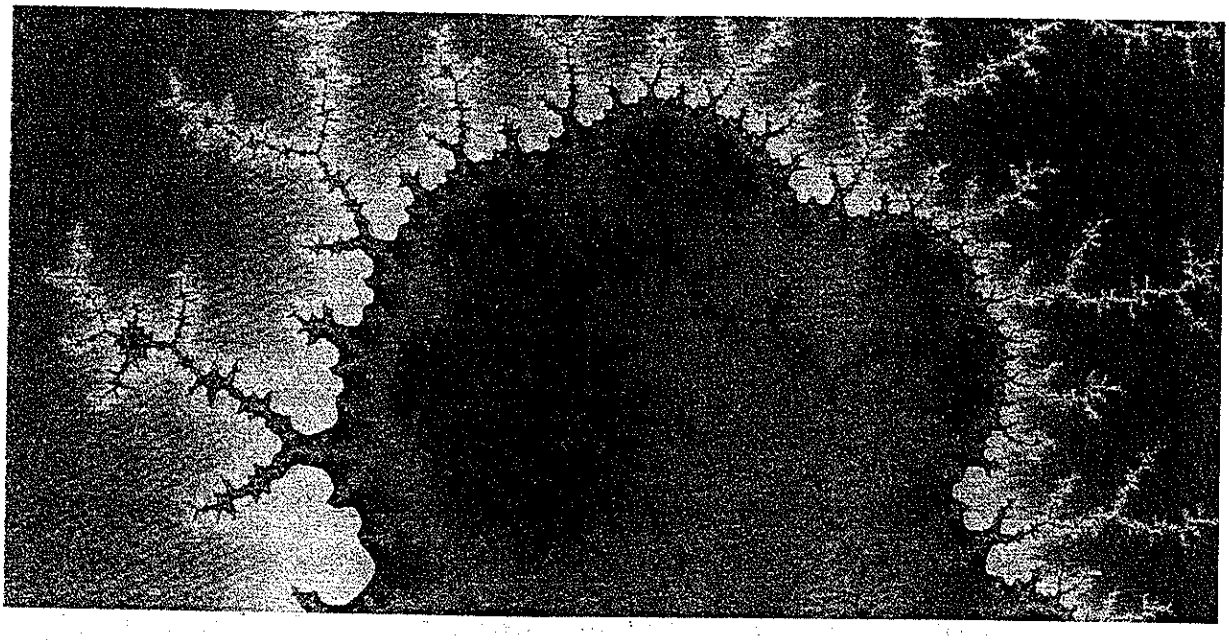


21
Máquinas de estado

21.1. Presentación del capítulo

En este capítulo tratamos las máquinas de estado. Son una forma importante de modelar el comportamiento dinámico de clasificadores.

El capítulo comienza con una introducción a las máquinas de estado, una explicación de los dos tipos diferentes de las máquinas de estado, máquinas de estado y clases, y la sintaxis de la máquina de estado. Luego se centra en los componentes básicos de las máquinas de estado: estados, transiciones y eventos.

21.2. Máquinas de estado

Tanto los diagramas de actividad como los diagramas de máquina de estado modelan aspectos del comportamiento dinámico de un sistema, pero tienen semántica y propósitos muy diferente en modelado. Los diagramas de actividad están basados en Petri Nets (véase el capítulo 14) y tienden a utilizarse para modelar procesos de negocio en los que participan varios objetos. Las máquinas de estado UML están basadas en el trabajo de Harel [Harel 1] y tienden a utilizarse para modelar la historia del ciclo de vida de un solo objeto reactivo como una máquina de estado finita, una máquina que puede existir en un número finito de estados. La máquina realiza transiciones entre estos estados en respuesta a eventos de una forma bien definida.

