

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

TERMINO 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*.

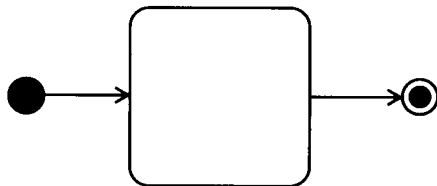
Tenga en cuenta que un diagrama de estados es intrínsecamente distinto, de manera muy significativa, de uno de clase, de objeto o de un caso de uso. Los diagramas que ya ha visto modelan el comportamiento de un sistema, o al menos un grupo de clases, objetos o casos de uso. Un diagrama de estados muestra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

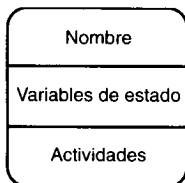


Adición de detalles al icono de estado

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.

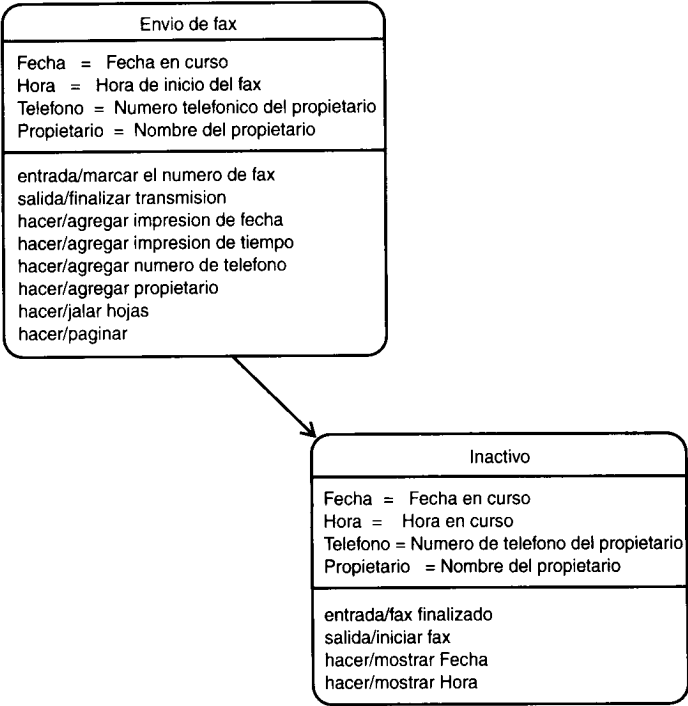


Las variables de estado como cronómetros o contadores son, en ocasiones, de ayuda. Las actividades constan de sucesos y acciones: tres de las más utilizadas son *entrada* (qué sucede cuando el sistema entra al estado), *salida* (qué sucede cuando el sistema sale del estado), y *hacer* (qué sucede cuando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

TERMINO 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:

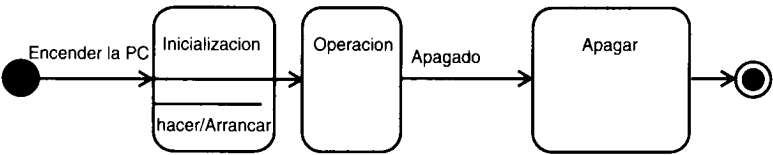
- Inicialización
- Operación
- Apagar

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.

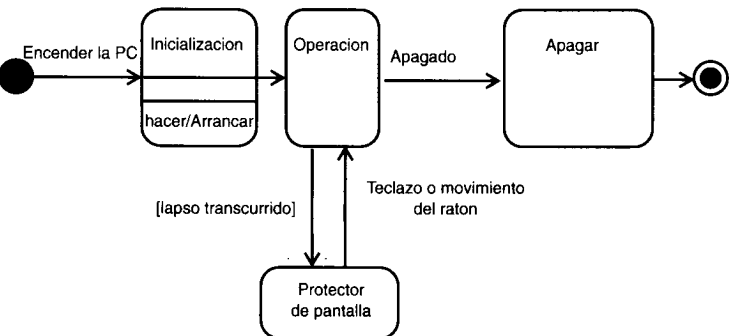
El intervalo se especifica en el panel de control de su sistema operativo (Windows en este caso). Por lo general es de 15 minutos. Cualquier opresión de una tecla o movimiento del ratón provocará una transición del estado Protector de pantallas al estado Operación.

