarMillaje() es protegida.

La realización, como podría imaginar, implica que el nivel público se aplique a cualquier operación en una interfaz. La protección de operaciones mediante cualquiera de los otros niveles tal vez no tendría sentido, dado que una interfaz se orienta a ser realizada por diversas clases.

Para indicar el nivel público, anteceda el atributo u operación con un signo de suma (+), para revelar un nivel protegido, antecédalo con un símbolo de número (#), y para indicar el nivel privado, antecédalo con un guión (-). La figura 5.8 muestra los atributos y operaciones públicos, protegidos y privados tanto en una televisión como en un automóvil.

FIGURA 5.8

Los atributos y operaciones públicos y privados, tanto de una televisión como de un automóvil.



Ámbito

TERMINO NUEVO

El ámbito es otro concepto referente a los atributos y operaciones, y la forma en que se relacionan dentro de un sistema. Hay dos tipos de ámbitos, el de instancia y el de archivador. En el primero cada instancia cuenta con su propio valor en un atributo u operación. En un ámbito de *archivado*, sólo habrá un valor del atributo u operación en todas las instancias de la clase. Un atributo u operación con el ámbito de archivador, aparece con su nombre subrayado. Este tipo de ámbito se utiliza con frecuencia cuando un grupo específico de instancias (ningunas otras) tienen que compartir los valores exactos de un atributo privado. El ámbito de instancia es, por mucho, el tipo más común de ámbito.

Resumen

Para completar sus nociones de clases y la forma en que se conectan, es necesario comprender algunas relaciones adicionales. Una agregación establece una asociación para conformar un todo: una clase "todo" se genera de clases que la componen. Un componente en una agregación puede ser parte de más de un todo. Una composición es una conformación muy íntimamente ligada con la agregación en el sentido de que un componente de una composición puede ser parte solamente de un todo. La representación del UML de las agregaciones es similar a la representación de las composiciones. La línea de asociación que une la parte con un todo tiene un rombo. En una agregación, el rombo no está relleno, en tanto que en una composición sí lo está.

Un diagrama de contexto enfoca la atención en una clase específica dentro de un sistema. Un diagrama de contexto de composición es como un mapa detallado de un mapa mayor. Muestra un diagrama de clases anidado dentro de un gran símbolo rectangular de clase. Un diagrama de contexto de sistema muestra la forma en que el diagrama de clases compuestas se relaciona con otros objetos del sistema.

Una realización es una asociación entre una clase y una interfaz, una colección de operaciones que cierta cantidad de clases podrá utilizar. Una interfaz se representa como una clase sin atributos. Para distinguirla de una clase cuyos atributos hayan sido omitidos del diagrama, el estereotipo «interfaz» aparecerá por encima del nombre de la interfaz. Otra posibilidad es la de anteceder el nombre de la interfaz con una "I" mayúscula. La realización se representa en el UML mediante una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que conecta a la clase con la interfaz. Otra forma para representar una realización es con una línea continua que conecte a una clase con un pequeño círculo, para que el círculo se interprete como la interfaz.

En términos de visibilidad, todas las operaciones en una interfaz son *públicas*, de modo que cualquier clase podrá utilizarlas. Los otros dos niveles de visibilidad son *protegido* (la funcionalidad se extiende a las clases secundarias de aquella que contiene los atributos y operaciones) y *privado* (atributos y operaciones que se pueden utilizar sólo dentro de la clase que los contiene). Un signo de suma (+) denota a la visibilidad pública, el símbolo de número (#) la protegida y el guión (-) la privada.

El ámbito es otro aspecto de los atributos y operaciones. En un ámbito de instancia, cada objeto de una clase cuenta con su propio valor en un atributo u operación. En un ámbito de archivador, sólo hay un valor para un atributo u operación en particular a través de un conjunto de objetos de una clase. Los objetos que no estén en este conjunto no podrán acceder al valor contenido en el ámbito de archivador.

Preguntas y respuestas

P ¿Se considera transitiva a la agregación? Es decir, si la clase 3 es un componente de la clase 2, y la clase 2 es un componente de la clase 1, ¿la clase 3 será un componente de la clase 1?

R Así es, la agregación es transitiva. En nuestro ejemplo, los botones y la bola del ratón son parte del ratón, a la vez que son parte de la computadora.

P ¿La palabra "interfaz" implica "interfaz de usuario" o GUI?

R No. Es algo más genérico. Una interfaz es tan sólo un conjunto de operaciones que una clase presenta a las demás clases. De hecho, una de estas operaciones podría ser (aunque no necesariamente) la del usuario.

Taller

El cuestionario y los ejercicios verificarán y fortalecerán su conocimiento respecto al tema de las agregaciones, composiciones, contextos e interfaces. Las respuestas las podrá ver en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cuál es la diferencia entre una agregación y una composición?
2. ¿Qué es la realización?
3. Mencione los tres niveles de visibilidad y describa lo que significa cada uno de ellos.

Ejercicios

1. Cree un diagrama de contexto de composición de una revista. Tome en cuenta la tabla de contenido, la editorial, los artículos y las columnas. Luego, cree un diagrama de contexto del sistema que muestre a la revista junto con el suscriptor y el comprador en el puesto de revistas.

2. En la actualidad, el tipo más popular de GUI es la interfaz WIMP (ventanas, iconos, menús y puntero, por sus siglas en inglés). Dibuje un diagrama de clases de la interfaz WIMP, y haga uso de todo el conocimiento adecuado del UML que ha adquirido hasta ahora. Además de las clases indicadas en las siglas, incluya los elementos relacionados como las barras de desplazamiento y el cursor, así como cualquiera de las otras clases necesarias.

HORA 6

Introducción a los casos de uso

Ahora que ha visto lo correspondiente a las clases y sus relaciones, es el momento de volver nuestra atención a otra área principal del UML: los casos de uso.

En esta hora se tratarán los siguientes temas:

- Qué son los casos de uso
- Importancia de los casos de uso
- Inclusión de los casos de uso
- Extensión de los casos de uso
- Inicio de un análisis de un caso de uso

En las tres horas anteriores hemos visto los diagramas que proporcionan una idea estática de las clases en un sistema. Ahora veremos a los diagramas que establecen una idea dinámica y mostraremos la forma en que el sistema y sus clases cambian con el tiempo. Las ideas estáticas ayudan a que un analista se comunique con un cliente. La idea dinámica, como verá, ayudará al analista a comunicarse con un grupo de desarrolladores, y ayudará a estos últimos a crear programas.

El cliente y el equipo de desarrollo conforman un importante conjunto de integrantes en un sistema. No obstante, una parte de igual importancia no se ha tomado en cuenta: el usuario. Ni la idea estática ni la dinámica mostrarán el comportamiento del sistema desde el punto de vista del usuario. Comprender tal punto de vista es clave para generar sistemas que sean tanto útiles como funcionales; esto es, que cumplan con los requerimientos y que sea fácil (e, incluso, divertido) trabajar con ellos.

El modelado de un sistema desde el punto de vista de un usuario es el trabajo de los casos de uso. En esta hora comprenderá todo lo relacionado con los casos de uso y su función. En la siguiente hora aprenderá a utilizar el diagrama de casos de uso del UML para visualizar un caso de uso.

Qué son los casos de uso

Recientemente adquirí una máquina de fax. Cuando fui a comprarla, en un almacén de venta de equipo para oficinas, encontré una enorme gama de opciones. ¿Cómo hice para decidirme por una en particular? Me pregunté exactamente qué es lo que deseaba hacer con una máquina de fax. ¿Qué características deseaba? ¿Cuáles funciones necesitaba que tuviera? ¿Deseaba utilizar papel común o térmico? ¿Quería generar copias? ¿Conectarlo a mi computadora? ¿Utilizarlo como digitalizador? ¿Tendría que enviar faxes a tal velocidad que necesitaría una función de marcado rápido? ¿Querría utilizar la máquina de fax para diferenciar entre una llamada telefónica y un fax entrante?

Todos seguimos un procedimiento como éste cuando realizamos una compra que no sea impulsiva. Lo que hacemos es seguir un tipo de *análisis del caso de uso*: nos preguntamos cómo utilizaremos el producto o sistema que queremos comprar, de modo que podamos obtener algo que cumpla con nuestras necesidades. Lo importante es saber cuáles son esos requerimientos.

Este tipo de análisis es particularmente crucial para la fase de análisis del desarrollo de un sistema. La forma en que los usuarios utilicen un sistema le da la pauta para lo que diseñará y creará.

El caso de uso es una estructura que ayuda a los analistas a trabajar con los usuarios para determinar la forma en que se usará un sistema. Con una colección de casos de uso se puede hacer el bosquejo de un sistema en términos de lo que los usuarios intenten hacer con él.

TÉRMINO NUEVO Imagínese al caso de uso como una colección de situaciones respecto al uso de un sistema. Cada escenario describe una secuencia de eventos. Cada secuencia se inicia por una persona, otro sistema, una parte del hardware o por el paso del tiempo. A las entidades que inician secuencias se les conoce como *actores*. El resultado de la secuencia debe ser algo utilizable ya sea por el actor que la inició, o por otro actor.

Importancia de los casos de uso

Así como el diagrama de clases es un buen medio para estimular a un cliente a que hable respecto a un sistema desde su propio punto de vista, el caso de uso es una excelente herramienta para estimular a que los usuarios potenciales hablen, de un sistema, desde sus propios puntos de vista. No siempre es fácil para los usuarios explicar cómo pretenden utilizar un sistema. Puesto que el desarrollo tradicional de los sistemas era, con frecuencia, algo así como una ciencia oculta, con muy poca información para los usuarios, a aquellos que osaban preguntar se les daba información muy poco explícita o ciertamente confusa respecto a lo que utilizarían.

La idea es involucrar a los usuarios en las etapas iniciales del análisis y diseño del sistema. Esto aumenta la probabilidad de que el sistema sea de mayor provecho para la gente a la que supuestamente ayudará, en lugar de ser un manojo de expresiones de computación incomprensibles e inmanejables por los usuarios finales.

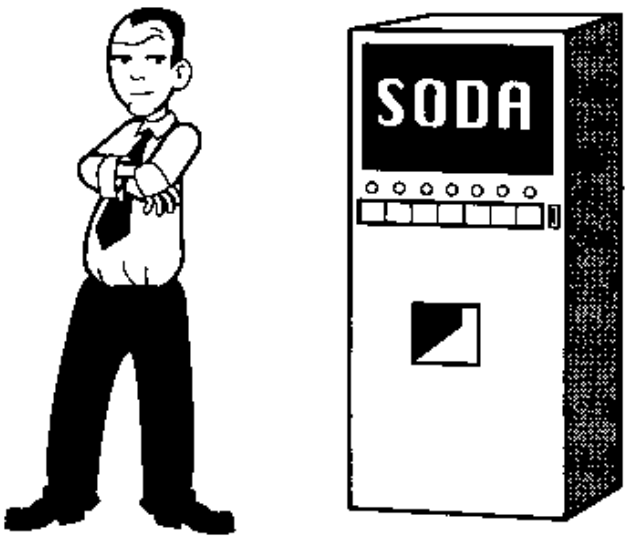
Un ejemplo: la máquina de gaseosas

Suponga que empezará a diseñar una máquina despachadora de gaseosas. Para obtener el punto de vista del interesado, entrevistará a varios usuarios potenciales respecto a la manera en que utilizarán dicha máquina.

Dado que la función principal de una máquina de gaseosas es permitir a un cliente adquirir una lata de gaseosa, probablemente las personas le dirán que se enfrentará a diversos escenarios —un caso de uso, en otras palabras— que podría etiquetar como "Comprar gaseosa". Examinemos cada posible escenario en este caso de uso. Recuerde que tales escenarios podrían aparecer durante la conversación con los usuarios.

FIGURA 6.1

Un caso de uso establece un conjunto de escenarios para realizar algo útil para un actor. En este ejemplo, un caso de uso es "Comprar gaseosa".



El caso de uso "Comprar gaseosa"

El actor, en este caso de uso, es un cliente que desea comprar una lata de gaseosa. El escenario iniciará cuando el cliente inserte dinero, posteriormente realizará una selección; y si todo funciona bien, la máquina contará con, al menos, una lata de la gaseosa elegida, misma que pondrá al alcance del cliente.

Además de la secuencia, hay otros aspectos del escenario anterior que merecen cierta consideración. ¿Qué condiciones llevaron al cliente a iniciar el escenario en el caso de uso "Comprar gaseosa"? La sed es la más obvia. ¿Qué se obtiene como resultado de tal escenario? Nuevamente, lo obvio es que el cliente tenga una gaseosa en su poder.

¿Lo que he descrito es la única posibilidad de "Comprar gaseosa"? Habría otras cuestiones que saltarían a la vista; por ejemplo, es posible que la máquina no tenga la gaseosa que desee el cliente; también es posible que el cliente no tenga el importe exacto de la gaseosa. ¿Cómo diseñaría a la máquina de gaseosas para controlar tales escenarios?

Veamos el caso en que la máquina se haya quedado sin gaseosa, otra secuencia de pasos en el caso de uso "Comprar gaseosa". Imagínelo como una ruta alternativa dentro del caso de uso. El cliente inicia el caso de uso al insertar dinero en la máquina y posteriormente hace una selección. La máquina no cuenta con ninguna lata de la gaseosa seleccionada, por lo que mostrará un mensaje al cliente que indicará que no tiene de esa marca. Lo ideal sería que el mensaje le pida al cliente que haga otra selección. La máquina también debería dar la opción de devolver el dinero al cliente. En este punto, el cliente selecciona otra marca que la maquina entregará (siempre y cuando cuente con provisiones de esta marca), o devolverá el dinero. La condición previa es un cliente sediento y el resultado es una lata de gaseosa o la devolución del dinero.

Claro que el escenario de quedarse sin gaseosa sería posible: el mensaje "No hay de esta marca" podría aparecer en cuanto las provisiones de la máquina se acabaran y permanecer a la vista hasta que la máquina sea reabastecida. En tal caso, el usuario podría no insertar el dinero en primera instancia. El cliente para el que usted diseñará la máquina podría preferir el primer escenario: si el cliente ya insertó dinero, la tendencia podría ser hacer otra selección en lugar de pedir a la máquina que lo devuelva.

Analicemos ahora el escenario de la cantidad de dinero incorrecta. Nuevamente, el usuario inicia el caso de uso en la forma usual y posteriormente hace una selección. Asumamos que la máquina tiene provisión de la marca elegida. En la máquina hay una reserva de moneda fraccionaria y devuelve la diferencia al despachar la gaseosa. Si la

máquina no cuenta con una reserva de moneda fraccionaria, devolverá el dinero y mostrará un mensaje que pida al usuario el importe exacto. La condición previa es la ya indicada. El resultado será una lata de gaseosa junto con el cambio, o la devolución del dinero originalmente depositado.

Otra posibilidad es que tan pronto como se agote la moneda fraccionaria, aparezca un mensaje que informe a los clientes que se requiere el importe exacto. El mensaje permanecería a la vista hasta que la máquina sea reabastecida con moneda fraccionaria.

Casos de uso adicionales

Ya ha examinado a la máquina de gaseosas desde el punto de vista de un usuario: el cliente. Hay otros usuarios que intervienen, como el proveedor que tiene que reabastecer a la máquina, el recolector de dinero (que tal vez sea el mismo que el proveedor) que tiene que recoger el dinero acumulado en la alcancía de la máquina, etcétera. Esto nos indica que debemos crear al menos dos casos de uso: "Reabastecer" y "Recolectar dinero", cuyos detalles surgirán durante las entrevistas con los proveedores y los recolectores.

Veamos el caso de uso de "Reabastecer". El proveedor inicia este caso de uso dado que algún intervalo (digamos, dos semanas) ha pasado. El representante del proveedor le quita el seguro a la máquina (tal vez mediante una llave y un cerrojo, pero eso entra dentro de la implementación), jala la puerta para abrir la máquina, y llena el compartimiento de cada marca hasta su capacidad. El representante también rellena la reserva de moneda fraccionaria. Luego, cierra el frente de la máquina y vuelve a poner el seguro. La condición previa es el paso del intervalo, el resultado es que el proveedor cuenta con un nuevo conjunto de ventas potenciales.

Para el caso de uso de "Recolectar el dinero", el recolector inicia debido también a que ha pasado cierto tiempo. La persona deberá seguir la misma secuencia que en "Reabastecer" para abrir la máquina. El recolector sacará el dinero de la máquina y seguirá los pasos de "Reabastecer" para cerrar y poner el seguro a la máquina. La condición previa es el paso del intervalo y el resultado es el dinero en las manos del recolector.

Vea que cuando derivamos un caso de uso, no nos preocupamos por la forma de implementarlo. En nuestro ejemplo, no nos interesamos en los aspectos internos de la máquina de gaseosas. Tampoco por la forma en que funcione el mecanismo de refrigeración, o por la forma en que la máquina controle la cantidad de dinero que contenga. Tan sólo intentamos ver la forma en que la máquina lucirá para alguien que tenga que utilizarla.

El objetivo es derivar una colección de casos de uso que, finalmente, mostraremos a las personas que diseñen la máquina de gaseosas y a las personas que la construirán. Por añadidura, nuestros casos de uso reflejan lo que los clientes, recolectores y proveedores desean, por lo que el resultado será una máquina que todos esos grupos puedan utilizar con facilidad.

Inclusión de los casos de uso

En los casos de uso "Reabastecer" y "Recolectar dinero", tal vez distinguió ciertos pasos en común. Ambos empezaban con abrir la máquina, y finalizaban con el cierre de la máquina y su aseguramiento. ¿Podríamos eliminar la duplicación de pasos de un caso de uso al otro?

Sí podemos. La forma de hacerlo es tomar cada secuencia de pasos en común y conformar un caso de uso adicional a partir de ellos. Combinemos los pasos necesarios para "quitar el seguro" y "abrir la máquina" y llamémoslos "Exhibir el interior" y los pasos "cerrar la máquina" y "asegurarla" en otro caso de uso llamado "Cubrir el interior".

Con estos nuevos casos de uso a la mano, el caso de uso "Reabastecer" iniciaría con el caso de uso "Exhibir el interior". Luego, el representante del proveedor seguiría los pasos ya indicados, y concluiría con el caso de uso "Cubrir el interior". De forma similar, el caso de uso "Recolectar dinero" iniciaría con "Exhibir el interior", procedería como se indicó, y finalizaría con el caso de uso "Cubrir el interior".

Como ve, "Reabastecer" y "Recolectar dinero" incluyen los nuevos casos de uso. Por ello, a esta técnica de aprovechamiento de un caso de uso se le conoce como *inclusión de un caso de uso.*

La inclusión de un caso de uso también se conoce como *usar un caso de uso.* Creo que el término *incluir* tiene dos ventajas. La primera, es más clara: los pasos en un caso de uso, incluyen los de otro. La segunda, se evita la confusión potencial de las palabras "usar" y "uso" en un contexto tan estrecho. Así, no tendremos que decir "promover el uso mediante el uso reiterativo de un caso de uso".

Extensión de los casos de uso

Es posible volver a utilizar un caso de uso de una forma distinta a una inclusión. En ocasiones crearemos un caso de uso agregándole algunos pasos a un caso de uso existente.

Regresemos al caso de uso "Reabastecer". Antes de colocar nuevas latas de gaseosas en la máquina, suponga que el representante del proveedor nota las marcas que se han vendido bien, así como las que no se han vendido tan bien. En lugar de sólo reabastecer todas las marcas, el representante podría sacar aquellas que no se han vendido bien, y reemplazarlas por latas de las marcas que han probado ser más populares. De esta forma, tendría que indicar al frente de la máquina el nuevo surtido de marcas disponibles.

Si agregamos estos pasos a "Reabastecer", tendremos un nuevo caso de uso que llamaríamos "Reabastecer de acuerdo a las ventas". Este nuevo caso de uso es una extensión del original, acción a la que se le conoce como *extensión de un caso de uso.*

Inicio del análisis de un caso de uso

En nuestro caso, nos hemos involucrado directamente con los casos de uso y nos hemos enfocado en algunos de ellos. En el mundo real, por lo general, seguirá un conjunto de procedimientos cuando empiece un análisis de casos de uso.

Empezará con entrevistas a los clientes (y entrevistas con expertos) que lo lleven a los diagramas iniciales de clases que indicamos en la hora 3. Esto le dará cierta idea del área en la que trabajará y una familiaridad con los términos que utilizará. Posteriormente, contará con un fundamento para hablar con los usuarios.

Entrevistará a los usuarios (preferentemente en grupos) y les pedirá que le indiquen todo lo que ellos harían con el sistema que usted diseñará. Sus respuestas conformarán un conjunto candidato de casos de uso. Luego, es importante describir brevemente cada caso de uso. También tendrá que derivar una lista de todos los actores que iniciarán y se beneficiarán de los casos de uso. Cuenta más información obtenga en esta fase, aumentará su aptitud para hablar con los usuarios en su propio idioma.

Los casos de uso aparecerán en varias fases del proceso de desarrollo. Le ayudarán con el diseño de una interfaz del usuario, coadyuvarán con las opciones de desarrollo de los programadores y establecerán las bases para probar el sistema recién generado.

Para mayor información en el tema del análisis de los casos de uso, va a tener que aplicar el UML, y ello se hará en la siguiente hora.

Resumen

El caso de uso es una estructura para describir la forma en que un sistema lucirá para los usuarios potenciales. Es una colección de escenarios iniciados por una entidad llamada actor (una persona, un componente de hardware, un lapso u otro sistema). Un caso de uso debería dar por resultado algo de valor ya sea para el actor que lo inició o para otro.

Es posible volver a utilizar casos de uso. Una forma ("inclusión") es utilizar los pasos de un caso de uso como parte de la secuencia de pasos de otro caso de uso. Otra forma ("extensión") es crear un nuevo caso de uso mediante la adición de pasos a un caso de uso existente.

La entrevista directa con los usuarios es la mejor técnica para derivar casos de uso. Cuando se deriva un caso de uso, es importante destacar las condiciones para iniciar el caso de uso, y los resultados obtenidos como consecuencia del mismo.

Hará las entrevistas a los usuarios después de entrevistar a los clientes y generar una lista de prospectos de clases. Esto le dará un fundamento en la terminología que utilizará para hablar con los usuarios. Es una buena idea entrevistar a un grupo de usuarios. El objetivo es derivar un conjunto candidato de casos de uso y todos los posibles actores.

Preguntas y respuestas

P En realidad ¿para qué necesito el concepto del caso de uso? ¿Qué no sólo podríamos preguntar a los usuarios lo que deseen ver en el sistema y dejarlo así?

R En realidad, no. Tenemos que crear una estructura de lo que los usuarios nos digan, y los casos de uso la proporcionan. La estructura se vuelve útil cuando tiene que llevar los resultados de sus entrevistas con los usuarios y comunicarlos a los clientes y desarrolladores.

P ¿Qué tan difícil es derivar los casos de uso?

R De acuerdo con mi experiencia, el listado de casos de uso —al menos los de alto nivel— no es muy complejo. Hay ciertas dificultades al profundizar en cada una e intentar lograr que los usuarios listen los pasos de cada escenario. Cuando genere un sistema que reemplace una manera existente de hacer las cosas, los usuarios típicamente ya sabrán los pasos bastante bien y los habrán utilizado con tanta regularidad que se les dificultará estructurarlos. Es una buena idea tener un panel de usuarios, ya que la discusión en grupo por lo general trae consigo ideas que un usuario en particular podría tener problemas para expresar.

Taller

Esta hora se basó en teoría más que en el UML. En este taller, el objetivo será comprender los conceptos teóricos y aplicarlos en diversos contextos. La práctica, que veremos en la siguiente hora, le ayudará a reafirmar los conceptos cuando aprenda a visualizarlos mediante el UML. Las respuestas aparecen en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cómo se llama a la entidad que inicia un caso de uso?
2. ¿Qué se entiende con "incluir un caso de uso"?
3. ¿Qué se entiende con "extender un caso de uso"?
4. ¿Un caso de uso es lo mismo que un escenario?

Ejercicios

1. En el caso del ejemplo de la máquina de gaseosas, cree otro caso de uso que incluya a los casos de uso "Exhibir el interior" y "Cubrir el interior".
2. Los casos de uso pueden ayudarle a analizar un negocio y un sistema. Imagine a una gran tienda de equipos de cómputo que venda hardware, periféricos y software. ¿Quiénes serían los actores? ¿Cuáles serían algunos de los principales casos de uso? ¿Cuáles serían algunos de los escenarios dentro de cada caso de uso?

HORA 7

Diagramas de casos de uso

El caso de uso es un poderoso concepto que ayuda a un analista a comprender la forma en que un sistema deberá comportarse. Le ayuda a obtener los requerimientos desde el punto de vista del usuario. Es necesario aprender a visualizar los conceptos del caso de uso que conoció en la hora anterior.

En esta hora se tratarán los siguientes temas:

- Representación de un modelo de caso de uso
- Concepción de las relaciones entre casos de uso
- Diagramas de casos de uso en el proceso de análisis
- Aplicación de los modelos de caso de uso
- Verá la idea general del UML

El caso de uso es muy poderoso, pero lo es aún más cuando se visualiza por medio del UML. Esta visualización le permitirá mostrar los casos de uso a los usuarios para que ellos le puedan dar mayor información. Es un hecho que los usuarios con frecuencia saben más de lo que dicen: el caso de uso ayuda a romper el hielo. A su vez, una representación visual le ayuda a combinar los diagramas de casos de uso con otro tipo de diagramas.

Una de las finalidades del proceso de análisis de un sistema es generar una colección de casos de uso. La idea es tener la posibilidad de catalogar y

hacer referencia a esta colección, que sirve como el punto de vista de los usuarios acerca del sistema. Cuando llegue el momento de actualizar el sistema, el catálogo de casos de uso funcionará como un fundamento para obtener los requerimientos de la actualización.

Representación de un modelo de caso de uso

Hay un actor que inicia un caso de uso y otro (posiblemente el que inició, pero no necesariamente) que recibirá algo de valor de él. La representación gráfica es directa. Una elipse representa a un caso de uso, una figura agregada representa a un actor. El actor que inicia se encuentra a la izquierda del caso de uso, y el que recibe a la derecha. El nombre del actor aparece justo debajo de él, y el nombre del caso de uso aparece ya sea dentro de la elipse o justo debajo de ella. Una línea asociativa conecta a un actor con el caso de uso, y representa la comunicación entre el actor y el caso de uso. La línea asociativa es sólida, como la que conecta a las clases asociadas.

Uno de los beneficios del análisis del caso de uso es que le muestra los confines entre el sistema y el mundo exterior. Generalmente, los actores están fuera del sistema, mientras que los casos de uso están dentro de él. Utilizará un rectángulo (con el nombre del sistema en algún lugar dentro de él) para representar el confín del sistema. El rectángulo envuelve a los casos de uso del sistema.

Término Nuevo Los actores, casos de uso y líneas de interconexión componen un modelo de caso de uso. La figura 7.1 le muestra estos símbolos.

Figura 7.1

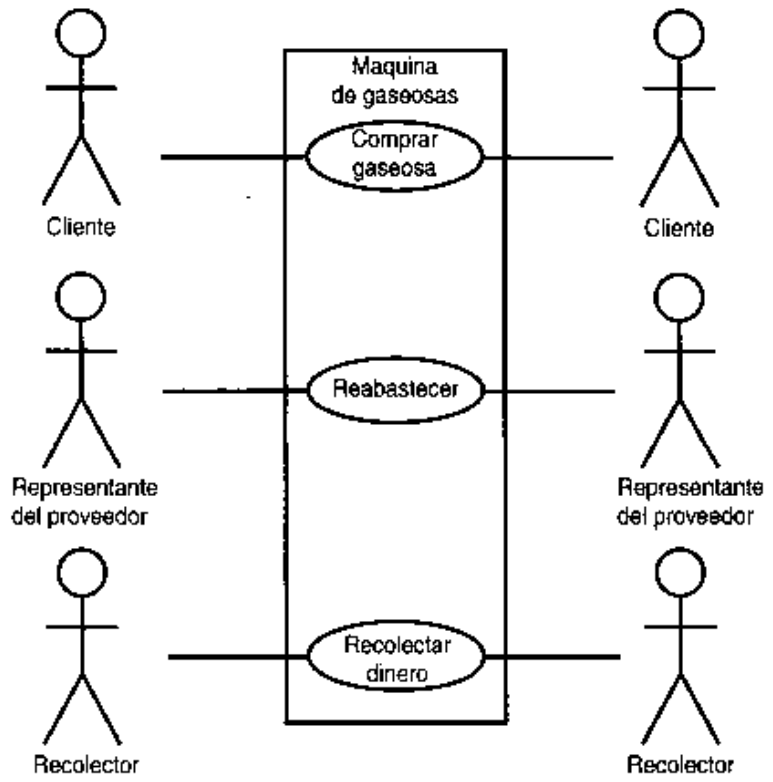
En un modelo de caso de uso, una figura agregada representa a un actor, una elipse a un caso de uso y una línea asociativa representa la comunicación entre el actor y el caso de uso.

Una nueva visita a la máquina de gaseosas

Apliquemos los símbolos al ejemplo de la hora anterior. Como recuerda, desarrolló los casos de uso para una máquina de gaseosas. El caso de uso "Comprar gaseosa" se encuentra dentro del sistema junto con "Reabastecer" y "Recolectar dinero". Los actores son el Cliente, Representante del proveedor y el Recolector. La figura 7.2 le muestra un modelo UML de caso de uso para una máquina de gaseosas.

FIGURA 7.2

Un modelo de caso de uso proveniente de la máquina de gaseosas de la hora 6.



Secuencia de pasos en los escenarios

Cada caso de uso es una colección de escenarios y cada escenario es una secuencia de pasos. Como puede ver, tales pasos no aparecen en el diagrama. No se encuentran en notas adjuntas a los casos de uso. Aunque el UML no lo prohíbe, la claridad es clave en la generación de cualquier diagrama y el adjuntar notas a cada caso de uso podría volverlo confuso. ¿Cómo y dónde haría la secuencia de pasos?

El uso de los diagramas de casos de uso será, por lo general, parte de un documento de diseño que el cliente y el equipo de diseño tomarán como referencia. Cada diagrama tendrá su propia página, de igual manera, cada escenario de caso de uso tendrá su propia página, donde se listará en modo de texto a:

El actor que inicia al caso de uso

Condiciones previas para el caso de uso

Pasos en el escenario

Condiciones posteriores cuando se finaliza el escenario

El actor que se beneficia del caso de uso

También puede enumerar las conjeturas del escenario (por ejemplo, que un cliente a la vez utilizará la máquina de gaseosas) y una breve descripción de una sola frase del escenario.

La hora 6, "Introducción a los casos de uso", presentó algunos escenarios alternativos del caso de uso "Comprar gaseosa". En su descripción, también podría poner estos escenarios de manera separada ("Sin el producto" y "Cambio incorrecto"), o podría considerarlos como excepciones al primer escenario del caso de uso. La forma exacta de hacerlo sólo le concernirá a usted, su cliente y los usuarios.

Para mostrar los pasos en un escenario, hay otra posibilidad que es utilizar un diagrama de actividades UML sobre el cual hablaremos en la hora 11, "Diagramas de actividades".

Concepción de las relaciones entre casos de uso

TERMINO NUEVO

El ejemplo de la hora 6 también mostró dos formas en que los casos de uso se pueden relacionar entre sí. Una de ellas, la *inclusión*, le permite volver a utilizar los pasos de un caso de uso dentro de otro. La otra, *extensión*, le permite crear un caso de uso mediante la adición de pasos a uno existente.

TERMINO NUEVO

Existen otros dos tipos de relaciones que son generalización y agrupamiento. Como en las clases, la *generalización* cuenta con un caso de uso que se hereda de otro. El *agrupamiento* es una manera sencilla de organizar los casos de uso.

Inclusión

Examinemos los casos de uso "Reabastecer" y "Recolectar dinero" del ejemplo de la hora 6. Ambos se inician mediante la apertura de la máquina, y finalizan con el cierre y sellado de la misma. El caso de uso "Exhibir el interior" se creó para capturar el primer par de pasos; y "Cubrir el interior" para el segundo. Tanto "Reabastecer", como "Recolectar dinero" incluyen este par de casos de uso.

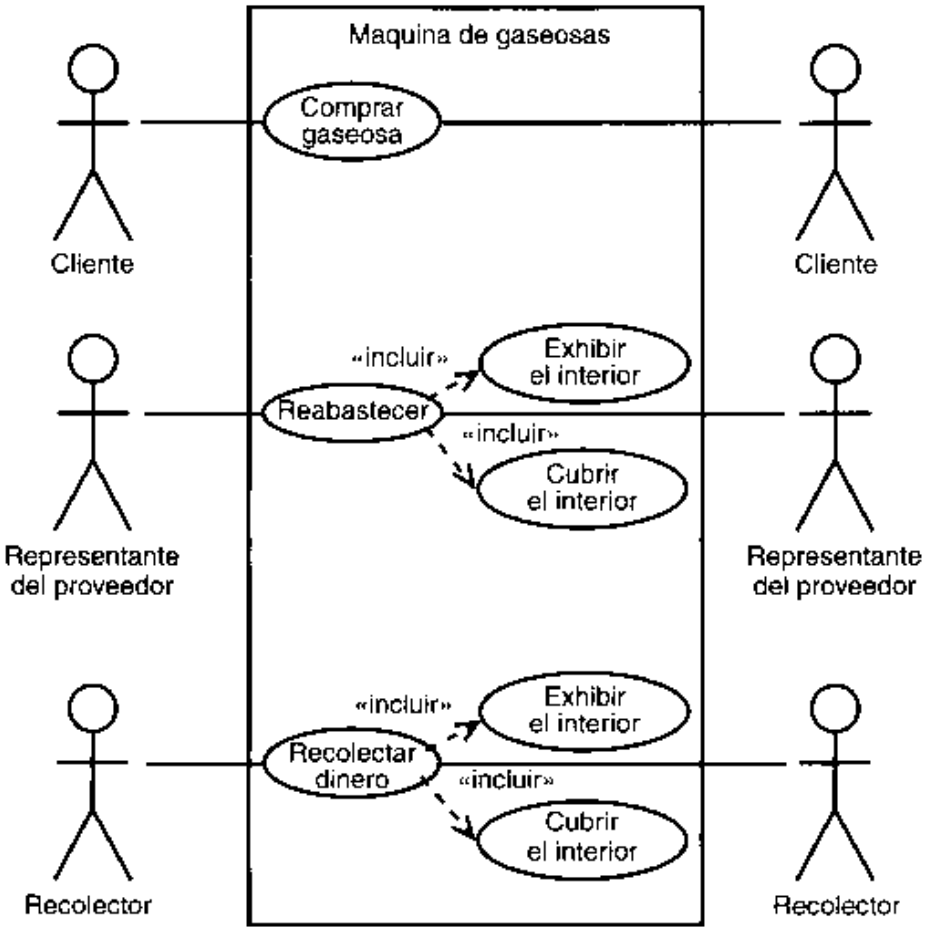
Para representar a la inclusión utilizará el símbolo que usó para la dependencia entre clases: una línea discontinua con una punta de flecha que conecta las clases apuntando hacia la clase dependiente. Justo sobre la línea, agregará un estereotipo: la palabra "incluir" bordeada por dos pares de paréntesis angulares. La figura 7.3 le muestra la relación de «inclusión» en el modelo de caso de uso de la máquina de gaseosas.

Tenga en cuenta que un caso de uso incluido nunca aparecerá solo. Simplemente funciona como parte de un caso de uso que lo incluya.

En la notación de texto que sigue los pasos en la secuencia, indicará los casos de uso incluidos. El primer paso en el caso de uso "Reabastecer" podría ser «incluir» (Exhibir el interior).

FIGURA 7.3

El modelo de caso de uso en la máquina de gaseosas con la inclusión.



Extensión

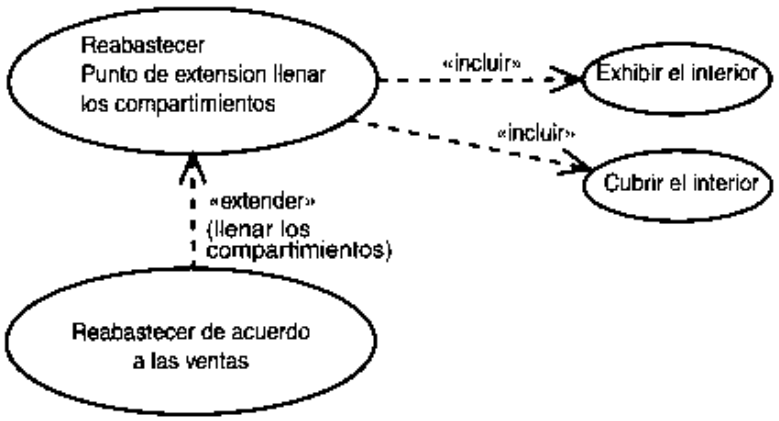
TERMINO NUEVO La hora 6 mostró que el caso de uso "Reabastecer" podría ser la base de otro caso de uso: "Reabastecer de acuerdo a las ventas". En lugar de sólo reabastecer la máquina de gaseosas para que todas las marcas tengan la misma cantidad de latas, el representante podría anotar aquellas que se venden mejor y reabastecer acorde con ello. Por lo que podemos decir que el nuevo caso de uso *extiende* al original dado que agrega otros pasos a la secuencia del caso de uso original, que se conoce como el caso de uso *base*.

TERMINO NUEVO La extensión sólo se puede realizar en puntos indicados de manera específica dentro de la secuencia del caso de uso base. A estos puntos se les conoce como *puntos de extensión*. En el caso de uso Reabastecer, los nuevos pasos (anotar las ventas y abastecer de manera acorde) se darían luego que el representante haya abierto la máquina y esté listo para llenar los compartimientos de las marcas de gaseosas. En este ejemplo, el punto de extensión es "Llenar los compartimientos".

Como la inclusión, podrá concebir la extensión con una línea de dependencia (línea discontinua con punta una punta de flecha), junto con un estereotipo que muestra "extender" entre paréntesis angulares. Dentro del caso de uso básico, el punto de extensión aparecerá debajo del nombre del caso de uso. La figura 7.4 le muestra la relación de extensión para "Reabastecer" y "Reabastecer de acuerdo a las ventas", junto con la inclusión de relaciones para "Reabastecer" y "Recolectar dinero".

FIGURA 7.4

Un diagrama de casos de uso que muestra la extensión y la inclusión.



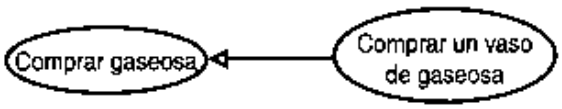
Generalización

Las clases pueden heredarse entre sí y esto también se aplica a los casos de uso. En la herencia de los casos de uso, el caso de uso secundario hereda las acciones y significado del primario, y además agrega sus propias acciones. Puede aplicar el caso de uso secundario en cualquier lugar donde aplique el primario.

En el ejemplo, deberá imaginar un caso de uso "Comprar un vaso de gaseosa" que se hereda de "Comprar gaseosa". El caso de uso secundario tiene acciones como "agregar hielo" y "mezclar marcas de gaseosas". Modelará la generalización de casos de uso de la misma forma que lo hace con las clases: con líneas continuas y una punta de flecha en forma de triángulo sin rellenar que apunta hacia el caso de uso primario, como se muestra en la figura 7.5.

FIGURA 7.5

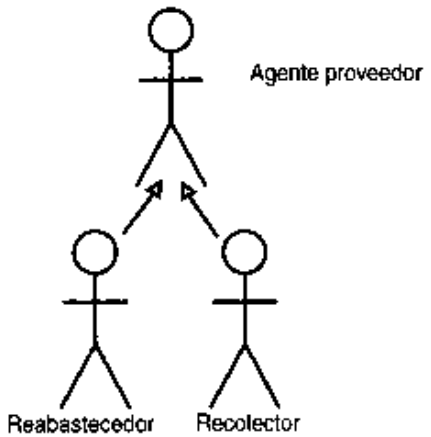
Un caso de uso puede heredar el sentido y comportamiento de otro.



La relación de generalización puede establecerse entre actores, así como entre casos de uso. Quizá tenga personificados al representante del proveedor, al recolector y al agente del proveedor. Si cambia el nombre del representante como Reabastecedor, tanto éste como el Recolector serán secundarios del Agente Proveedor, como muestra la figura 7.6.