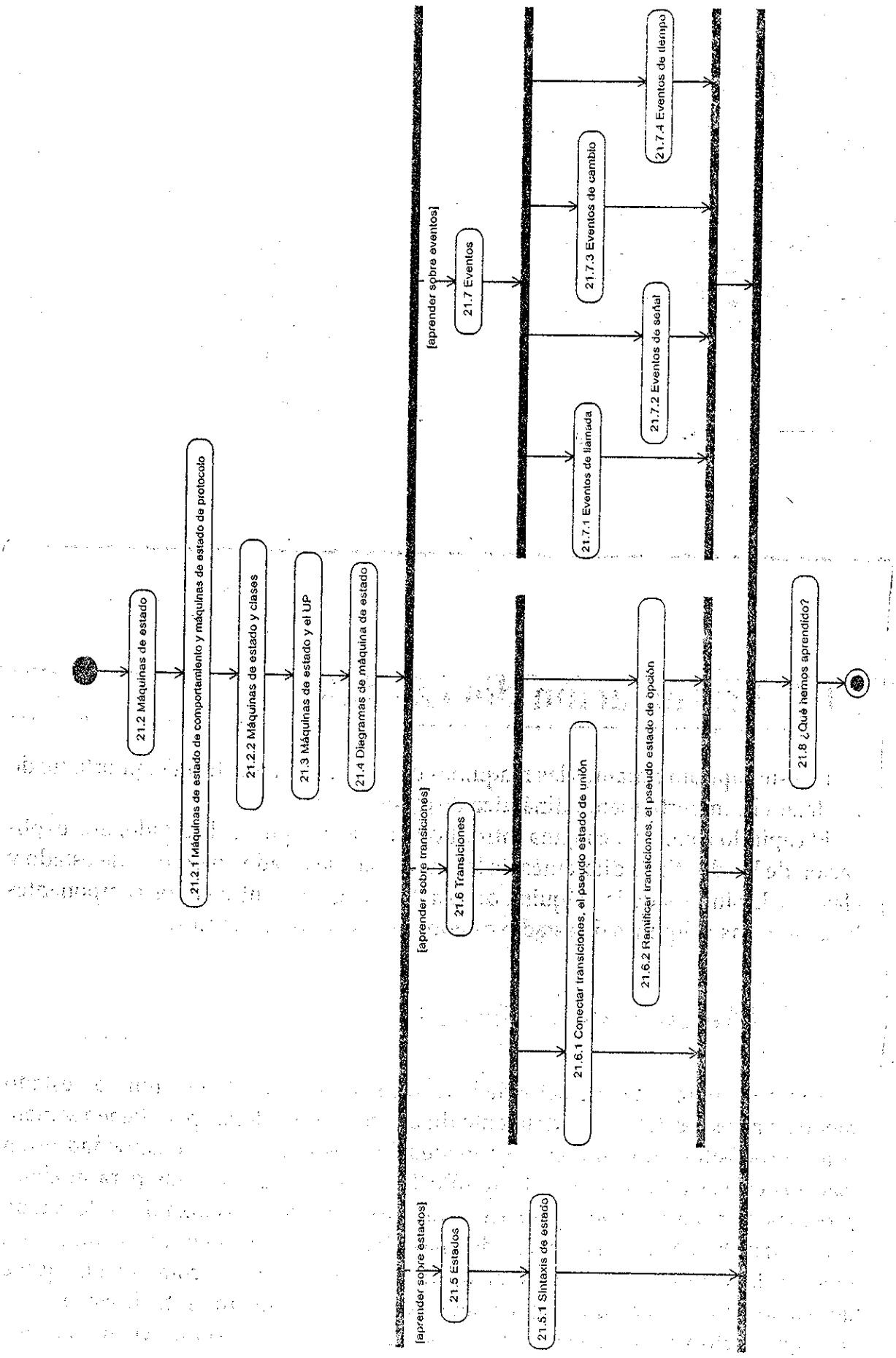

21.2 Máquinas de estado
21.2.1 Máquinas de estado de comportamiento y máquinas de estado de protocolo
21.2.2 Máquinas de estado y clases
21.3 Máquinas de estado y el UP
21.4 Diagramas de máquina de estado
[aprender sobre estados]
21.5 Estados
21.5.1 Sintaxis de estado
[aprender sobre transiciones]
21.6 Transiciones
21.6.1 Conectar transiciones, el pseudo estado de unión
21.6.2 Ramificar transiciones, el pseudo estado de opción
[aprender sobre eventos]
21.7 Eventos
21.7.1 Eventos de llamada
21.7.2 Eventos de señal
21.7.3 Eventos de cambio
21.7.4 Eventos de tiempo
21.8 ¿Qué hemos aprendido?

Figura 21.1.

Los tres elementos clave de las máquinas estado son estados, eventos y transiciones:

- **Estado:** "una condición o situación durante la vida de un objeto durante el cual se cumple alguna condición, realiza alguna actividad o espera algún evento". [Rumbaugh 1].
- **Evento:** "la especificación de una ocurrencia que tiene ubicación en tiempo y espacio" [Rumbaugh 1].
- **Transición:** El movimiento de un estado a otro en respuesta a un evento.

Examinamos los estados, eventos y transiciones en mucho más detalle más adelante en este capítulo.

Un objeto reactivo es un objeto, en el sentido amplio del término, que proporciona el contexto para una máquina estado. Los objetos reactivos:

- Responden a eventos externos (por ejemplo, eventos fuera del contexto del objeto).
- Pueden generar y responder a eventos internos.
- Tienen un ciclo de vida modelado como una progresión de estados, transiciones y eventos.
- Pueden tener un comportamiento que depende de comportamiento pasado.

El mundo real está lleno de objetos reactivos que se pueden modelar con máquinas de estado. En el modelado UML, las máquinas de estado están normalmente definidas en el contexto de un clasificador determinado. La máquina de estado modela el comportamiento común para todas las instancias de ese clasificador. Puede utilizar máquinas de estado para modelar el comportamiento dinámico de clasificadores como:

- Clases.
- Casos de uso.
- Subsistemas.
- Sistemas enteros.

21.2.1. Máquinas de estado de comportamiento y máquinas de estado de protocolo

La especificación UML 2 nos dice que existen dos tipos de máquinas de estado que comparten una sintaxis común:

- Máquinas de estado de comportamiento.
- Máquinas de estado de protocolo.

Las máquinas de estado de comportamiento utilizan estados, transiciones y eventos para definir el comportamiento del clasificador de contexto. Las máquinas de estado de comportamiento solamente se pueden utilizar cuando el clasificador de contexto tiene un comportamiento a modelar de algún tipo. Algunos clasificadores, por ejemplo, interfaces y puertos, no tienen comportamiento, simplemente definen un protocolo de uso. Los estados en las máquinas de estado de comportamiento pueden especificar una o más acciones que se ejecutan cuando se entra en el estado, se está en el estado, o se sale de éste.