TERMINO 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”).

TERMINO 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

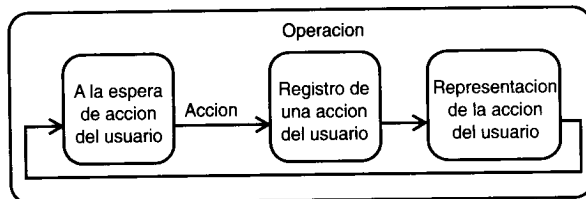
Como su nombre lo indica, los subestados secuenciales suceden uno detrás de otro. Si retomamos los subestados mencionados con anterioridad dentro del estado Operación de la GUI, tendrá la siguiente secuencia:

- 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.



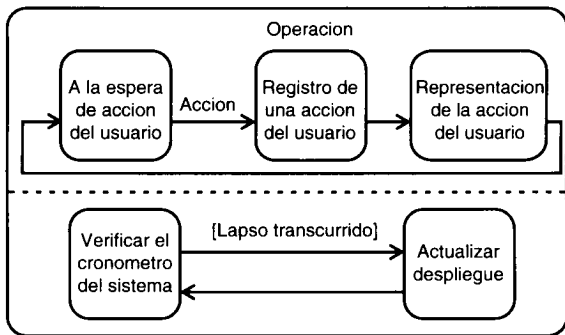
Subestados concurrentes

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.



TERMINO 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

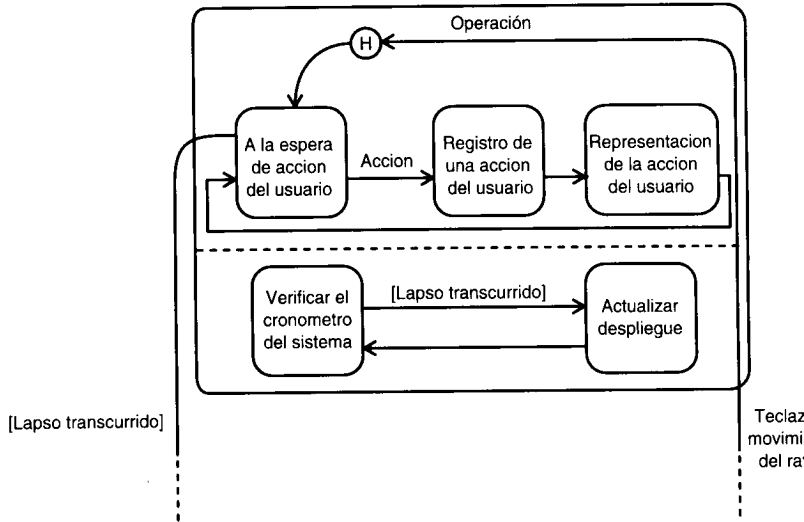
Cuando se activa su protector de pantallas y mueve su ratón para regresar al estado Operación ¿qué ocurre? ¿Acaso su pantalla retoma el estado inicial, como si apenas se hubiera encendido? ¿O lucirá tal como la dejó antes de que se activara el protector de pantallas?

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.



TERMINO 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*).



TERMINO 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).

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.

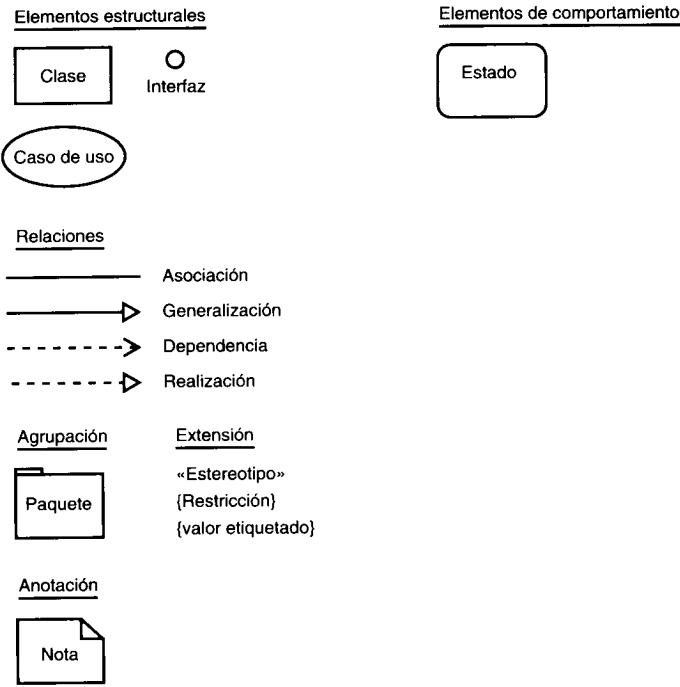
Los desarrolladores, en particular, deben saber la forma en que los objetos se supone que se comportarán, ya que son ellos quienes tendrán que establecer tales comportamientos en el software. No es suficiente con implementar un objeto: los desarrolladores deben hacer que tal objeto haga algo. Los diagramas 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 “sí” 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