FIGURA 7.6

Como las clases y los casos de uso, los actores pueden estar en una relación de generalización.



Agrupamiento

En ciertos diagramas de casos de uso, podría tener varios casos de uso que querrá organizar. Esto puede ocurrir cuando un sistema consta de varios subsistemas. Otra posibilidad sería cuando entrevista a los usuarios para obtener los requerimientos de un sistema. Cada requerimiento podría ser representado como un caso de uso por separado. Necesitará alguna forma de ordenar los requerimientos por categorías.

La forma más directa de organizar sería agrupar en un paquete los casos de uso que se relacionen. Recuerde que un paquete aparece como una carpeta tabular. Los casos de uso agrupados aparecerán dentro de la carpeta.

Diagramas de casos de uso en el proceso de análisis

Con el ejemplo dado y con el cual ha trabajado, aplicó directamente la simbología del caso de uso. Ahora nos regresaremos un poco y colocaremos los casos de uso en el contexto de un esfuerzo de análisis.

Las entrevistas al cliente deberán iniciar el proceso. Estas entrevistas producirán diagramas de clases que fungirán como las bases de su conocimiento para el dominio del sistema (el área en el cual resolverá los problemas). Una vez que conozca la terminología general del área del cliente, estará listo para hablar con los usuarios.

Las entrevistas con los usuarios comienzan en la terminología del dominio, aunque deberán alternarse hacia la terminología de los usuarios. Los resultados iniciales de las entrevistas deberán revelar a los actores y casos de uso de alto nivel que describirán los requerimientos funcionales en términos generales. Esta información establece los confines y ámbito del sistema.

Las entrevistas posteriores con los usuarios profundizarán en estos requerimientos, lo que dará por resultado modelos de casos de uso que mostrarán los escenarios y las secuencias detalladamente. Esto podría resultar en otros casos de uso que satisfagan las relaciones de inclusión y extensión. En esta fase, es importante confiar en su comprensión del dominio (a partir de los diagramas de clases derivados de las entrevistas con el cliente). Si no comprende adecuadamente el dominio, podría crear demasiados casos de uso y demasiados detalles (situación que podría, definitivamente, obstaculizar el diseño y el desarrollo).

Aplicación de los modelos de caso de uso

Para ayudarle a comprender con más profundidad los modelos de casos de uso y cómo aplicarlos, vamos a ver un ejemplo más complejo que una máquina de gaseosas. Suponga que deberá diseñar una red de área local (LAN) para una firma de consultoría, y que tendrá que comprender la funcionalidad para poder crearla. ¿Cómo empezaría?

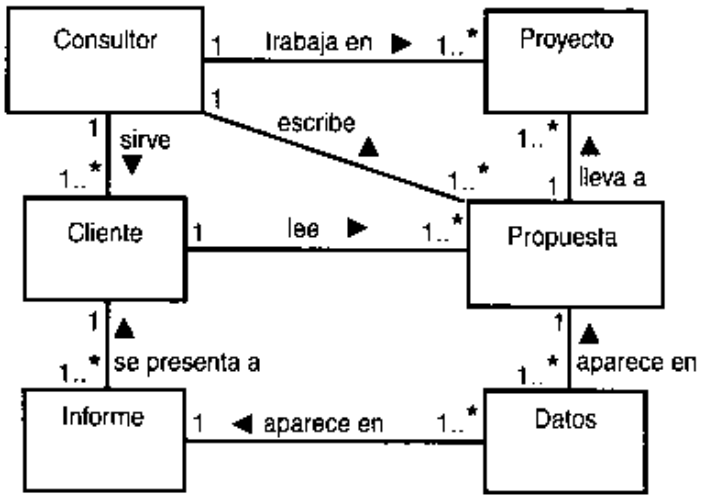
Una LAN es una red de comunicaciones que una organización utiliza en un ámbito limitado. Permite a los usuarios compartir recursos e información.

Comprensión del dominio

Empecemos con las entrevistas al cliente para crear un diagrama de clases que refleje cómo es la vida en el mundo de la consultoría. El diagrama de clases podría incluir las siguientes clases: Consultor, Cliente, Proyecto, Propuesta, Datos e Informe. La figura 7.7 le muestra la forma en que podría lucir el diagrama.

FIGURA 7.7

Un diagrama de clases para el mundo de la consultoría.



Comprensión de los usuarios

Ahora que el dominio está a la mano, vuelva su atención a los usuarios debido a que el objetivo será entender los tipos de funcionalidad por crear en el sistema.

En el mundo real, entrevistaría a los usuarios. En este ejemplo, basará sus ideas en cierto conocimiento general de las LANs y del dominio. No obstante, tenga presente que en el análisis de sistemas del mundo real, nada puede sustituir a las entrevistas con las personas.

Un grupo de usuarios serán consultores, otros podrían ser oficinistas. Entre otros usuarios en potencia se encontrarán funcionarios corporativos, vendedores, administradores de red, administradores de oficina y administradores de proyectos. (¿Se le ocurren otros?)

En este punto, sería conveniente a mostrar a los usuarios en una jerarquía de generalización, como se observa en la figura 7.8.

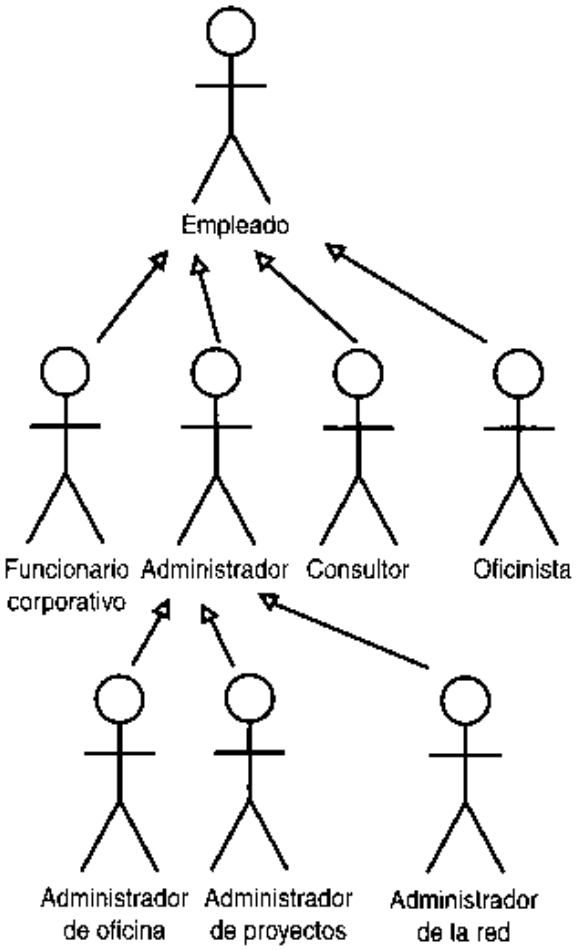
Comprensión de los casos de uso

¿Qué hay de los casos de uso? Hay algunas posibilidades: "Establecer niveles de seguridad", "Crear una propuesta", "Almacenar una propuesta", "Utilizar correo electrónico", "Compartir información de la base de datos", "Realizar la contabilidad", "Conectarse a la LAN desde fuera de ella", "Conectarse a Internet", "Compartir información de la base

de datos", "Indizar las propuestas", "Utilizar propuestas previas" y "Compartir impresoras". De acuerdo con esta información, la figura 7.9 le muestra el diagrama de casos de uso de alto nivel que hemos generado.

FIGURA 7.8

La jerarquía de usuarios que interactuarán con la LAN.



Este conjunto de casos de uso constituye los requerimientos funcionales de la LAN.

Profundización

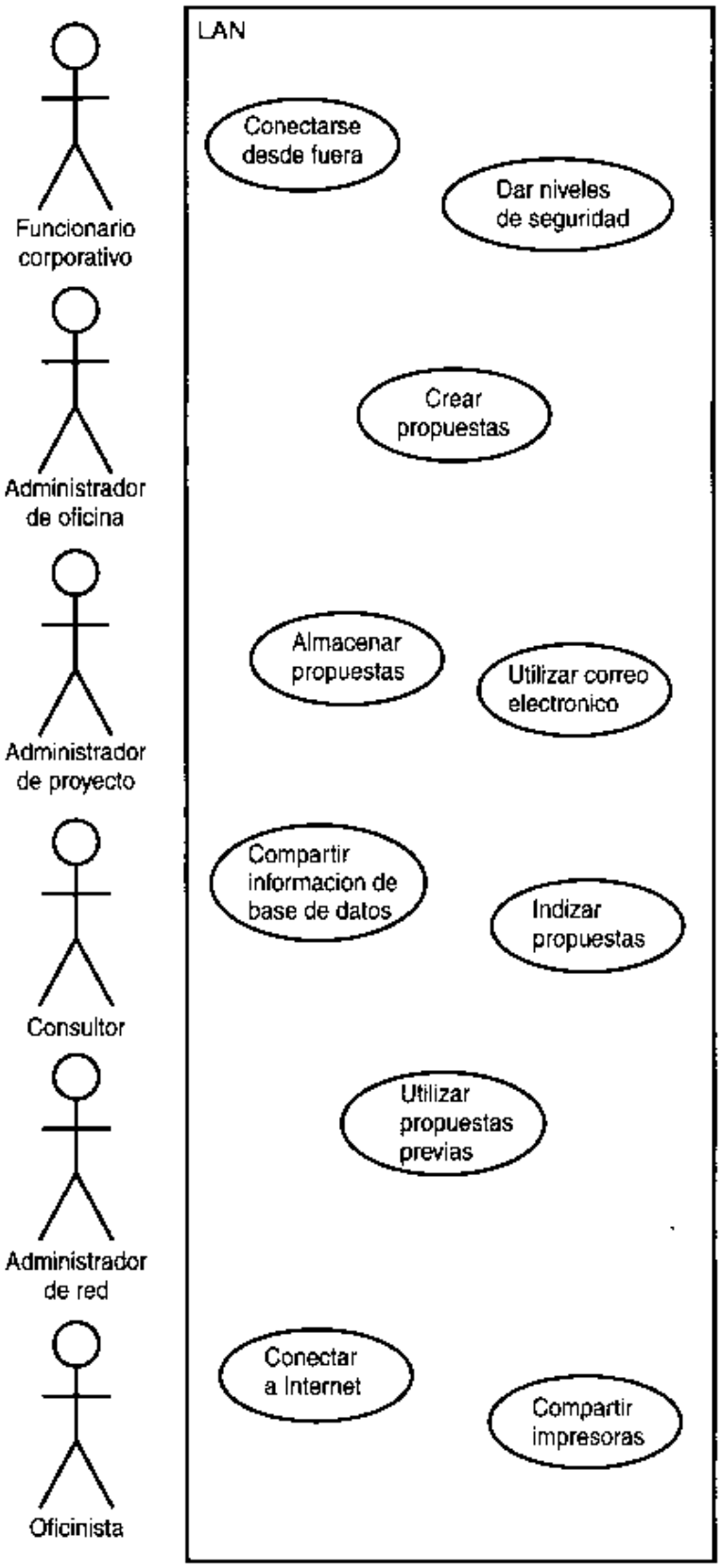
Elaboremos uno de los casos de uso de alto nivel y generemos un modelo de caso de uso. Una actividad extremadamente importante en una firma de consultoría es la generación de propuestas, así que examinemos el caso de uso "Crear una propuesta".

Las entrevistas con los consultores probablemente le indicarán cuántos pasos se necesitan en este caso de uso. Para empezar, el actor inicial es un consultor. El consultor tiene que iniciar una sesión en la LAN y ser verificado como usuario válido. Luego tendrá que utilizar algún software integrado para oficina (procesador de textos, hoja de cálculo y gráficos) para escribir la propuesta. En el proceso, el consultor podría volver a utilizar porciones de propuestas previas. La firma de consultoría podría tener una directiva de que un funcionario corporativo y otros dos consultores revisen una propuesta antes de que llegue a manos del cliente. Para ello, el consultor almacena la propuesta en un área central accesible mediante la LAN, y envía a los correos electrónicos de los tres revisores un mensaje que indique que la propuesta se encuentra lista, así como su ubicación.

Luego de recibir los comentarios y hacer las modificaciones necesarias (nuevamente, con el software integrado para oficina), el consultor imprime la propuesta y la envía por correo al cliente. Cuando todo termina, el consultor se retira de la red. El consultor habrá completado una propuesta y es el actor que se beneficia del caso de uso.

FIGURA 7.9

Un diagrama de casos de uso de alto nivel que representa una LAN para una firma de consultoría.



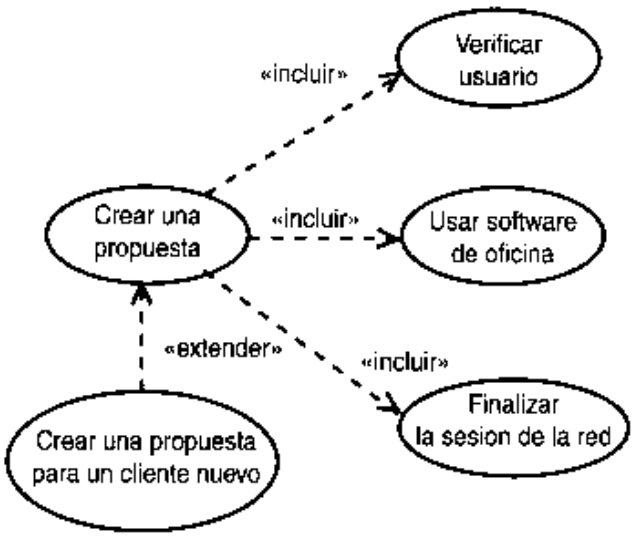
En la secuencia anterior, es claro que ciertos pasos se repetirán de un caso de uso a otro, y ello le llevará a otros casos de uso (posiblemente incluidos) en los que tal vez no había pensado. Iniciar una sesión y ser verificado son dos pasos que pueden incluir varios casos de uso. Por ello, creará un caso de uso "Verificar usuario" que incluya "Crear una propuesta". Otro par de casos de uso son "Utilizar software de oficina" y "Finalizar sesión de la red".

Podrían aparecer otros detalles del proceso de una propuesta cuando vea que las propuestas elaboradas para los clientes nuevos son diferentes a las de los clientes constantes. En sí, las propuestas a los nuevos clientes probablemente incluyen información promocional de la empresa. Con los clientes constantes, no será necesario enviar tal información. Así pues, otro nuevo caso de uso, "Crear una propuesta para un cliente nuevo" extenderá a "Crear una propuesta".

La figura 7.10 le muestra el diagrama de casos de uso que resulta de este análisis del caso de uso "Crear una propuesta".

FIGURA 7.10

El caso de uso "Crear una propuesta" en la LAN de una firma de consultoría.



Este ejemplo aterriza un punto importante, uno que había destacado anteriormente: El análisis del caso de uso describe el comportamiento de un sistema. No toca a la implementación. ¡Esto es particularmente importante en este caso, dado que el diseño de una LAN supera, por mucho, el alcance de este libro!

Dónde estamos

Este es un buen momento para ver la estructura general del UML dado que ya ha avanzado en dos de sus principales aspectos: la orientación a objetos y el análisis de casos de uso. Ha visto sus fundamentos y simbología, así como explorado algunas aplicaciones.

En las horas 2 a la 7 ha trabajado con:

Clases	Agregaciones
Objetos	Composiciones
Interfaces	Estereotipos
Casos de uso	Restricciones
Actores	Notas
Asociaciones	Paquetes
Generalizaciones	Extensiones
Realizaciones	Inclusiones

Intentemos dividir este conjunto de elementos en categorías.

Elementos estructurales

Las clases, objetos, actores, interfaces y casos de uso son cinco de los elementos estructurales en el UML. Aunque tienen diversas diferencias (mismas que, como ejercicio, deberá indicar), son similares en el sentido de que representan partes ya sea físicas o conceptuales de un modelo. Conforme avance en la parte I, verá otros elementos estructurales.

Relaciones

La asociación, generalización, dependencia y realización, son las relaciones en el UML. (La inclusión y extensión son dos tipos de dependencias.) Sin las relaciones, los modelos UML no serían más que listas de elementos estructurales. Las relaciones conectan a tales elementos y de ese modo conectan los modelos con la realidad.

Agrupamiento

El paquete es el único elemento de agrupamiento en el UML, éste le permite organizar los elementos estructurales en un modelo. Un paquete puede contener cualquier tipo de elemento estructural, y diferentes tipos a la vez.

Anotación

La nota es el elemento de anotación del UML; éstas le permiten adjuntar restricciones, comentarios, requerimientos y gráficos explicativos a sus modelos.

Extensión

Los estereotipos o clisés son dos estructuras que el UML proporciona para extender el lenguaje. Le permiten crear nuevos elementos además de los existentes, de modo que pueda modelar de forma adecuada la sección de realidad en la que se centrará su sistema.

...y más

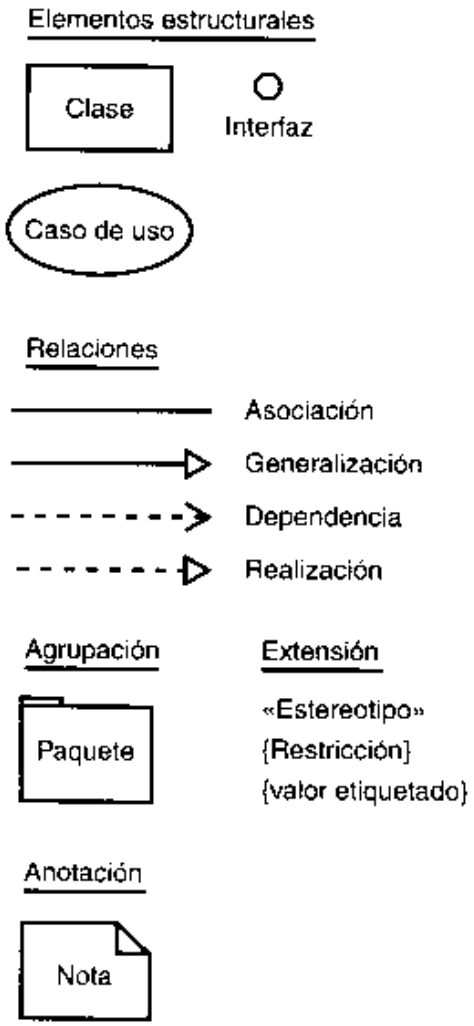
Además de los elementos estructurales, relaciones, agrupamientos, anotaciones y extensiones, el UML cuenta con otra categoría: elementos de comportamiento. Tales elementos le muestran la forma en que las partes de un modelo (como los objetos) cambian con el tiempo. Aún no sabe utilizarlos, pero los verá en la siguiente hora.

El Panorama

Ahora ya tiene una idea de la forma en que el UML se organiza. La figura 7.11 esquematiza esta organización por usted. Conforme vea las siguientes horas de la parte I, tenga esta organización en mente. Le hará adiciones conforme avance y este "panorama" le mostrará dónde agregar el nuevo conocimiento que adquiera.

FIGURA 7.11

La organización del UML, en términos de los elementos que ha utilizado hasta ahora.



Resumen

El caso de uso es una poderosa herramienta para obtener los requerimientos funcionales. Los diagramas de casos de uso agregan mayor poder: debido a que conciben los casos de uso, facilitan la comunicación entre los analistas y los usuarios, y entre los analistas y los clientes. En un diagrama, el símbolo del caso de uso es una elipse. El símbolo de un actor es una figura adjunta. Una línea asociativa conecta a un actor con el caso de uso. Los casos de uso están, por lo general, dentro de un rectángulo que representan el confín del sistema.

La inclusión se representa por una línea de dependencia con un estereotipo «incluir». La extensión se representa por una línea de dependencia con un estereotipo «extender». Las otras dos relaciones entre casos de uso son generalización, en la que un caso de uso hereda el sentido y acciones de otro, y el agrupamiento, mismo que organiza un conjunto de casos de uso. La generalización se representa por la misma línea que muestra la herencia entre clases. El agrupamiento se representa por el icono del paquete.

Los diagramas de casos de uso figuran con fuerza en el proceso de análisis. Se empieza con entrevistas con los clientes para obtener diagramas de clases. Éstos proporcionan una base para entrevistar a los usuarios. Tales entrevistas dan por resultado un diagrama de casos de uso de alto nivel que muestra los requerimientos funcionales del sistema. Para crear los modelos de caso de uso, profundice en cada caso de uso de alto nivel. Los diagramas resultantes de caso de uso darán los fundamentos para el diseño y desarrollo.

Ahora que ha visto la orientación a objetos y los casos de uso, está listo para ver el panorama del UML. Los elementos que ha aprendido de las horas 2 a 7 se encuentran en estas categorías: elementos estructurales, relaciones, organización, anotación y extensión. En la siguiente hora verá un elemento de la categoría restante: elementos de comportamiento. Tenga en mente este panorama para que se le facilite el aprendizaje del UML.

Preguntas y respuestas

P Me di cuenta que en el diagrama de casos de uso de alto nivel no mostró las asociaciones entre los actores y los casos de uso. ¿A qué se debe?

R El diagrama de casos de uso de alto nivel surge en las etapas iniciales de las entrevistas con los usuarios. En este punto, esto es más o menos un ejercicio de recopilación de ideas y el objetivo es encontrar los requerimientos generales, ámbito y confines del sistema. Las asociaciones tendrán mayor sentido cuando posteriores entrevistas con los clientes le lleven a profundizar en cada requerimiento y que los modelos de casos de uso tomen forma.

P ¿Por qué es importante tener en cuenta tal "panorama" del UML? ¿No bastaría con que sepa utilizar cada tipo de diagrama?

R Si usted comprende la organización del UML, podrá manejar situaciones que no haya encontrado antes. Podrá reconocer cuando un elemento UML existente no haga el trabajo, y sabrá cómo construir uno nuevo. También sabrá cómo crear un diagrama híbrido si llegara a ser la única forma de presentar claramente un modelo.

Taller

En este taller continuará con el conocimiento obtenido en la hora 6 y lo usará como base para el conocimiento de la hora 7. El objetivo es utilizar su nuevo conocimiento para concebir los casos de uso y sus relaciones. Las respuestas aparecen en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. Mencione dos ventajas de concebir un caso de uso.
2. Describa la generalización y el agrupamiento, las relaciones entre los casos de uso que ha visto durante esta hora. Mencione dos situaciones en las que usted agruparía los casos de uso.
3. ¿Cuáles son las similitudes entre las clases y los casos de uso? ¿Cuáles las diferencias?

Ejercicios

1. Bosqueje el diagrama de un modelo de caso de uso para un control remoto de una televisión. Asegúrese de incluir todas las funciones del control remoto como casos de uso para su modelo.
2. En el segundo ejercicio de la hora 6 indicó a los actores y casos de uso de un almacén de cómputo. Esta vez, dibuje un diagrama de casos de uso de alto nivel con base en el trabajo que realizó en tal ejercicio. Luego, genere un modelo de caso de uso para al menos uno de los casos de uso de alto nivel. En su trabajo, intente incorporar las relaciones «incluir» o «extender» que sean necesarias.

HORA 8

Diagramas de estados

Hasta ahora ha comprendido los importantes elementos estructurales del UML. Ahora verá un elemento que le muestra cómo modificar los procedimientos con el tiempo.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de estados
- Sucesos, acciones y condiciones de seguridad
- Subestados: secuenciales y concurrentes
- Estados históricos
- Por qué son importantes los diagramas de estados
- Adición del diagrama de estados al panorama del UML

TÉRMINO NUEVO

Al finalizar la hora anterior, dije que aquí trataría una nueva categoría de elementos con la cual no había trabajado, el elemento de comportamiento, éste muestra la forma en que las partes de un modelo UML cambian con el tiempo. Verá un miembro en particular de esta categoría, el diagrama de estados.

Cada año trae consigo nuevos estilos en ropas y automóviles, las estaciones cambian el color de las hojas de los árboles y cada año que pasa deja entrever el crecimiento y madurez de los niños. Al pasar el tiempo y conforme suceden las cosas, hay cambios que afectan a los objetos que nos rodean.

Esto también se aplica en cualquier sistema. Conforme el sistema interactúa con los usuarios y (posiblemente) con otros sistemas, los objetos que lo conforman pasarán por cambios necesarios para ajustar las interacciones. Si va a modelar sistemas, necesitará contar con un mecanismo para los cambios en el modelo.

Qué es un diagrama de estados

Una manera para caracterizar un cambio en un sistema es decir que los objetos que lo componen modificaron su *estado* como respuesta a los sucesos y al tiempo. He aquí algunos ejemplos rápidos:

Cuando acciona el interruptor, la fuente de luz cambia su estado de apagada a encendida.

Cuando presiona un botón de un control remoto, una televisión cambia su estado para mostrarle un canal u otro.

Luego de un lapso adecuado, una lavadora cambia su estado de "lavar" a "enjuagar".

TERMINO NUEVO El *diagrama de estados* UML captura este tipo de cambios. Presenta los estados en los que puede encontrarse un objeto junto con las transiciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado.

TERMINO NUEVO Un diagrama de estados también se conoce como un *motor de estado*.

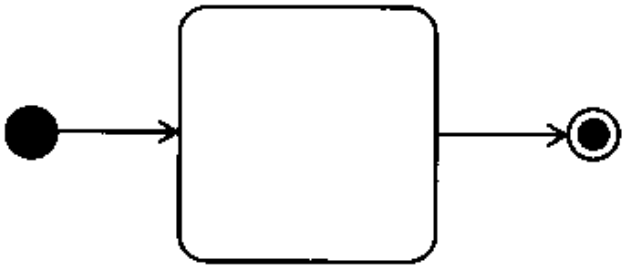
Tenga en cuenta que un diagrama de estados es intrínsecamente distinto, de manera muy significativa, de uno de clase, de objeto o de un caso de uso. Los diagramas que ya ha visto modelan el comportamiento de un sistema, o al menos un grupo de clases, objetos o casos de uso. Un diagrama de estados muestra las condiciones de un solo objeto.

Simbología

La figura 8.1 le muestra el rectángulo de vértices redondeados que representa a un estado, junto con una línea continua y una punta de flecha, mismas que representan a una transición. La punta de la flecha apunta hacia el estado donde se hará la transición. La figura también muestra un círculo relleno que simboliza un punto inicial y la diana que representa a un punto final.

FIGURA 8.1

Los símbolos UML en un diagrama de estados. El icono para el estado es un rectángulo de vértices redondeados, y el símbolo de una transición es una línea continua y una punta de flecha. Un círculo relleno se interpreta como el punto inicial de una secuencia de estados, y una diana representa al punto final.



Adición de detalles al icono de estado

El UML le da la opción de agregar detalles a la simbología. Así como es posible dividir un símbolo de clase en tres áreas (nombre, atributos y operaciones), puede dividir el icono de estado de igual forma. El área superior contendrá el nombre del estado (que tiene que establecer ya sea que haya la subdivisión o no), el área central contendrá las variables de estado, y el área inferior las actividades. La figura 8.2 le muestra estos detalles.

FIGURA 8.2

Puede subdividir el símbolo del estado en áreas que muestren el nombre, variables y actividades del estado.

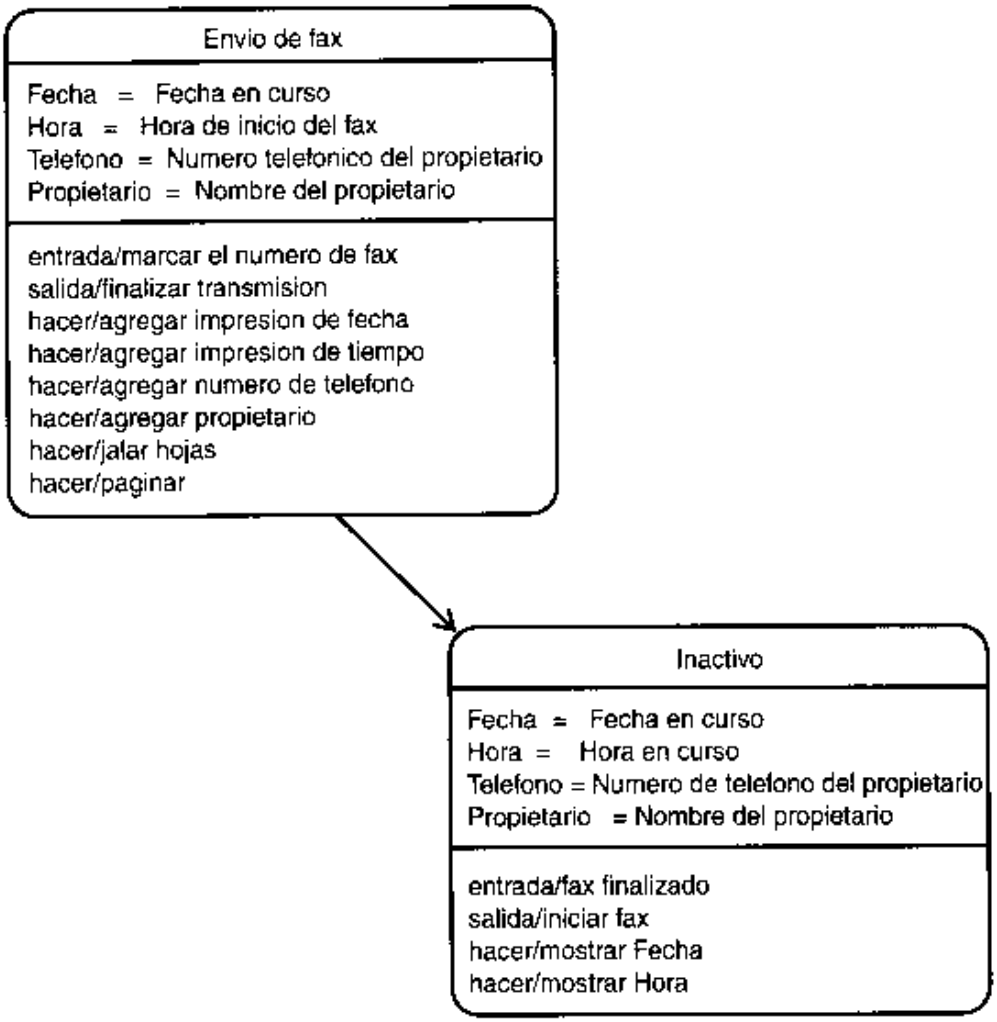
Las variables de estado como cronómetros o contadores son, en ocasiones, de ayuda. Las actividades constan de sucesos y acciones: tres de las más utilizadas son *entrada* (qué sucede cuando el sistema entra al estado), *salida* (qué sucede cuando el sistema sale del estado), y *hacer* (qué sucede cuando el sistema está en el estado). Puede agregar otros conforme sea necesario.

Un máquina de fax sirve como ejemplo de un objeto que puede pasar por diversas variables y actividades de estado. Cuando se envía un fax —esto es, cuando se encuentra en estado de envío de fax— la máquina de fax anota la fecha y hora en que inició el envío (los valores de las variables de estado "fecha" y "hora"), y también anota su número telefónico así como el nombre del propietario (los valores de las variables de estado "teléfono" y "propietario"). Al encontrarse en este estado, la máquina se encarga de agregar un registro de fecha y hora al fax, número telefónico y nombre del propietario. En otras actividades de este estado, la máquina jalará las hojas, paginará el fax y finalizará la transmisión.

Mientras se encuentre en el estado de inactividad, la máquina de fax mostrará la fecha y la hora en una pantalla. La figura 8.3 le muestra el diagrama de estados.

FIGURA 8.3

La máquina de fax es un buen ejemplo de un estado con variables y actividades.



Sucesos y acciones

TÉRMINO NUEVO

También puede agregar ciertos detalles a las líneas de transición. Puede indicar un suceso que provoque una transición (*desencadenar un suceso*), y la actividad de cómputo (*la acción*) que se ejecute y haga que suceda la modificación del estado. A los sucesos y acciones los escribirá cerca de la línea de transición mediante una diagonal para separar un suceso desencadenado de una acción. En ocasiones un evento causará una transición sin una acción asociada, y algunas veces una transición sucederá dado que un estado finalizará una actividad (en lugar de hacerlo por un suceso). A este tipo de transición se le conoce como *transición no desencadenada*. La GUI (interfaz gráfica de usuario) con que interactúe le dará ejemplos de detalles de la transición. Por el momento, asumamos que la GUI puede establecerse en uno de tres estados:

Inicialización

Operación

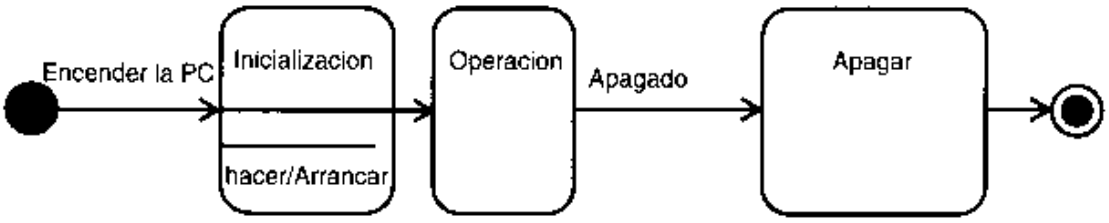
Apagar

Cuando encienda su equipo, se ejecutará un proceso de arranque. Al encender la PC se desencadena un suceso que provoca que la GUI aparezca luego de una transición desde el estado de Inicialización, y el arranque es una acción que se realiza durante tal transición.

Como resultado de las actividades en el estado de inicialización, la GUI entra al modo de Operación. Cuando desea apagar su PC, desencadena un suceso que provoca la transición hacia el estado de Apagado, y con ello la PC se apaga. La figura 8.4 muestra el diagrama de estados que captura los estados y transiciones en la GUI.

FIGURA 8.4

Los estados y transiciones de una interfaz gráfica del usuario incluyen el desencadenamiento de eventos, acciones y transiciones no desencadenadas.



Condiciones de seguridad

La anterior secuencia de estados de la GUI deja mucho que desear. Por ejemplo: si deja solo su equipo o si realizara alguna actividad en la que no tocará al ratón o al teclado, podría aparecer un protector de pantallas que evitaría el desgaste de su pantalla. Para decir esto en términos de cambio de estado, si ha pasado cierto tiempo sin que haya interacción con el usuario, la GUI hará una transición del estado Operación a un estado que no aparece en la figura 8.4: el estado de Protector de pantallas.

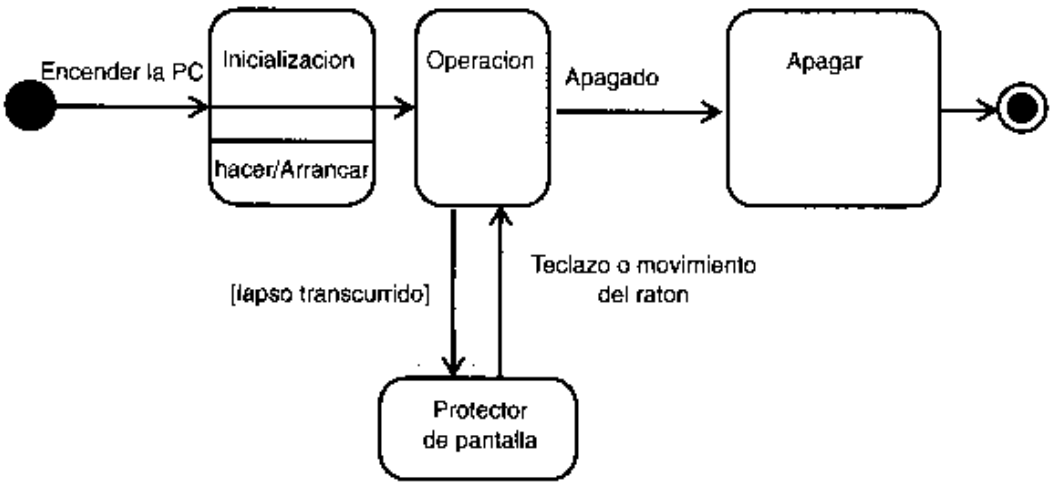
El intervalo se especifica en el panel de control de su sistema operativo (Windows en este caso). Por lo general es de 15 minutos. Cualquier opresión de una tecla o movimiento del ratón provocará una transición del estado Protector de pantallas al estado Operación.

TÉRMINO NUEVO

El intervalo de 15 minutos es una *condición de seguridad*: cuando se llega a ella, se realiza la transición. La figura 8.5 muestra el diagrama de estados de la GUI con el estado Protector de pantalla y la condición de seguridad añadida. Vea que la condición de seguridad se establece como expresión booleana.

FIGURA 8.5

El diagrama de estados para la GUI, con el estado Protector de pantalla y la condición de seguridad.



Subestados

Nuestro modelo de la GUI aún está algo vacío. El estado Operación, en particular, es mucho más sustancioso de lo que he indicado en las figuras 8.4 y 8.5.

Cuando la GUI está en el estado Operación, hay muchas cosas que ocurren tras bambalinas, aunque no sean particularmente evidentes en la pantalla. La GUI aguarda de forma constante a que usted haga algo (oprimir una tecla, mover el ratón u oprimir uno de sus botones). Luego, deberá registrar tales acciones y modificar lo que se despliega para reflejarlas en la pantalla (como mover el cursor cuando usted mueva el ratón, o mostrar una "a" cuando usted oprima la tecla "a").

TÉRMINO NUEVO

Con ello la GUI atravesará por varios cambios mientras se encuentre en el estado Operación. Tales cambios serán cambios de estado. Dado que estos estados se encuentran dentro de otros, se conocerán como *subestados*. Hay dos tipos de subestados: *secuencial* y *concurrente*.

Subestados secuenciales

Como su nombre lo indica, los subestados secuenciales suceden uno detrás de otro. Si retomamos los subestados mencionados con anterioridad dentro del estado Operación de la GUI, tendrá la siguiente secuencia:

A la espera de acción del usuario

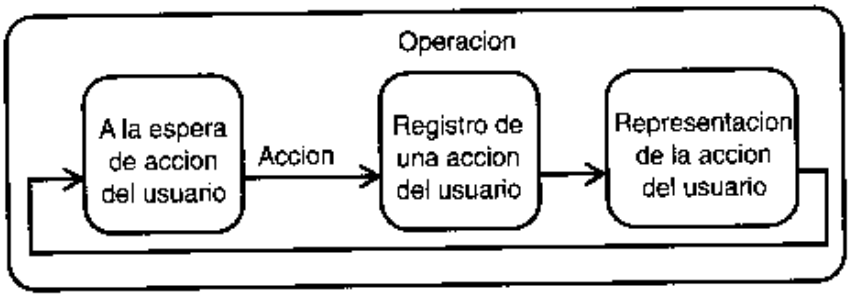
Registro de una acción del usuario

Representación de la acción del usuario

La acción del usuario desencadena la transición a partir de A la espera de acción del usuario hacia Registro de una acción del usuario. Las actividades dentro del Registro trascienden de la GUI hacia la Representación de la acción del usuario. Después del tercer estado, la GUI vuelve a iniciar A la espera de acción del usuario. La figura 8.6 le muestra cómo representar los subestados secuenciales dentro del estado Operación.

FIGURA 8.6

Subestados secuenciales dentro del estado Operación de la GUI.



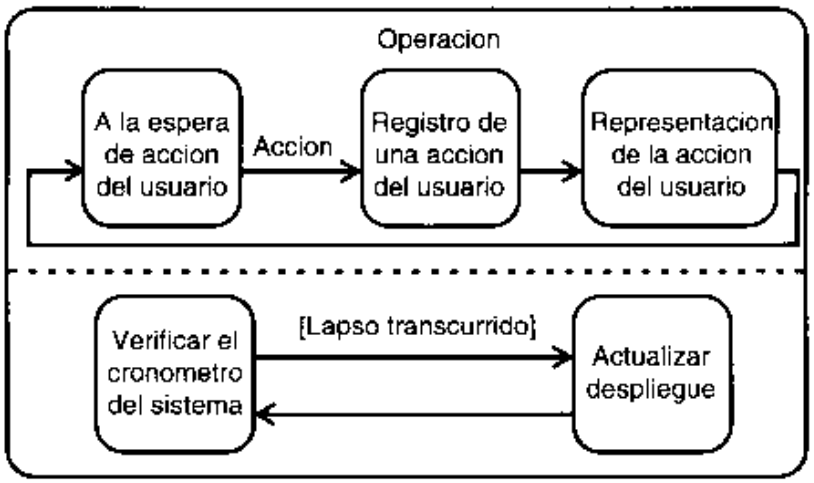
Subestados concurrentes

Dentro del estado Operación, la GUI no sólo aguarda a que usted haga algo. También verifica el cronómetro del sistema y (posiblemente) actualiza el despliegue de una aplicación luego de un intervalo específico. Por ejemplo, una aplicación podría incluir un reloj en pantalla que tuviera que actualizar la GUI.