Las máquinas de estado de protocolo utilizan estados, transiciones y eventos para definir el protocolo del clasificador de contexto. Este protocolo incluye lo siguiente:

- Las condiciones bajo las cuales se pueden invocar operaciones en el clasificador y sus instancias.
- Los resultados de llamadas de operación.
- El orden de llamadas de operación.

Las máquinas de estado de protocolo no dicen nada sobre la implementación de este comportamiento; solamente definen cómo el comportamiento aparece ante una entidad externa. Las máquinas de estado de protocolo se pueden utilizar para definir el protocolo para todos los clasificadores, incluidos aquellos que no tienen implementación. Los estados en máquinas de estado de protocolo no pueden especificar acciones, éste es el trabajo de las máquinas de estado de comportamiento.

En la práctica, los modeladores raramente distinguen entre máquinas de estado de comportamiento y de protocolo. Sin embargo, si lo desea, puede utilizar la palabra clave `{protocolo}` detrás del nombre de la máquina de estado de protocolo.

21.2.2. Máquinas de estado y clases

Las máquinas de estado se utilizan más comúnmente para modelar el comportamiento dinámico de clases y eso es en lo que nos centramos en el resto de este capítulo. Toda clase puede tener una sola máquina estado de comportamiento que modela todos los posibles estados, eventos y transiciones para todas las instancias de esa clase.

Toda clase puede también tener una o más máquinas de estado de protocolo, aunque éstas se utilizan más a menudo con clasificadores sin comportamiento como interfaces y puertos. Una clase hereda las máquinas de estado de protocolo de sus padres. Si una clase tiene más de una máquina de estado, deben ser coherentes entre sí.

Las máquinas de estado de comportamiento y protocolo para una clase deberían especificar el comportamiento y protocolo requerido por todos los casos de uso en el que participan las instancias de la clase. En caso que encuentre que un caso de uso requiere un protocolo o comportamiento que no se captura en una máquina de estado, esto indica que las máquinas de estado están en cierto sentido incompletas.

21.3. Máquinas de estado y el UP

Como los diagramas de actividad, no existe ningún lugar donde las máquinas de estado encajen en el UP. Puede utilizarlas en el workflow de análisis para modelar el ciclo de vida de clases que tienen estados interesantes como `Pedido` y `CuentaBancaria` y también puede utilizarlas en el workflow de diseño para modelar elementos como concurrencia y beans de sesión sin estado de Java. Incluso las hemos utilizado en el workflow de requisitos cuando tratábamos de entender un caso de uso complejo.

Como siempre, la pregunta es ¿añadirá valor a su modelo crear una máquina estado para algo? Si crear una máquina de estado le ayuda a entender un ciclo de vida complejo o comportamiento, merece la pena hacerlo. De lo contrario, no debería preocuparse.

Probablemente encontrará que utiliza máquinas de estado al final de la fase de elaboración y al principio de la fase de construcción. Eso es cuando está tratando de entender las clases en su sistema con suficiente detalle para que se puedan implementar. Algunas veces, una máquina de estado puede ser una ayuda valiosa para esto.

Desde nuestra perspectiva, el principal problema con las máquinas de estado es probarlas. El workflow de prueba de UP está fuera del alcance de este libro, pero creemos que es importante decir algo sobre las máquinas de estado de prueba ya que éste es un aspecto de las pruebas que los analistas y diseñadores tienen que hacer. Cuando crea una máquina de estado, ¿cómo sabe que es correcta? Con la mayoría de las herramientas de modelado UML, no tiene otra opción que realizar un ensayo manual donde alguien pretende que esté esa máquina de estado de modo que pueda ver cómo reacciona bajo diferentes circunstancias. A veces es mejor trabajar en un pequeño grupo con el propietario de la máquina de estado y otros modeladores y expertos.

Sin embargo, la mejor forma de crear y comprobar las máquinas de estado es simularlas. Existen varias herramientas disponibles que le permiten hacer esto, por ejemplo, RealTime Studio de Artisan Software (`www.artisansw.com`). Con la simulación, puede ejecutar la máquina de estado para ver cómo se comporta. Algunas herramientas también le permiten generar código de las máquinas de estado. Para el modelado de negocio, dichas herramientas pueden ser excesivas, pero para los sistemas incorporados en tiempo real, donde los objetos pueden tener un comportamiento y ciclos de vida complejos, pueden ser una verdadera ventaja.

21.4. Diagramas de máquina de estado

Para ilustrar los diagramas de máquina de estado consideremos un sencillo ejemplo del mundo real. Uno de los objetos más sencillos y más obvios del mundo real que constantemente pasa en ciclo por una máquina de estado es una bombilla.

La figura 21.2 muestra cómo puede enviar eventos a una bombilla utilizando un enchufe. Los dos eventos que puede enviar son `encender` (este evento modelo el suministro de corriente eléctrica a la bombilla) y `apagar` (que corta la corriente).