TERMINO 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

TERMINO 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.



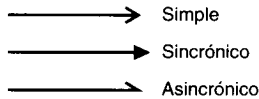
Mensaje

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.



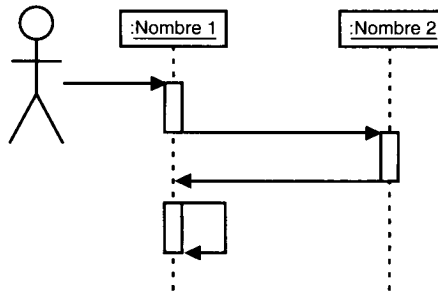
Tiempo

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 interacciones 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.

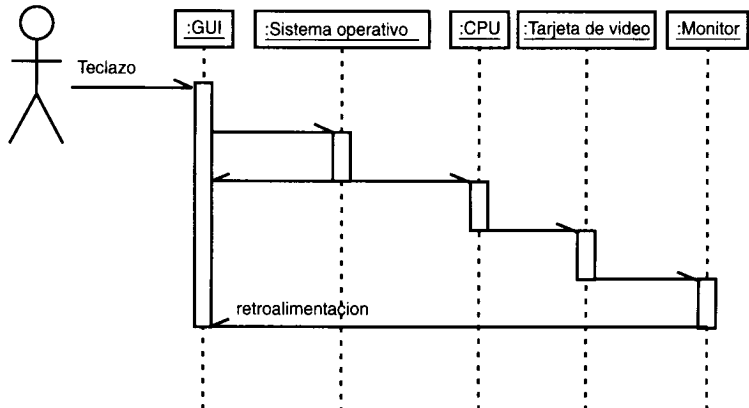
Todo esto ocurre con tanta rapidez que olvidamos que todo ello se realiza. (¡Si acaso pensábamos que ocurría!)

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

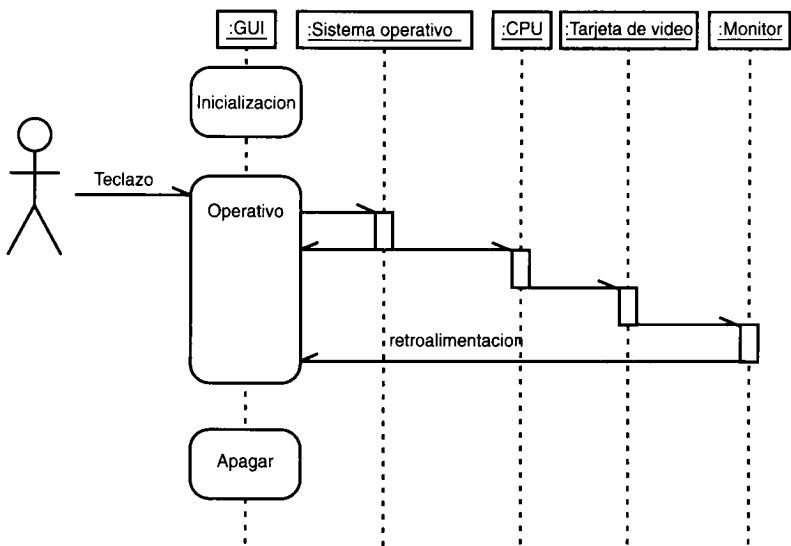
Un diagrama de secuencias que muestra la forma en que la GUI interacciona con otros objetos.