Todo esto sucede al mismo tiempo que la secuencia que ya indiqué. Aunque cada secuencia es, claro, un conjunto de subestados secuenciales, las dos secuencias son concurrentes entre sí. Puede representar la concurrencia con una línea discontinua entre los estados concurrentes, como en la figura 8.7.

FIGURA 8.7

Los subestados concurrentes suceden al mismo tiempo. Una línea discontinua los separa.



TÉRMINO NUEVO

La separación del estado Operación en dos componentes podría recordarle algo. ¿Recuerda cuando traté el tema de las adiciones y composiciones? Cuando cada componente sea parte de un "todo", tratará con una composición. Las partes concurrentes del estado Operación tienen el mismo tipo de relación con él. Por ello, el estado Operación es un *estado compuesto*. Un estado que consta sólo de subestados secuenciales, también es un estado compuesto.

Estados históricos

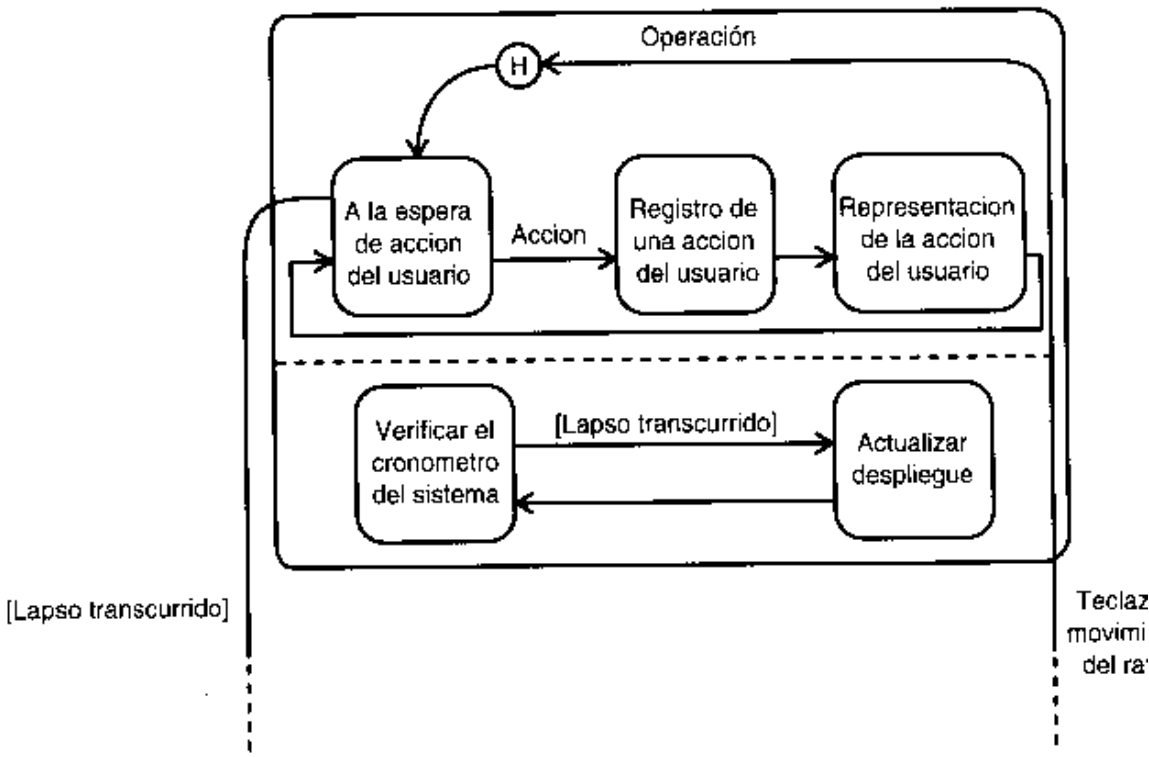
Cuando se activa su protector de pantallas y mueve su ratón para regresar al estado Operación ¿qué ocurre? ¿Acaso su pantalla retoma el estado inicial, como si apenas se hubiera encendido? ¿O lucirá tal como la dejó antes de que se activara el protector de pantallas?

Es obvio, si el protector de pantallas provocara que la pantalla regresara a su estado inicial de operación, la idea del protector de pantallas sería contraproducente. Los usuarios podrían perder su trabajo y tendrían que reiniciar una sesión nuevamente.

El diagrama de estados históricos captura esta idea. El UML proporciona un símbolo que muestra que un estado compuesto recuerda su subestado activo cuando el objeto trasciende fuera del estado compuesto. El símbolo es la letra "H" encerrada en un círculo que se conecta por una línea continua al subestado por recordar, con una punta de flecha que apunta a tal subestado. La figura 8.8 muestra este símbolo en el estado Operación.

FIGURA 8.8

El estado histórico, simbolizado con la "H" dentro del círculo, le muestra que un estado compuesto recuerda su subestado activo cuando el objeto trasciende fuera de tal estado compuesto.



TÉRMINO NUEVO En el diagrama de estados no he tratado con las ventanas que están abiertas por otras ventanas (es decir, con subestados anidados en otros). Cuando un estado histórico recuerda los subestados en todos los niveles de anidación (como el estado Operación de Windows lo hace), el estado histórico es *profundo*. Si sólo recuerda el subestado principal, el estado histórico será *superficial*. Un estado histórico profundo se representa agregando un asterisco (*) a la "H" en el círculo (H*).

TÉRMINO NUEVO El estado histórico y el estado inicial (representados por el círculo relleno) son conocidos como *pseudoestados*. No tienen variables de estados ni actividades, por lo que no son estados "completos".

Mensajes y señales

En el ejemplo, el suceso desencadenado que provocó la transición de Protector de pantalla a Operación es la opresión de una tecla, un movimiento del ratón o una opresión de uno de sus botones. Cualquiera de estos sucesos es, en efecto, un mensaje del usuario a la GUI. Esto es un concepto importante dado que los objetos se comunican mediante el envío de mensajes entre sí. En este caso, el suceso desencadenado es un mensaje de un objeto (el usuario) a otro (la GUI).

TÉRMINO NUEVO Un mensaje que desencadena una transición en el diagrama de estados del objeto receptor se conoce como *señal*. En el mundo de la orientación a objetos, el envío de una señal es lo mismo que crear un objeto Señal y transmitirlo al objeto receptor. El objeto Señal cuenta con propiedades que se representan como atributos. Dado que una señal es un objeto, es posible crear jerarquías de herencia de señales.

Por qué son importantes los diagramas de estados

El diagrama de estados del UML proporciona una gran variedad de símbolos y abarca varias ideas (todas para modelar los cambios por los que pasa un objeto). Este tipo de diagrama tiene el potencial de convertirse en algo complejo con pasmosa rapidez. ¿En verdad es necesario?

De hecho, así es. Es necesario contar con diagramas de estados dado que permiten a los analistas, diseñadores y desarrolladores comprender el comportamiento de los objetos de un sistema. Un diagrama de clases y el diagrama de objetos correspondiente sólo muestra los aspectos estáticos de un sistema. Muestran las jerarquías y asociaciones, y le indican qué son las operaciones. Pero no le muestran los detalles dinámicos de las operaciones.

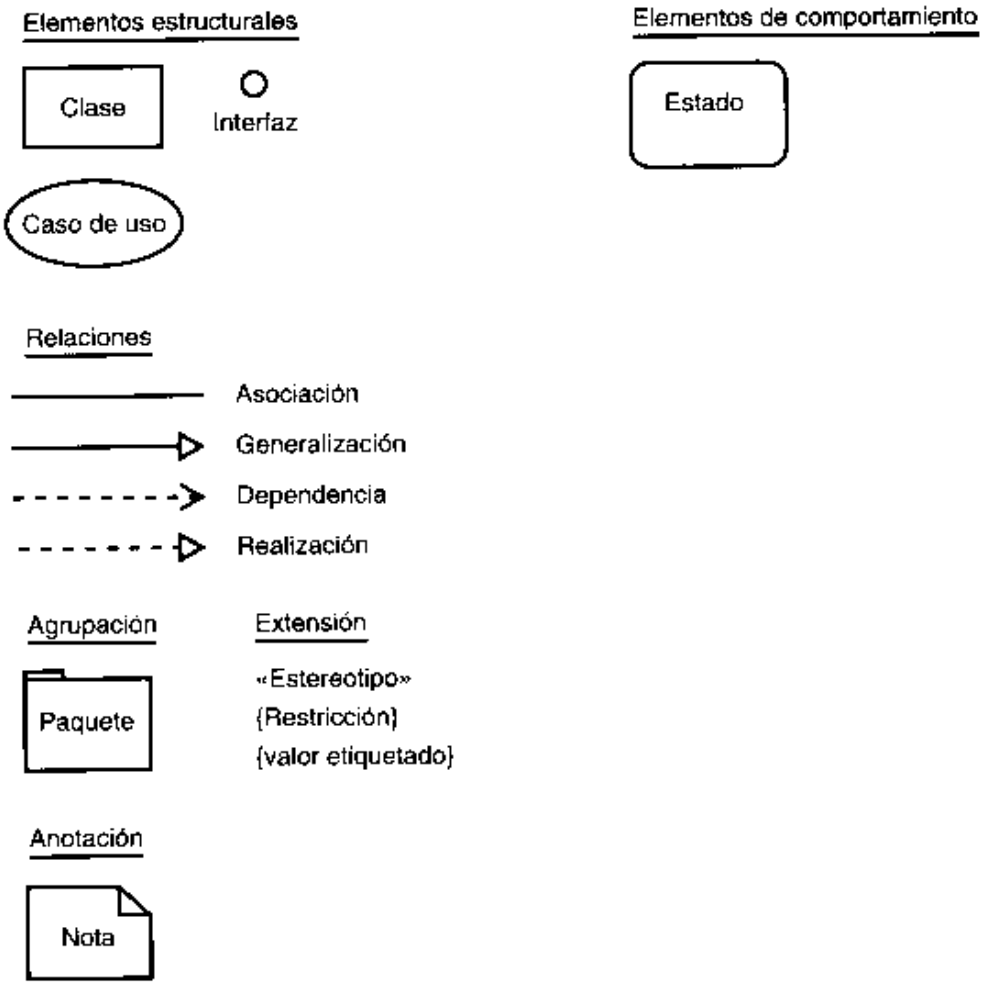
Los desarrolladores, en particular, deben saber la forma en que los objetos se supone que se comportarán, ya que son ellos quienes tendrán que establecer tales comportamientos en el software. No es suficiente con implementar un objeto: los desarrolladores deben hacer que tal objeto haga algo. Los diagramas de estados se aseguran que no tendrán que adivinar lo que se supone que harán los objetos. Con una clara representación del comportamiento del objeto, aumenta la probabilidad de que el equipo de desarrollo produzca un sistema que cumpla con los requerimientos.

Adiciones al panorama

Ahora puede agregar los "Elementos de comportamiento" al panorama del UML. La figura 8.9 le muestra la imagen con el diagrama de estados incluido.

FIGURA 8.9

El panorama del UML ahora incluye un elemento de comportamiento: el diagrama de estados. La organización del UML, en términos de los elementos que ha utilizado hasta ahora.



Resumen

Los objetos en los sistemas modifican sus estados como respuestas a sucesos y al tiempo. El diagrama de estados de UML captura estos cambios de estado. Un diagrama de estados se enfoca en los cambios de estado en un solo objeto. Un rectángulo de vértices redondeados representa a un estado, y una línea continua con una punta de flecha representa una transición de un estado a otro.

El símbolo del estado contiene el nombre del mismo y puede tener variables y actividades del estado. Una transición puede suceder como respuesta a un suceso desencadenado, e implicar una respuesta o acción. Una transición también puede ocurrir por la actividad en un estado: una transición que ocurre de esta forma se conoce como *transición no desencadenada*. Finalmente, una transición puede ocurrir cuando se cumple una condición particular, o *condición de seguridad*.

En ocasiones, un estado consta de subestados. Los subestados pueden ser secuenciales (ocurrir uno después del otro) o concurrentes (ocurrir al mismo tiempo). Un estado que consta de subestados se conoce como estado compuesto. Un estado histórico indica que un estado compuesto recordará su subestado cuando el objeto trascienda de este estado compuesto. Un estado histórico puede ser superficial o profundo. Tales términos son propios de los subestados anidados. Un estado histórico *superficial* recuerda sólo el subestado principal. Un estado histórico *profundo* recuerda todos los niveles de los subestados.

Cuando un objeto envía un mensaje que desencadena una transición en el diagrama de estados de otro objeto, tal mensaje es una *señal*. Una señal es, por sí misma, un objeto, y podrá crear una jerarquía de herencia de las señales.

Es necesario contar con los diagramas de estados porque facilitan la comprensión de los objetos de un sistema a los analistas, diseñadores y desarrolladores. Los desarrolladores, en particular, deben saber cómo se supone que se comportarán los objetos, dado que serán quienes tengan que establecer estos comportamientos en el software. No es suficiente implementar un objeto: los desarrolladores tienen que hacer que tal objeto haga algo.

Preguntas y respuestas

P ¿Cuál es la mejor manera de empezar a crear un diagrama de estados?

R Es muy parecido a crear un diagrama de clases o un modelo de caso de uso. En el diagrama de clases, listará todas las clases y luego realizará las asociaciones entre ellas. En el diagrama de estados, primero listará los estados del objeto, y luego se enfocará en las transiciones. Conforme avance en cada transición, deberá prever si un suceso desencadenado lo activará y si se realizará alguna acción.

P ¿Cada diagrama de estados debe tener un estado final (el que se representa por la diana)?

R No. Un objeto que nunca queda inactivo jamás tendrá un estado final.

P ¿Alguna sugerencia para diseñar un diagrama de estados?

R Intente arreglar los estados y transiciones para minimizar el cruzamiento de líneas. Uno de los objetivos de este diagrama (y de cualquier otro) se centra en la claridad. Si las personas no pueden comprender los modelos que cree, nadie los usará y sus esfuerzos (no importa qué tan minuciosos hayan sido) habrán sido infructuosos.

Taller

El cuestionario y los ejercicios le harán trascender al estado "Aprendizaje de diagramas de estados". Como siempre, encontrará las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionarios

1. ¿De qué forma difiere un diagrama de estados de uno de clases, de objetos o de caso de uso?
2. Defina los siguientes términos: *transición*, *suceso* y *acción*.
3. ¿Qué es una *transición no desencadenada*?
4. ¿Cuál es la diferencia entre los subestados secuenciales y los concurrentes?

Ejercicios

1. Suponga que diseñará un tostador. Cree el diagrama de estados que controle los estados del pan en el tostador. Incluya los sucesos desencadenados, acciones y condiciones de seguridad necesarios.
2. Cada vez que un objeto envíe una señal, se creará un objeto Señal y será transmitido. En Windows, hay varias señales posibles a partir de la GUI. Suponga que la señal (el tipo de señal que envíe a Windows) sea una clase. (¿Qué tipo de clase es?) Cree un diagrama de clases de las posibles señales y muestre toda la herencia inherente.
3. La figura 8.7 le muestra los subestados concurrentes dentro del estado Operación de la GUI. Dibuje un diagrama del estado Protector de pantalla que incluya los subestados concurrentes.

HORA 9

Diagramas de secuencias

Los diagramas de estados se enfocan a los diferentes estados de un objeto. Esto es sólo una pequeña parte del cuadro. El diagrama de secuencias del UML establece el siguiente paso y le muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de secuencias
- La GUI
- Diagramas de instancias y diagramas genéricos
- Uso de "si" y "mientras"
- Creación de un objeto en la secuencia
- Representación de la recursividad
- Diagramas de secuencias en el panorama del UML

Los diagramas de estados que vio en la hora anterior se centran en un objeto y muestran los cambios por los que pasa dicho objeto.

El UML le permite expandir su campo de visión y le muestra la forma en que un objeto interacciona con otros. En este campo de visión expandido, incluirá una importante dimensión: el tiempo. La idea primordial es que las interacciones entre los objetos se realizan en una secuencia establecida y que la secuencia se toma su tiempo en ir del principio al fin. Al momento de crear un sistema tendrá que especificar la secuencia, y para ello, utilizará al diagrama correspondiente.

Qué es un diagrama de secuencias

TÉRMINO NUEVO

El *diagrama de secuencias* consta de objetos que se representan del modo usual: rectángulos con nombre (subrayado), mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical.

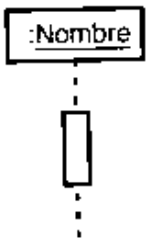
Objetos

TÉRMINO NUEVO

Los objetos se colocan cerca de la parte superior del diagrama de izquierda a derecha y se acomodan de manera que simplifiquen al diagrama. La extensión que está debajo (y en forma descendente) de cada objeto será una línea discontinua conocida como la *línea de vida* de un objeto. Junto con la línea de vida de un objeto se encuentra un pequeño rectángulo conocido como *activación*, el cual representa la ejecución de una operación que realiza el objeto. La longitud del rectángulo se interpreta como la duración de la activación. La figura 9.1 muestra un objeto, su línea de vida y su activación.

FIGURA 9.1

Representación de un objeto en un diagrama de secuencias.



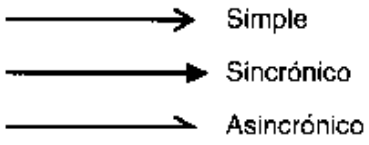
Mensaje

Un mensaje que va de un objeto a otro pasa de la línea de vida de un objeto a la de otro. Un objeto puede enviarse un mensaje a sí mismo (es decir, desde su línea de vida hacia su propia línea de vida).

Un mensaje puede ser *simple*, *sincrónico*, o *asincrónico*. Un mensaje simple es la transferencia del control de un objeto a otro. Si un objeto envía un mensaje sincrónico, esperará la respuesta a tal mensaje antes de continuar con su trabajo. Si un objeto envía un mensaje asincrónico, no esperará una respuesta antes de continuar. En el diagrama de secuencias, los símbolos del mensaje varían, por ejemplo, la punta de la flecha de un mensaje simple está formada por dos líneas, la punta de la flecha de un mensaje sincrónico está rellena y la de un asincrónico tiene una sola línea, como se aprecia en la figura 9.2.

FIGURA 9.2

Símbolos para los mensajes en un diagrama de secuencias.



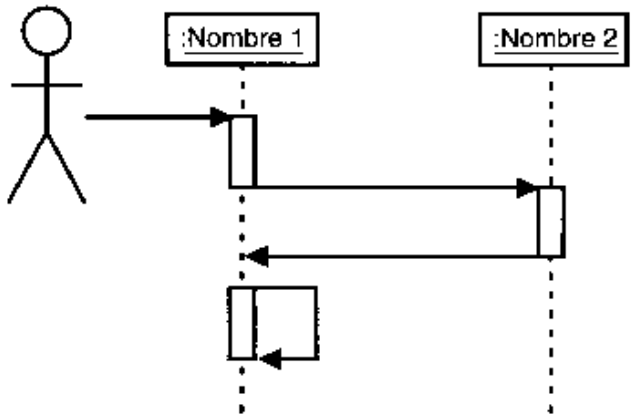
Tiempo

El diagrama representa al tiempo en dirección vertical. El tiempo se inicia en la parte superior y avanza hacia la parte inferior. Un mensaje que esté más cerca de la parte superior ocurrirá antes que uno que esté cerca de la parte inferior.

Con ello, el diagrama de secuencias tiene dos dimensiones. La dimensión horizontal es la disposición de los objetos, y la dimensión vertical muestra el paso del tiempo. La figura 9.3 muestra al conjunto básico de símbolos del diagrama de secuencias, con los símbolos en funcionamiento conjunto. La figura muestra a un actor que inicia la secuencia, aunque, en sentido estricto, la figura adjunta no es parte del conjunto de símbolos del diagrama de secuencias.

FIGURA 9.3

En un diagrama de secuencias los objetos se colocan de izquierda a derecha en la parte superior. Cada línea de vida de un objeto es una línea discontinua que se desplaza hacia abajo del objeto. Una línea continua con una punta de flecha conecta a una línea de vida con otra, y representa un mensaje de un objeto a otro. El tiempo se inicia en la parte superior y continúa hacia abajo. Aunque un actor es el que normalmente inicia la secuencia, su símbolo no es parte del conjunto de símbolos del diagrama de secuencias.



Para ver en acción a esta importante herramienta del UML, apliquémosla en los ejemplos que hemos visto en las horas anteriores. Cada aplicación le mostrará algunos conceptos importantes que se relacionan con los diagramas de secuencias.

La GUI

En la hora anterior desarrolló los diagramas de estados que muestran los cambios por los que pasa una GUI. Ahora dibujará un diagrama de secuencias que represente las interactividades de la GUI con otros objetos.

La secuencia

Suponga que el usuario de una GUI presiona una tecla alfanumérica; si asumimos que utiliza una aplicación como un procesador de textos, el carácter correspondiente deberá aparecer de inmediato en la pantalla. ¿Qué ocurre tras bambalinas para que esto suceda?

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de vídeo.
5. La tarjeta de vídeo envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

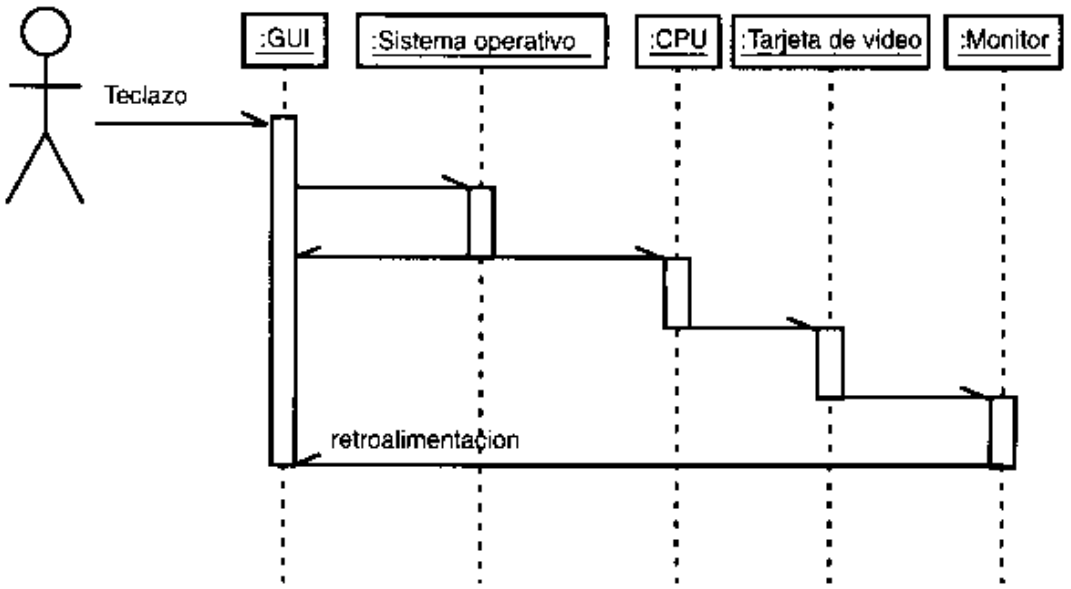
Todo esto ocurre con tanta rapidez que olvidamos que todo ello se realiza. (¡Si acaso pensábamos que ocurría!)

El diagrama de secuencias

La figura 9.4 representa el diagrama de secuencias de la GUI. Como ve, los mensajes son asincrónicos: ninguno de los componentes aguarda nada antes de continuar. Al trabajar con algunas aplicaciones de Windows, tal vez haya sentido algunos de los efectos de la comunicación asincrónica, particularmente en una máquina lenta. Cuando teclea en un procesador de textos, en ocasiones no ve aparecer en la pantalla el carácter correspondiente a la tecla que haya oprimido sino hasta después de haber oprimido algunas más.

FIGURA 9.4

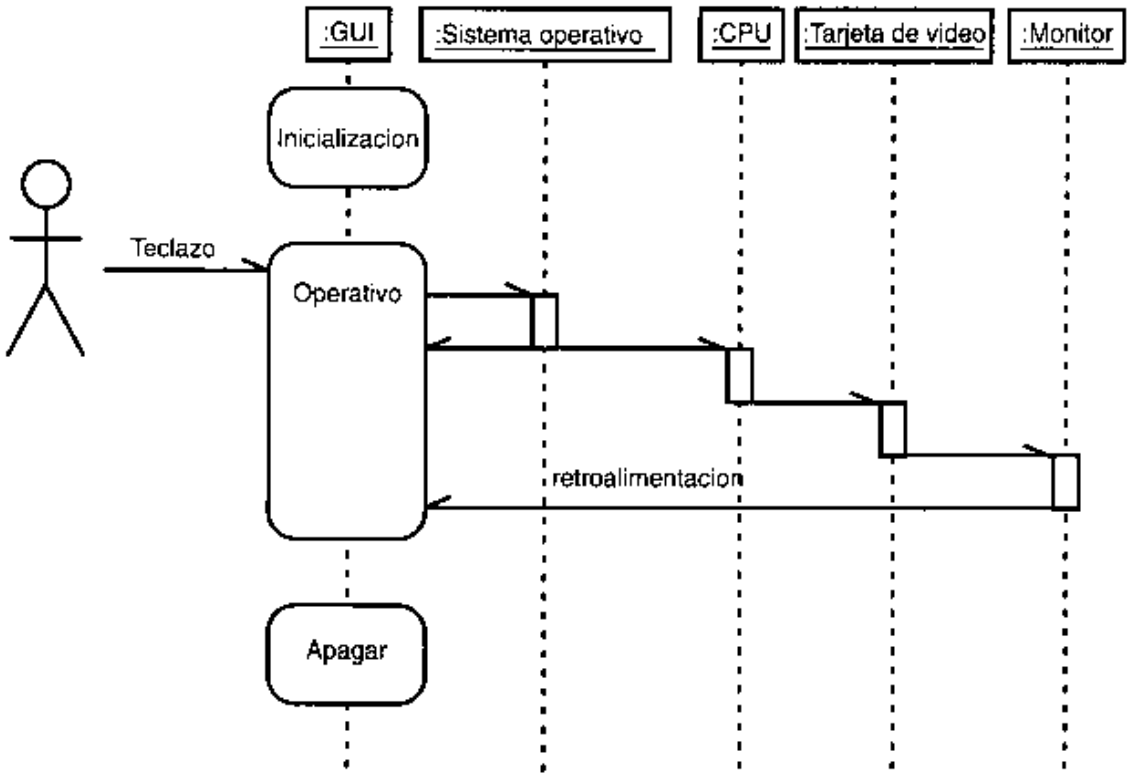
Un diagrama de secuencias que muestra la forma en que la GUI interacciona con otros objetos.



En ocasiones, es muy instructivo mostrar los estados de uno o varios de los objetos en el diagrama de secuencias. Dado que ya ha analizado los estados de la GUI (en la hora anterior), esto es fácil de hacer. La figura 9.5 le muestra un híbrido: el diagrama de secuencias de la GUI con los estados de la GUI. Observe que la secuencia se origina y finaliza en el estado Operativo de la GUI, como podría esperarlo.

FIGURA 9.5

Un diagrama de secuencias puede mostrar los estados de un objeto.



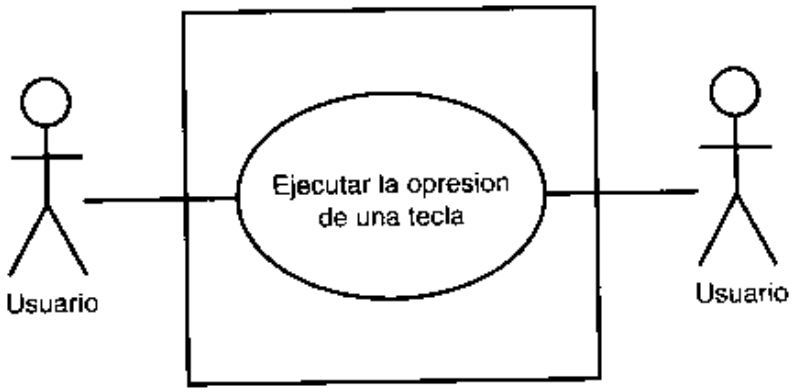
En un diagrama de secuencias, otra forma de mostrar el cambio de estado de un objeto es incluir al objeto más de una vez en el diagrama.

El caso de uso

¿Qué es exactamente lo que representa un diagrama de secuencias? En este ejemplo, muestra las interacciones de objetos que se realizan durante un escenario sencillo: la opresión de una tecla. Este escenario podría ser parte de un caso de uso llamado "Ejecutar la opresión de una tecla" (vea la figura 9.6). Al representar gráficamente las interacciones del sistema en el caso de uso, el diagrama de secuencias habrá, en efecto, "delineado" el caso de uso dentro del sistema.

FIGURA 9.6

El caso de uso representado gráficamente por el diagrama de secuencias de la figura 9.4.



En el siguiente ejemplo, examinaré detalladamente la relación entre los casos de uso y los diagramas de secuencias.

Instancias y genéricos

El ejemplo anterior comenzó con un diagrama de estados. Este otro ejemplo empieza con un caso de uso. "Comprar gaseosa" fue uno de los casos de uso del ejemplo de la máquina de gaseosas en las horas 6, "Introducción a los casos de uso", y 7, "Diagramas de casos de uso".

Un diagrama de secuencias de instancias

En el mejor escenario del caso de uso "Comprar gaseosa", recuerde que el actor es un cliente que desea adquirir una lata de gaseosa. El cliente inicia el escenario mediante la inserción de dinero en la máquina. Luego hace una selección. Dado que hablamos del mejor escenario, la máquina tiene al menos una lata de la gaseosa elegida y por lo tanto presenta una lata fría al cliente.

Asumamos que en la máquina de gaseosas hay tres objetos que realizan la tarea que nos ocupa: la fachada (la interfaz que la máquina de gaseosas presenta al usuario), el registrador de dinero (que lo recolecta), y el dispensador (que entrega la gaseosa). También daremos por hecho que el registrador de dinero controlará al dispensador. La secuencia será como sigue:

1. El cliente inserta el dinero en la alcancía que se encuentra en la fachada de la máquina.
2. El cliente hace su elección.
3. El dinero viaja hacia el registrador.
4. El registrador verifica si la gaseosa elegida está en el dispensador.
5. Dado que es el mejor escenario, asumimos que sí hay gaseosas, y el registrador actualiza su reserva de efectivo.
6. El registrador hace que el dispensador entregue la gaseosa en la fachada de la máquina.

TÉRMINO NUEVO

Dado que el diagrama de secuencias sólo se centra en un escenario (una instancia) en el caso de uso "Comprar gaseosa", se conoce como *diagrama de secuencias de instancias*. La figura 9.7 le muestra este diagrama. Vea que el diagrama muestra mensajes sencillos. Cada mensaje mueve el flujo de control de un objeto a otro.

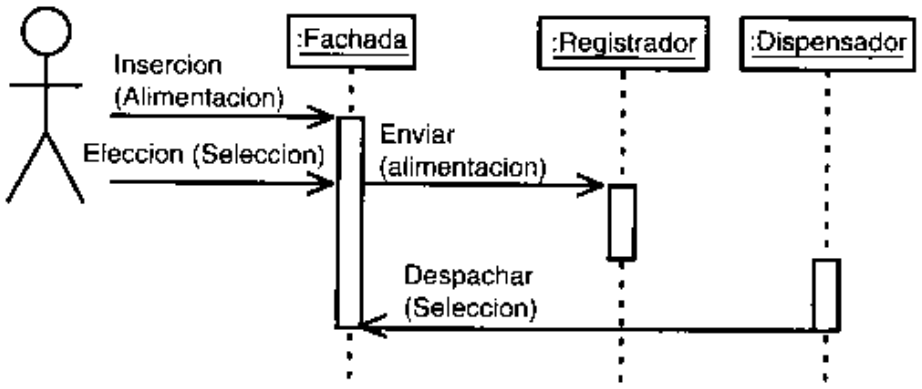


FIGURA 9.7

Este diagrama de secuencias modela tan sólo el mejor escenario del caso de uso "Comprar gaseosa". Por lo tanto, es un diagrama de secuencias de instancias.

Un diagrama de secuencias genérico

Como recordará, el caso de uso "Comprar gaseosa" tenía dos escenarios alternos. Uno de ellos se refería al hecho de que la máquina no tuviera la gaseosa seleccionada y el otro cuando el cliente no contaba con el dinero exacto. Si tomara en cuenta todos los escenarios de un caso de uso al momento de crear un diagrama de secuencias, se trataría de un *diagrama de secuencias genérico*.

En este caso podrá generar el diagrama de secuencias genérico a partir del diagrama de secuencias de instancias. Para ello tendrá que justificar el control del flujo. Esto es, tendrá que representar las condiciones y consecuencias de "Monto incorrecto" y "Sin gaseosa".

Para el escenario relacionado con "Monto incorrecto".

1. El registrador verifica si la alimentación del usuario concuerda con el precio de la gaseosa.
2. Si el monto es mayor que el precio, el registrador calcula la diferencia y verifica si cuenta con cambio.

3. Si se puede devolver la diferencia, el registrador devuelve el cambio al cliente y todo transcurre como antes.
4. Si la diferencia no se encuentra en la reserva del cambio, el registrador regresará el monto alimentado y mostrará un mensaje que indique al cliente que inserte el monto exacto.
5. Si la cantidad insertada es menor al precio, el registrador no hace nada y la máquina esperará más dinero.

Si diseña una máquina de gaseosas para un cliente, tal vez tenga que tomar una decisión de diseño respecto al paso 5. Podrá hacer que la máquina aguarde cierto tiempo, calcule la diferencia entre el precio y el monto insertado, y que muestre un mensaje que solicite al cliente que inserte la diferencia.

Como parte de la decisión, tendrá que responder a estas preguntas: ¿Qué tanto le importará esta facultad al cliente? ¿Cuánto costaría implementar la tecnología que lograra lo anterior?

Éste es un buen ejemplo de la forma en que un diagrama de secuencias puede influir en el proceso de análisis.

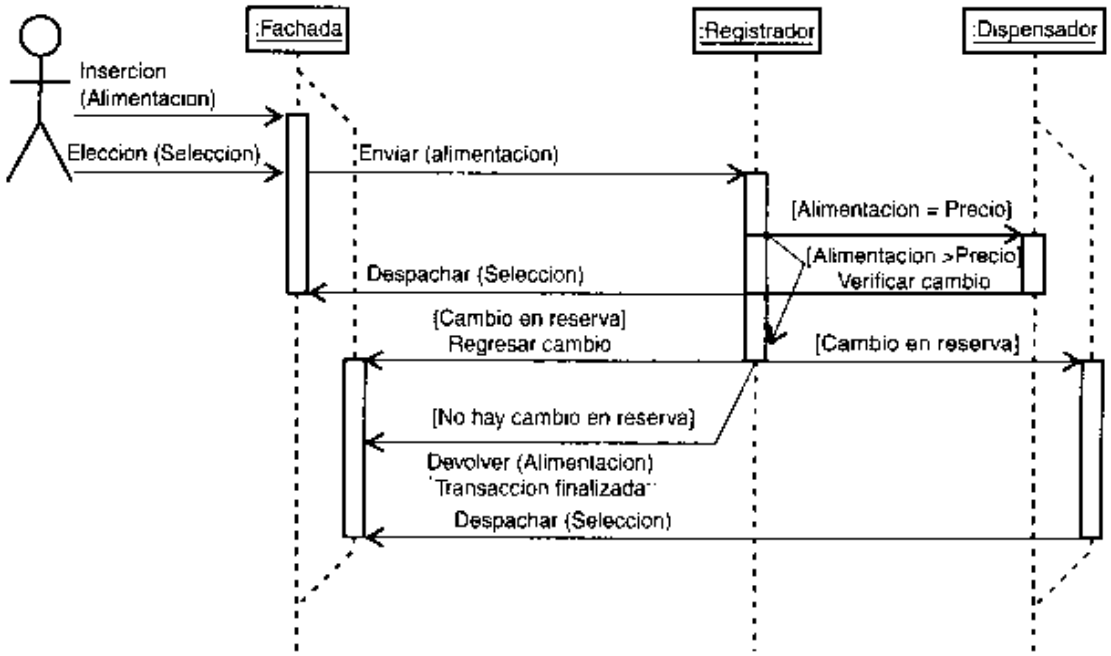
Para representar cada condición en la secuencia, tal condición la colocará en un "si" (un si condicional) entre corchetes. Arriba de las flechas de mensaje apropiadas, agregue [alimentación > precio], [alimentación – precio no presente] y [alimentacion – precio presente].

Cada condición causará una bifurcación del control en el mensaje, que separará al mensaje en rutas distintas. Como cada ruta irá al mismo objeto, la bifurcación causará una "ramificación" del control en la línea de vida del objeto receptor, y separará las líneas de vida en rutas distintas. En algún lugar de la secuencia, las ramas del mensaje confluirán, como las bifurcaciones en las líneas de vida.

La figura 9.8 muestra un diagrama luego de agregar el escenario de "Monto incorrecto".

FIGURA 9.8

El diagrama de secuencias luego de agregar el escenario de "Monto incorrecto" al caso de uso "Comprar gaseosa".



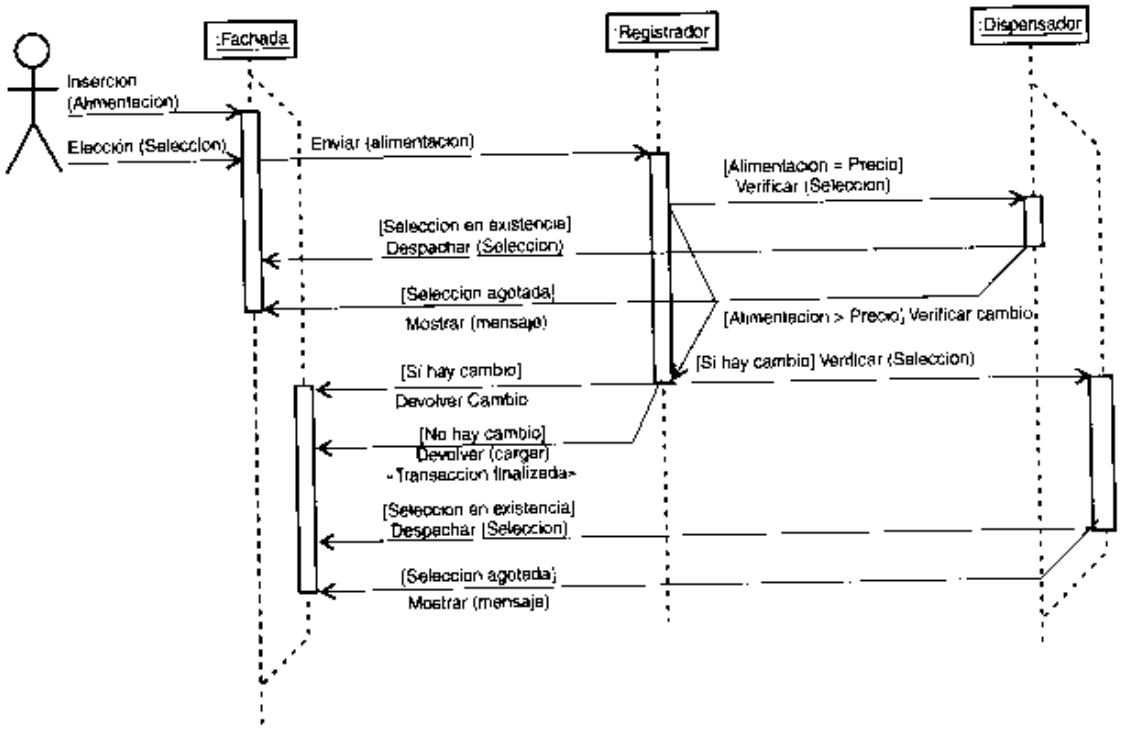
Ahora agregaremos el escenario "Sin gaseosa".