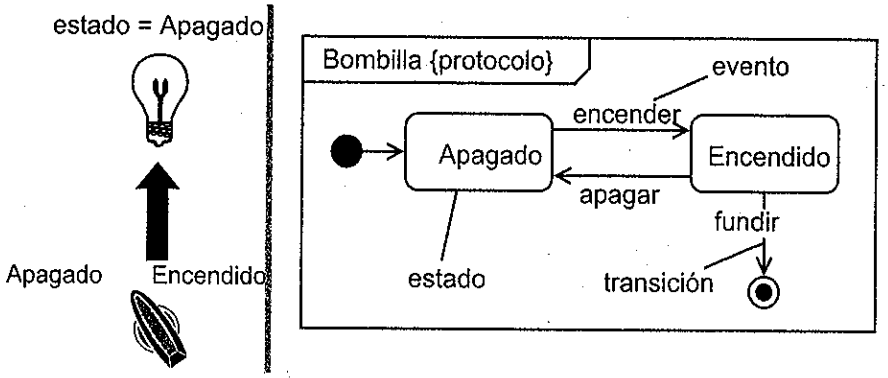


Figura 21.2.

Un diagrama de máquina de estado contiene exactamente una máquina de estado para un solo objeto reactivo. En este caso, el objeto reactivo es un sistema que engloba la bombilla, el enchufe y el suministro eléctrico. El diagrama de máquina de estado se puede dibujar en un marco explícito como ilustra la figura 21.2, o puede existir dentro de marcos implícitos proporcionados por una herramienta de modelado.

Puede prefijar el nombre de la máquina de estado con `Maquina Estado` si lo desea, pero esto no es necesario ya que las máquinas de estado tienen una sintaxis bastante identificable.

- Los estados son rectángulos redondeados aparte del estado inicial de partida (círculo relleno) y estado de parada (diana).
- Las transiciones indican posibles rutas entre estados y se modelan por una fecha.
- Los eventos se escriben sobre las transiciones que activan.

La semántica básica es también bastante sencilla. Cuando un objeto reactivo en estado `A` recibe el evento `unEvento` puede pasar al estado `B`.

Toda máquina de estado debería tener un estado inicial de partida (círculo relleno) que indica el primer estado de la secuencia y, a menos que los estados pasen en ciclo interminablemente, también deberían tener un estado final (diana) que termina la secuencia de transiciones. Normalmente, pasa del pseudo estado inicial al estado "real" de la máquina de estado. El pseudo estado inicial se utiliza como un marcador para el principio de las series de transición de estado.

En la figura 21.2 cuando el enchufe se gira a la posición "Encendido", el evento `encender` se envía a la bombilla. Ahora, en las máquinas de estado los eventos se consideran instantáneos. Es decir, tarda cero veces que el evento enviado desde el enchufe llegue a la bombilla. Los eventos instantáneos proporcionan una simplificación importante a la teoría de la máquina de estado que la hace mucho más

manejable. Sin eventos instantáneos podríamos tener condiciones de carrera donde dos eventos compiten desde su origen para alcanzar el mismo objeto reactivo. Tendríamos que modelar esta condición de carrera como algún tipo de máquina de estado. La bombilla recibe el evento `encender` y cambia el estado a `Encendido` en respuesta al evento. Los objetos pueden cambiar de estado al recibir un evento. Cuando el evento `apagar` se envía a la bombilla, ésta cambia el estado a `Apagado`.

En algún momento puede ocurrir el evento `fundir` (cuando la bombilla se funde). Esto termina la máquina de estado. Examinamos cada elemento de la máquina de estado en detalle en los siguientes apartados.

21.5. Estados

El *The UML Reference Manual* [Rumbaugh 1] define un estado como "una condición o situación durante la vida de un objeto durante la cual cumple cierta condición, realiza cierta actividad o espera algún evento". El estado de un objeto varía con el tiempo, pero en cualquier punto está determinado por:

- Los valores del atributo del objeto.
- Las relaciones que tiene con otros objetos.
- Las actividades que realiza.

Con el tiempo, los objetos se envían mensajes entre sí y estos mensajes son eventos que pueden causar cambios en el estado del objeto. Es importante pensar detenidamente en lo que queremos decir con "estado". En el caso de la bombilla, podríamos decidir que cualquier cambio a cualquiera de los átomos en la bombilla constituye un nuevo estado. Esto es perfectamente correcto, pero nos proporcionaría una infinidad de estados, la mayoría de los cuales serían virtualmente idénticos.

Sin embargo, desde el punto de vista del usuario de una bombilla, los únicos estados que marcan una diferencia son `Encendido` y `Apagado` y el estado final cuando la bombilla se funde. Ésta es la clave para modelar estados con éxito; necesita identificar estados que marquen una diferencia para su sistema.

Como otro ejemplo, considere la clase `Color` que se muestra en la figura 21.3.

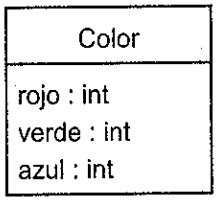


Figura 21.3.

Si asumimos que rojo, verde y azul pueden adoptar los valores 0-255, entonces basándose en los valores de estos atributos, los objetos de esta clase pueden tener 256*256*256=16777216 estados posibles. Sin embargo, debemos hacernos la pre-

gunta: ¿cuál es la diferencia semántica clave entre cada uno de esos estados? La respuesta es ninguna. Cada uno de los 16777216 posibles estados representa un color y eso es todo. De hecho, la máquina de estado para esta clase es muy aburrida y puede modelar todas las posibilidades con un solo estado.