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.





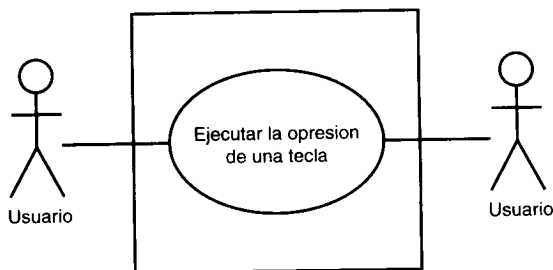
En un diagrama de secuencias, otra forma de mostrar el cambio de estado de un objeto es incluir al objeto más de una vez en el diagrama.

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:

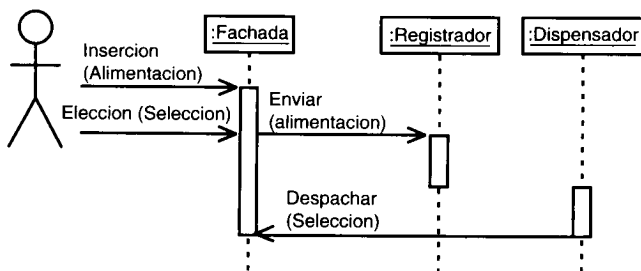
1. El cliente inserta el dinero en la alcancía que se encuentra en la fachada de la máquina.
2. El cliente hace su elección.
3. El dinero viaja hacia el registrador.
4. El registrador verifica si la gaseosa elegida está en el dispensador.
5. Dado que es el mejor escenario, asumimos que sí hay gaseosas,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

TERMINO 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.

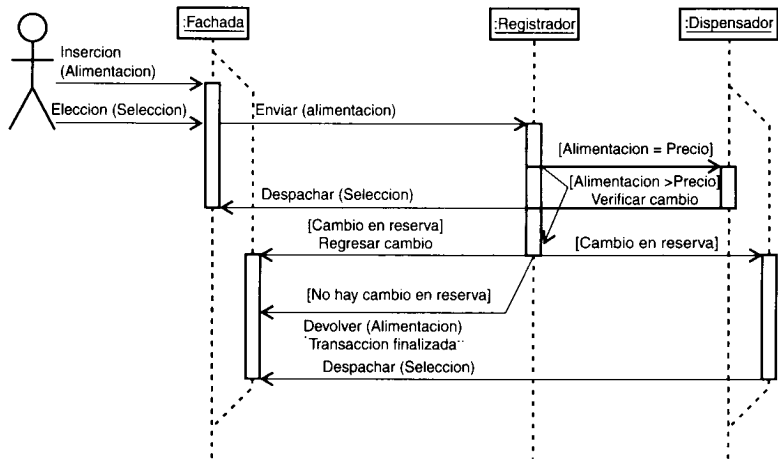
Para representar cada condición en la secuencia, tal condición la colocará en un “si” (un si condicional) entre corchetes. Arriba de las flechas de mensaje apropiadas, agregue [alimentación > precio], [alimentación – precio no presente] y [alimentación – precio presente].

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”.



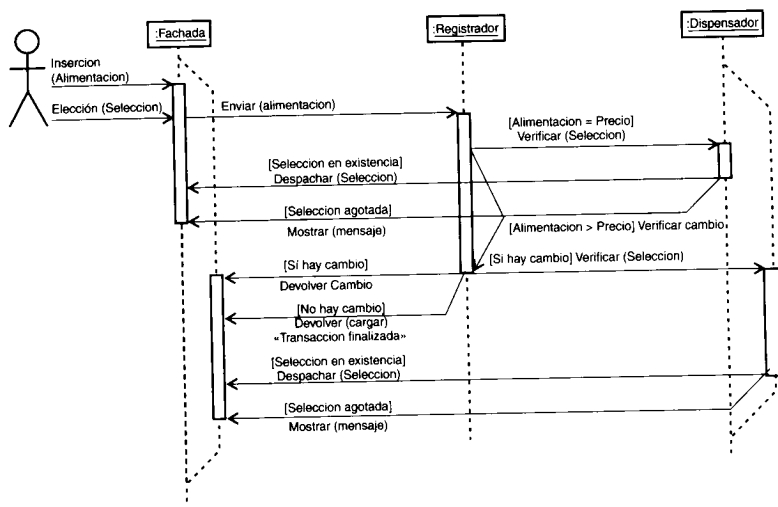
Ahora agregaremos el escenario “Sin gaseosa”.