1. Una vez que el cliente elige una marca agotada, la máquina mostrará un mensaje de "Agotado".
2. La máquina mostrará un mensaje que solicitará al cliente que haga otra elección.
3. El cliente tendrá la opción de oprimir un botón para que se le regrese su dinero.
4. Si el cliente elige una marca en existencia, todo procederá como en el mejor escenario si el monto insertado es correcto. Si no lo es, la máquina seguirá por el escenario del "Monto incorrecto".
5. Si el cliente elige otra marca agotada, el proceso se repetirá hasta que el cliente elija una marca en existencia o presione un botón que le regrese su dinero.

La figura 9.9 le muestra el diagrama de secuencias genérico de la máquina de gaseosas con los escenarios "Monto incorrecto" y "Sin marca".

FIGURA 9.9

El diagrama de secuencias genérico de la máquina de gaseosas luego de agregarle el escenario "Sin marca" a la figura 9.8.



Si empieza a pensar que un diagrama de secuencias está implícito en cada caso de uso, ya tiene la idea.

Creación de un objeto en la secuencia

En los ejemplos que hemos visto ha analizado distintos tipos de mensajes, diagramas de secuencias genérico y de instancias, así como estructuras de control. Otro concepto importante relacionado con los diagramas de secuencias, particularmente cuando diseñe software, es la creación de objetos.

Con frecuencia se da el caso de que un programa orientado a objetos debe crear un objeto. Recuerde que en términos del software, una clase es una plantilla para crear un objeto (como un molde de galletas para crear una galleta). ¿Cómo representaría la creación de un objeto cuando represente una secuencia de interacciones entre objetos?

El caso de uso "Crear una propuesta" del ejemplo de la LAN en una firma de consultoría nos muestra una instancia de la creación de objetos. En este ejemplo concebirá la LAN en el entendido de que todo se realiza mediante la red. Si damos por hecho que el consultor ya ha iniciado una sesión en la LAN, la secuencia que modelará quedará como sigue:

1. El consultor querrá volver a utilizar partes de una propuesta existente y busca en un área centralizada de la red una propuesta adecuada.
2. Si el consultor encuentra una propuesta adecuada, abrirá el archivo y, en el proceso, abrirá el software integrado para la oficina relacionada. El consultor guardará el archivo con un nuevo nombre, con lo que creará un archivo nuevo para la nueva propuesta.

3. Si el consultor no encuentra una propuesta, abrirá la aplicación de oficina y creará un archivo para la propuesta.
4. Al trabajar en la propuesta, el consultor utilizará las aplicaciones del software integrado para oficina.
5. Cuando el consultor finalice la propuesta, la guardará en el área de almacenamiento centralizada.

Además de la creación de objetos (en este caso, de archivos), esta secuencia trae consigo el uso de "si" así como de un ciclo "mientras".

TERMINO NUEVO

Primero veamos lo relacionado con la creación de objetos. Cuando una secuencia da por resultado la creación de un objeto, tal objeto se representará de la forma usual: como un rectángulo con nombre. La diferencia es que no lo colocará en la parte superior del diagrama de secuencias, sino que lo colocará junto con la dimensión vertical, de modo que su ubicación corresponda al momento en que se cree. El mensaje que creará al objeto se nombrará "Crear()". Los paréntesis implican una operación: en un lenguaje orientado a objetos, una operación *constructor* genera un objeto.

En lugar de usar "Crear()" o "Create()" para etiquetar la flecha de un mensaje de creación de un objeto, podría valerse de un estereotipo llamado «Crear».

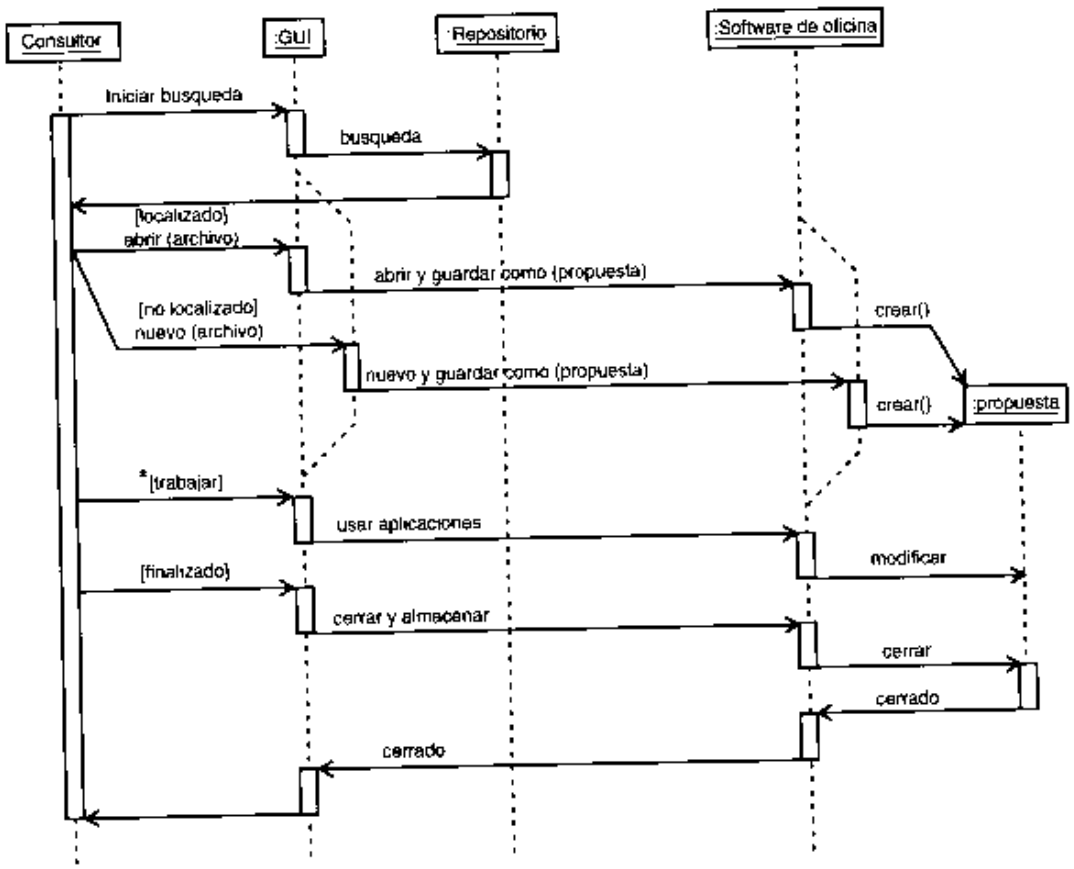
En el caso de "mientras", a este control de flujo lo representará colocando la condición mientras ("mientras se trabaja en una propuesta") entre corchetes, con un asterisco (*) antes del primer corchete.

La figura 9.10 le muestra el diagrama de secuencias del caso de uso "Crear una propuesta".

Este ejemplo representa una abstracción en la que he omitido detalles que no nos competen en lo particular. Esto lo he hecho de dos formas. Primero, obvié los detalles de la LAN, como lo mencioné. También vea que la GUI es un objeto en el diagrama de secuencias, y que no he incluido toda la complejidad del caso de uso "Teclazo" del ejemplo anterior. Los detalles de la interacción de la GUI con el sistema operativo, la CPU y el monitor no son importantes en este caso.

FIGURA 9.10

El diagrama de secuencias del caso de uso "Crear una propuesta".



Cómo representar la recursividad

TERMINO NUEVO

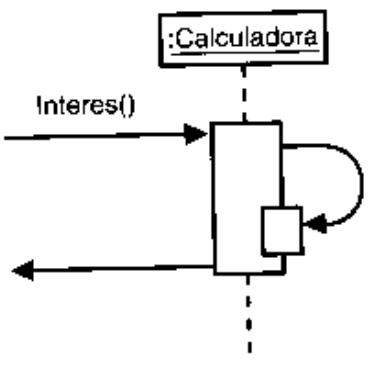
En ocasiones un objeto cuenta con una operación que se invoca a sí misma. A esto se le conoce como *recursividad*, y es una característica fundamental de varios lenguajes de programación.

He aquí un ejemplo. Suponga que uno de los objetos en su sistema sea una calculadora, y que una de sus operaciones sea el cálculo de intereses. Para calcular el interés compuesto para un periodo que incluya a varios periodos, la operación del cálculo de intereses del objeto tendrá que invocarse a sí misma varias veces.

Para representar esto en el UML, dibujará una flecha de mensaje fuera de la activación que signifique la operación, y un pequeño rectángulo sobrepuesto en la activación. Dibuje una flecha de modo que apunte al pequeño rectángulo, y una que regresa al objeto que inició la recursividad. La figura 9.11 muestra lo anterior.

FIGURA 9.11

Cómo representar la recursividad en un diagrama de secuencias.

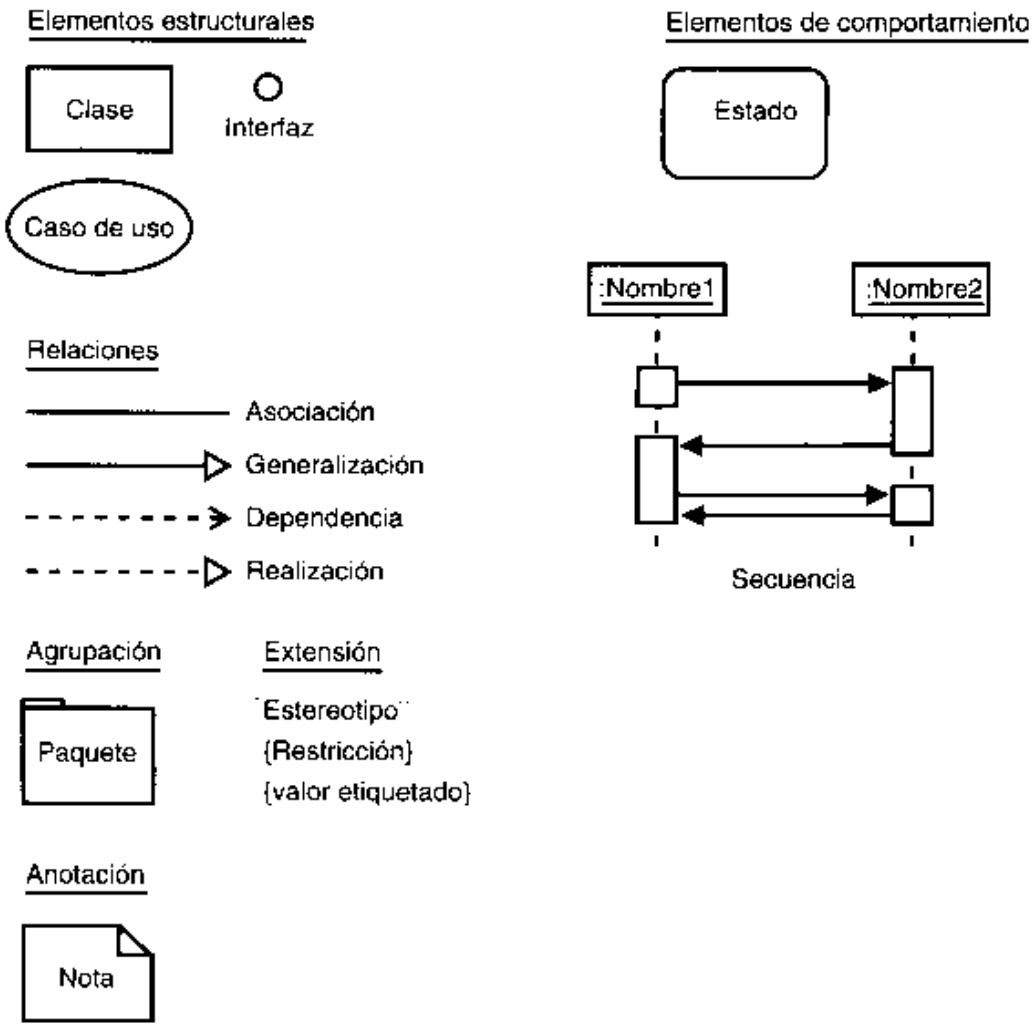


Adiciones al panorama

Ahora podrá agregar otro diagrama a su panorama del UML. Dado que se refiere al comportamiento de los objetos, el diagrama de secuencias iría bajo la categoría "Elementos de comportamiento". La figura 9.12 actualiza su creciente panorama.

FIGURA 9.12

El panorama del UML con la adición del diagrama de secuencias.



Resumen

El diagrama de secuencias UML agrega la dimensión del tiempo a las interactividades de los objetos. En el diagrama, los objetos se colocan en la parte superior y el tiempo avanza de arriba hacia abajo. La línea de vida de un objeto desciende de cada uno de ellos. Un pequeño rectángulo de la línea de vida de un objeto representa una *activación* (la ejecución de una de las operaciones del objeto). Puede incorporar los estados de un objeto colocándolos junto a su línea de vida.

Los mensajes (simples, sincrónicos y asincrónicos) son flechas que conectan a una línea de vida con otra. La ubicación del mensaje en la dimensión vertical representará el momento en que sucede dentro de la secuencia. Los mensajes que ocurren primero están más cerca de la parte superior del diagrama, y los que ocurren después cerca de la parte inferior.

Un diagrama de secuencias puede mostrar ya sea una instancia (un escenario) de un caso de uso, o puede ser genérico e incorporar todos los escenarios de un caso de uso. Los diagramas de secuencias genéricos con frecuencia dan la oportunidad de representar instrucciones condicionales y ciclos "mientras". Bordee a cada condición con corchetes, y haga lo mismo en un ciclo "mientras" pero anteceda al corchete izquierdo con un asterisco.

Cuando una secuencia incluya la creación de un objeto, lo representará como un rectángulo de la forma acostumbrada. Su posición en la dimensión vertical representarán el momento en que se creó.

En ciertos sistemas, una operación puede invocarse a sí misma. A esto se le conoce como *recursividad*. Se representa con una flecha que sale de la activación hacia sí misma, y un pequeño rectángulo sobrepuesto a la activación.

Preguntas y respuestas

P El diagrama de secuencias parece que podría ser útil para más que tan sólo el análisis de sistemas. ¿Puedo usarlo para mostrar la interactividad en una empresa?

R Así es. Los objetos pueden ser los actores principales, y los mensajes pueden ser simples transferencias de control.

P Usted indicó la forma de representar la creación de objetos en un diagrama de secuencias. ¿Los objetos también se destruyen, y si es así, cómo lo represento?

R Los objetos, en efecto, se destruyen. Podrá representar la destrucción de un objeto con una "X" al final de la línea de vida correspondiente a tal objeto.

Taller

Ahora que ha vuelto hacia los objetos y ha visto sus interactividades, venga y responda algunas preguntas y realice algunos ejercicios para reafirmar su conocimiento de los diagramas de secuencias. Encontrará las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. Defina mensaje *sincrónico* y mensaje *asincrónico*.
2. En un diagrama de secuencias genérico ¿cómo representaría el control de flujo implícito en una instrucción condicional?
3. ¿Cómo representaría el control de flujo implícito en una instrucción de ciclo "mientras"?
4. En un diagrama de secuencias ¿cómo representaría a un objeto recién creado?

Ejercicios

1. Cree un diagrama de secuencias de instancias que muestre lo que ocurre cuando envía con éxito un fax. Esto es, modele las interactividades entre objetos en el mejor escenario del caso de uso "enviar fax" de una máquina de fax. Incluya los objetos de la máquina que envía, la que recibe, el fax y un "intercambio" central que encause a los faxes y a las llamadas telefónicas.
2. Cree un diagrama de secuencias genérico que incluya escenarios infructuosos (línea ocupada, error de la máquina que envía), así como del mejor escenario indicado en el ejercicio 1.

HORA 10

Diagramas de colaboraciones

Ahora veremos lo correspondiente a un diagrama que es similar al que vimos en la hora anterior. Este también muestra la colaboración entre los objetos, pero de una forma significativamente diferente del diagrama de secuencias.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de colaboraciones
- Cómo aplicar un diagrama de colaboraciones
- Uso de "si" y "mientras"
- Anidamiento
- Objetos activos y concurrencia
- Sincronización
- Dónde encajan los diagramas de colaboraciones en el UML

Los diagramas de colaboraciones muestran la forma en que los objetos colaboran entre sí, tal como sucede con un diagrama de secuencias. Muestran los objetos junto con los mensajes que se envían entre ellos. Si el diagrama de secuencias hace eso, ¿por qué el UML requeriría otro diagrama?, ¿qué no son lo mismo?, ¿no es una pérdida de tiempo?

Ambos tipos de diagrama son similares. De hecho, son *semánticamente equivalentes*. Esto significa que representan la misma información, y podrá convertir un diagrama de secuencias en un diagrama de colaboraciones equivalente y viceversa.

Como se infiere, es útil contar con ambas formas. Los diagramas de secuencias destacan la sucesión de las interacciones. Los diagramas de colaboraciones destacan el contexto y organización general de los objetos que interactúan. He aquí otra forma de encontrar la diferencia: el diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.

Qué es un diagrama de colaboraciones

Un diagrama de objetos muestra a los objetos como tales y sus relaciones entre sí. Un diagrama de colaboraciones es una extensión de uno de objetos. Además de las relaciones entre objetos, el diagrama de colaboraciones muestra los mensajes que se envían los objetos entre sí. Por lo general, evitará la multiplicidad dado que podría ser fuente de confusión.

Para representar un mensaje, dibujará una flecha cerca de la línea de asociación entre dos objetos, esta flecha apunta al objeto receptor. El tipo de mensaje se mostrará en una etiqueta cerca de la flecha; por lo general, el mensaje le indicará al objeto receptor que ejecute una de sus operaciones. El mensaje finalizará con un par de paréntesis, dentro de los cuales colocará los parámetros (en caso de haber alguno) con los que funcionará la operación.

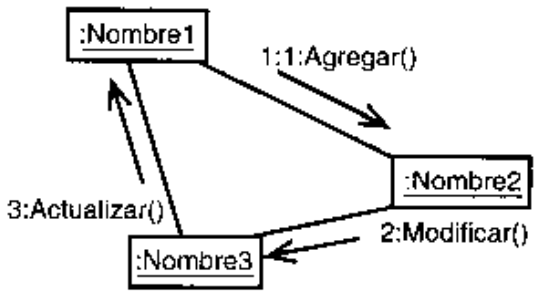
Mencioné que podrá convertir cualquier diagrama de secuencias en diagrama de colaboraciones y viceversa. Por medio de esto podrá representar la información de secuencia en un diagrama de colaboraciones. Para ello, agregará una cifra a la etiqueta de un mensaje, misma que corresponderá a la secuencia propia del mensaje. La cifra y el mensaje se separan mediante dos puntos (:).

La figura 10.1 le muestra la simbología del diagrama de colaboraciones.

Aprovechemos la equivalencia de ambos tipos de diagramas. Para desarrollar los conceptos de los diagramas de colaboraciones volveremos a ver los ejemplos que revisamos la hora anterior. Conforme lo haga, verá más conceptos.

FIGURA 10.1

Simbología del diagrama de colaboraciones.



La GUI

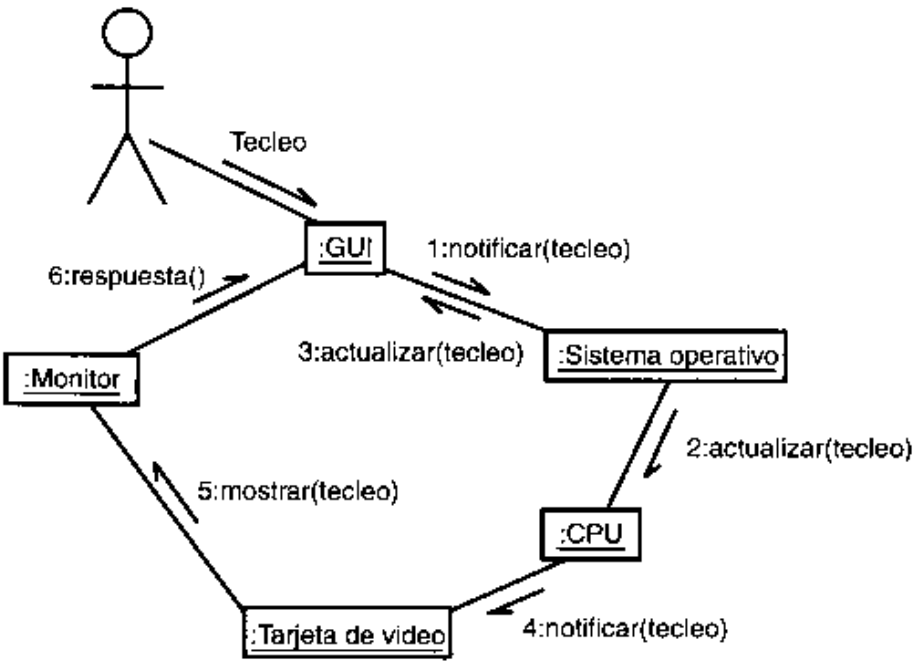
Este ejemplo es el caso más directo. Un actor inicia la secuencia de interacción al oprimir una tecla, con lo que los mensajes ocurrirán de manera secuencial. Tal secuencia (a partir de la hora anterior) es:

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de vídeo.
5. La tarjeta de vídeo envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

La figura 10.2 muestra la forma de representar esta secuencia de interacciones en un diagrama de colaboraciones. El diagrama muestra la figura agregada que representa al usuario que inicia la secuencia, aunque esta figura no es parte de la simbología de este diagrama.

FIGURA 10.2

Un diagrama de colaboraciones para el ejemplo de la GUI.



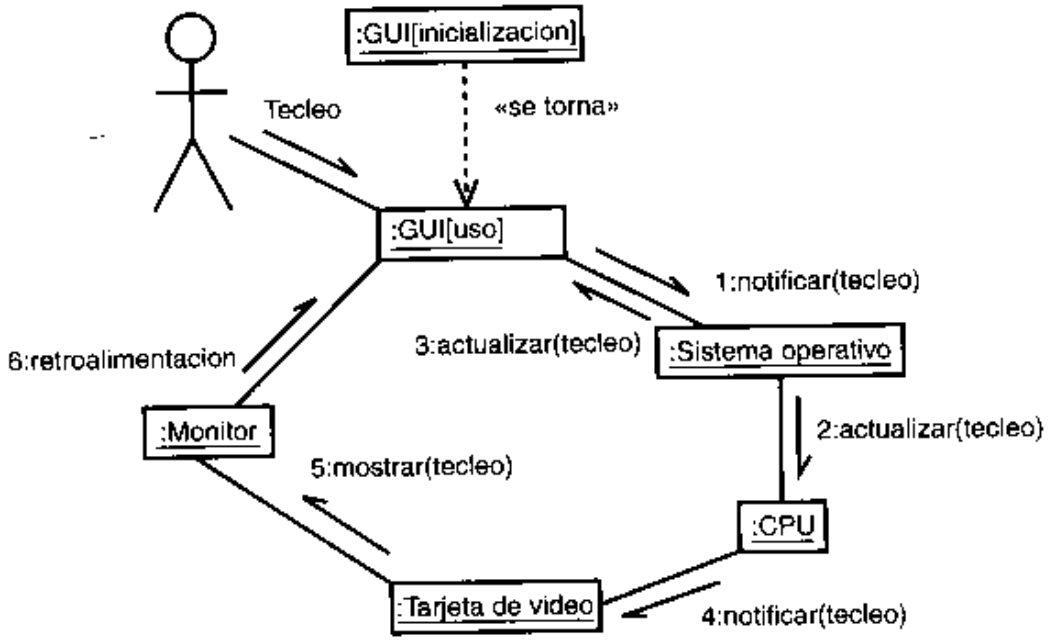
Cambios de estado

Puede mostrar los cambios de estado en un objeto en un diagrama de colaboraciones. En el rectángulo del objeto indique su estado. Agregue otro rectángulo al diagrama que haga las veces del objeto e indique el estado modificado. Conecte a los dos con una línea discontinua y etiquete la línea con un estereotipo «se torna».

La figura 10.3 ilustra un cambio de estado para la GUI, que muestra que el estado de inicialización se convierte en el estado operativo.

FIGURA 10.3

Un diagrama de colaboraciones puede incorporar cambios de estado.



La máquina de gaseosas

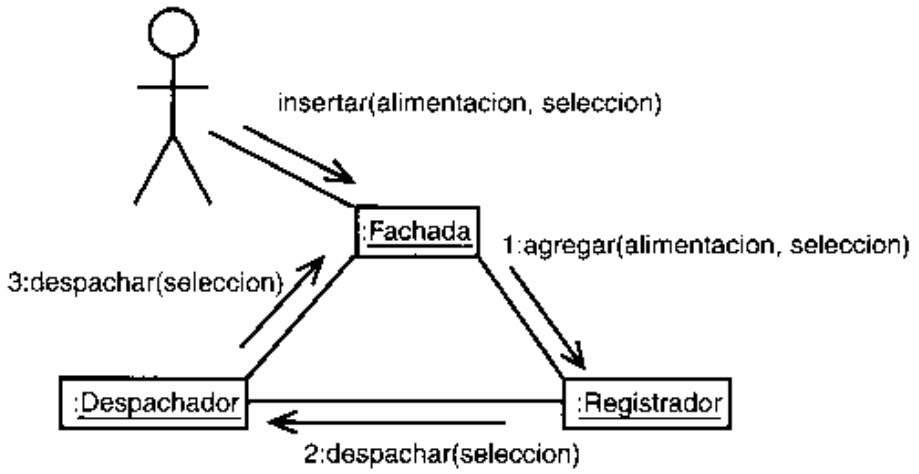
Las cosas se hacen más interesantes cuando aplica las condiciones a una situación real, como lo hizo en la hora anterior con la máquina de gaseosas. Iniciemos con la mejor situación del caso de uso "Comprar gaseosa", donde la secuencia es:

1. El cliente inserta el dinero en la alcancía que se encuentra en la fachada de la máquina.
2. El cliente hace su elección.
3. El dinero viaja hacia el registrador.
4. El registrador verifica si la gaseosa elegida está en el dispensador.
5. Dado que es la mejor situación, asumimos que sí hay gaseosas, y el registrador actualiza su reserva de efectivo.
6. El registrador hace que el dispensador entregue la gaseosa en la fachada de la máquina.

El diagrama de colaboraciones es directo, como lo muestra la figura 10.4.

FIGURA 10.4

El diagrama de colaboraciones para el mejor caso de "Comprar gaseosa".



Ahora, agreguemos el caso de "cantidad incorrecta de dinero". El diagrama tiene que contabilizar varias condiciones:

1. El usuario ha introducido más dinero que el necesario para la compra.
2. La máquina cuenta con la cantidad adecuada de cambio.
3. La máquina no tiene la cantidad correcta de cambio.

Usted representará las condiciones de la misma forma en que las representó en el diagrama de secuencias. Colocará la condición entre corchetes, misma que se antecede a la etiqueta. Lo importante es coordinar las condiciones con la numeración.

Esto podría ser algo complicado, por lo que haremos el diagrama por secciones. Empezaremos con la condición donde el usuario ha insertado más dinero del indicado en el precio y el registrador cuenta con el cambio adecuado. Agregará el paso de la máquina al devolver el cambio al cliente, y agregará las condiciones entre corchetes. El paso que devuelve el cambio es una consecuencia del que verifica si hay cambio. Para indicar esto en el paso de devolver cambio utilizará el mismo número del mensaje que verifica el cambio, y agregará un punto decimal y un uno. A esto se le conoce como *anidación*. La figura 10.5 le muestra los detalles.

¿Qué ocurre cuando la máquina no cuenta con el cambio correcto? Tendrá que mostrar un mensaje que lo indique, devuelva el dinero y pida al usuario que inserte el importe correcto. Así, la transacción habrá finalizado.

Cuando agregue esta condición, agregará una bifurcación en el control de flujo. Numerará esta bifurcación como un mensaje anidado. Dado que es el segundo mensaje anidado, habrá un 2 luego del punto decimal. Finalmente, y debido a que la transacción habrá finalizado, hará clara esta situación mediante la adición de un estereotipo "transacción finalizada" en este mensaje, y otro en el mensaje que despacha la gaseosa. La figura 10.6 presentará la situación.

FIGURA 10.5

El diagrama de colaboraciones con parte de la situación "monto de dinero inadecuado".

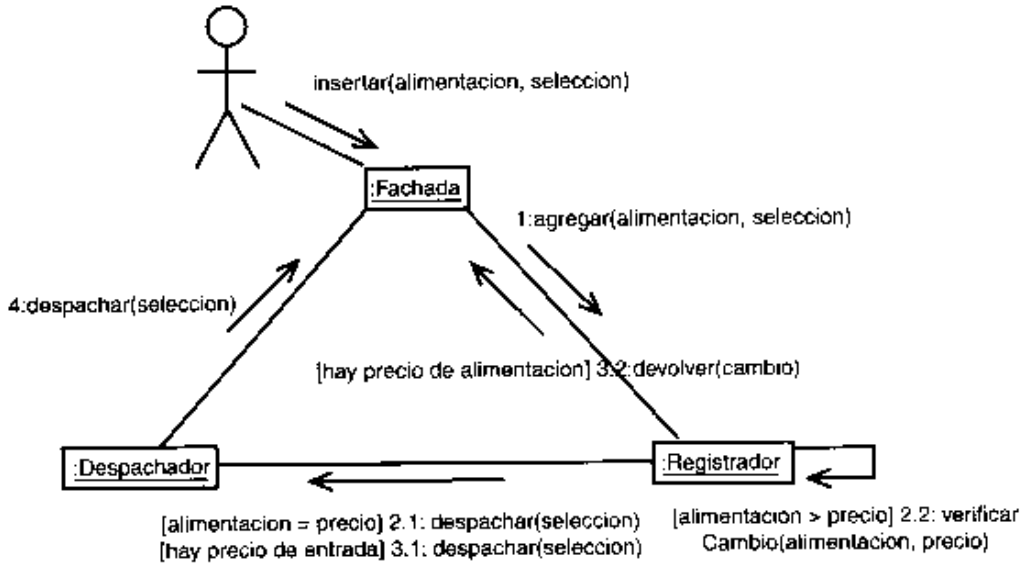
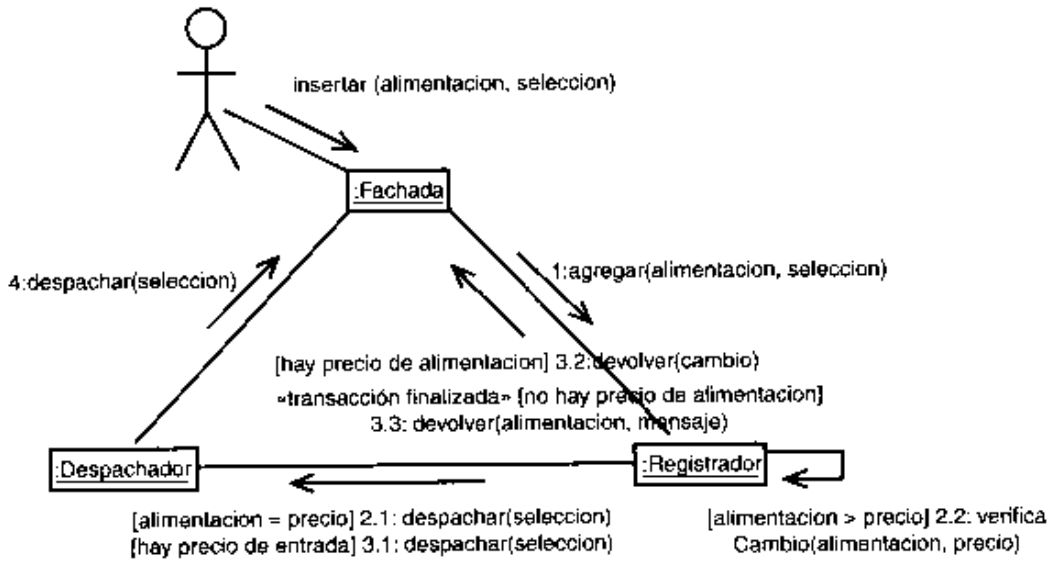


FIGURA 10.6

El diagrama de colaboraciones "Comprar gaseosa" con toda la situación "monto de dinero inadecuado".



En el taller, al finalizar esta hora, habrá un ejercicio que le pedirá que complete el diagrama de colaboraciones mediante la adición de la situación "no hay gaseosa".

Creación de un objeto

Para mostrar la creación de objetos, volveré al caso de uso "Crear propuesta" de la firma de consultoría. Una vez más, la secuencia que modelará será:

1. El consultor buscará en el área de almacenamiento centralizada de la red una propuesta adecuada en la cual basarse.
2. Si el consultor localiza una propuesta adecuada, la abrirá y en el proceso abrirá la aplicación de oficina. El consultor guardará el archivo bajo un nuevo nombre, con lo que creará un nuevo archivo para la nueva propuesta.
3. Si el consultor no encuentra una propuesta, abrirá la aplicación de oficina y generará un nuevo archivo.

4. Al trabajar en la propuesta, el consultor utilizará los componentes de la aplicación de oficina.
5. Cuando el usuario finalice la propuesta, la guardará en el área de almacenamiento centralizada.

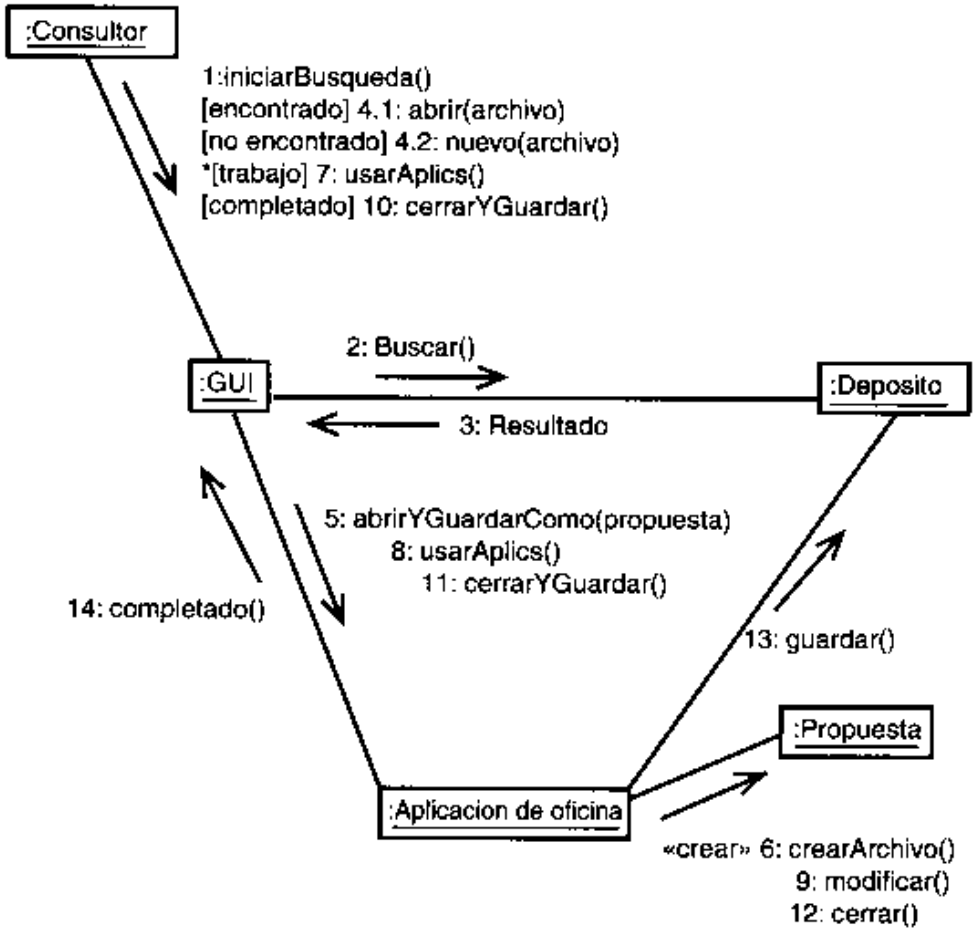
Para mostrar la creación de un objeto, agregará un estereotipo "crear" al mensaje que genera al objeto.

Una vez más, utilizará instrucciones "si" (if) y mensaje anidados. También trabajará con un ciclo "mientras" (while). Como en el diagrama de secuencias, para representar a "mientras", colocará esta condición entre corchetes y antecederá al del lado izquierdo con un asterisco.

La figura 10.7 le muestra este diagrama de colaboraciones completo con la creación del objeto y "mientras".

FIGURA 10.7

El diagrama de colaboraciones "Crear una propuesta".



Algunos conceptos más

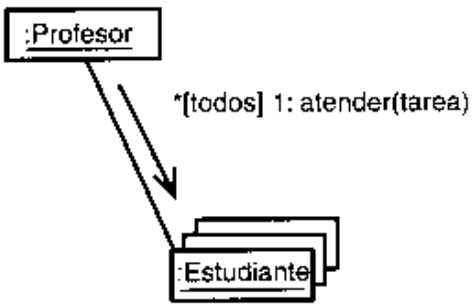
Aunque ha visto algunas bases, no ha visto todo lo relacionado con los diagramas de colaboraciones. Los conceptos en esta sección son un poco esotéricos, pero podrían serle útiles en sus esfuerzos para analizar sistemas.

Varios objetos receptores en una clase

En ocasiones un objeto envía un mensaje a diversos objetos de la misma clase. Por ejemplo: un profesor le pide a un grupo de estudiantes que entreguen una tarea. En el diagrama de colaboraciones, la representación de los diversos objetos es una pila de rectángulos que se extienden "desde atrás". Agregará una condición entre corchetes precedida por un asterisco para indicar que el mensaje irá a todos los objetos. La figura 10.8 le muestra los detalles.

FIGURA 10.8

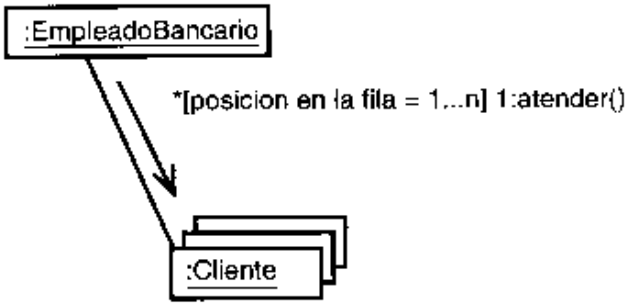
Un objeto que envía un mensaje a diversos objetos de una clase.



En algunos casos, el orden del mensaje enviado es importante. Por ejemplo, un empleado bancario dará servicio a cada cliente conforme fue llegando a la fila. Esto lo representará con un "mientras" cuya condición implicará orden (como en "posición en la fila = 1 ... n") junto con el mensaje y la pila de rectángulos (vea la figura 10.9).

FIGURA 10.9

Un objeto que envía un mensaje a varios otros en un orden específico.



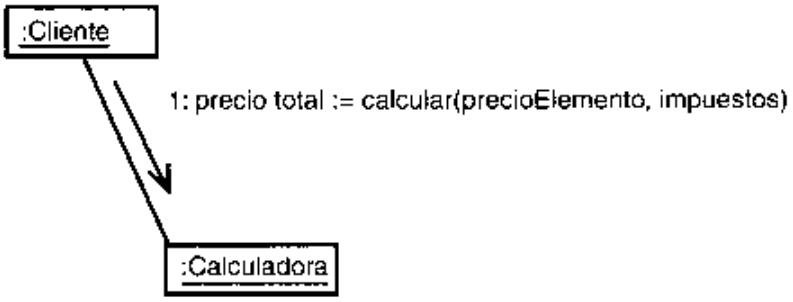
Representación de los resultados

Un mensaje podría ser una petición a un objeto para que realice un cálculo y devuelva un valor. Un objeto Cliente podría solicitar a un objeto Calculadora que calcule el precio total que sea la suma del precio de un elemento y el impuesto.

El UML le da una sintaxis para representar esta situación. Deberá escribir una expresión que tenga el nombre del valor devuelto a la izquierda, seguido de ":=", a continuación el nombre de la operación y las cantidades con que opera para producir el resultado. En este ejemplo, la expresión podría ser precioTotal := calcular(precioElemento, impuesto).
La figura 10.10 le muestra la sintaxis de un diagrama de colaboraciones.

FIGURA 10.10

Un diagrama de colaboraciones que incluye la sintaxis de un resultado.



TERMINO NUEVO A la parte que está a la derecha de la expresión se le conoce como *firma del mensaje*.