En resumen, tiene que existir "diferencia semántica que marque una diferencia" entre estados para que pueda modelarlos en una máquina de estado. Deben añadir valor a su modelo. Verá ejemplos de máquinas de estado que añaden valor en este capítulo y en el siguiente.

21.5.1. Sintaxis de estado

La sintaxis de estado de UML se resume en la figura 21.4.

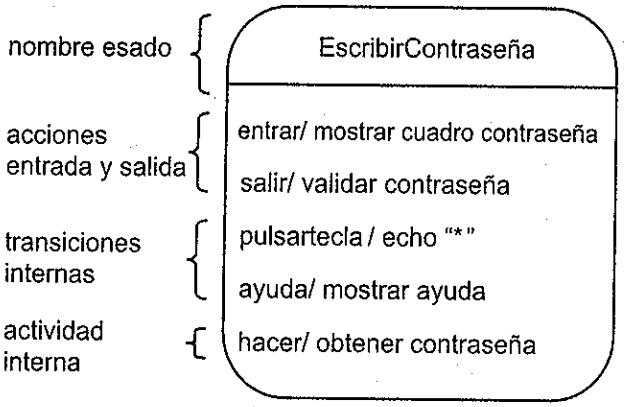


Figura 21.4.

Todo estado en una máquina estado de comportamiento puede contener cero o más acciones y actividades. Los estados en máquinas de estado de protocolo no tienen acciones o actividades.

Las acciones se consideran instantáneas y sin interrupción, mientras que las actividades adoptan una cantidad de tiempo finito y se pueden interrumpir. Toda acción en un estado está asociada con una transición interna que está activada por un evento. Puede haber cualquier número de acciones y transiciones internas dentro de un estado.

Una transición interna le permite capturar el hecho de que algo que merece modelarse ha ocurrido pero no provoca una transición a un nuevo estado. Por ejemplo, en la figura 21.4 pulsar una de las teclas en el teclado es un evento destacable pero no provoca una transición fuera del estado `EscribirContraseña`. Lo modelamos como un elemento interno, `pulsartecla`, que causa una transición interna que activa la acción `echo"*"`.

Dos acciones especiales, la acción de entrada y la acción de salida, están asociadas con los eventos especiales `entrar` y `salir`. Estos dos eventos tienen semántica especial. El evento `entrar` ocurre instantáneamente y automáticamente al entrar

en el estado; es lo primero que ocurre cuando se entra en el estado y hace que la acción de entrada asociada se ejecute. El evento `salir` es lo último que sucede instantáneamente y automáticamente al salir del estado y hace que la acción de salida asociada se ejecute.

Las actividades, por otro lado, adoptan una cantidad finita de tiempo y se pueden interrumpir al recibir un evento. La palabra clave `hacer` indica una actividad. Mientras que las acciones siempre terminan porque son atómicas, es posible interrumpir una actividad antes de que se haya terminado de procesar.

21.6. Transiciones

La sintaxis de transición de UML para máquinas de estado de comportamiento se resume en la figura 21.5.

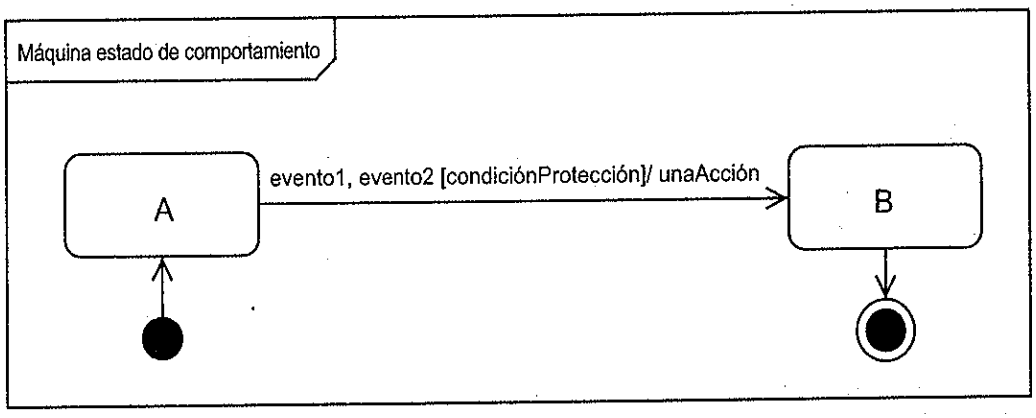


Figura 21.5.

Las transiciones en una máquina de estado de comportamiento tienen una sintaxis sencilla que se puede utilizar para transiciones externas (mostradas por una flecha) o transiciones internas (anidadas dentro de un estado). Toda transición tiene tres elementos opcionales.

1. **Cero o más eventos:** Éstos especifican ocurrencias externas o internas que pueden activar la transición.
2. **Cero o una condición de protección:** Ésta es una expresión booleana que debe evaluar en verdadero antes de que pueda ocurrir la transición. Se sitúa detrás de los eventos.
3. **Cero o más acciones:** Esto es parte de un trabajo asociado con la transición y ocurre cuando se activa la transición.