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.

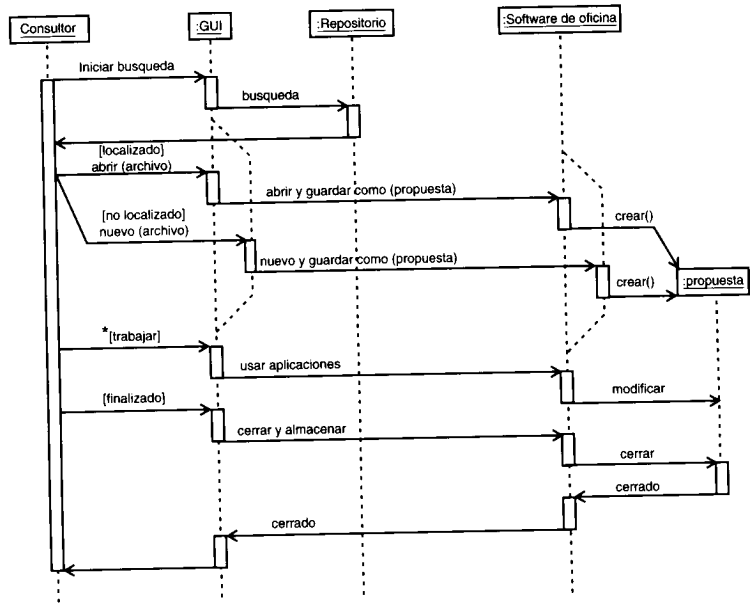
La figura 9.10 le muestra el diagrama de secuencias del caso de uso “Crear una propuesta”.



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

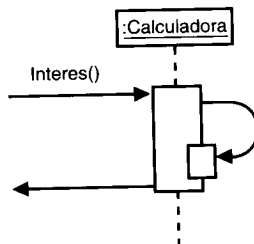
En ocasiones un objeto cuenta con una operación que se invoca a sí misma. A esto se le conoce como *recursividad*, y es una característica fundamental de varios lenguajes de programación.

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.

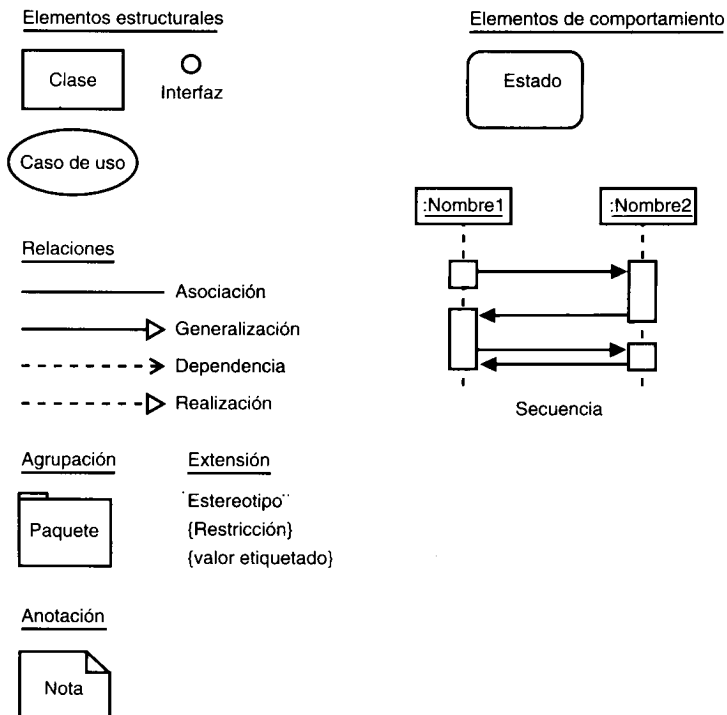


Adiciones al panorama

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.