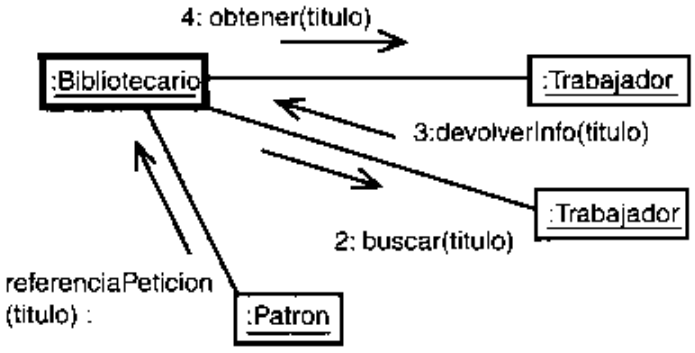
Objetos activos

TERMINO NUEVO En algunas interacciones, un objeto específico controla el flujo. Este *objeto activo* puede enviar mensajes a los objetos pasivos e interactuar con otros objetos activos. En una biblioteca, un bibliotecario relaciona las peticiones a partir de un patrón, verifica la información de referencia en una base de datos, devuelve una respuesta al peticionario, asigna personas para reabastecer los libros, entre otras cosas. Un bibliotecario también interactúa con otros que realicen las mismas operaciones. Al proceso de que dos o más objetos activos hagan sus tareas al mismo tiempo, se le conoce como *concurrencia*.

El diagrama de colaboraciones representa a un objeto activo de la misma manera que a cualquier otro objeto, excepto que su borde será grueso y más oscuro. (Vea la figura 10.11.)

FIGURA 10.11

Un objeto activo controla el flujo en una secuencia. Se representa como un rectángulo con un borde grueso en negro.



Sincronización

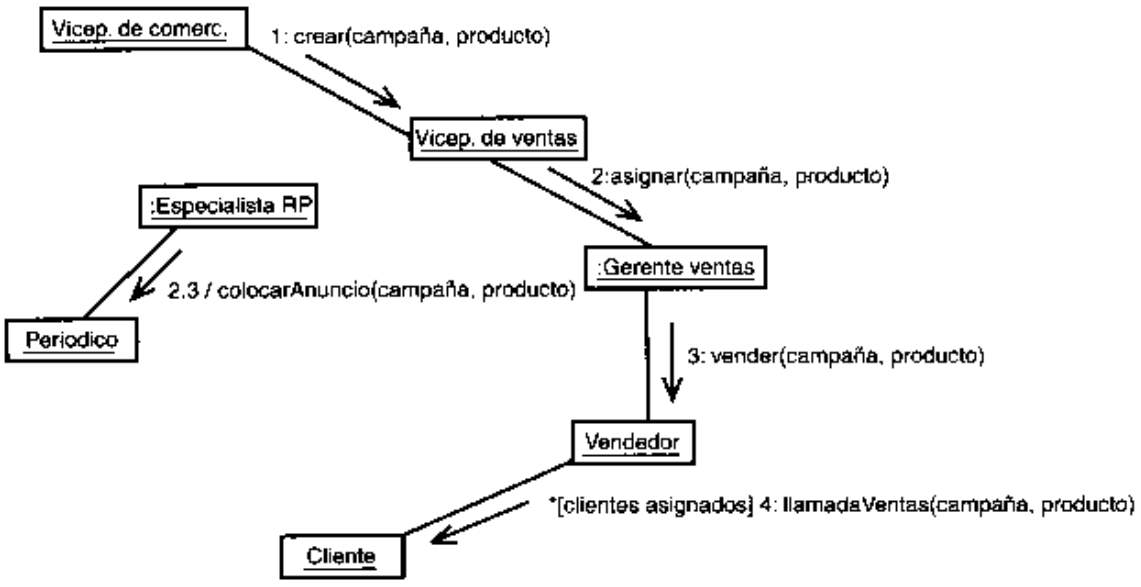
Otro caso con el que se puede encontrar es que un objeto sólo puede enviar un mensaje después de que otros mensajes han sido enviados. Es decir, el objeto debe "sincronizar" todos los mensajes en el orden debido.

Un ejemplo aclarará esto. Suponga que sus objetos son personas en un corporativo, y que están ocupados en la campaña de un nuevo producto. He aquí la secuencia de interacciones:

1. El vicepresidente de comercialización le pide al de ventas que cree una campaña para un producto en particular.
2. El vicepresidente de ventas crea la campaña y la asigna al gerente de ventas.
3. El gerente de ventas instruye a un agente de ventas para que venda el producto de acuerdo con la campaña.
4. El agente de ventas hace llamadas para vender el producto a los clientes en potencia.
5. Luego de que el vicepresidente de ventas ha dado la comisión y el gerente de ventas ha expedido la directiva (esto es, cuando se han completado los pasos 2 y 3), un especialista en relaciones públicas de la corporación hará una llamada al periódico local y colocará un anuncio de la campaña.

¿Cómo representará la posición del paso cinco en la secuencia? Nuevamente, el UML le da una sintaxis. En lugar de anteceder este mensaje con una etiqueta numérica, lo antecederá con una lista de mensajes que tendrán que completarse antes de que se realice el paso cinco. La lista de elementos se separará mediante una coma, y finalizará con una diagonal. La figura 10.12 le muestra el diagrama de colaboraciones en este ejemplo.

FIGURA 10.12
La sincronización de mensajes en un diagrama de colaboraciones.

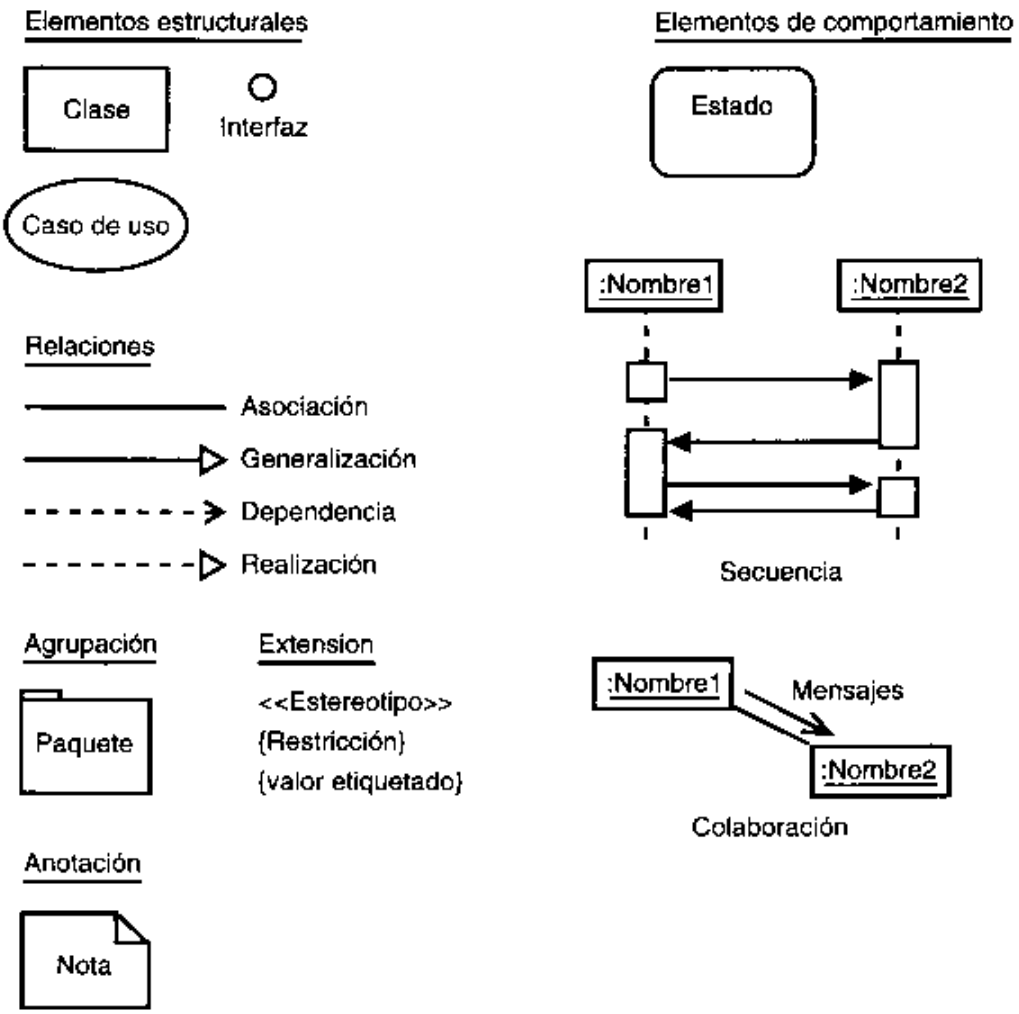


Adiciones al panorama

Ahora podrá agregar el diagrama de colaboraciones a su panorama del UML. Es otro elemento de comportamiento, como se aprecia en la figura 10.13.

FIGURA 10.13

El panorama del UML, que incluye al diagrama de colaboraciones.



Resumen

Un diagrama de colaboraciones es otra forma de presentar la información en un diagrama de secuencias. Ambos tipos de diagramas son semánticamente equivalentes y se recomienda usar ambos cuando construya el modelo de un sistema. El diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.

El diagrama de colaboraciones muestra las asociaciones entre objetos, así como los mensajes que pasan de un objeto a otro. El mensaje se representa con una flecha junto a la línea de asociación, y una etiqueta numerada que muestra el contenido del mensaje. El número representa el turno del mensaje en la secuencia.

Las condicionales se representan como antes, mediante la colocación de la instrucción condicional entre corchetes. Para representar un ciclo "mientras", anteceda al corchete izquierdo con un asterisco.

Algunos mensajes provienen de otros. El esquema de numeración de las etiquetas representa esto de forma muy similar a los manuales técnicos que muestran sus encabezados y subtítulos: con un sistema de numeración que utiliza puntos decimales para representar los niveles del anidamiento.

Los diagramas de colaboraciones le permiten modelar varios objetos receptores en una clase, ya sea que los objetos reciban o no los mensajes en un orden específico. También podrá representar objetos activos que controlen el flujo de los mensajes, así como los mensajes que se sincronizan con otros.

Preguntas y respuestas

P ¿Realmente tengo que incluir a ambos diagramas (el de colaboraciones y el de secuencias) en la mayoría de los modelos UML que genere?

R Se recomienda hacerlo. Ambos tipos de diagramas podrán estimular diversas ideas de los procesos durante el segmento de análisis en el proceso de desarrollo. El diagrama de colaboraciones clarifica las relaciones entre los objetos debido a que incluye los vínculos entre ellos. El de secuencia se enfoca en la secuencia de las interacciones. A su vez, la organización de su cliente podría incluir personas cuya idea de los procesos podría diferir entre ellos. Cuando tenga que presentar su modelo, un tipo de diagrama podría comprenderse mejor para ciertas personas.

Taller

Ahora que ha comprendido los diagramas de secuencias y a sus hermanos, los de colaboración, pruebe y fortalezca su conocimiento con el cuestionario y los ejercicios. Como siempre, verá las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cómo representa a un mensaje en un diagrama de colaboraciones?
2. ¿Cómo mostraría información secuencial en un diagrama de colaboraciones?
3. ¿Cómo mostraría los cambios de estado?
4. ¿Qué se entiende por la "equivalencia semántica" de dos tipos de diagramas?

Ejercicios

1. En el ejemplo de la máquina de gaseosas, sólo mostré un diagrama de colaboraciones equivalente al diagrama de secuencias de instancia de la situación "importe incorrecto". Cree un diagrama de colaboraciones que corresponda al diagrama de secuencias genérico de la hora 9 para el caso de uso "Comprar gaseosa". Esto es, agregue la situación "Gaseosa agotada" al diagrama de colaboraciones de la figura 10.5.

2. En el diagrama de colaboraciones del caso de uso "Crear propuesta", el consultor busca en el área central de almacenamiento una propuesta adecuada para volverla a utilizar. Imagine a "buscar" como un mensaje enviado para buscar en una secuencia de archivos, y utilice las técnicas de modelado de la sección "Algunos conceptos más" para cambiar el diagrama de colaboraciones en la figura 10.6.

HORA 11

Diagramas de actividades

Ahora veremos un tipo de diagrama que podría parecerle familiar, este diagrama le muestra los pasos en una operación o proceso.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de actividades
- Aplicación de los diagramas de actividades
- Marcos de responsabilidad
- Adiciones al panorama

Si alguna vez ha tomado algún curso básico de programación, ya conocerá los diagramas de flujo. Siendo uno de los primeros modelos visuales que se aplicaron a la computación, el diagrama de flujo muestra una secuencia de pasos, procesos, puntos de decisión y bifurcaciones. A los programadores novatos se les invita a que utilicen este diagrama para conceptualizar problemas y derivar sus soluciones. La idea es convertir al diagrama de flujo

en la base del código. Con sus diversas características y tipos de diagramas, el UML es en cierta medida un diagrama de flujo con esteroides.

El diagrama de actividades del UML, tema de esta hora, es muy parecido a los viejos diagramas de flujo. Le muestra los pasos (conocidos como *actividades*) así como puntos de decisión y bifurcaciones. Es útil para mostrar lo que ocurre en un proceso de negocios u operación. Los encontrará como parte integral del análisis de un sistema.

Qué es un diagrama de actividades

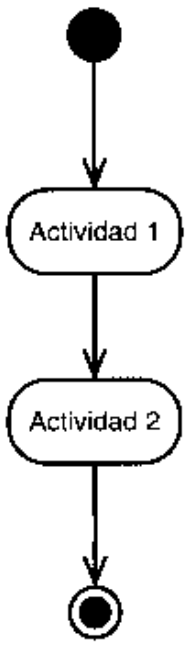
Para empezar, un diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso. Es una extensión de un diagrama de estados, mismo que ya conoció. El diagrama de estados muestra los estados de un objeto y representa las actividades como flechas que conectan a los estados. El diagrama de actividades resalta, precisamente, a las actividades.

A cada actividad se le representa por un rectángulo con las esquinas redondeadas (más angosto y ovalado que la representación del estado). El procesamiento dentro de una actividad se lleva a cabo y, al realizarse, se continúa con la siguiente actividad. Una flecha representa la transición de una a otra actividad. Al igual que el diagrama de estados, el de actividad cuenta con un punto inicial (representado por un círculo relleno) y uno final (representado por una diana).

La figura 11.1 le muestra el punto inicial y final, así como dos actividades y una transición.

FIGURA 11.1

Transición de una actividad a otra.



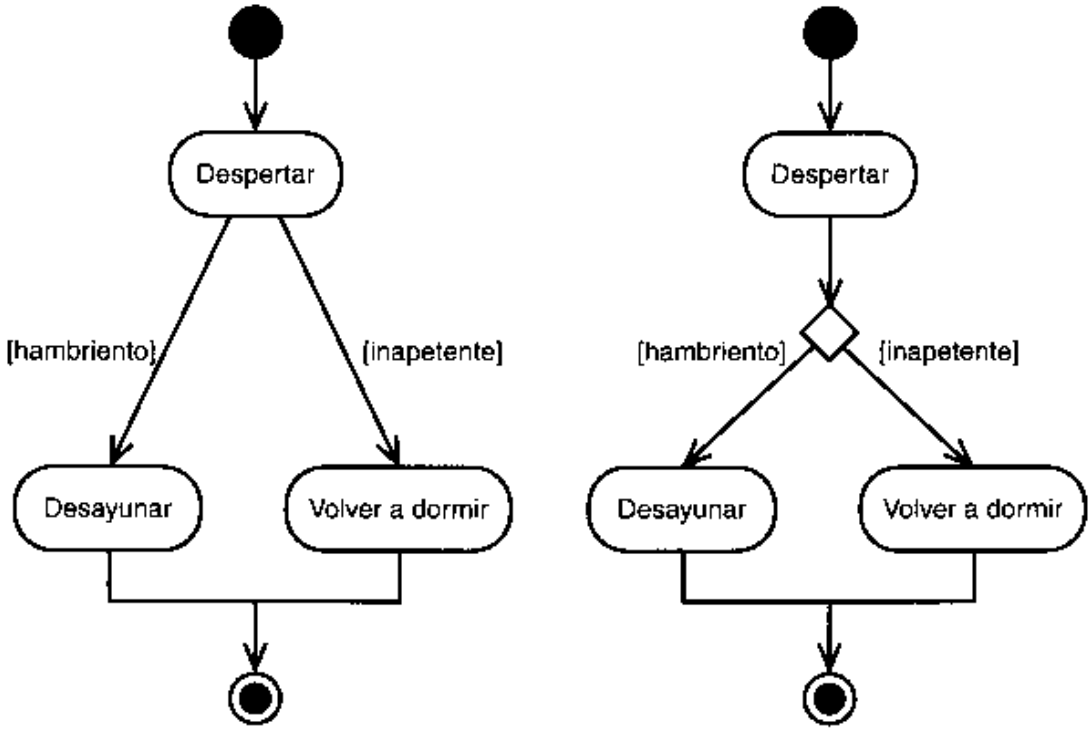
Decisiones, decisiones, decisiones

Casi siempre una secuencia de actividades llegará a un punto donde se realizará alguna decisión. Ciertas condiciones le llevarán por un camino y otras por otro (pero ambas son mutuamente exclusivas).

Podrá representar un punto de decisión de una de dos formas: la primera es mostrar las rutas posibles que parten directamente de una actividad y la segunda es llevar la transición hacia un rombo —reminiscencias del símbolo de decisión en un diagrama de flujo— y que de allí salgan las rutas de decisión (como usuario de los antiguos diagramas de flujo, prefiero la segunda opción). De cualquier forma, indicará la condición con una instrucción entre corchetes junto a la ruta correspondiente. La figura 11.2 le muestra las posibilidades.

FIGURA 11.2

Dos formas de mostrar una decisión.

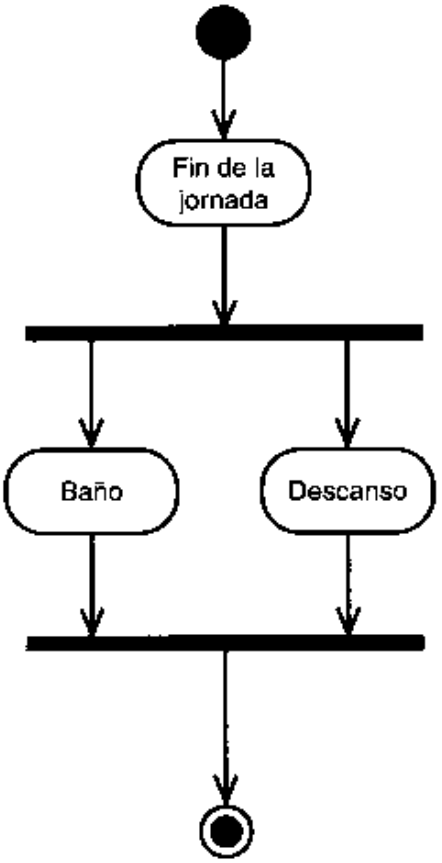


Rutas concurrentes

Conforme modele actividades tendrá la oportunidad de separar una transición en dos rutas que se ejecuten al mismo tiempo (es decir, de forma concurrente) y luego se reúnan. Para representar esta división, utilizará una línea gruesa perpendicular a la transición y las rutas partirán de ella. Para representar la reincorporación, ambas rutas apuntarán a otra línea gruesa (vea la figura 11.3).

FIGURA 11.3

Representación de una transición que se bifurca en dos rutas que se ejecutan de forma concurrente y, luego, se reincorporan.



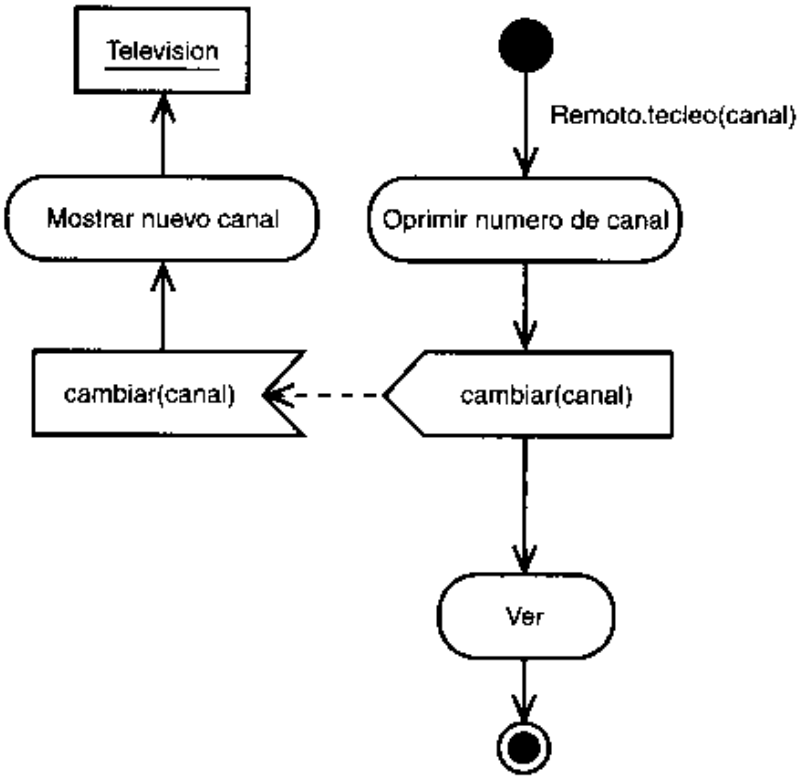
Indicaciones

Durante una secuencia de actividades, es posible enviar una indicación. Cuando se reciba, la indicación provocará que se ejecute una actividad. El símbolo para enviar una indicación es un pentágono convexo, y el que la recibe es un pentágono cóncavo. La figura 11.4 le ayudará a clarificar la idea.

En términos del UML el pentágono convexo simboliza al *envío de un evento*; el cóncavo simboliza la *recepción del evento*.

FIGURA 11.4

Envío y recepción de una indicación.



Aplicación de los diagramas de actividades

Veamos algunos ejemplos. Para empezar, diagramará una operación y posteriormente un proceso.

Una operación: Fibs

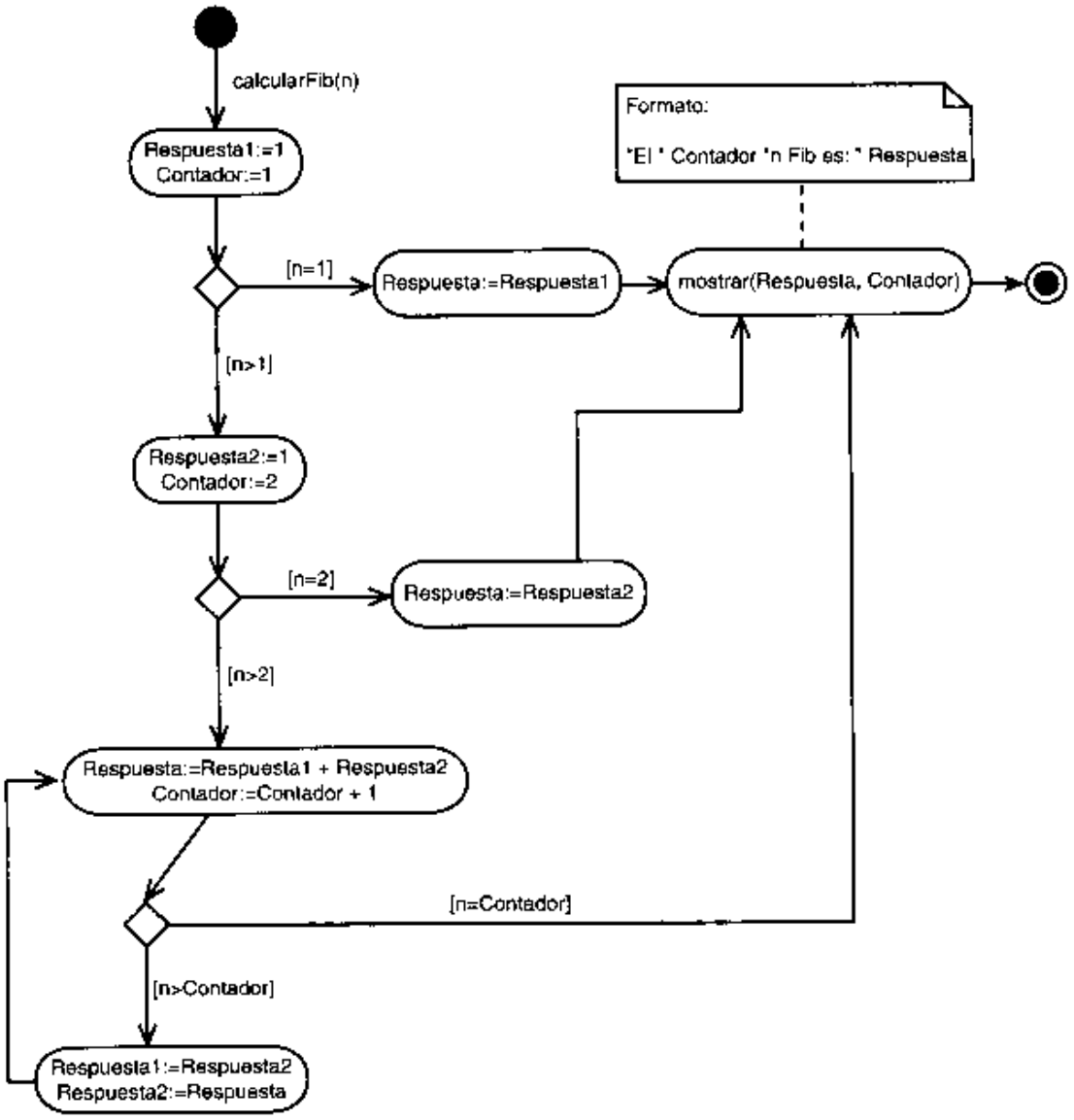
¿Había visto la siguiente serie de números? 1, 1, 2, 3, 5, 8, 13,.... se conoce como la "serie de Fibonacci", pues este matemático medieval la escribió hace unos 800 años. Cada número es un "fib", así que el primer fib [o fib(1)] es 1, fib(2) es 1, fib(3) es 2, y así sucesivamente. La regla de cada fib, excepto en los dos primeros, es la suma del anterior par de fibs, por ejemplo, fib(8) tendría un valor de 21.

Suponga que una de sus clases es una calculadora, y que una de sus operaciones fuese la de calcular el enésimo fib y mostrarlo. A esta operación podría llamarla *calcularFib(n)*. Creemos un diagrama de actividades que modele a esta operación.

Requerirá algunas variables como son: un contador para llevar un control para verificar si se ha llegado al enésimo fib, una variable para conglomerar los resultados, y dos más para almacenar dos fibs que tendrá que sumar entre sí. La figura 11.5 le muestra el diagrama de actividades que realizaría la tarea.

FIGURA 11.5

Un diagrama de actividades para calcularFib(n), una operación que calcula el enésimo número de Fibonacci.



Proceso de creación de un documento

Ahora volvamos nuestra atención de una operación a un proceso. Imagine las actividades necesarias para utilizar una aplicación de oficina para crear un documento. Una posible secuencia podría ser:

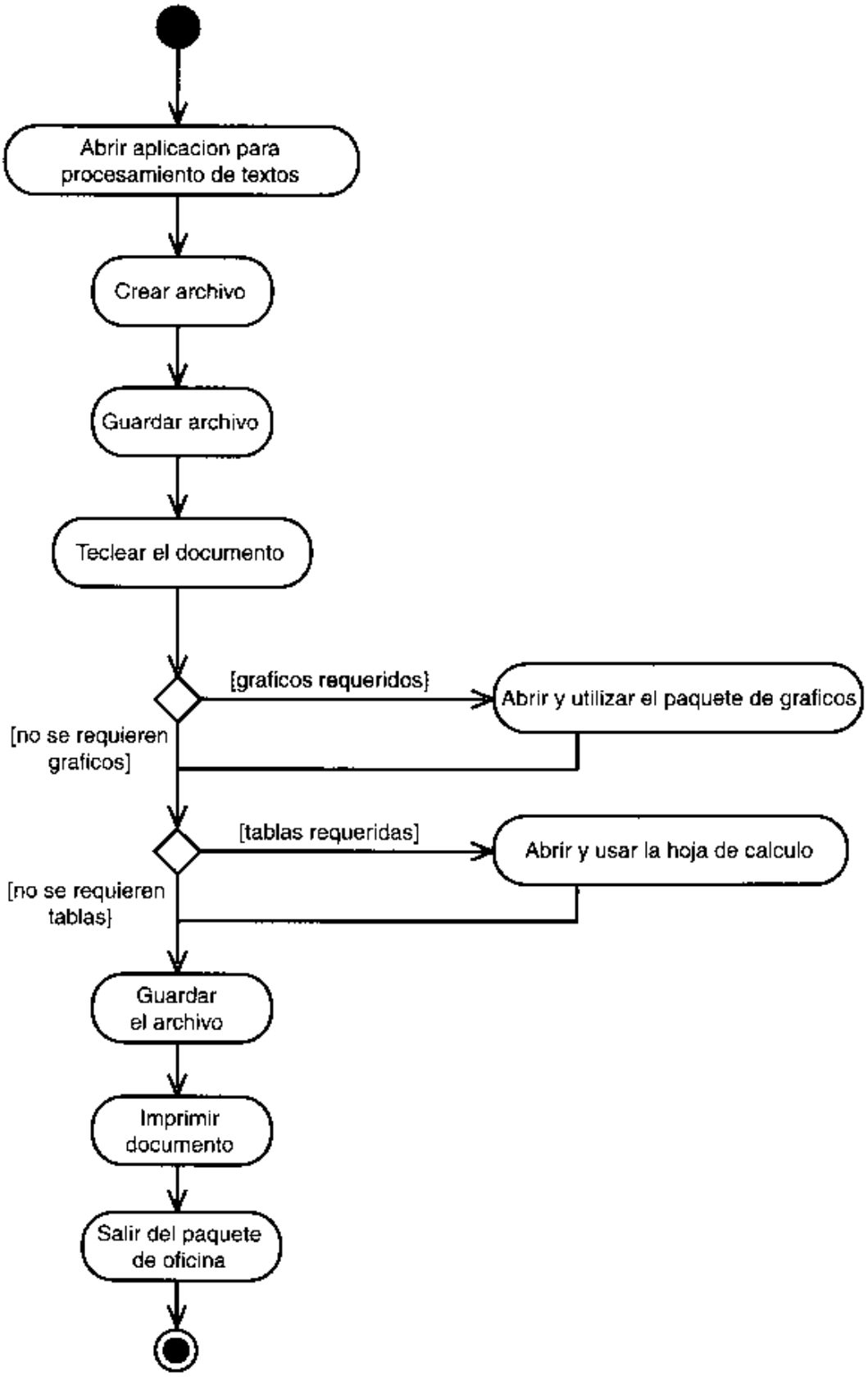
1. Abrir la aplicación para procesamiento de textos.
2. Crear un archivo.
3. Guardar el archivo con un nombre único en una carpeta.
4. Teclear el documento.
5. Si se necesitan ilustraciones, se abre la aplicación relacionada, se generan los gráficos y se colocan en el documento.
6. Si se necesita una hoja de cálculo, se abre la aplicación relacionada, se crea la hoja correspondiente y se coloca en el documento.

7. Se guarda el archivo.
8. Se imprime el documento.
9. Se sale de la aplicación de oficina.

El diagrama de actividades de esta secuencia aparece en la figura 11.6.

FIGURA 11.6

Un diagrama de actividades para el proceso de creación de un documento.



Marcos de responsabilidad

Uno de los aspectos más útiles del diagrama de actividades es su facultad para expandirse y mostrar quién tiene las responsabilidad en un proceso.

Veamos el caso de una firma de consultoría y el proceso de negociación involucrado en una junta con un cliente. Las actividades podrían ocurrir como sigue:

1. Un vendedor hace una llamada al cliente y concierta una cita.
2. Si la cita es en la oficina del consultor, los técnicos corporativos prepararán una sala de conferencias para hacer una presentación.
3. Si es en la oficina del cliente, un consultor preparará una presentación en una laptop.
4. El consultor y el vendedor se reunirán con el cliente en el sitio y a la hora convenidos.
5. El vendedor crea una minuta.
6. Si la reunión ha planteado la solución de un problema, el consultor creará una propuesta y la enviará al cliente.

Un diagrama de actividades estándar podría lucir como en la figura 11.7.

TÉRMINO NUEVO

El diagrama de actividades agrega la dimensión de visualizar responsabilidades. Para ello, separará el diagrama en segmentos paralelos conocidos como *marcos de responsabilidad*. Cada marco de responsabilidad muestra el nombre de un responsable en la parte superior, y presenta las actividades de cada uno. Las transiciones pueden llevarse a cabo de un marco a otro. La figura 11.8 muestra la versión con marcos de responsabilidad del diagrama de actividades de la figura 11.7.

Ambos diagramas de actividades de "Reunión con un cliente nuevo" muestran la creación de una propuesta como actividad. En cada caso, tal actividad podría tener una nota adjunta que cite al diagrama de actividades para la creación del documento.

FIGURA 11.7

Un diagrama de actividades para el proceso de negociación en una junta con un cliente.

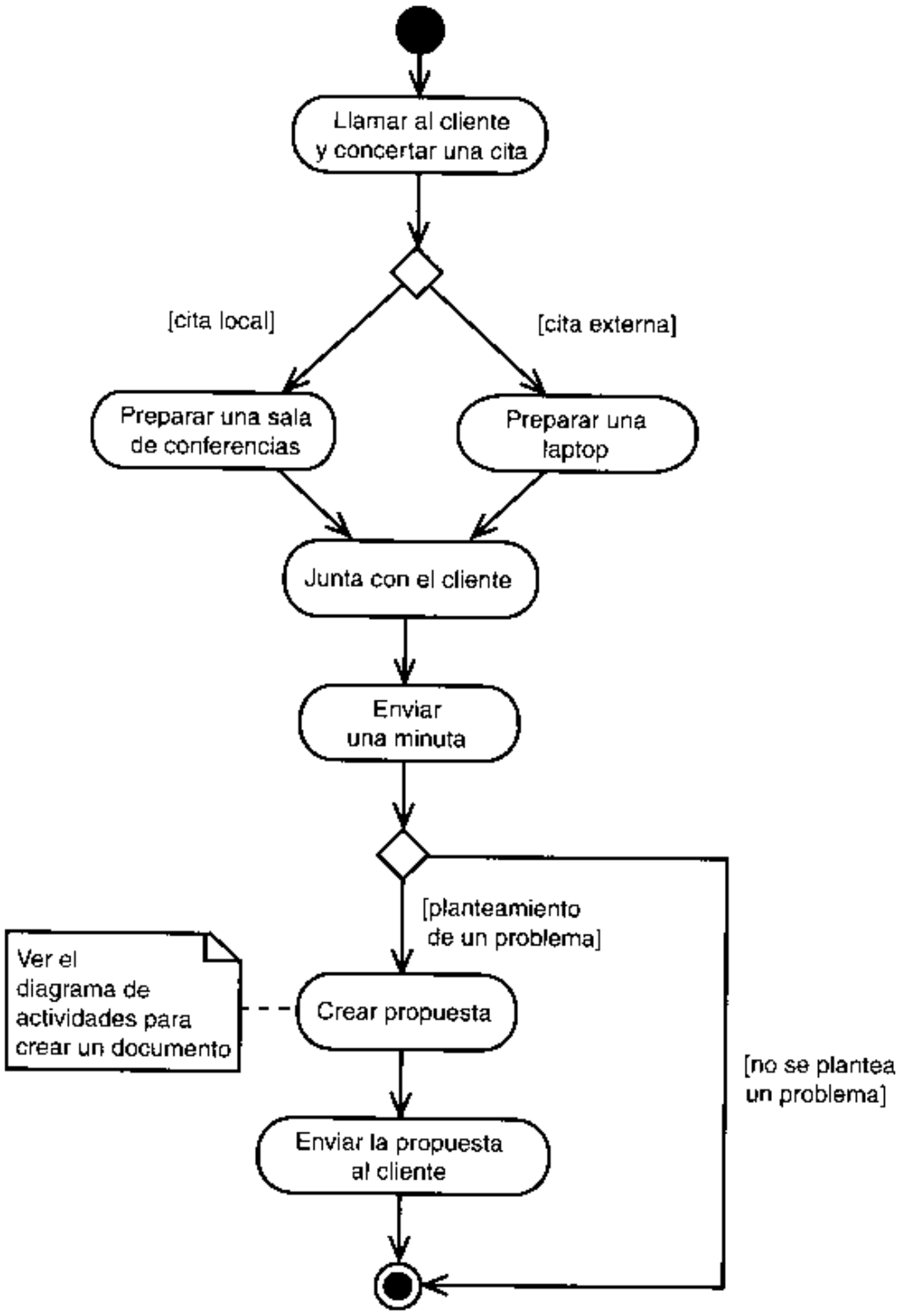
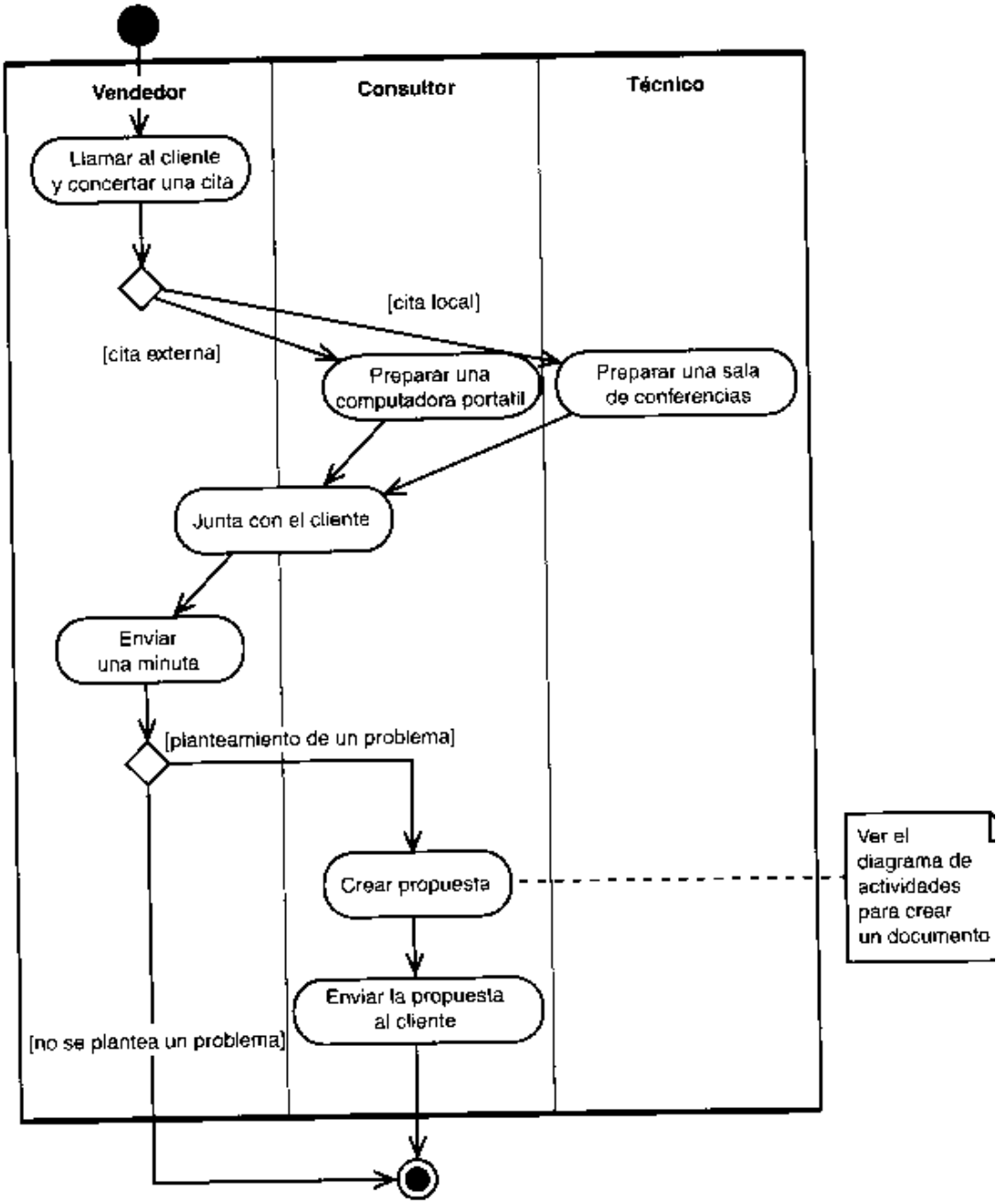


FIGURA 11.8

La versión con marcos de trabajo de diagrama de actividades de la figura 11.7 que muestra quién es el responsable de cada actividad.