Puede leer la figura 21.5 de la siguiente manera. "En (`evento1` OR `evento2`) si (`condiciónProtección` es verdadera) entonces realizar `unaAcción` e inmediatamente entrar en estado `B`". La acción puede implicar variables en el ámbito de la máquina de estado. Por ejemplo:

```
acciónRealizada(eventoAcción)/comando =eventoAcción.obtenerComandoAcción()
```

En este ejemplo `acciónRealizada(eventoAcción)` es un evento generado al pulsar un botón en un GUI de Java. Al recibir este evento, ejecutamos una acción que almacena el nombre del botón en la variable `comando`.

Las transiciones en las máquinas de estado de protocolo tienen una sintaxis algo diferente, como se ilustra en la figura 21.6.

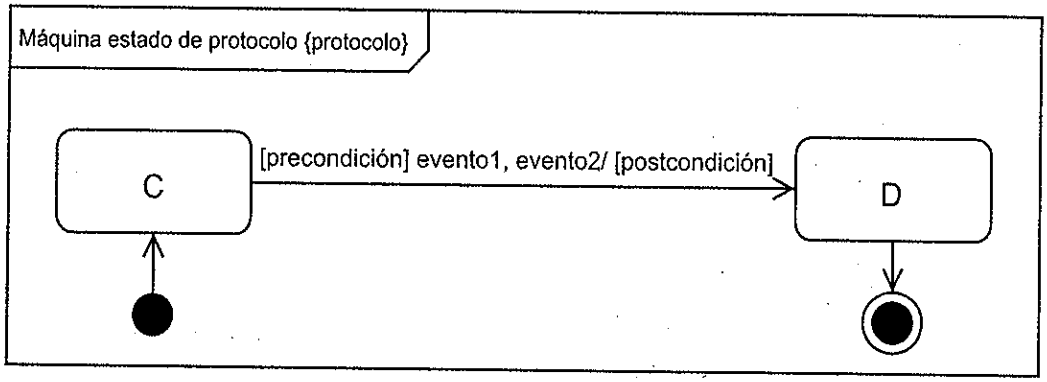


Figura 21.6.

- No existe acción ya que estamos especificando un protocolo en lugar de una implementación.
- La condición de protección se reemplaza por precondiciones y postcondiciones. Observe que la precondición se sitúa delante de los eventos y la postcondición detrás de la barra inclinada.

En las máquinas de estado de comportamiento y protocolo, si una transición no tiene evento, es una transición automática. Una transición automática no espera a un evento y se activa cuando su condición de protección o precondición es verdadera.

21.6.1. Conectar transiciones, el pseudo estado de unión

Las transiciones se pueden conectar por pseudo estados de unión. Éstos representan puntos donde las transiciones se unen o ramifican. Se representan como círculos rellenos que tienen una o más transiciones de entrada y una o más transiciones de salida. El ejemplo en la figura 21.7 muestra una máquina de estado para la clase `Préstamo` que hemos incorporado en el capítulo 18. `Préstamo` modela el préstamo de un libro de una biblioteca. La máquina de estado para `Préstamo` tiene una sencilla unión. Éste es el uso más común de los pseudo estados de unión.

Un pseudo estado de unión puede tener más de una transición de salida. Si éste es el caso, toda transición de salida debe protegerse por una condición de protección mutuamente exclusiva de modo que solamente se pueda activar una transición de salida. Un ejemplo se muestra en la figura 21.8, donde hemos ampliado la máquina de estado para la clase `Préstamo` para gestionar el caso en el que un `Préstamo` se puede ampliar. Una regla de negocio es que un libro prestado se tiene

que presentar en la biblioteca para ampliar su préstamo, de modo que los eventos `devolverLibro` son todavía válidos.

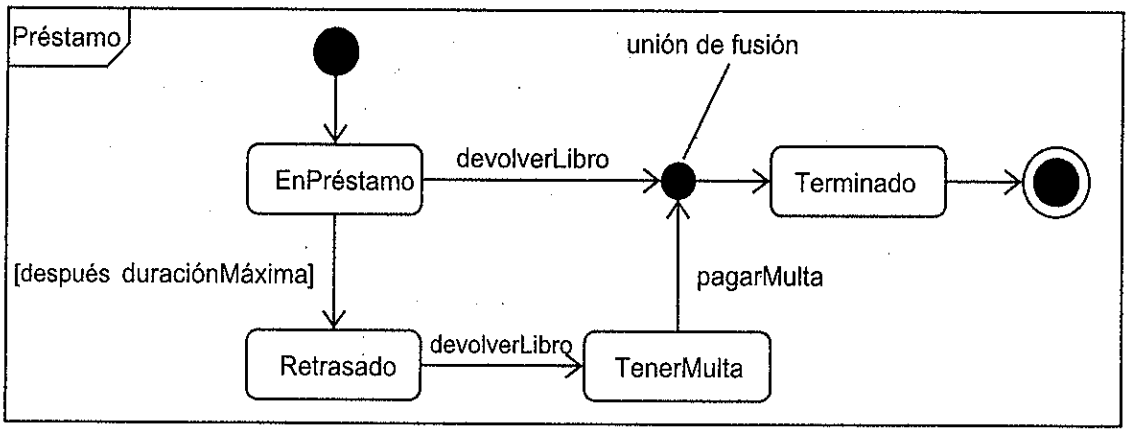


Figura 21.7.