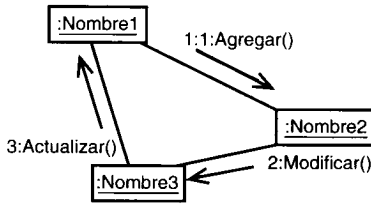
Mencioné que podrá convertir cualquier diagrama de secuencias en diagrama de colaboraciones y viceversa. Por medio de esto podrá representar la información de secuencia en un diagrama de colaboraciones. Para ello, agregará una cifra a la etiqueta de un mensaje, misma que corresponderá a la secuencia 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

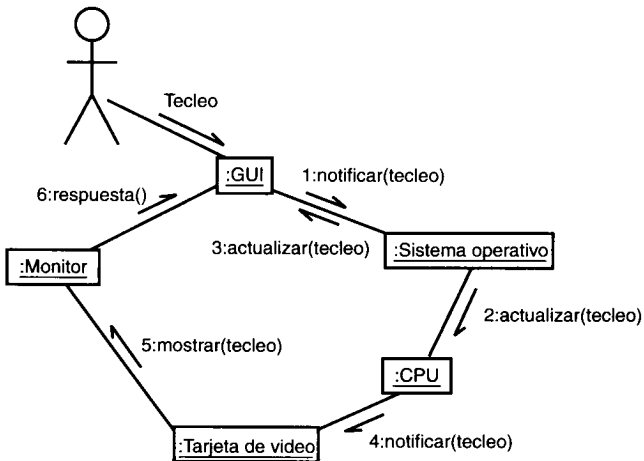
Este ejemplo es el caso más directo. Un actor inicia la secuencia de interacción al oprimir una tecla, con lo que los mensajes ocurrirán de manera secuencial. Tal secuencia (a partir de la hora anterior) es:

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.



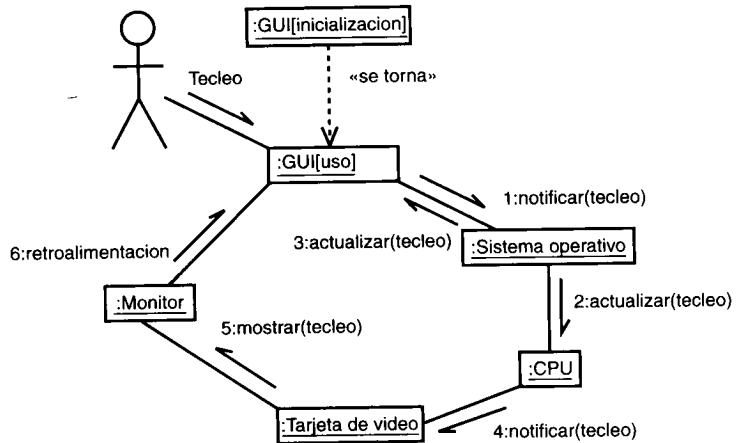
Cambios de estado

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

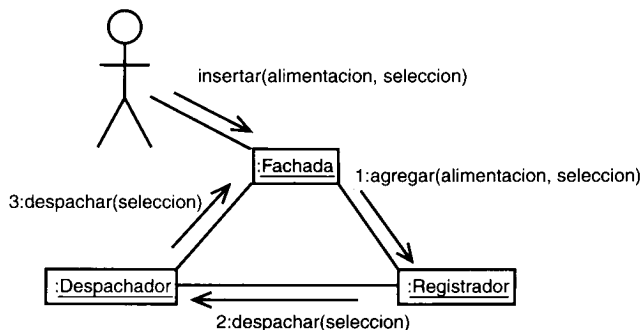
Las cosas se hacen más interesantes cuando aplica las condiciones a una situación real, como lo hizo en la hora anterior con la máquina de gaseosas. Iniciemos con la mejor situación del caso de uso “Comprar gaseosa”, donde la secuencia es:

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”.