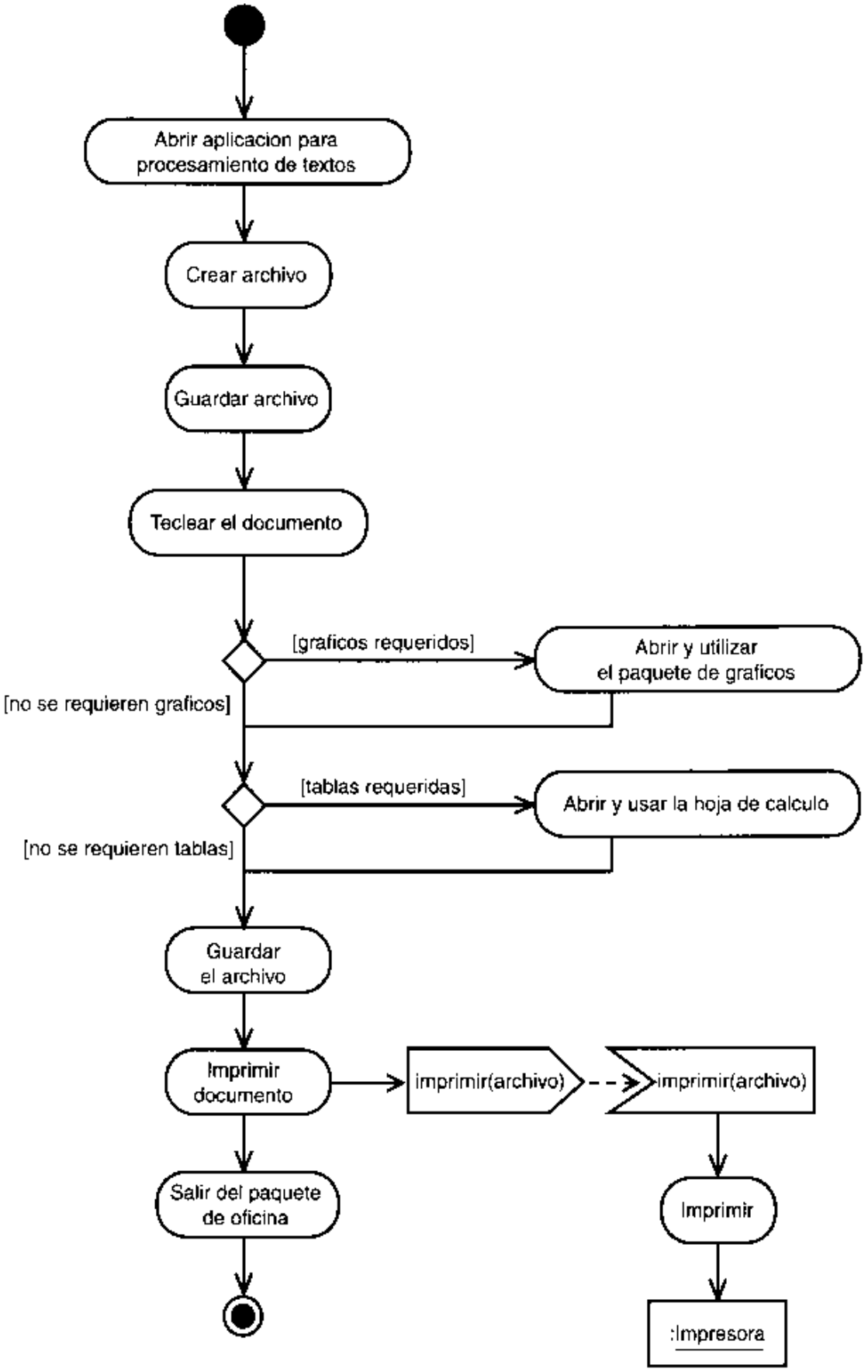
Diagramas híbridos

Recordemos el diagrama de actividades para la creación de un documento. Podrá depura la actividad de la impresión de un documento. En lugar de sólo mostrar una actividad "Imprimir documento", podría ser un poco más específico. La impresión se realiza dado que una señal dentro del archivo de documento se transmite desde la aplicación para el procesamiento de textos a la impresora, misma que la recibe y la imprime.

La figura 11.9 le muestra que podrá representar esto con los símbolos para la transmisió y recepción de señales, junto con un objeto Impresora que reciba al símbolo y realice su tarea de impresión. Éste es un ejemplo de diagrama híbrido, dado que contiene símbolo que normalmente asociaría con diferentes tipos de diagramas.

FIGURA 11.9

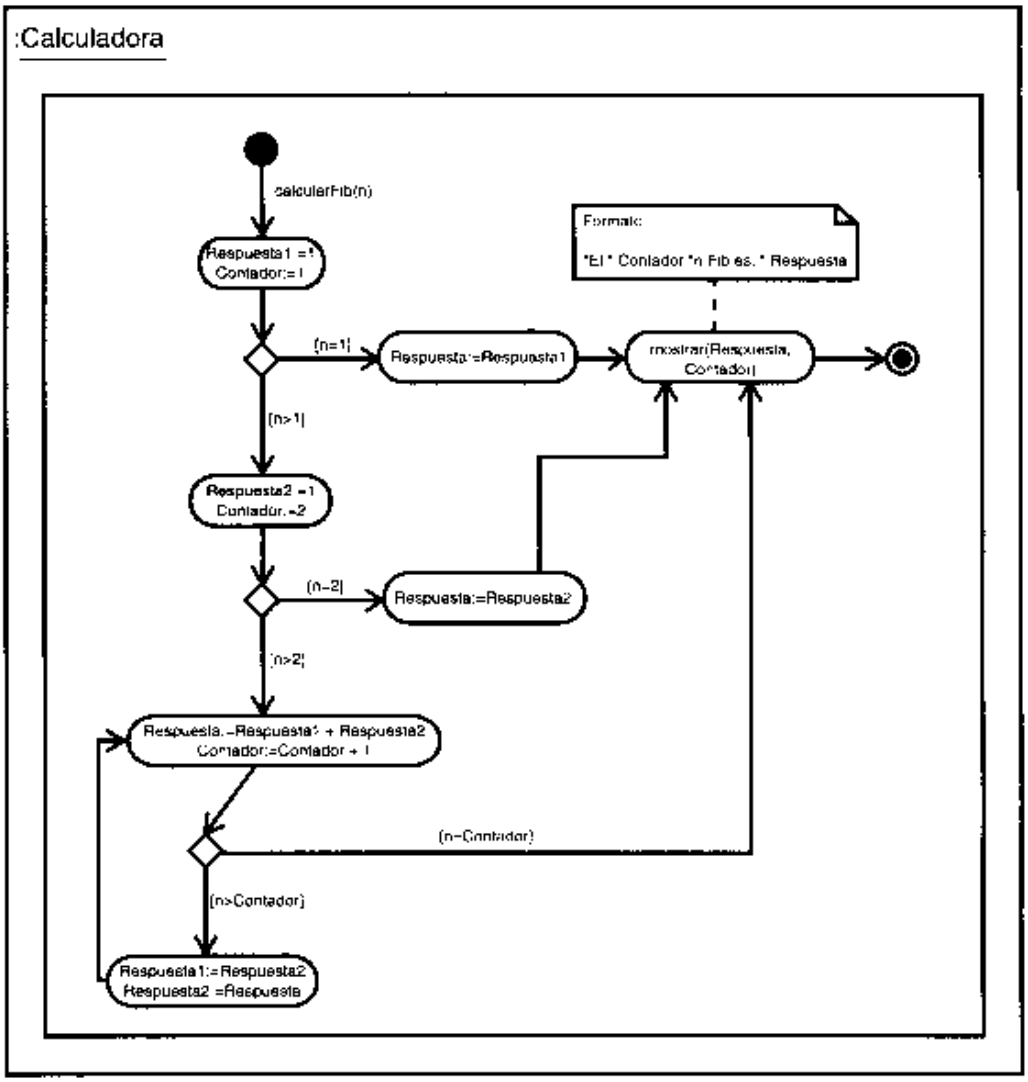
La depuración de la actividad "Imprimir documento" nos otorga un diagrama híbrido.



He aquí otra posibilidad de diagrama híbrido: podrá mostrar un diagrama de actividades para realizar una operación dentro de un símbolo de objeto, y mostrar al objeto que recibe una petición para ejecutar la operación. Suponga que modeló al objeto Calculadora, el cual calcula los números de Fibonacci. Los desarrolladores podrían encontrar útil que usted lo representara con un diagrama híbrido como el de la figura 11.10.

FIGURA 11.10

Un diagrama híbrido puede mostrar un diagrama de actividades dentro de un objeto.

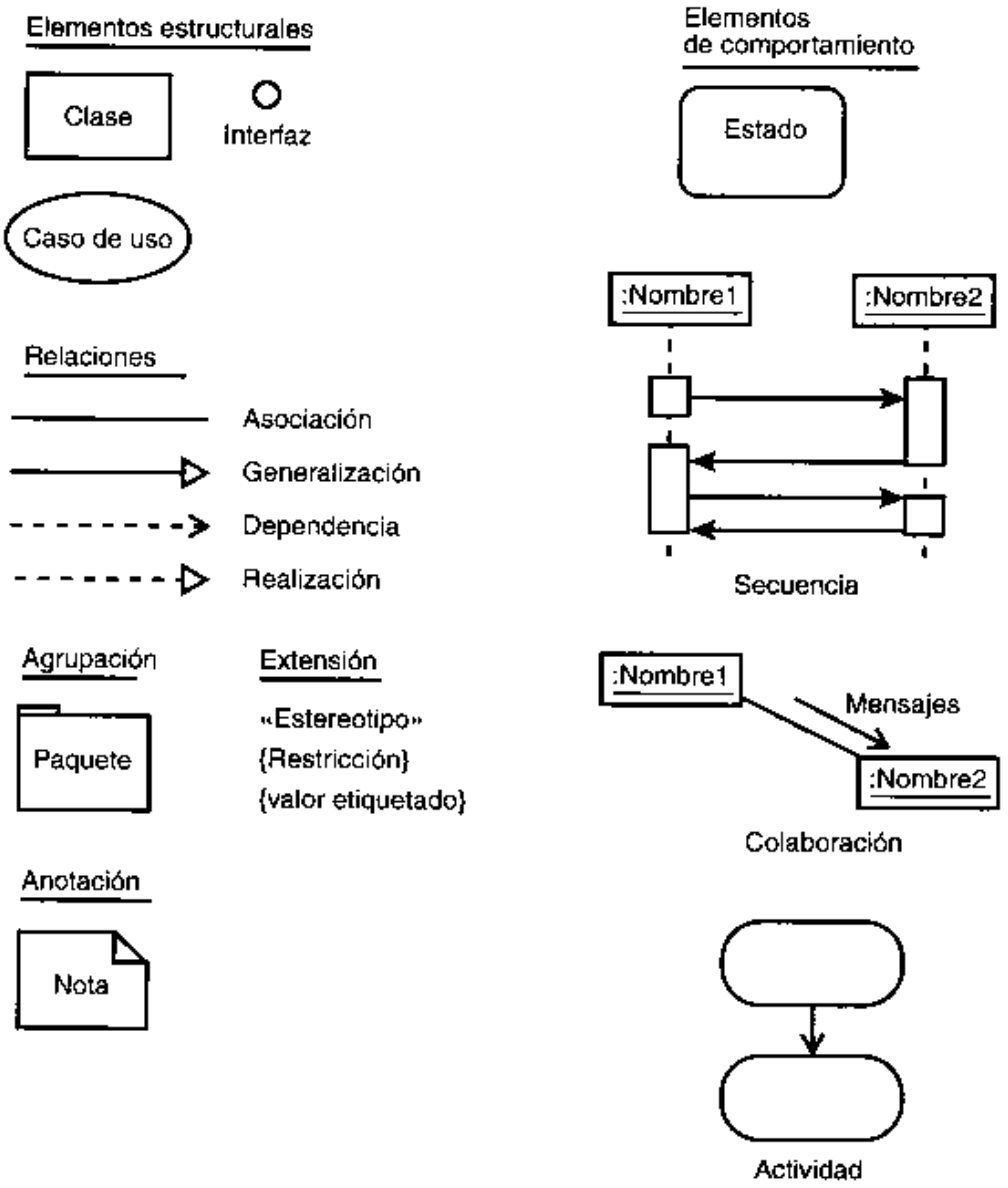


Adiciones al panorama

La figura 11.11 muestra el panorama del UML, donde se incluye el diagrama de actividades. Este diagrama es un elemento de comportamiento.

FIGURA 11.11

El panorama del UML ahora incluye al diagrama de actividades.



Resumen

El diagrama de actividades del UML es muy parecido a un diagrama de flujo. Muestra los pasos, puntos de decisión y bifurcaciones. Este tipo de diagrama es útil para representar las operaciones de un objeto y los procesos de negocios.

El diagrama de actividades es una extensión del diagrama de estados. Los diagramas de estados destacan los estados y representan actividades como flechas entre los estados. Los de actividad se enfocan en las actividades. Cada actividad se representa como un rectángulo con esquinas redondeadas, más ovalados en apariencia que la representación de un estado. El diagrama de actividades utiliza los mismos símbolos que el de estados para los puntos de inicio y final.

Cuando una ruta se divide en dos o más, tal dispersión se representa con una línea gruesa perpendicular a las rutas, mismas que se reúnen en una línea similar. Dentro de un diagrama de secuencias puede mostrar una señal, cuya transmisión se representa con un pentágono convexo, y la recepción con uno cóncavo.

En un diagrama de actividades, puede representar las actividades de acuerdo con la responsabilidad asignada. Esto lo haría con marcos de responsabilidad, mismos que son segmentos paralelos que corresponden a los responsables de realizar cada tarea.

Es posible combinar al diagrama de actividades con símbolos de otros diagramas con lo que se producirán diagramas híbridos.

Preguntas y respuestas

P Ésta es otra de esas preguntas de "¿Realmente lo necesito?". Con todo lo que nos muestra un diagrama de estados, ¿realmente necesito los de actividad?

R Yo le recomiendo que incluya los diagramas de actividades en su análisis. Pueden poner en claro algunos procesos y operaciones para usted y sus clientes. También son muy útiles para los desarrolladores. Es muy probable que un buen diagrama de actividades sea de gran utilidad para que un desarrollador codifique una operación.

P Ha mostrado dos tipos de diagramas híbridos. ¿El UML establece limitaciones en los tipos de híbridos que puede crear?

R No, en absoluto. El UML no intenta ser restrictivo. Aunque tiene algunas reglas sintácticas, la idea es que los analistas generen un modelo que transmita una idea consistente a los clientes, diseñadores y desarrolladores; no la de satisfacer ceñidas reglas lingüísticas. Si puede generar un diagrama híbrido que ayude a que todos los involucrados comprendan un sistema, hágalo.

Taller

El cuestionario y los ejercicios le harán razonar los diagramas de actividades y su utilidad. Las respuestas se encuentran en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cuáles son las dos formas de representar a un punto de decisión?
2. ¿Qué es un marco de responsabilidad?
3. ¿Cómo representaría la transmisión y recepción de una indicación?

Ejercicios

1. Cree un diagrama de actividades que muestre el proceso que realizaría al encender su automóvil. Empiece al colocar la llave en el switch, finalice con el motor en funcionamiento y tome en cuenta todo lo que haría si el motor no arrancara de inmediato.
2. ¿Qué podría agregar al diagrama de actividades en el proceso de negociación de la junta con un cliente nuevo?
3. Si distribuye tres piedras de modo que una de ellas se ubique en una fila y las otras dos en otra, formarían un triángulo. Si hace lo mismo con seis piedras, de manera que se ubiquen: una en una fila, dos en la siguiente, y tres en la última, también formarían un triángulo. Por ello, a los número 3 y 6 se les conoce como *números triangulares*. El siguiente número triangular es 10, el que le sigue es 15, y así por el estilo. El primer número triangular es 1. Cree dos diferentes diagramas de actividades para un proceso que calcule el enésimo número triangular. Para uno, inicie con *n* y continúe en orden regresivo. Para el otro, inicie con 1 y continúe en orden progresivo.
4. He aquí un ejercicio para quienes les gustan las matemáticas. Si se sintió a gusto con el ejercicio 3, tal vez le guste este otro. Si no, tan sólo continúe con la siguiente hora (¡podría intentar diagramar lo que dije en las dos últimas oraciones!). En la geometría coordinada, se representa un punto en el espacio al mostrar su posición x y y. Por ello, puede decir que la ubicación del punto 1 es X1, Y1. La del punto 2 es X2, Y2. Para encontrar la distancia entre estos puntos, eleve al cuadrado X2-X1 y luego Y2-Y1. Sume ambos cuadrados y calcule la raíz cuadrada de la suma. Cree un diagrama de actividades para una operación *distancia (X1, Y1, X2, Y2)* que localice la distancia entre dos puntos.

HORA 12

Diagramas de componentes

En las horas anteriores ha visto diagramas que tratan temas conceptuales. Un diagrama de clases representa un concepto, es decir, la abstracción de elementos que confluyen en una categoría; un diagrama de estados también representa un concepto, como son los cambios en el estado de un objeto.

Ahora verá el uso de un diagrama que es algo distinto a los que ha visto, y aprenderá lo correspondiente a un diagrama UML que representa a una entidad real: un componente de software.

En esta hora se tratarán los siguientes temas:

- Qué es un componente
- Componentes e interfaces
- Qué es un diagrama de componentes
- Aplicación de los diagramas de componentes
- Diagramas de componentes en el panorama del UML

Qué es un componente

Un componente de software es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista. ¿Qué puede tomarse como componente? Una tabla, archivo de datos, ejecutable, biblioteca de vínculos dinámicos, documentos y cosas por el estilo.

¿Cuál es la relación entre un componente y una clase? Imagine a un componente como la personificación en software de una clase. La clase representa una abstracción de un conjunto de atributos y operaciones. Un importante punto por recordar de los componentes y clases es que un componente puede ser la implementación de más de una clase.

Si el componente se encuentra en una computadora y es la parte funcional del sistema, ¿para qué preocuparse por él? Tendrá que modelar componentes y sus relaciones para que:

1. Los clientes puedan ver la estructura del sistema finalizado.
2. Los desarrolladores cuenten con una estructura con la cual trabajar en adelante.
3. Quienes escriban las notas técnicas y la documentación puedan entender de qué escribirán.
4. Usted se aliste para volver a utilizar los componentes.

Exploremos el último punto. Uno de los aspectos más importantes de los componentes es el potencial que tienen de volver a ser utilizados. Con las necesidades actuales de los negocios de soluciones rápidas, entre más rápido presente un sistema para producción, mayor será su competitividad. Si puede crear un componente para un sistema y puede volver a utilizarlo en otro, usted habrá contribuido a esa competitividad. Tómese el tiempo y esfuerzo para modelar un componente que lo ayude a que esta reutilización pueda llevarse a cabo.

Recordaremos la reutilización al final de la siguiente sección.

Componentes e interfaces

Cuando trate con los componentes, tendrá que tratar con sus interfaces; al tratar el tema de las clases y los objetos, hablé de las interfaces. Como recordará en la hora 2, "Orientación a objetos", un objeto oculta al mundo exterior lo que hace (lo que se conoce como *encapsulación*, *encapsulamiento* u *ocultamiento de información*). El objeto tiene que presentar un "rostro" al mundo exterior, para que los demás objetos (incluso, potencialmente, los humanos) puedan pedirle que ejecute sus operaciones. A este "rostro" se le conoce como *interfaz* del objeto.

TÉRMINO NUEVO

Di mayores detalles de esta idea en la hora 5, "Agregación, composición, interfaces y relación". Como lo mencioné en su momento, diversas clases podrían no estar relacionadas con una clase principal (como en la herencia), pero sus acciones podrían incluir algunas de las mismas operaciones con las mismas firmas. Podrá reutilizar

este conjunto de operaciones de clase en clase. La *interfaz* es la construcción UML que le permite hacer esto. Una interfaz es un conjunto de operaciones que especifica algo respecto al comportamiento de una clase. Imagine a una interfaz como una clase que sólo contiene operaciones (no atributos). Para resumir: la interfaz es un conjunto de operaciones que presenta una clase a otras.

Al tratar las interfaces en la hora 5, también mencioné que la relación entre una clase y su interfaz se conoce como *realización*.

¡Un momento! Parecería que modelar una interfaz es la práctica de modelar un concepto. Al principio de esta hora, le dije que cuando modele un componente no será algo conceptual, dado que se encontrará en una computadora. ¿Dónde está la conexión?

En sí, una interfaz puede ser física o conceptual. La interfaz que utiliza una clase es la misma que la que utiliza su implementación de software (un componente). Como modelador, esto significa que la de la misma forma en que represente una interfaz para una clase representará una interfaz para un componente. Aunque la simbología del UML distingue entre una clase y un componente, no hace distinción entre una interfaz conceptual y una física.

TÉRMINO NUEVO Hay un punto importante a este respecto: sólo podrá ejecutar las operaciones de un componente a través de su interfaz. De la misma manera que en el caso de una clase y su interfaz, la relación entre un componente y su interfaz se conoce como *realización*.

TÉRMINO NUEVO Hay otro punto por destacar: un componente puede hacer disponible su interfaz para que otros componentes puedan utilizar las operaciones que contiene. Es decir, un componente puede acceder a los servicios de otro componente. El componente que proporciona los servicios se dice que provee una *interfaz de exportación*. Al que accede a los servicios se dice que utiliza una *interfaz de importación*.

Sustitución y reutilización

Las interfaces se destacan de forma importante en los conceptos primordiales de sustitución y reutilización de componentes. Puede sustituir un componente con otro si el nuevo contiene las mismas interfaces que el anterior. Podrá reutilizar un componente en otro sistema si éste puede acceder al componente reutilizado mediante sus interfaces. Puede diseñar un componente para ser reutilizado en proyectos de desarrollo a lo largo de su empresa si quiere depurar sus interfaces para que un amplio rango de componentes puedan acceder a ellos.

Es aquí donde son útiles las interfaces en el modelado. Puede simplificarse la vida de un desarrollador que intente sustituir o reutilizar un componente si la información de su interfaz se encuentra disponible como un modelo. Si no, el desarrollador tendrá que pasar por el largo proceso de hacer un seguimiento del código.

Tipos de componentes

Conforme avance en su carrera como modelador, se encontrará con tres tipos de componentes:

1. *Componentes de distribución*, que conforman el fundamento de los sistemas ejecutables (por ejemplo, DLL, ejecutables, controles ActiveX y Java Beans).
2. *Componentes para trabajar en el producto*, a partir de los cuales se han creado los componentes de distribución (como archivos de base de datos y de código).
3. *Componentes de ejecución*, creados como resultado de un sistema en ejecución.

Si es usted un usuario de Windows, encontrará ejemplos de los tres tipos de componentes cuando utilice la ayuda, aunque tal vez no lo sepa. El componente de distribución es el archivo .HLP (Archivo de Ayuda): Cuando haga clic en uno, se abrirá el cuadro de diálogo de los temas de la ayuda y empezará a buscar el tema que elija. La primera vez que haga clic en la ficha Buscar, verá una pequeña animación que le indicará que se está creando un índice. Un archivo .CNT (tema de contenido) describe el esquema del contenido. Dado que este archivo le ayuda a crear un componente de distribución (puede utilizar la página de índice por siempre), es un componente para trabajar en el producto. A su vez, el índice creado se encuentra en un archivo .FTS (búsqueda de texto completo). Finalmente, la primera vez que abra la ayuda, se creará un archivo .GID (índice general), que es resultado de un análisis del sistema de ayuda de Windows que agiliza el acceso a los temas del archivo de ayuda. Con ello, los archivos .FTS y .GID son componentes de ejecución.

Qué es un diagrama de componentes

Un diagrama de componentes contiene, obviamente, componentes, interfaces y relaciones. También pueden aparecer otros tipos de símbolos que ya haya visto.

Representación de un componente

El símbolo principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo. La figura 12.1 le muestra este símbolo. Debe colocar el nombre del componente dentro del símbolo. El nombre es una cadena.

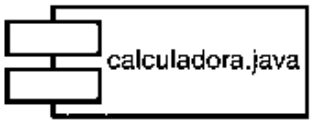
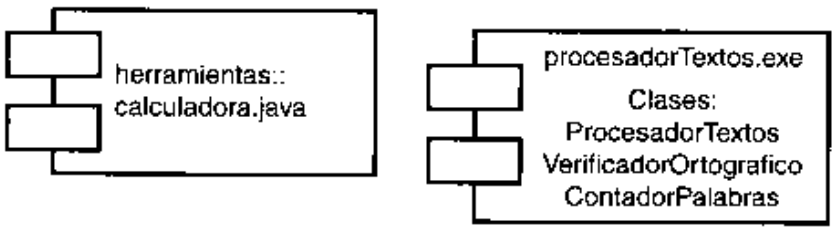


FIGURA 12.1
El símbolo que representa a un componente.

La figura 12.2 le muestra que si el componente es miembro de un paquete, puede utilizar el nombre del paquete como prefijo para el nombre del componente. También puede agregar información que muestre algún detalle del componente.

FIGURA 12.2

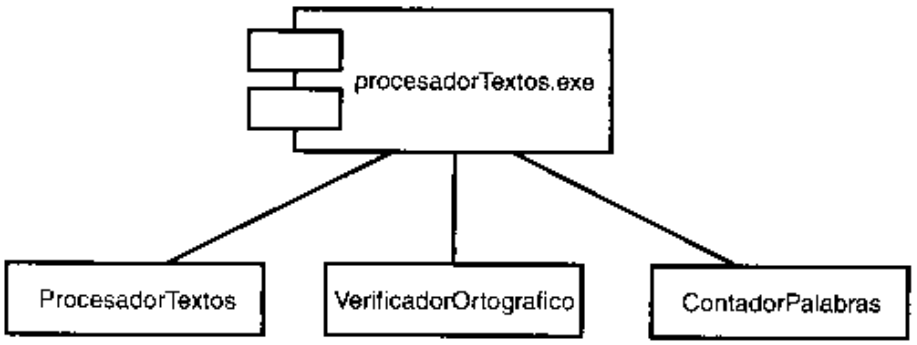
Adición de información al símbolo del componente.



El símbolo a la derecha de la figura 12.2 le muestra las clases que implementa un componente en particular. La figura 12.3 le muestra otra forma de hacer esto, aunque esta técnica por lo general desordena el diagrama. Vea las relaciones de dependencia entre el componente y las clases.

FIGURA 12.3

Símbolos de las relaciones entre un componente y las clases que implementa.

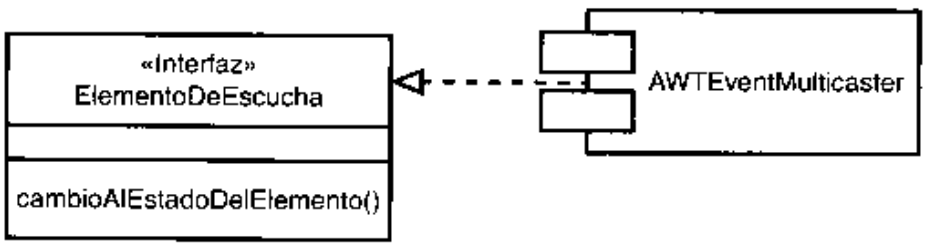


Cómo representar las interfaces

Existen dos formas para representar a un componente y sus interfaces: la primera muestra la interfaz como un rectángulo que contiene la información que se le relaciona, se conecta al componente por la línea discontinua y una punta de flecha representada por un triángulo sin rellenar que visualiza la realización (vea la figura 12.4).

FIGURA 12.4

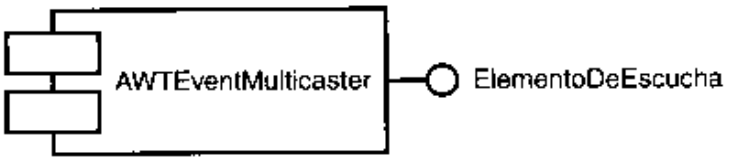
Puede representar a una interfaz como un rectángulo con información, conectado al componente por una flecha de realización.



La figura 12.5 le muestra la segunda forma de representar a un componente y sus interfaces; esta forma es representativa, ya que representará la interfaz como un pequeño círculo que se conecta al componente por una línea continua. En este contexto, la línea representa la relación de realización (compare a las figuras 12.4 y 12.5 con las figuras 5.6 y 5.7).

FIGURA 12.5

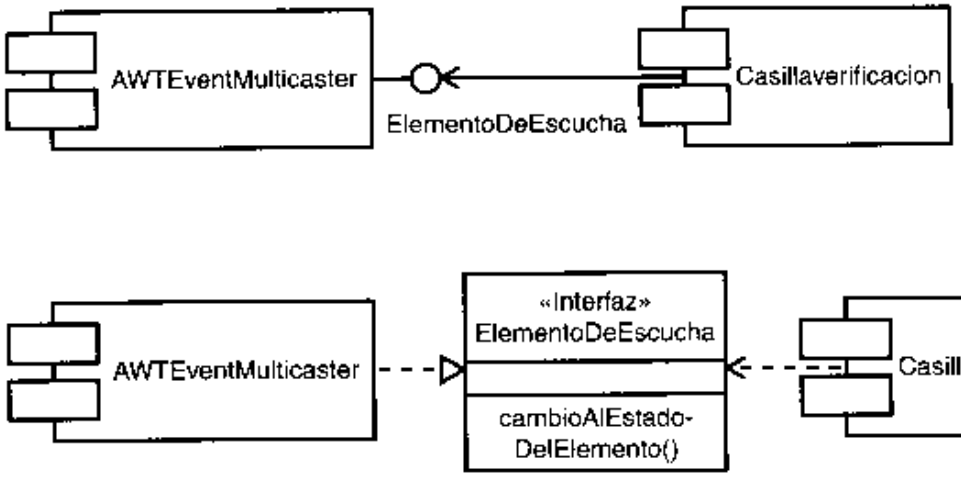
Puede representar a una interfaz como un pequeño círculo, conectado al componente por una línea continua que, en este contexto, se interpreta como realización.



Además de la realización, puede representar a la dependencia, que es la relación entre un componente y una interfaz de importación. Como recordará, la dependencia se vislumbra como una línea discontinua con una punta de flecha. Puede mostrar la realización y la dependencia en el mismo diagrama, como se ve en la figura 12.6.

FIGURA 12.6

Una interfaz que realiza un componente y otra de la que depende.



Aplicación de los diagramas de componentes

Algunos ejemplos le ayudarán a empezar a usar los diagramas de componentes. El primero es un modelo de una página Web que representa a un componente Java. El siguiente también es un modelo de una página Web pero éste utiliza controles ActiveX. Finalizará con un modelo de un paquete de Microsoft conocido como PowerToys. Este paquete, que puede obtener del sitio de Microsoft, le permite modificar aspectos de Win32.

Una página Web con un subprograma Java

Este ejemplo modela un programa tomado del excelente y entretenido libro de Rogers Cadenhead llamado *Aprendiendo programación con Java 1.1 en 24 horas* (Prentice Hall

Hispanoamericana, 1997). El ejemplo aparece en la hora 22, "Escriba un juego para Web". Rogers muestra cómo generar un applet (o subprograma) que ejecuta el juego de dados "Craps" en una página Web, y utiliza una clase llamada "Die" (para crear los dados) de la hora 21, "A jugar con Java". Para ver los detalles del código, tendrá que leer el libro. Aquí sólo nos concentraremos en los componentes.

Término Nuevo Un applet es un pequeño programa hecho en Java que funciona en una página Web.

La página Web se llama Craps.html. El código fuente del applet se encuentra en el archivo Craps.java, y el código objeto es el archivo Craps.class. El código fuente de la clase Die se encuentra en Die.java y el código objeto en Die.class. Los cinco archivos se encuentran en el mismo directorio —que llamaremos Tirodedados (que no es el nombre que Roger le dio)—.

Craps.html depende, obviamente, de Craps.class y Die.class. Cada archivo .class es un componente y cada uno es la implementación de una clase. Lo que no es muy obvio (tendría que ver el código fuente para descubrirlo) es que tanto Craps.java como Die.java importan (utilizan las clases de) java.awt, un grupo de clases que muestran y controlan la GUI (el "awt" significa: "Conjunto de herramientas abstractas para manejar ventanas").

En el contexto de Java, la *importación* permite al desarrollador utilizar sólo el nombre de una operación cuando la escribe en un programa, en lugar de utilizar toda la ruta de la operación (que podría ser muy larga). La importación no "incrusta" una clase en otra. Tan sólo permite escribir algo más corto.

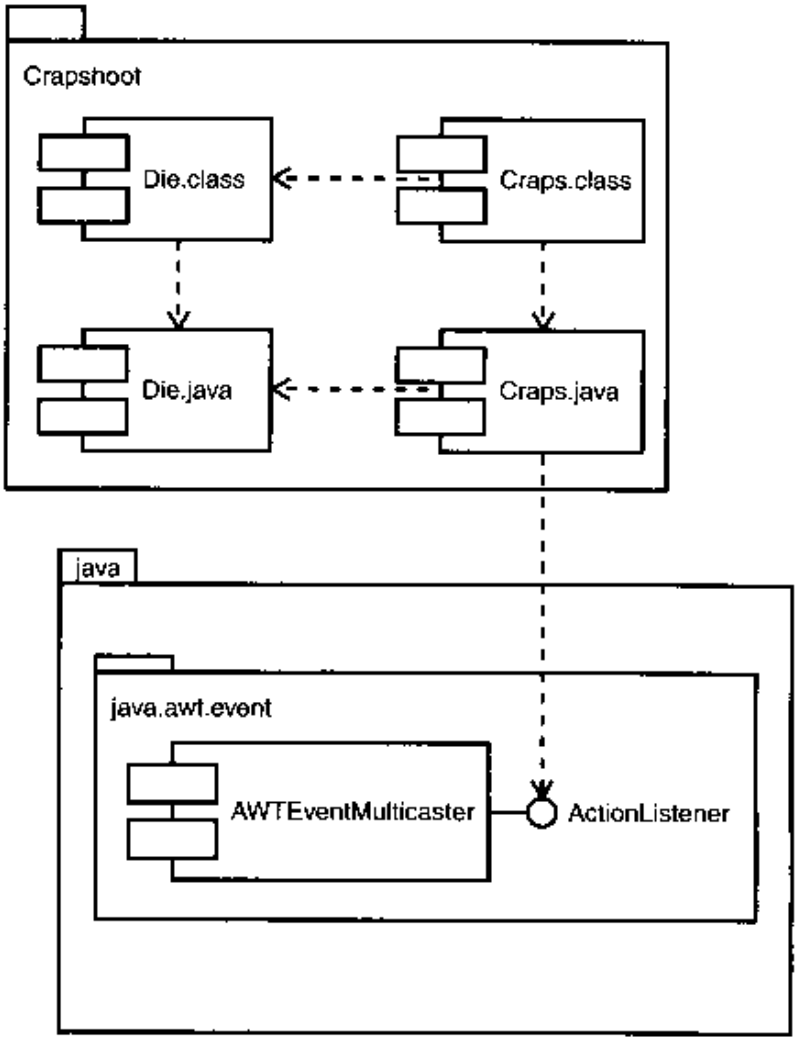
Craps.java es un applet, y por ello se hereda desde la clase java.applet.Applet. Finalmente, Craps.java importa a java.awt.event e implementa una interfaz ActionListener (para responder a los eventos generados por el usuario, como hacer clic con el ratón).

Dentro del código, la interfaz ActionListener proporciona un botón para que el usuario haga clic para que se "tiren los dados". Al ver la referencia de Java, notará que la clase java.awt.AWTEventMulticaster implementa esta interfaz.

¿Sabe qué? ¡Esto sería más fácil de comprender si viera el modelo UML! La figura 12.7 le muestra el diagrama de componentes. Un paquete corresponde al directorio en donde se encuentran los archivos, el otro al JDK (Conjunto de herramientas para desarrollo en Java).

FIGURA 12.7

El diagrama para el juego de dados basado en la Web de Rogers Cadenhead.



TERMINO NUEVO Este ejercicio —generar un modelo a partir de un código existente— se conoce como *ingeniería inversa*.

Una página Web con controles ActiveX

ActiveX es el medio de Microsoft para agregar componentes a las aplicaciones. Con tantos tipos de componentes ActiveX (controles) disponibles, podrá encontrar alguno que haga casi todo lo que requiera una aplicación. Una propiedad de un componente ActiveX es su número de identificación hexadecimal único de 32 bits, conocido como CLSID (identificador de la clase).

Si sus requerimientos son especiales, podrá generar su propio componente ActiveX en Visual Basic o Visual C++. Luego podrá reutilizarlo de aplicación en aplicación.

En las páginas Web, los componentes ActiveX se encuentran y trabajan con el código escrito en algún lenguaje para secuencias de comandos como VBScript. En este ejemplo, la página Web cuenta con un control Timer ActiveX, dos cuadros combinados ActiveX y tres botones ActiveX. La página Web permite a un usuario establecer los parámetros para

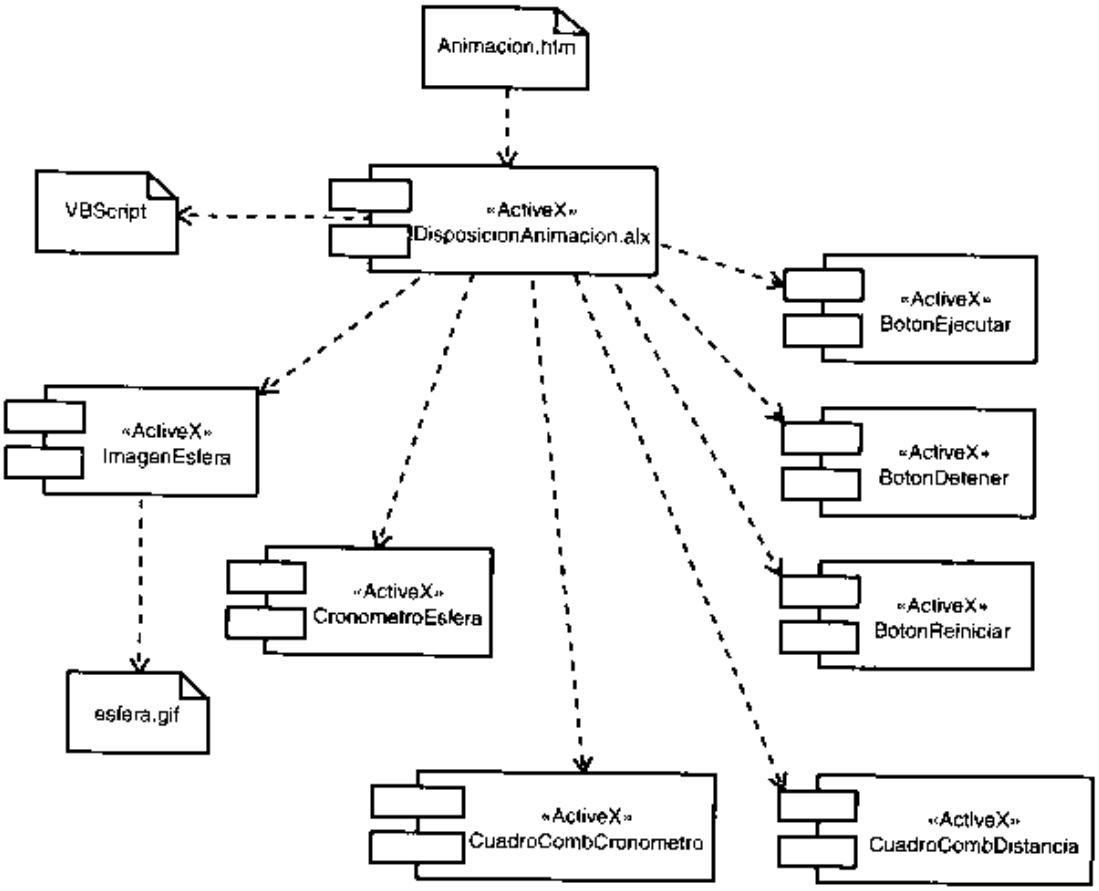
animar el movimiento de una esfera (una imagen .gif) por la pantalla. De un cuadro combinado, el usuario selecciona la cantidad de píxeles por movimiento. Del otro se selecciona la cantidad de milisegundos entre movimientos. Un botón iniciará el movimiento, el otro lo detendrá y el tercero restaurará la esfera a su posición inicial. El cronómetro moverá la esfera cuando pase la cantidad de milisegundos elegida por el usuario.