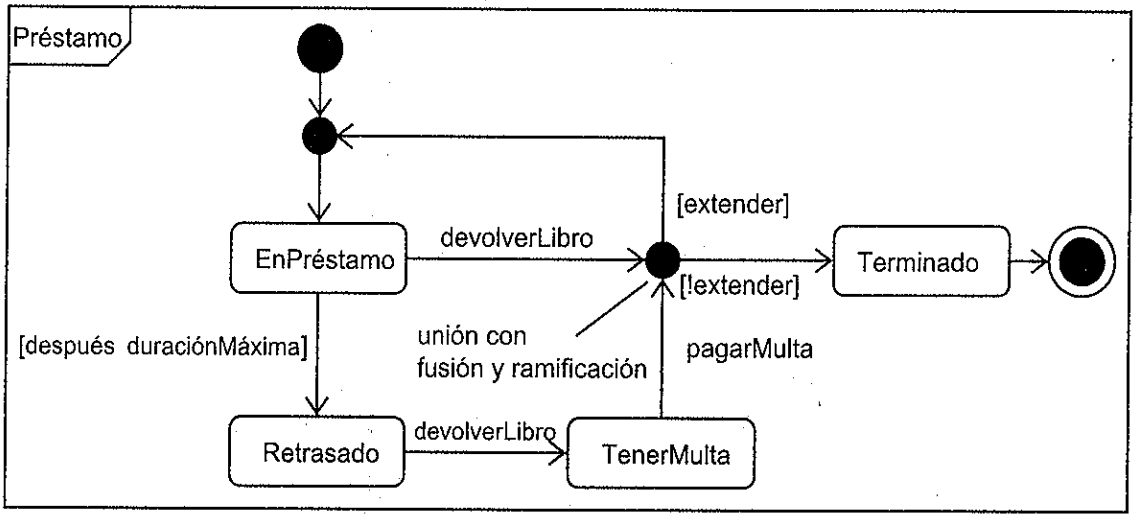


Figura 21.8.

21.6.2. Ramificar transiciones, el pseudo estado de opción

Si quiere mostrar una sencilla ramificación sin una unión, debería utilizar un pseudo estado de opción, como se muestra en la figura 21.9.

El pseudo estado de opción le permite dirigir el flujo por la máquina de estado de acuerdo a condiciones que especifica en sus transiciones de salida. Por ejemplo, la figura 21.9 muestra una máquina de estado de comportamiento para una sencilla clase `PréstamoBanco`. Al recibir el evento `aceptarPago`, el objeto `PréstamoBanco` pasa del estado `Impagado` a uno de los tres estados `TotalmentePagado`, `SobrePagado` y `ParcialmentePagado`, dependiendo de la cantidad del `pago` comparado con el `saldo` del `PréstamoBanco`. Las condiciones en las transiciones de salida del pseudo estado de opción deben ser mutuamente exclusivas para asegurarse de que solamente una de ellas puede lanzarse en cualquier momento.

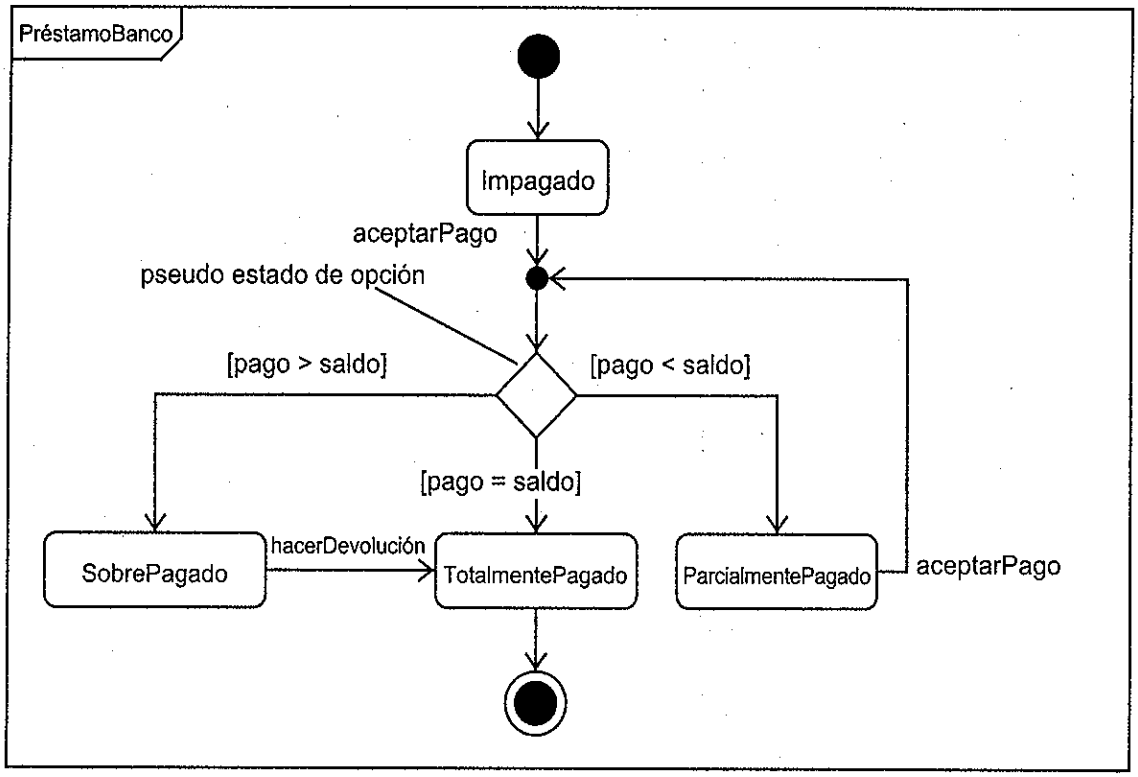


Figura 21.9.