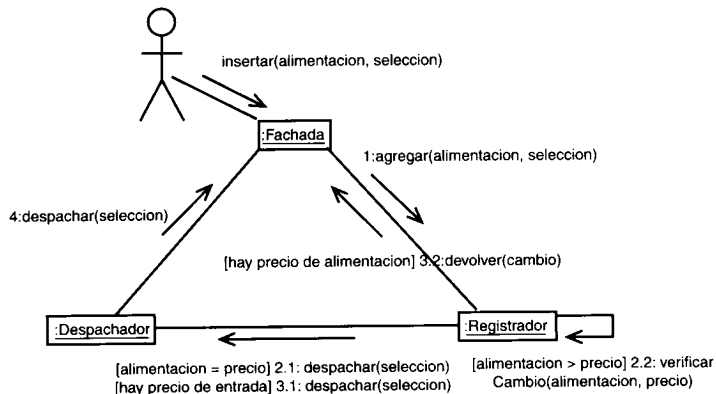
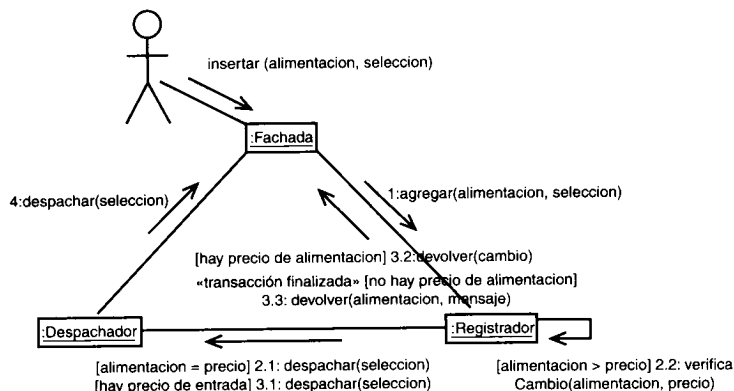


FIGURA 10.6

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.

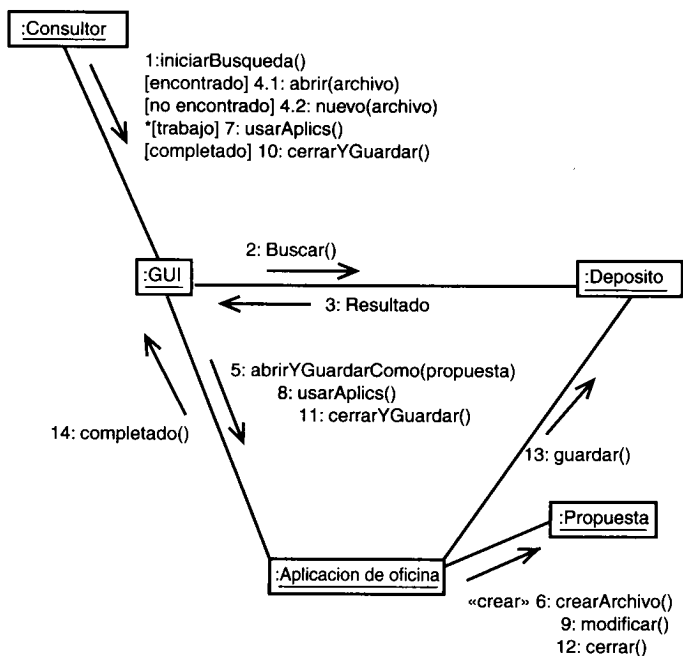
Para mostrar la creación de un objeto, agregará un estereotipo “crear” al mensaje que genera al objeto.

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

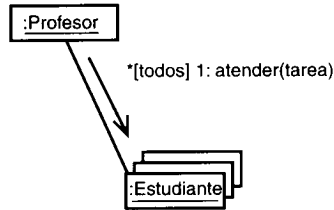
Aunque ha visto algunas bases, no ha visto todo lo relacionado con los diagramas de colaboraciones. Los conceptos en esta sección son un poco esotéricos, pero podrían serle útiles en sus esfuerzos para analizar sistemas.

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”. Agregaré 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

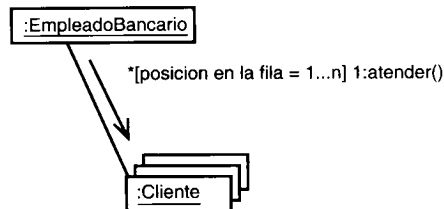
Un objeto que envía un mensaje a diversos objetos de una clase.