Los controles ActiveX se encuentran en un componente separado conocido como *Disposición* (Layout). La página HTML y la disposición se encuentran en el mismo directorio.

La figura 12.8 le muestra el diagrama de componentes para esta página. Observe el uso del símbolo de anotación para representar al VBScript. Aunque esto no es absolutamente necesario, destaca una diferencia entre el lenguaje de secuencias de comandos y los componentes compilados ActiveX.

FIGURA 12.8

El diagrama de componentes para una página Web con componentes ActiveX.



PowerToys

Si utiliza cualquier versión de Win32, ya conocerá las horribles flechas pequeñas que se encuentran en la esquina inferior izquierda de cada icono de acceso directo. Microsoft tiene un paquete llamado PowerToys que le permite eliminarlas y hacer varias otras cosas con la GUI, mediante una aplicación llamada TweakUI que es parte del paquete.

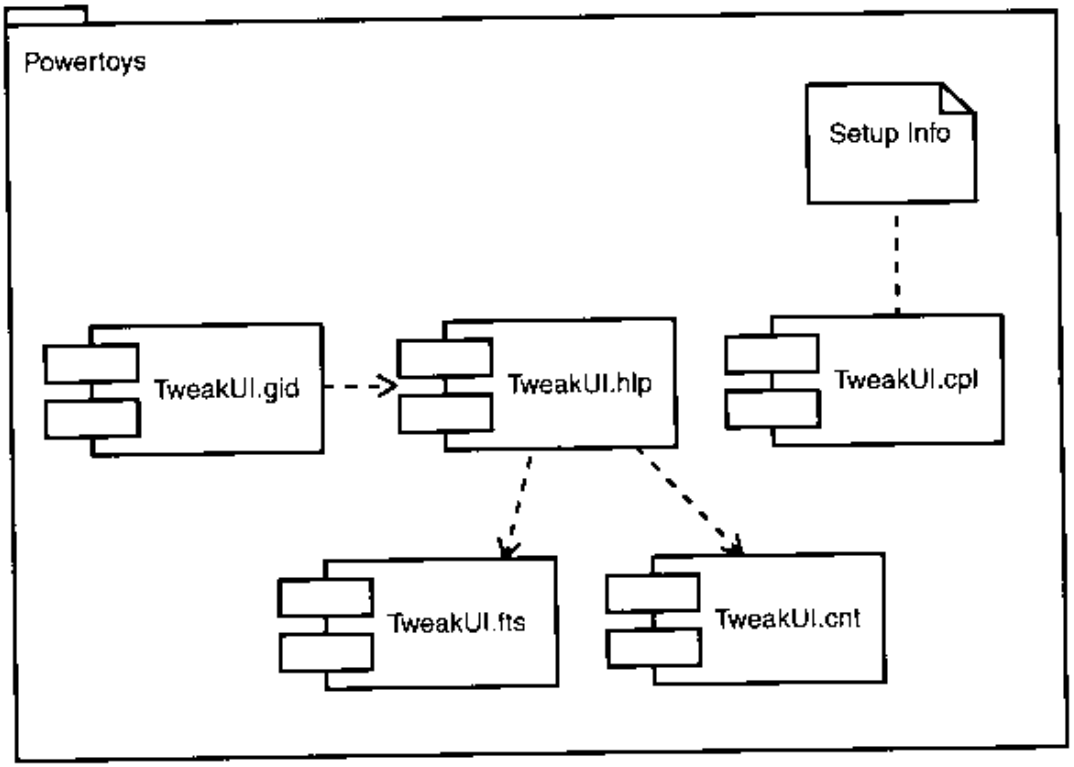
Puede obtener a PowerToys del sitio Web de Microsoft. Es gratis. Cuando lo obtenga y descomprima, verá varios archivos con extensiones .dll. También verá un archivo de

ayuda y otro .CNT. Haga clic en el archivo de ayuda y generará un archivo .GID. Utilice la característica Buscar y creará un archivo .FTS.

La figura 12.9 le muestra un diagrama de componentes que modela a TweakUI en el paquete PowerToys, mismo que muestra las dependencias entre los diversos tipos de componentes.

FIGURA 12.9

Modelado de TweakUI en el paquete PowerToys.

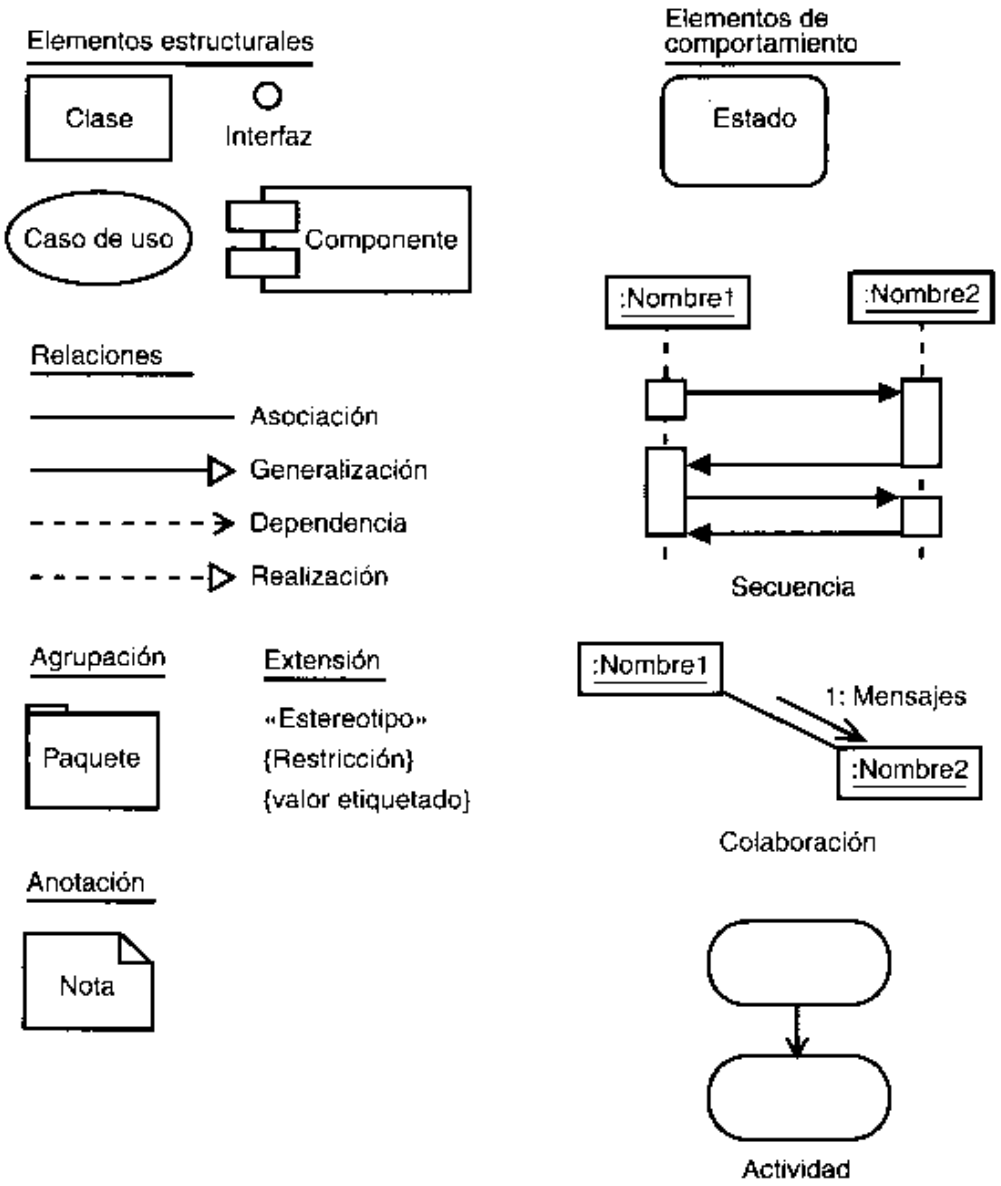


Diagramas de componentes en el panorama

Ya casi tiene todo su panorama. La figura 12.10 incluye el diagrama de componentes, que se enfoca en una arquitectura de software del sistema. En la siguiente hora, verá cómo modelar la arquitectura de hardware.

FIGURA 12.10

Su panorama del UML ahora incluye al diagrama de componentes.



Resumen

El diagrama de componentes UML es un conglomerado de figuras de los diagramas que ya ha visto. En lugar de representar una entidad conceptual como una clase o estado, un diagrama de componentes representa a un elemento real: un componente de software. Estos componentes se encuentran en las computadoras, no en la mente del analista.

Un componente puede accederse a través de su interfaz, una colección de operaciones. La relación entre un componente y su interfaz se llama *realización*. Un componente puede acceder los servicios de otro. Cuando se hace, utiliza una *interfaz de importación* . El componente que realiza la interfaz con tales servicios proporciona una *interfaz de exportación.*

La representación de un componente es un rectángulo con otros dos rectángulos pequeños sobrepuestos en su lado izquierdo. Puede representar una interfaz de dos formas: la primera es un rectángulo que contiene información de la interfaz y se conecta con el componente mediante una línea discontinua con una punta de flecha representada por triángulo sin relleno. La otra es un pequeño círculo conectado al componente con una línea continua. Ambos tipos de conexión pretenden mostrar una relación de realización.

Preguntas y respuestas

P En un diagrama de componentes, ¿cuál será la regla de oro para usar símbolos que no representen a componentes?

R Esto lo hará cuando desee indicar algo que sea ciertamente distinto de un componente compilado. No es necesario, pero podría ayudar a tener otro punto de vista. Podría utilizar el símbolo de la anotación para representar archivos de encabezado, dll, o archivos de secuencias de comandos. Otra posibilidad es la de utilizar el símbolo regular del componente con un estereotipo que indique el tipo de archivo.

P Ha utilizado a VBScript como un componente de la página Web. El código de VBScript consta de varios procedimientos. ¿No podría modelar cada uno como componente?

R Sí, podría. No obstante, podría desordenar su modelo si depurara al VBScript (o JavaScript) hasta tal nivel, así que podría, mejor, agregar una nota que abunde al respecto.

Taller

En este taller reforzará su conocimiento respecto a los componentes y cómo modelarlos. Uno de los ejercicios trata de los conceptos de una página Web, el otro de DLL. El Apéndice A, "Respuestas a los cuestionarios", será el componente del libro que contenga las respuestas.

Cuestionario

1. ¿Cuáles son los tres tipos de componentes?
2. ¿Cómo llamaría a la relación entre un componente y su interfaz?
3. ¿Cuáles son las dos formas de representar a esta relación?
4. ¿Qué es una interfaz de exportación? ¿Que es interfaz de importación?

Ejercicios

1. Adentrémonos en la ingeniería inversa para modelar una página Web. Visite `http://www.pearson.com.mx`, la página Web de Pearson Educación (la empresa que amablemente publicó el libro que ahora lee). Utilice el menú Ver de su explorador para seleccionar la opción que le deje a la vista el código fuente. No encontrará ningún componente ActiveX o applet de Java, pero verá diversos archivos .gif y cierto código en JavaScript. No incluya todos los archivos .gif en su modelo, sólo algunos para reafirmar su comprensión. Vaya al principio del código y verá una referencia a una hoja de estilo. La referencia se encuentra en un elemento LINK de HTML. Este es un archivo por separado que contiene la información de estilo para la página Web. Asegúrese de incluirla en su modelo.

2. En el diagrama de colaboraciones del caso de uso "Crear propuesta", el consultor busca en el área central de almacenamiento una propuesta adecuada para volverla a utilizar. Imagine a "buscar" como un mensaje enviado para buscar en una secuencia de archivos, y utilice las técnicas de modelado de la sección "Algunos conceptos más" para cambiar el diagrama de colaboraciones en la figura 10.6.

HORA 13

Diagramas de distribución

Hasta ahora nos hemos concentrado en el entorno conceptual, aunque en la hora anterior vimos los modelos de la arquitectura de software. Es momento de concentrarnos en el hardware. Como podrá ver, hemos trascendido desde los elementos (como las clases) que se encuentran en los análisis, hasta los componentes en los equipos de cómputo y al hardware existente.

Claro está que el hardware es un tema primordial en un sistema de varios componentes. En el mundo actual de la computación, un sistema podría abarcar diversos tipos de plataformas en ubicaciones dispersas. Un diseño sólido de distribución de hardware es básico para el diseño del sistema. El UML le da los símbolos para crear una imagen clara de la forma en que deberá lucir el hardware final.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de distribución
- Aplicación de los diagramas de distribución
- Los diagramas de distribución en el panorama del UML

Qué es un diagrama de distribución

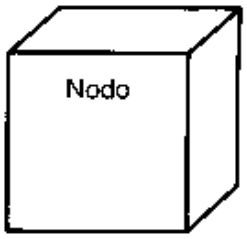
TERMINO NUEVO

El elemento primordial del hardware es un *nodo*, que es un nombre genérico para todo tipo de recurso de cómputo. Es posible usar dos tipos de nodos: un procesador, el cual puede ejecutar un componente, y un dispositivo que no lo ejecuta. Normalmente, un dispositivo (como impresora o monitor) tiene contacto de alguna forma con el mundo exterior.

En el UML, un cubo representa a un nodo. Deberá asignar un nombre para el nodo, y podrá utilizar un estereotipo para indicar el tipo de recurso que sea. La figura 13.1 le muestra un nodo.

FIGURA 13.1

Representación de un nodo en el UML.



El nombre es una cadena de texto. Si el nodo es parte de un paquete, su nombre puede contener también el del paquete. Puede dividir al cubo en compartimientos que agreguen información (como componentes colocados en el nodo), como en la figura 13.2.

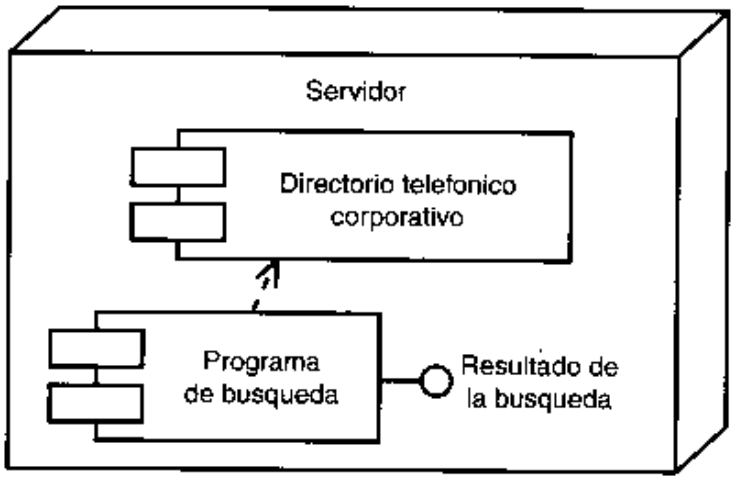
FIGURA 13.2

Cómo agregar información a un nodo.

Otra forma de indicar los componentes distribuidos es la de mostrarlos en relaciones de dependencia con un nodo (vea la figura 13.3).

FIGURA 13.3

Puede mostrar los componentes en relaciones de dependencia con un nodo.

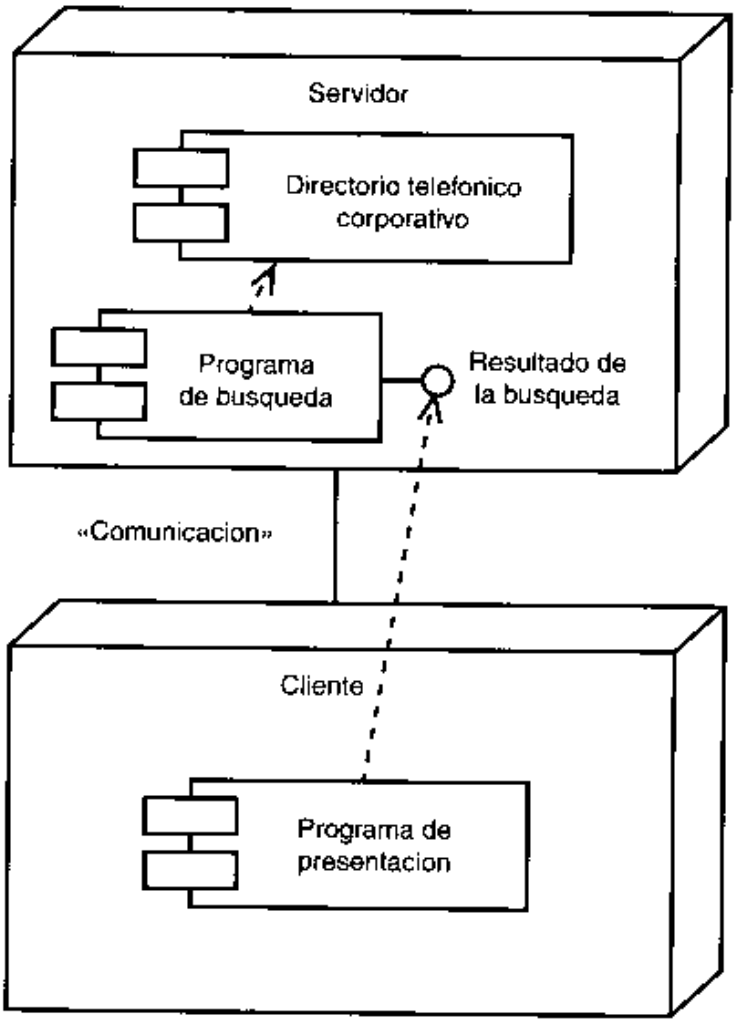


Una línea que asocie a dos cubos representará una conexión entre ellos. Podrá utilizar un estereotipo para dar información respecto a la conexión. La figura 13.4 proporciona ejemplos de conexiones entre nodos.

Tenga en cuenta que una conexión no es necesariamente un cable o alambre. También puede visualizar conexiones inalámbricas como las infrarrojas o satelitales.

FIGURA 13.4

Representación de conexiones entre nodos.



Aunque la conexión es el tipo común de asociación entre dos nodos, es posible utilizar otros (como la agregación o la dependencia). Podrá representarlas de las formas ya conocidas.

Aplicación de los diagramas de distribución

Un buen lugar para empezar es con un equipo de cómputo doméstico, por lo que el primer ejemplo es un diagrama de distribución del sistema que utilicé para escribir este libro.

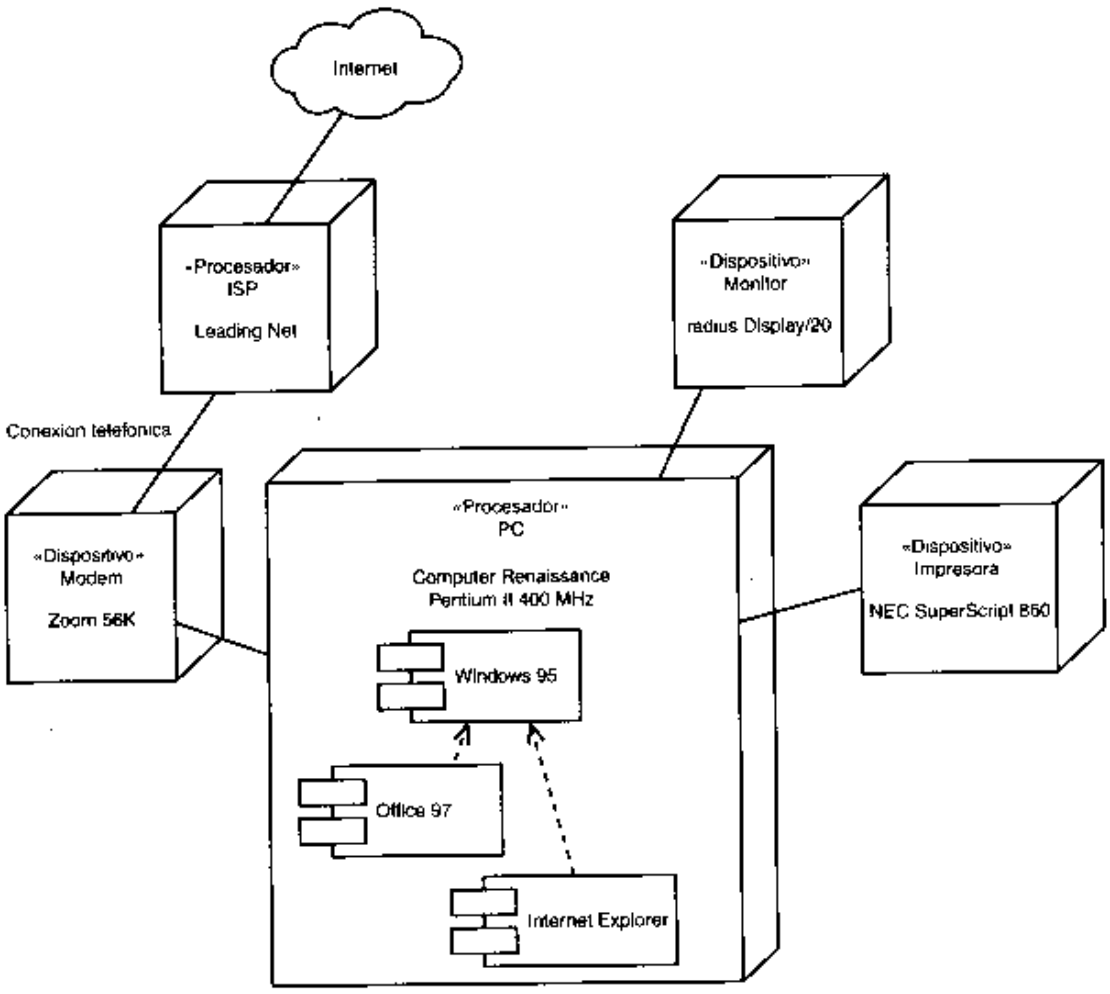
No obstante, como lo dije, los sistemas actuales de varios procesadores conectan nodos que podrían encontrarse lejanos entre sí. Para visualizar completamente este problema, necesitará también ver los ejemplos de los diagramas de distribución aplicados a las redes. Incluiré ejemplos que podrán servirle para adaptarlos a su propio entorno. Cada ejemplo incluye restricciones que reflejan las reglas de la red particular.

Un equipo doméstico

Para modelar mi equipo de cómputo, he incluido al procesador y los dispositivos, a la vez de que he modelado mi conexión telefónica con mi proveedor de servicios de Internet y su conexión. La nube que representa la Internet no es parte de la simbología del UML, pero es útil para clarificar el modelo. La figura 13.5 presenta el diagrama de distribución.

FIGURA 13.5

El diagrama de distribución de mi equipo de cómputo.



Una red token-ring

En una red token-ring, las computadoras equipadas con una NIC (tarjeta de interfaz de red) se conectan a una MSAU (unidad central de acceso a multiestaciones). Se conectan varias MSAU en una serie que podría parecer un anillo (por ello la parte "ring" del nombre). El anillo de MSAU se combina para fungir como un policía de tránsito, mediante una señal

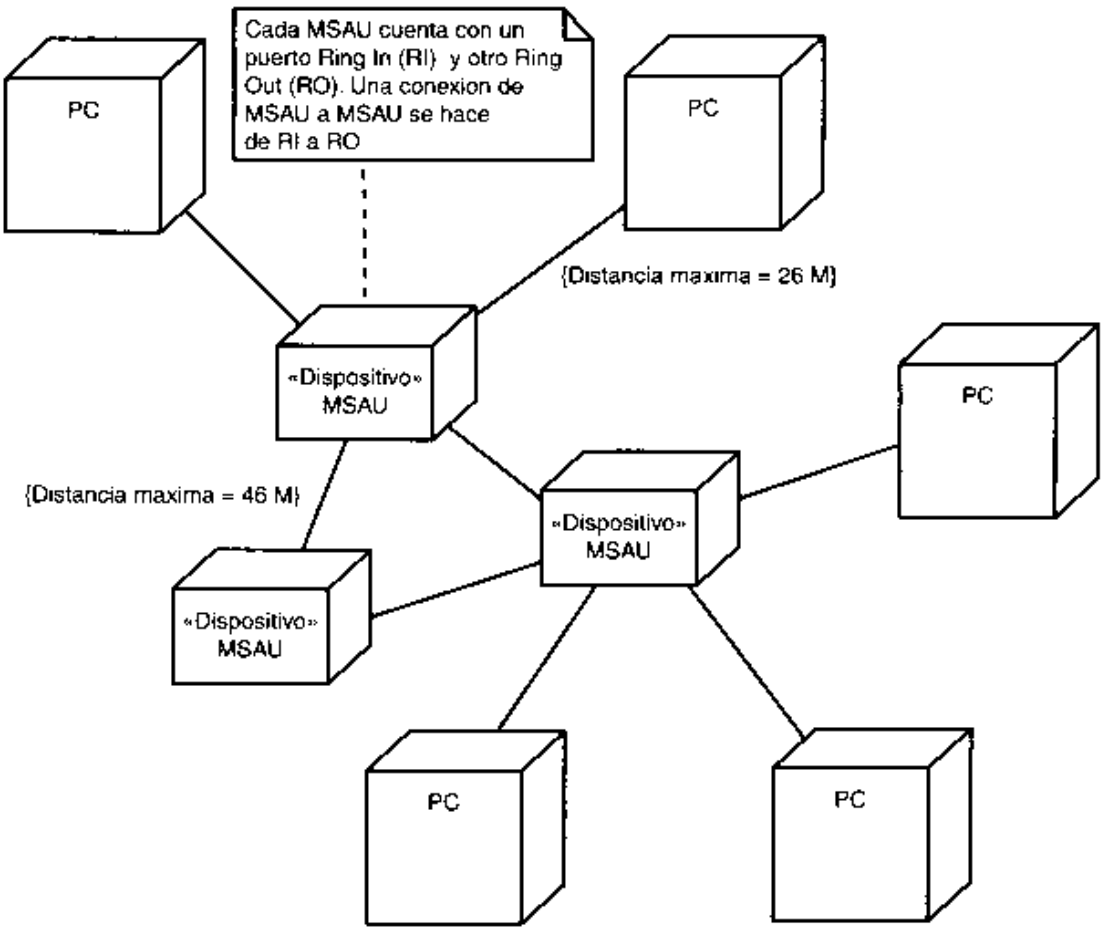
conocida como *token* que permite a cada equipo de cómputo saber cuándo puede transmitir información. Así es, el token va de equipo en equipo hasta que uno de ellos contenga información por enviar. En realidad, el token se mueve por el anillo de MSAU.

Cuando se obtiene el token, sólo esa información del equipo puede ir por la red. Una vez que se envía, la información viaja hasta su destino. Cuando llega, se devuelve un acuse de recibo al equipo que la envió.

En este ejemplo, que se aprecia en la figura 13.6, he modelado una red que consta de tres MSAU y sus respectivos equipos.

FIGURA 13.6

El diagrama de distribución de una red token-ring que consta de tres MSAU.



ARCnet

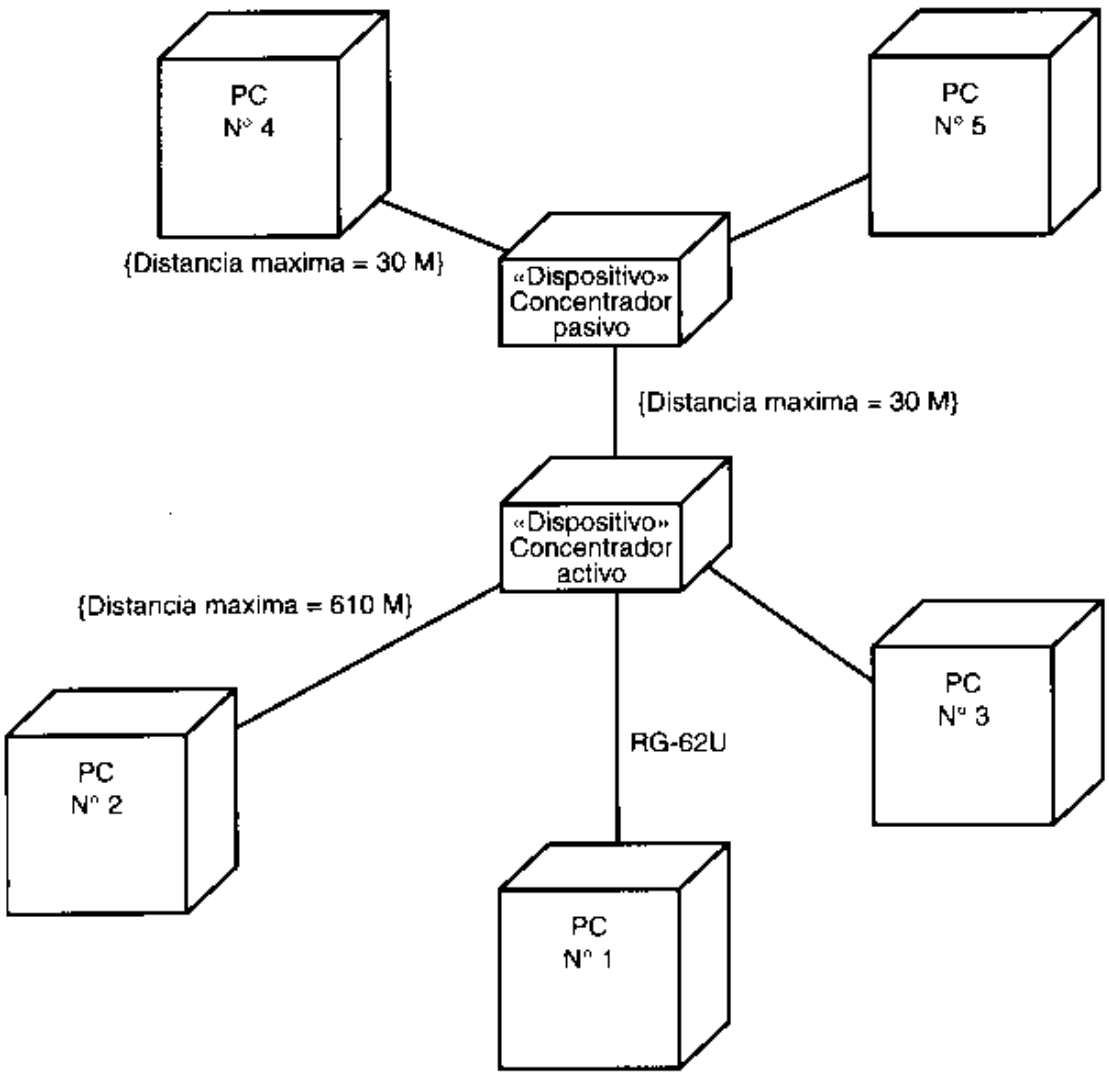
Como en una red token-ring, una red ARCnet (Red de Cómputo de Recursos Adjuntos) implica pasar un token o señal de un equipo a otro. La diferencia es que en ARCnet cada equipo tiene asignado un número. El orden numérico determina cuál equipo obtendrá al token. Cada equipo se conecta a un concentrador o hub que podrá ser activo (amplificará la información que llega antes de transmitirla) o pasivo (transmitirá la información sin amplificarla).

A diferencia de los MSAU en una red token-ring, los concentradores ARCnet no mueven el token en un anillo. Los equipos se lo pasan entre sí.

La figura 13.7 modela una red ARCnet con un concentrador pasivo, uno activo y varios equipos.

FIGURA 13.7

Un diagrama de distribución de una red ARCnet.



Thin ethernet

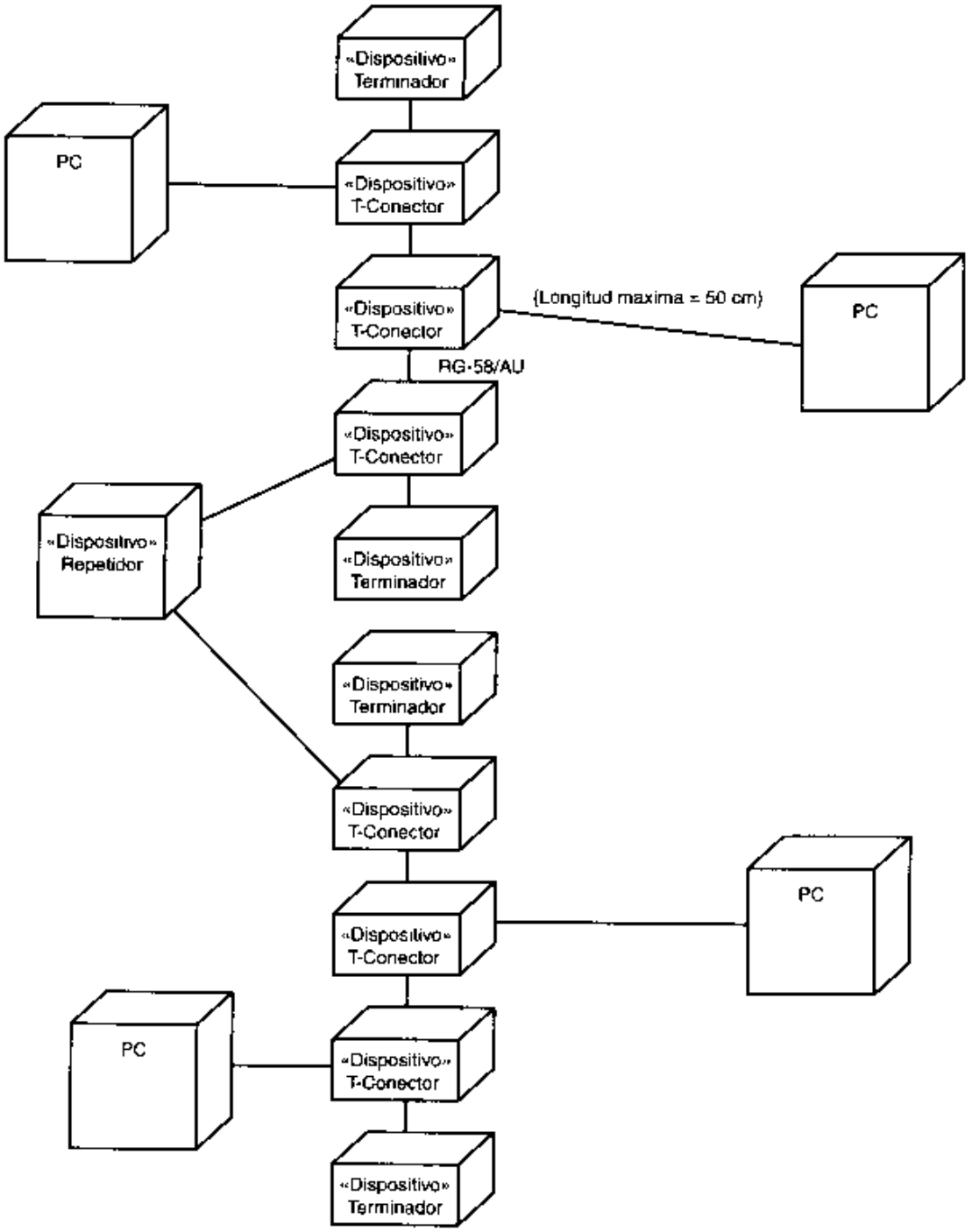
TÉRMINO NUEVO

La red thin ethernet es un tipo muy popular. Los equipos se conectan a un cable de red mediante dispositivos conocidos como conectores T. Un segmento de red puede unirse a otro mediante un *repetidor*, un dispositivo que amplifica una señal antes de transmitirla. También pueden hacerse conexiones de tipo RJ-45, aunque, en este caso, nos concentraremos tan sólo en la conexión T.

La figura 13.8 modela una red thin ethernet.

FIGURA 13.8

Diagrama de distribución de una red thin ethernet.



Red inalámbrica Ricochet de Metricom

Metricom, Inc, empresa localizada en Los Gatos, CA, cuenta con una solución inalámbrica por módem para obtener acceso móvil a Internet. Su módem inalámbrico se conecta al puerto serial de un equipo de cómputo y se comunica con su red Ricochet.

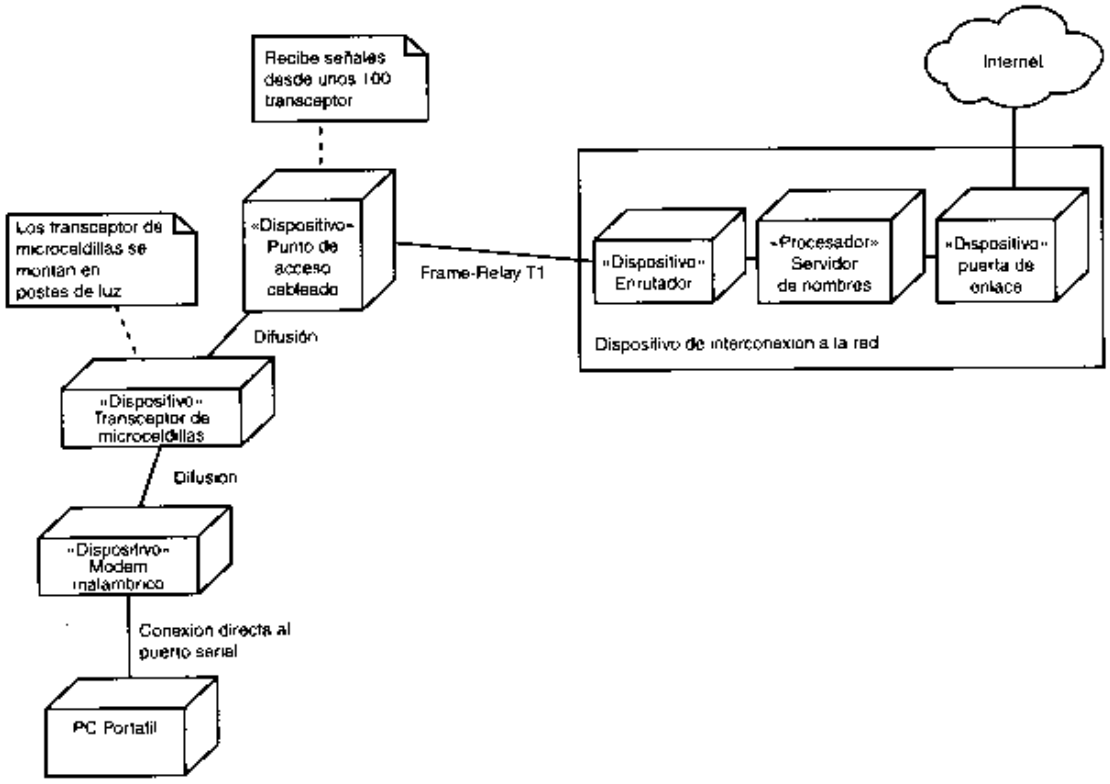
La red Ricochet consta de transmisores y receptores de radio, cuyo tamaño es de una caja de zapatos. Tales radios de microceldilla se montan en la parte superior de los postes de luz a distancias de 400 a 800 metros, en un patrón de tablero de ajedrez. Cada radio de microceldilla obtiene una pequeña cantidad de energía de su poste de luz si se equipa con un adaptador especial.

Los radios de microceldillas difunden señales a Puntos de acceso cableados que llevan la información a un NIF (dispositivo de interconexión a la red). El NIF consta de un servidor de nombres (una base de datos que valida las conexiones), un enrutador (dispositivo que enlaza a las redes entre sí), y una puerta de enlace (un dispositivo que traduce la información de un protocolo de comunicaciones a otro). La información se lleva del NIF a la Internet.

La figura 13.9 muestra un diagrama de distribución para esta red.

FIGURA 13.9

Red inalámbrica Ricochet de Metricom.