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

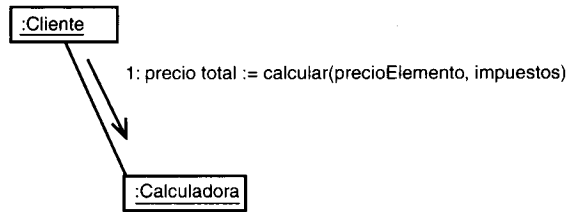
Un mensaje podría ser una petición a un objeto para que realice un cálculo y devuelva un valor. Un objeto Cliente podría solicitar a un objeto Calculadora que calcule el precio total que sea la suma del precio de un elemento y el impuesto.

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

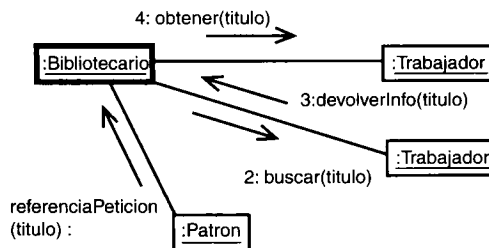
TERMINO NUEVO

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 *concurrency*.

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.

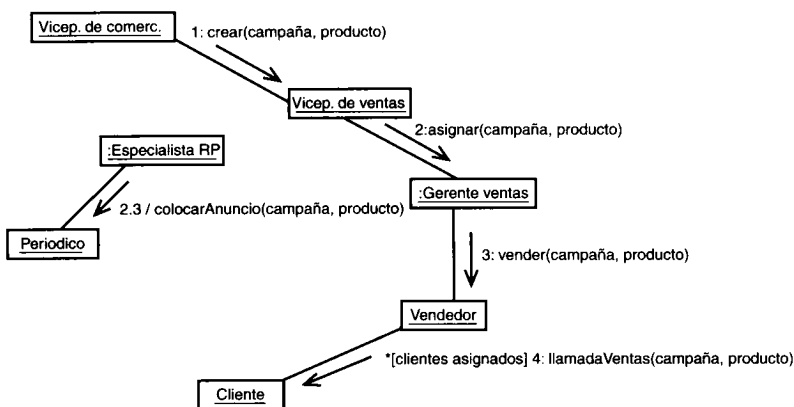
Un ejemplo aclarará esto. Suponga que sus objetos son personas en un corporativo, y que están ocupados en la campaña de un nuevo producto. He aquí la secuencia de interacciones:

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.

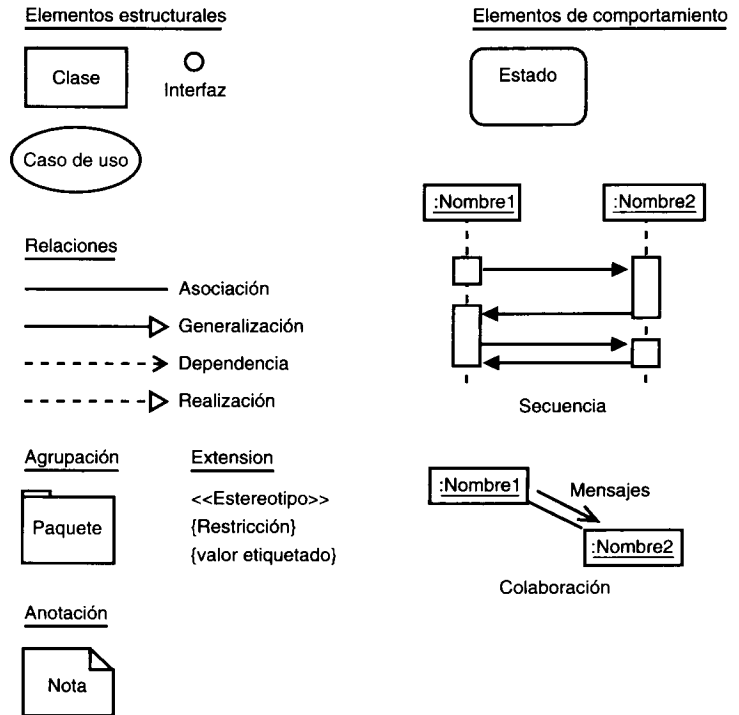


Adiciones al panorama

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.