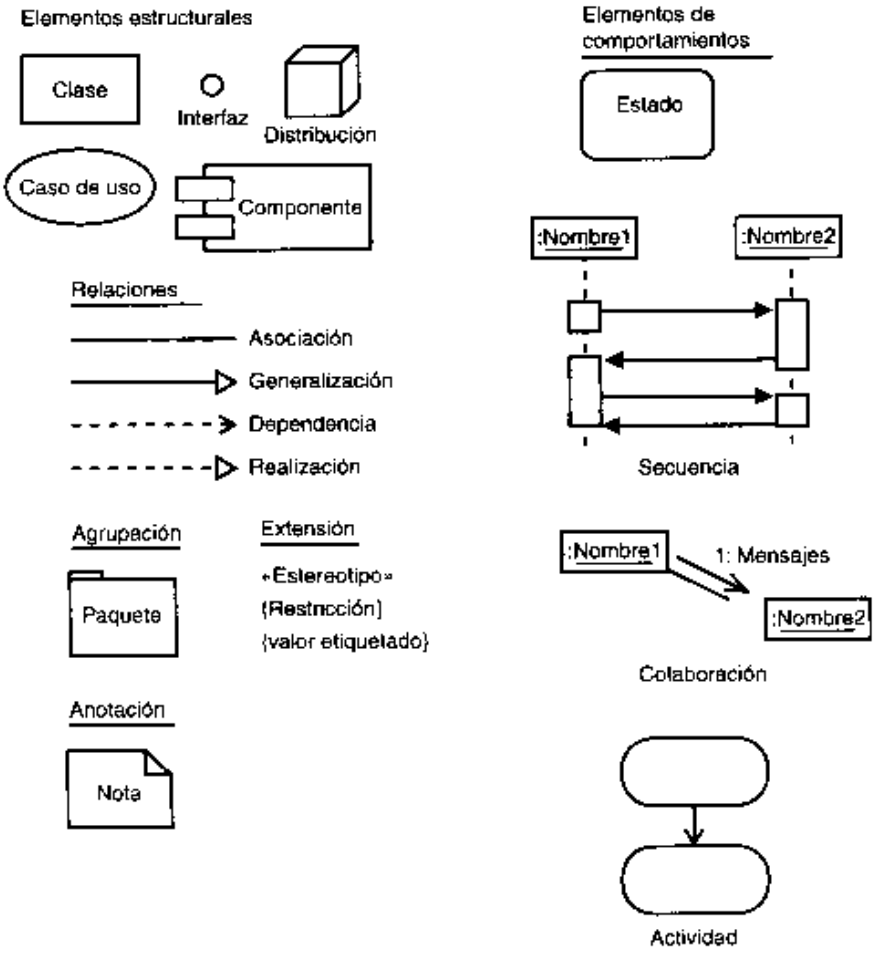


Los diagramas de distribución en el panorama

Ha llegado al final del conjunto de diagramas. El panorama incluye al diagrama de distribución, y ha quedado finalizado.

FIGURA 13.10

Su panorama del UML incluye al diagrama de distribución y está completo.



Resumen

El diagrama de distribución del UML ilustra la forma en que luce un sistema físicamente cuando sea conjugado. Un sistema consta de nodos, donde cada nodo se representa por un cubo. Una línea asocia a dos cubos y simboliza una conexión entre ellos. Los tipos de nodos son procesador (que puede ejecutar un componente) y dispositivo (que no lo puede hacer). Los dispositivos por lo general interactúan con el mundo.

Como puede imaginar, los diagramas de distribución son útiles para modelar redes. Los modelos presentados en esta hora incluyeron a redes token-ring, ARCnet, thin ethernet y la red inalámbrica Ricochet.

Preguntas y respuestas

P **Usted utilizó una nube para representar a la Internet, y dijo que no era parte de la simbología del UML. ¿Un modelador puede utilizar símbolos que no están en la simbología?**

R Así es. De hacerlo, no habrá policía UML que lo lleve a prisión. La idea es utilizar el UML para expresar una visión. En ninguna parte esto es tan útil como en los diagramas de distribución. Si tiene una imagen que pueda mostrar claramente los equipos de escritorio, portátiles, servidores y otros procesadores (o dispositivos), podrá utilizarlos en sus diagramas. Claro que estará creando un estereotipo gráfico. (Por cierto que el símbolo de la nube es una interesante nota al margen por aprender en el UML. Uno de los creadores del UML, Grady Booch, solía representar objetos como nubes en la simbología de su esquema de modelado antes de que se convirtiera en parte del equipo del UML.)

P **Suponga que cuenta con una gran cantidad de figuras para representar a ciertos objetos y no a otros. ¿Se pueden mezclar con los símbolos del UML?**

R Claro que puede. El objeto es dibujar diagramas para clarificar una visión, no para (perdón por el juego de palabras) nublarla.

Taller

Ahora que ha finalizado con todo el conjunto de diagramas del UML, pruebe su conocimiento respecto a la forma de representar hardware. Las respuestas se destacan en el apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Cómo representa a un nodo en un diagrama de distribución?
2. ¿Qué tipo de información puede aparecer en un nodo?
3. ¿Cuáles son los dos tipos de nodos?
4. ¿De qué forma funciona una red token-ring?

Ejercicios

1. Imagine a su equipo de cómputo doméstico como un conjunto de nodos. Dibuje un diagrama de distribución que incluya a su gabinete, monitor, impresora y cualquier otro periférico. Incluya cualquier estereotipo necesario y compartimientos para clarificar la información (posiblemente resultará en un modelo algo diferente al de mi equipo).
2. Es posible conectar una red a otra. Una forma de hacerlo es conectar a cada red con un enrutador y a cada uno de ellos con un (posiblemente muy grande) circuito de LAN a LAN. Dibuje un diagrama de distribución de una pequeña red token-ring que se conecte a una pequeña red thin ethernet.

HORA 14

Nociones de los fundamentos del UML

Ahora que ha visto los diagramas en el UML, ya está listo para ciertos conceptos fundamentales.

En esta hora se tratarán los siguientes temas:

- Estructura del UML
- Capa del metamodelado
- Extensión del UML
- Estereotipos, restricciones y valores etiquetados

Si éste fuera un texto académico en lugar de uno de autoaprendizaje, esta hora hubiese aparecido al principio de la parte I, en lugar de estar casi al final. Hice esto para darle la oportunidad de conocer las trincheras del UML (qué es y lo que hace). Así, ya estará listo para conocer los fundamentos y trabajar con ellos.

Ahora que ha visto los diagramas y sabe cómo utilizarlos, ¿para qué podría servirle esta hora? Si ya comprendió en qué se basa el UML, podrá extenderlo y adaptarlo cuando empiece a utilizarlo en el mundo real. Como cualquier analista de sistemas le diría: cada proyecto es diferente. No existe ningún texto de referencia, libro o manual que lo pueda preparar para todas las situaciones con las que se encontrará. No obstante, el tener una buena base de los conceptos fundamentales lo preparará para la mayoría de los sistemas que tenga que modelar.

Es como aprender un idioma extranjero. La mejor forma de hacerlo es sumergirse en él, como lo hizo en las horas 1 a la 13 (y como lo hará en la parte II, "Estudio de un caso"). Luego podrá empezar a captar las reglas gramaticales y sintácticas dado que estará preparado para comprenderlas (¡por desgracia, muchos cursos académicos de idiomas proceden en orden opuesto!).

Estructura del UML

Su panorama del UML le muestra las categorías de los diagramas y a éstos en cada categoría. Como lo dije en la hora 1, "Introducción al UML", necesitará todos los diagramas, ya que le permitirán ver su sistema desde diversos puntos. Debido a que hay varias personas a las que les interesa el sistema por distintas razones, deberá tener la capacidad de comunicar una visión consistente del sistema de diversas formas.

Aunque su panorama es útil como una forma de tener a la mano los elementos del UML, no funciona como una definición de éste. Los tres amigos estructuraron al UML de una manera formal para asegurarse que los elementos que habían creado pudieran mostrar una idea clara de un sistema propuesto, o completamente reestructurado.

El UML cuenta con una arquitectura de cuatro capas. Tales capas se distinguen por la generalidad de los elementos que en ellas residen.

En la actualidad, la palabra *arquitectura* nos brinca en el mundo del desarrollo de sistemas. Imagine una arquitectura como una forma de resumir un conjunto de decisiones respecto a la forma en que se organiza un sistema.
Tales decisiones se enfocan muy específicamente en los elementos del sistema: qué son, qué hacen, cómo se comportan, cómo se relacionan y se combinan.

En las horas anteriores estuvo operando en las dos capas más específicas. Cuando siguió un ejemplo o realizó un ejercicio que involucraba instancias específicas de un dominio en particular (como el diagrama de componentes de mi sistema de cómputo personal), se encontraba en la capa más específica. Esta capa se llama *capa de objetos del usuario*, donde "usuario" se refiere a quien utiliza el UML (no al que utiliza el sistema en sí).

TÉRMINO NUEVO

Estuvo en la siguiente capa cuando vio las clases, como en el ejemplo donde el analista habló con el entrenador de baloncesto para distinguir las clases en el dominio baloncesto. Los primeros estados del análisis tienen que ver con esta capa: trabajará con un experto o un cliente para obtener la información de un dominio, y con los usuarios potenciales para comprender los casos de uso que tendrán que tomarse en cuenta al generar el sistema. A esta capa se le denomina *capa de modelado*.

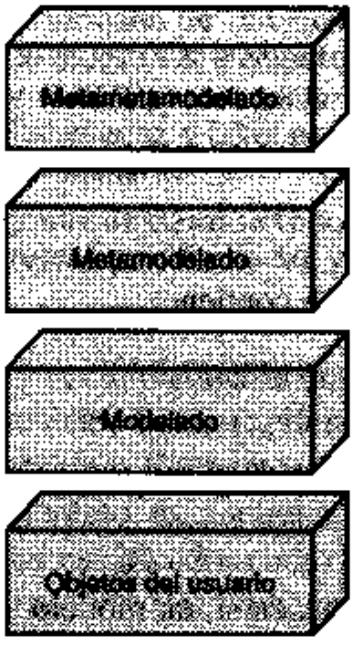
TÉRMINO NUEVO ¿En qué momento estuvo en una capa menor? Al principio de cada hora cuando aprendió un concepto como el de una clase o un nodo, estaba en la tercera de cuatro capas. Ésta define al lenguaje para un modelo específico. Luego de que obtenga algo de experiencia, se familiarizará tanto con el UML que esta tercera capa le será algo muy natural. Dado que esta capa define lo que va en un modelo, se conoce como *capa de metamodelado*.

Puesto que en su panorama se muestran los símbolos de las clases, nodos, componentes, casos de uso y cosas así, tal panorama pertenece a la capa de metamodelado.

TÉRMINO NUEVO ¿Y la cuarta capa? Durante su carrera como analista, posiblemente nunca tendrá que tratar con ella. Imagínela como una forma de definir un lenguaje que especifique clases, casos de uso, componentes y todos los demás elementos del UML con los que trabajará. Esto es más de la incumbencia de los teóricos que diseñan y comparan lenguajes que de los analistas que lo usan. Dado que esta capa define lo que va en el metamodelo, se conoce como *capa de metametamodelado*.

La figura 14.1 le muestra las cuatro capas.

FIGURA 14.1
Las cuatro capas del UML.



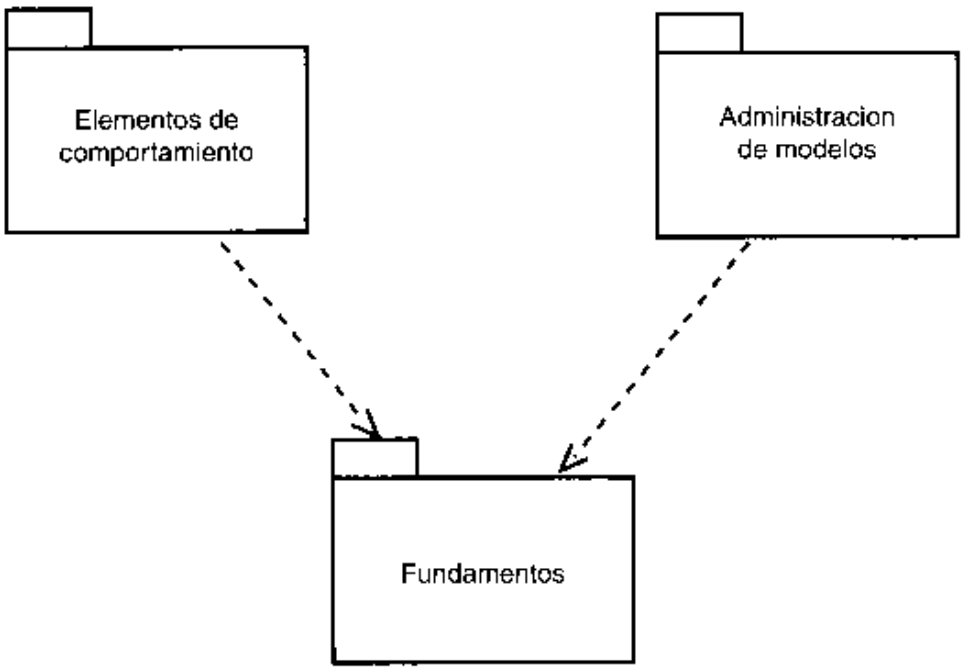
Capa del metamodelado: cercano y personal

Puesto que la capa de metamodelado es la base de su panorama, examinémosla más detalladamente.

Imagine a esta capa como una formación de tres paquetes: *Fundamentos, Elementos de comportamiento* y *Administración de modelos*. La figura 14.2 le muestra lo que quiero decir.

FIGURA 14.2

Los paquetes en la capa de metamodelado del UML.



Como en el caso de cualquier paquete, cada uno de estos grupos relacionan elementos entre sí. (¿Utilizamos al UML para modelar al UML? ¡Por supuesto!)

¿Cuáles son estos elementos? El paquete Fundamentos contiene:

- Núcleo
- Elementos auxiliares
- Tipos de datos
- Mecanismos de extensión

El paquete de Elementos de comportamiento contiene:

- Comportamiento en común
- Colaboraciones
- Casos de uso
- Máquinas de estado

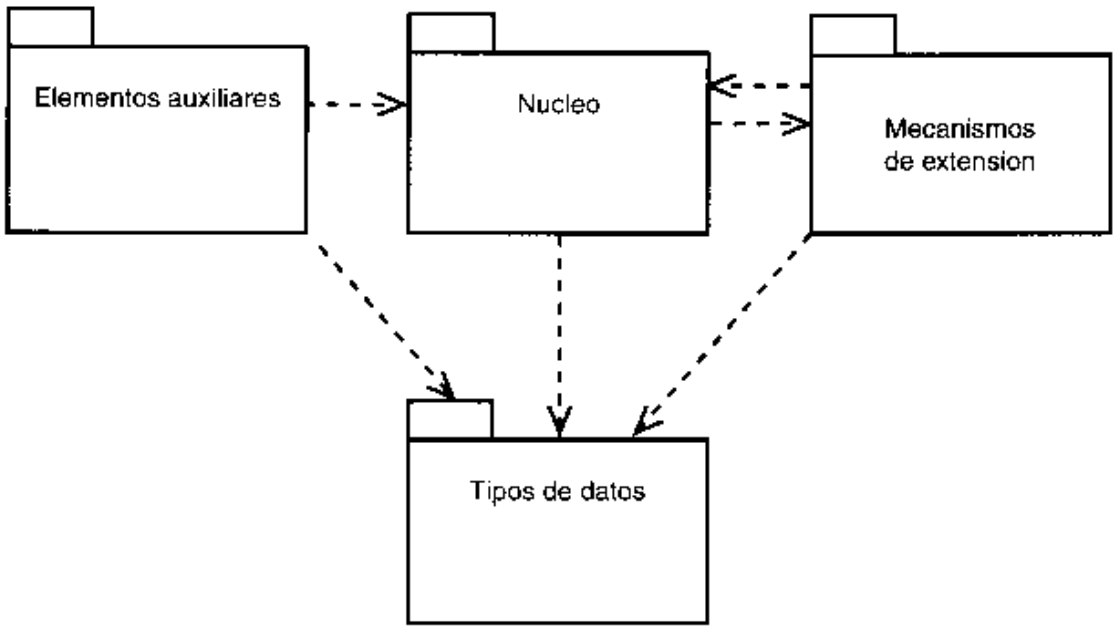
El paquete de Administración de modelos es, en sí, un modelo. Es un diagrama de clases que muestra cómo se relacionan los elementos del UML entre sí, como son los paquetes o subsistemas.

El paquete de Fundamentos

Hagamos una retrospectiva y entremos a otro nivel. Empezaré con el paquete de fundamentos, cuyos componentes aparecen en la figura 14.3.

FIGURA 14.3

Los paquetes del propio paquete de Fundamentos.



El núcleo define lo que necesita para crear un modelo UML. Cada uno de los elementos definidos es *abstracto* (lo que significa que no puede crear instancias de él) o *concreto* (con el que sí podrá). Entre los elementos abstractos se encuentran *ElementoDeModelo, ElementoGeneralizable y Clasificador*. Entre los concretos se encuentran *Clase, Interfaz, Asociación y Tipo de datos*.

Verá en esta sección que diré que un paquete "define", "establece" o "da los detalles formales de" un elemento (o concepto). Ello significa tres cosas: (1) el paquete muestra al elemento dentro de un diagrama de clases (otro ejemplo de usar al UML para modelar al UML), (2) el paquete contiene reglas para utilizar el elemento, y (3) el paquete proporciona información respecto al significado del elemento.

El diagrama de clases se conoce como *sintaxis abstracta*, las reglas se conocen como *reglas bien formadas*, y al significado se le conoce como *semántica*.

He aquí otra forma de ver la distinción entre abstracto y concreto. Nunca tendrá nada en su modelo a lo que usted llame explícitamente ElementoDeModelo o Clasificador (aunque claro, utilizará *tipos* ElementosDeModelo y Clasificadores todo el tiempo). Un clasificador, por ejemplo, es cualquier elemento que describe estructura y comportamiento. Imagine a un clasificador como una forma resumida de referirse a una *clase, componente, nodo, actor, interfaz, indicación, subsistema, caso de uso o tipo de datos*. Decir que algo se le aplica a un clasificador es equivalente a decir que se aplica a cualquiera de estos otros denominadores.

En tal caso, ¿los elementos concretos se derivan de los abstractos? Así es. ¿Ello significará que hablamos de clases y herencia? Por supuesto, pero dado que estamos en la capa de metamodelado, en realidad hablamos de *metaclases*. Por ello, un "clasificador" es una "metaclase". (¿Qué tanto sentido podría haber tenido esta frase en la hora 1?)

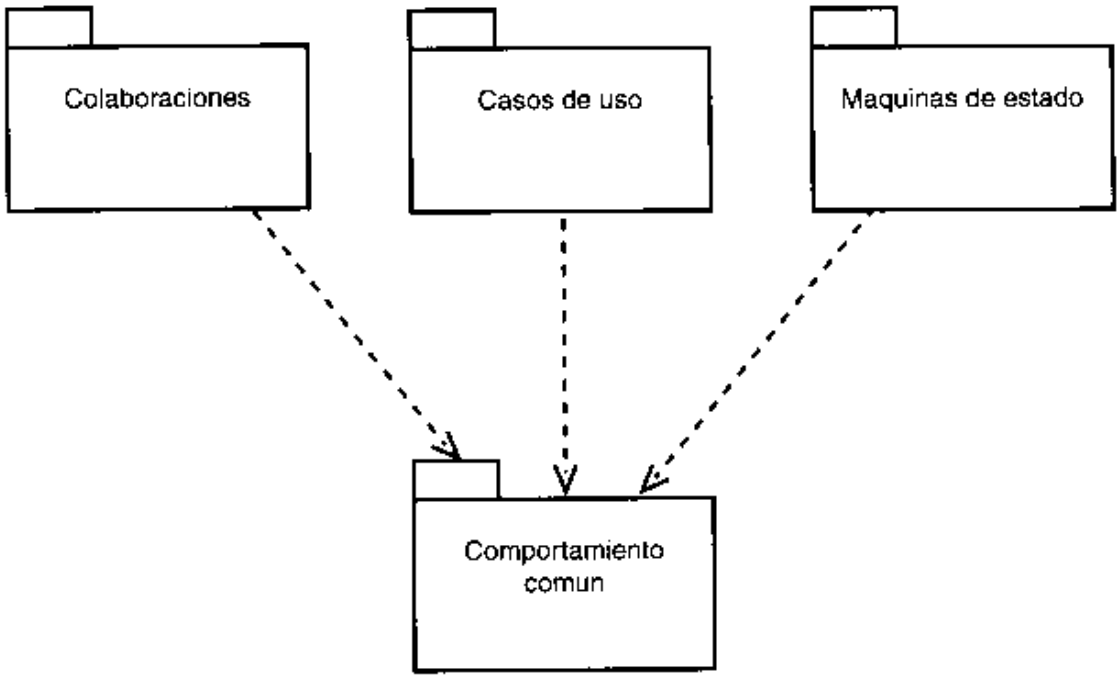
Continuemos con los otros paquetes dentro de los fundamentos. Elementos Auxiliares, un paquete que redeondea al Núcleo, define la Dependencia, Componentes y Nodos, entre otros. El paquete Tipo de Datos, especifica los tipos de datos que el UML utiliza, incluyendo los tipos primitivos (enteros, cadena y tiempo) y enumeraciones. Una enumeración como por ejemplo el Booleano lista los valores posible. El paquete de Mecanismos de Extensión le especifica cómo puede extender el UML e incluye algunas extensiones ya hechas. Analizaremos de manera más detallada este paquete más adelante, en la sección llamada Extensión del UML.

El paquete de los elementos de comportamiento

Este paquete es la parte del UML que se encarga de modelar el procedimiento de un sistema. Los paquetes de que consta aparecen en la figura 14.4.

FIGURA 14.4

Los paquetes que conforman al propio paquete de Elementos de comportamiento.



El paquete de comportamiento común proporciona los conceptos de los elementos dinámicos, y soporta otros paquetes como son: casos de uso, máquinas de estado y colaboraciones. Estos "conceptos" incluyen *Señal*, *Enlace* y *Punto final de asociación*.

El paquete de colaboraciones abarca un ámbito más amplio que tan sólo los diagramas de colaboraciones que utilizó en la hora 10, "Diagramas de colaboraciones". En este contexto, una "colaboración" describe la forma en que los clasificadores y sus asociaciones trabajan en conjunto para realizar una tarea en particular. Los diagramas de colaboraciones y los de caso de uso son parte de la perspectiva. La idea es que los clasificadores conformen al *contexto* de la colaboración; las asociaciones conforman a la *interacción*.

No es de extrañar que el paquete de casos de uso detalle los conceptos (como a un *Actor* y un *CasoDeUso*) que en él se encuentran. (Ambos conceptos son clasificadores, como lo indiqué en la sección anterior.) El objetivo general es tener la posibilidad de describir el comportamiento de un sistema sin entrar en detalles.

Tampoco debe sorprender que el paquete de Máquinas de estado dé los detalles formales de los conceptos que hay detrás de los diagramas de estados y de actividad que ya ha utilizado.

Administración de modelos

Este paquete define al *Modelo*, *Subsistema* y *Paquete*. La meta de estos elementos es agrupar los *ElementosDeModelo* de todo tipo.

Extensión del UML

Como ya lo ha visto en las horas anteriores, podrá pulir sus diagramas UML mediante la adición de detalles que expliquen mejor su significado. Los estereotipos, restricciones y valores etiquetados son herramientas útiles dentro del UML.

Puede crear una extensión sobre la marcha para agregar a su modelo cuestiones e ideas importantes de su dominio. Esto fue evidente la hora anterior: las restricciones en algunas de las comunicaciones entre nodos revelan las reglas de los diversos tipos de redes.

El UML incluye varios estereotipos, restricciones y valores etiquetados. Como ya lo mencioné, son parte del paquete Extensiones el cual, a su vez, se encuentra en el paquete Fundamentos de la capa de metamodelado. Cada una de estas extensiones incluidas es adecuada para uno (a veces dos) de los elementos del UML. Las siguientes secciones tratarán a tales extensiones.

Estereotipos

El propósito de un estereotipo es extender a un elemento del UML para que sea una instancia de una nueva metaclase, y se escribe entre dos pares de paréntesis angulares. Esto agrega una gran flexibilidad. Lo que significa que usted podrá utilizar un elemento existente del UML como base para crear sus propios elementos: elementos que capturen algún aspecto de su propio sistema o dominio de una forma en que no podrían hacerlo los elementos del UML.

La intención del estereotipo es permitir a la entidad recién creada que embone con los demás dentro de una herramienta de modelado. Las herramientas de modelado (como Rational Rose, SELECT Enterprise o Visual UML) tienen que almacenar y manejar las clases para la generación de código y la de informes. El mecanismo del estereotipo les permite hacerlo con sus creaciones.

El UML incluye un extenso conjunto de estereotipos generados. Podrá agregar de uno en uno o dos elementos. Las siguientes subsecciones organizan a los estereotipos en términos de los elementos con que confluyen.

Dependencia

La relación de dependencia puede tomar la mayor cantidad de estereotipos ya creados. Cada uno extiende una relación de dependencia entre un origen (el elemento del cual parte la flecha punteada) y un destino (el elemento al que apunta la flecha). Veamos rápidamente a cada dependencia estereotipada.

Una dependencia de tipo «se convierte en» muestra que el origen y el destino son el mismo objeto en distintos momentos. El origen se convierte en el destino con (posiblemente) diferentes roles y valores. «llamar» tiene una operación como origen y otra como su destino. En esta dependencia estereotipada, la operación de origen invoca a la de destino. Una dependencia «copiar» indica que el destino es una copia exacta del origen. En una dependencia «derivar», el origen se deriva del destino.

¿Recuerda el concepto de visibilidad de la hora 5, "Agregación, composición, interfaces y realización"? Si tiene una operación que sea privada dentro de una clase en particular, aún podrá hacerla accesible a otra clase. Coloque la otra clase (origen) y la operación (destino) en una dependencia «reunir» o «friend». El origen tendrá acceso al destino sin importar la visibilidad.

Una dependencia entre dos casos de uso también puede tener un estereotipo. Ya ha utilizado dos de ellos, «extender» y «usar», aunque sustituyó «incluir» por «usar». «extender» le indica que los comportamientos del caso de uso de destino se agregan al caso de uso de origen. «usar» indica que algunos casos de uso tienen cierto comportamiento en común, y este estereotipo le permite utilizar dicho comportamiento sin tener que repetirlo una y otra vez.

Una dependencia «importar» se establece entre dos paquetes. Este estereotipo agrega el contenido del destino al espacio de nombres del origen (el aspecto del paquete que agrupa los nombres que lo constituyen).

El estereotipo «instancia» indica que el origen es una instancia de su destino, que siempre será un clasificador. En una dependencia de «metadestino», tanto el destino como el origen son clasificadores, y el destino es la metaclase del origen.

En un «enviar», el origen es una operación y el destino es una señal. El estereotipo muestra que el origen envía la señal.

Clasificador

Los estereotipos extienden a los clasificadores de diversas formas. El estereotipo «metaclase» muestra que el clasificador al que está adjunto es una metaclase de otra clase. El «tipodeautoridad» indica que un clasificador tiene objetos que provienen de un antecesor en particular. También puede usar «tipodeautoridad» en una dependencia para mostrar que el destino es un tipo de potestad del origen. (Normalmente utilizará éste cuando modele bases de datos.)

Los estereotipos «proceso» y «subproceso» tienen que ver con el flujo de control. Ambos indican que su clasificador es una clase activa; es decir, sus objetos pueden iniciar la actividad de control. Un proceso puede consistir de varios subprocesos (flujos de control), y puede ejecutarse al mismo tiempo que otros procesos. Un subproceso puede ejecutarse junto con otros subprocesos en el mismo proceso.

Un clasificador con el estereotipo «utileria» es una colección titulada de atributos y operaciones que no son miembros de tal clasificador: un clasificador que no tiene instancias.

Finalmente, un clasificador puede tener al abuelo (¡casi literal!) de todos los estereotipos: «estereotipo». Este indica que el clasificador funciona como un estereotipo, y le permite modelar jerarquías de estereotipos.

Clase

Puede obtener algo más específico que con los clasificadores: también es posible extender a una clase. Un «tipo» es una clase que establece un dominio de objetos junto con atributos, operaciones y asociaciones. El «tipo» no contiene métodos (algoritmos ejecutables para sus operaciones).

Una «claseDeImplementacion» es lo contrario de un «tipo». Representa la implementación de una clase en un lenguaje de programación.

Generalización

Es una relación entre clasificadores, con su propio pequeño conjunto de estereotipos. «heredar» significa que las instancias del subtipo no pueden substituirse por instancias del supertipo. «subclase» hace lo mismo que en las clases: significa que las instancias de la subclase no son sustituibles por instancias de la superclase. «privado» denota una herencia exclusiva: oculta los atributos heredados y operaciones de una clase a sus ancestros.

Paquete

Los estereotipos de los paquetes son directos. Una «fachada» es un paquete que contiene referencias a elementos de otro paquete, y que no contiene elementos propios. Un «sistema» es una colección de modelos de un sistema. Un «cabo» es un paquete que proporciona sólo las partes públicas de otro paquete.

TÉRMINO NUEVO

Además de los elementos que he dicho, un paquete puede incluir patrones. Un patrón es un tipo de colaboración entre los elementos que ha probado su efectividad en diversas situaciones. Un «marcoDeTrabajo» es un paquete estereotipado que sólo contiene patrones.

Dado que los paquetes pueden encontrarse dentro de paquetes, sirve de mucho tener un estereotipo que indique cuál de los paquetes está en el nivel superior. Tal estereotipo es el «paqueteDeNivelSuperior». (¡Le dije que los estereotipos de los paquetes son directos!)

Componente

Los estereotipos para los componentes son aún más directos. Puede mostrar que un componente es un documento, un ejecutable, un archivo, una tabla de datos o una biblioteca. Los estereotipos respectivos son «documento», «ejecutable», «archivo», «tabla» y «biblioteca».

Algunos otros estereotipos

En esta subsección, he conjugado algunos estereotipos utilizados con otros elementos del UML.

Un comentario que aparece en una nota adjunta puede tener un estereotipo «requerimiento», que denota que el comentario establece un requisito para el elemento adjunto a la nota.

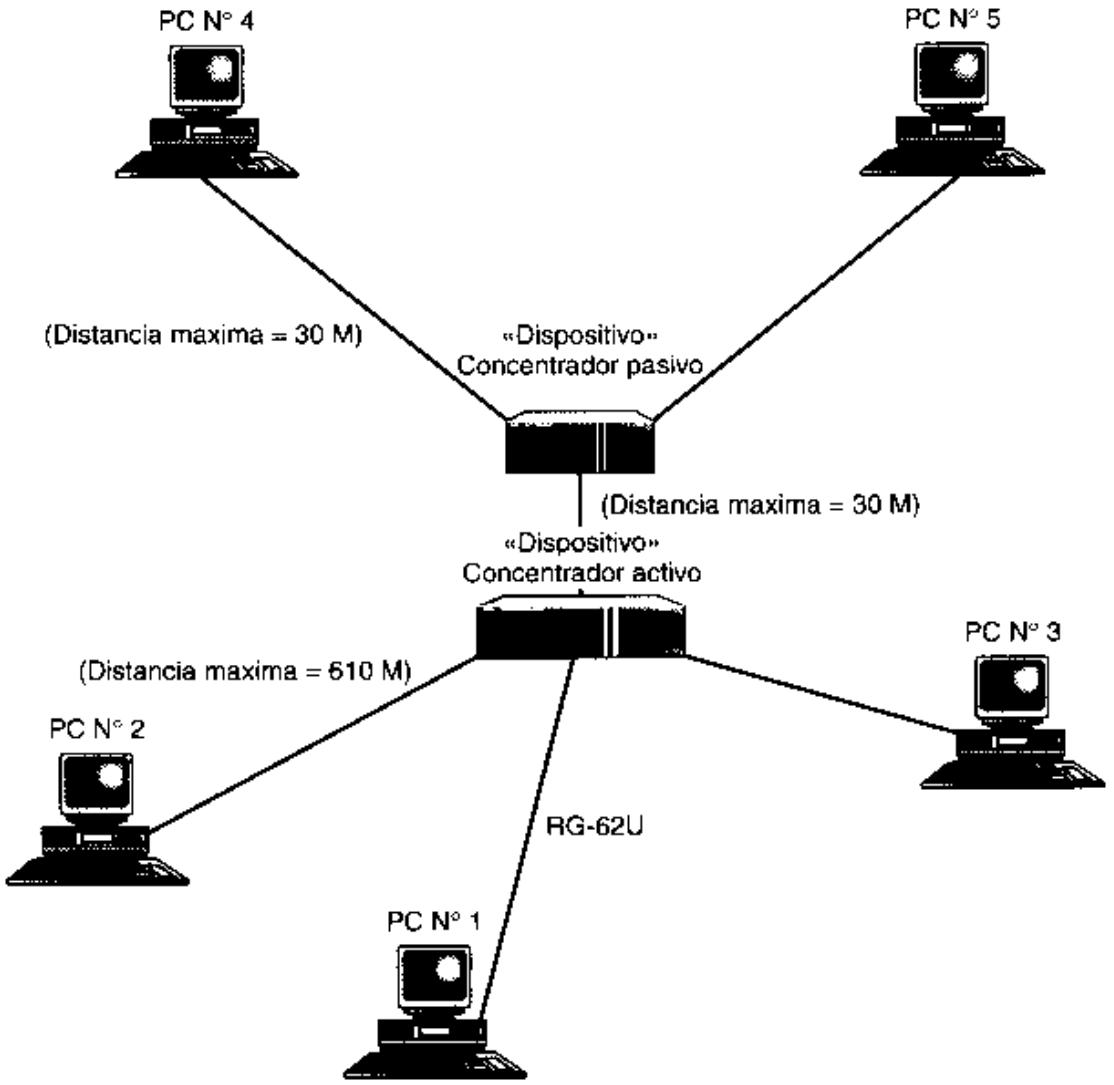
Dentro de una clase, una operación o un método pueden crear una instancia o destruirla. (Quizá haya visto métodos como *constructor* y *destructor* en Java.) Tales características se indican mediante «crear» y «destruir» respectivamente.

Las restricciones, que son el mecanismo de extensión que ahora trataré, pueden también funcionar con los estereotipos. En ocasiones usará una restricción para mostrar las condiciones previas a una operación; aunque algunas veces mostrará sus condiciones posteriores. Tales restricciones las estereotipará con «condicionPrevia» o «condicionPosterior». En ocasiones adjuntará una restricción a un conjunto de clasificadores o relaciones, y necesitará indicar que las condiciones de la restricción deberán tener todos los clasificadores, relaciones e instancias. Para ello, deberá estereotipar la restricción como «invariable».

Estereotipos gráficos

Algunas veces en su dominio, tal vez tendrá que obtener un nuevo símbolo o dos para transmitir algo a un cliente. Como lo mencioné en la hora anterior, los diagramas de distribución pueden encajar de forma importante en esto. Por lo general, hay disponibles figuras de procesadores y dispositivos, y pueden remplazar a los cubos que vio en la hora 13, "Diagramas de distribución". La figura 14.5 le muestra un ejemplo. Es una versión estilizada de la figura 13.7, un modelo de una ARCNet.

FIGURA 14.5
Un modelo estilizado de una ARCNet.



Restricciones

Las restricciones se encuentran entre llaves. Proporcionan las condiciones para las asociaciones, extremos de vínculos, generalizaciones y peticiones (transmisiones de señales o llamadas a operaciones).

La restricción {o} se aplica a un conjunto de asociaciones, y muestra que sólo una asociación podrá utilizarse para tal conjunto. Otra restricción basada en asociaciones, {implícito}, indica que una asociación es conceptual.

Los extremos de vínculos, que son puntos finales de vínculos entre objetos, pueden contener cualquier cantidad de restricciones. Cada restricción indica el porqué el objeto en el extremo del vínculo es visible. {parámetro} muestra que el objeto es un valor necesario relativo al vínculo. {propio} le indica que el objeto es el despachador de una petición. {global} y {local} indican el ámbito del objeto respecto al vínculo. {asociación} denota que el objeto es visible por su coalición.

Un conjunto de generalizaciones pueden ser {completo} (todos sus subtipos han sido especificados) o {incompleto} (aún pueden agregarse subtipos). Otro conjunto de generalizaciones pueden ser {traslapado} (más de un subtipo puede funcionar como un tipo de instancia) o {desarticulado} (sólo un subtipo puede ser un tipo de una instancia, lo que es predeterminado en la generalización).

Si una petición se envía a diversas instancias de destino en un orden no especificado, es una {difusión}. Si varias instancias devuelven valores, y la mayoría de tales valores determinan un solo valor, la restricción será un {voto}.

Valores etiquetados

Un valor etiquetado se escribe entre llaves. Consiste en una *etiqueta*, un signo = y un *valor*.

Los valores etiquetados ya elaborados son adecuados para los clasificadores, componentes, atributos, instancias y operaciones. Una etiqueta {documentación = }, se aplica a cualquier elemento. A la derecha del =, debe colocar una descripción, explicación o comentario respecto al elemento al que adjuntó este valor etiquetado.

Puede adjuntar una {ubicación = } a un clasificador o componente. Cuando lo adjunta a un clasificador, debe proporcionar al componente del cual es parte. Cuando lo hace a un componente, debe indicar el nodo donde se encuentra.

El valor etiquetado {persistencia = } puede ir en un atributo, clasificador o instancia. Denota la permanencia del estado del elemento al que lo ha adjuntado. Los valores posibles son *permanente* (el estado persiste cuando la instancia se destruye) y *transitorio* (cuando la instancia se destruye, también lo hace el estado).

{semántica = } especifica el significado de un clasificador o una operación. {responsabilidad} es una obligación de un clasificador.

Resumen

Esta hora trató los conceptos básicos del UML. El objetivo fue el de darle un conocimiento profundo que le permitirán aplicar al UML en situaciones reales que no siempre puedan reflejarse en los ejercicios de un libro de texto. Hemos tratado estos conceptos luego de todos los diagramas porque tenía que comprender los elementos del lenguaje antes de profundizar en sus fundamentos.

El UML consta de cuatro capas: objetos del usuario, modelado, metamodelado y metametamodelado (que van desde específicos hasta generales). Cuando analice un sistema, típicamente trabajará en las primeras dos capas. Cuando aprenda los conceptos del UML, por lo general se encontrará en la tercera. La cuarta capa está orientada a los teóricos y diseñadores del lenguaje, en lugar de los usuarios del lenguaje y analistas de sistemas.

El UML incluye varias extensiones propias. Cada uno de estos estereotipos, restricciones y valores etiquetados se orientan a ser usados por uno o dos de los símbolos del UML.

Si le hubiera dado todos estos conceptos fundamentales al iniciar en la hora 1 ¿los habría entendido?

Preguntas y respuestas

P Veo que el UML tiene varias reglas. ¿Quién las implementa?

R Como ya dije, la policía UML no le acecha para verificar que su modelo sea preciso. No obstante, una herramienta de modelado le podrá ayudar a ceñirse a las reglas. Por ejemplo: Visual UML tiene un archivo de estereotipos que usará de un modo sensible al contexto. Cuando intente colocar un estereotipo en un elemento en particular, sólo le permitirá seleccionar de los estereotipos adecuados para tal elemento, todos ellos en inglés. Además, le permitirá agregar estereotipos a su archivo de estereotipos.

P Los valores etiquetados parecen ser esotéricos. ¿Debo utilizarlos?

R Sí, y con frecuencia. Los valores etiquetados integrados, aunque son útiles, palidecen en comparación con los que usted definirá para sí. Puede utilizar un valor etiquetado para mantener información relacionada con la administración de un importante proyecto en su modelo: como los números de versión y los autores de las clases. Es decir, "estado", "versión" o "autor" podrían ser sus etiquetas, y usted establecerá los valores adecuados.

Taller

Este taller reafirmará su conocimiento de los fundamentos del UML. Utilice sus procesos de reflexión en el cuestionario y localice las respuestas en el Apéndice A, "Cuestionario". Ésta es una hora teórica, por lo que no he incluido ningún ejercicio.

Cuestionario

1. ¿Cuáles son las cuatro capas del UML?
2. ¿Qué es un clasificador?
3. ¿Porqué es importante el poder extender al UML?
4. ¿Cuáles son los mecanismos de extensión del UML?