21.7. Eventos

UML define un evento como "la especificación de una ocurrencia destacable que tiene ubicación en tiempo y espacio". Los eventos activan transiciones en máquinas de estado. Los eventos se pueden mostrar externamente en transiciones, como se muestra en la figura 21.10, o internamente dentro de estados, como se muestra en la figura 21.11.

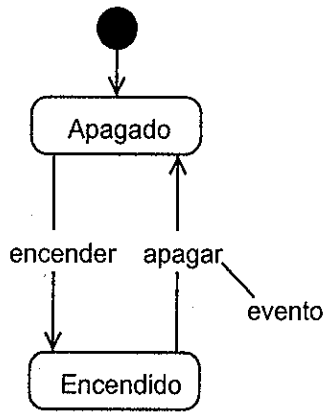


Figura 21.10.

Existen cuatro tipos de evento, cada uno de los cuales con semántica diferente:

- Evento de llamada.
- Evento de señal.

- Evento de cambio.
- Evento de tiempo.

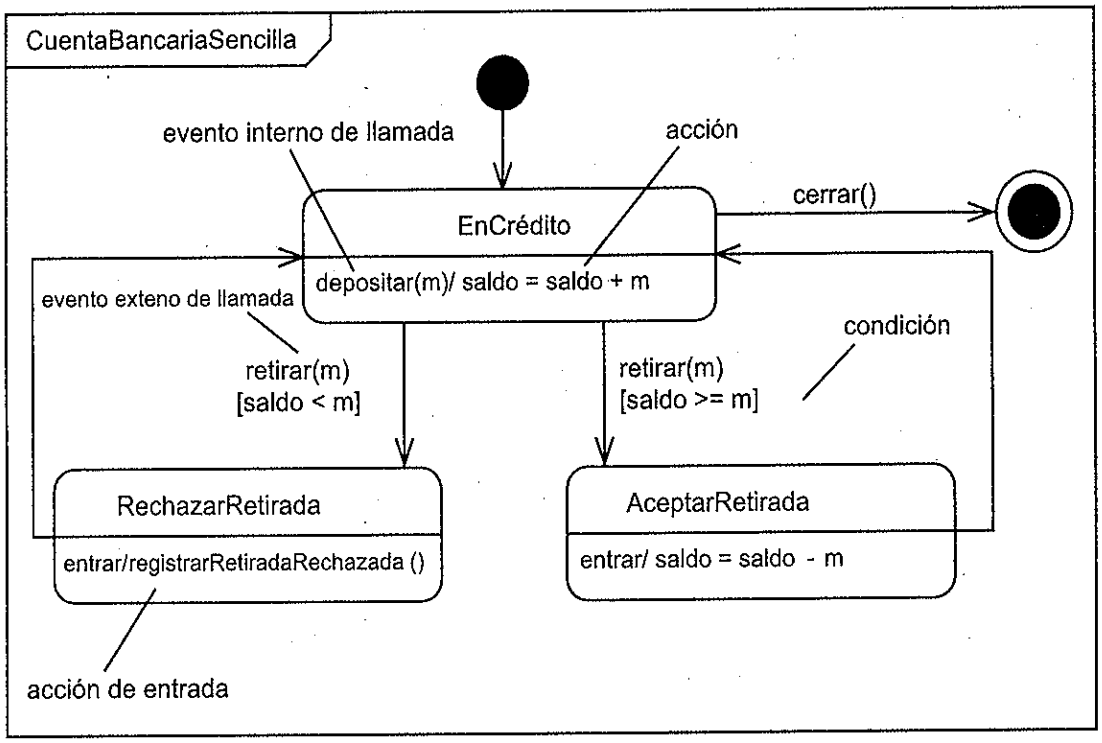


Figura 21.11.

21.7.1. Evento de llamada

Un evento de llamada es una petición de una operación específica a invocarse en una instancia de la clase de contexto.

Un evento de llamada debería tener la misma firma que una operación en la clase de contexto de la máquina de estado. Recibir un evento de llamada es un activador para que se ejecute la operación. Como tal, un evento de llamada es quizás el tipo más sencillo de evento.

El ejemplo en la figura 21.11 muestra un fragmento de la máquina de estado de una clase `CuentaBancariaSencilla`. Esta clase está sujeta a las siguientes restricciones de negocio:

- Las cuentas deben tener siempre un saldo mayor o igual a cero.
- La retirada de dinero se puede rechazar si hace que el saldo se quede por debajo de cero.

La figura muestra eventos de llamada internos y externos. Éstos corresponden a operaciones de la clase `CuentaBancariaSencilla`. Puede especificar una secuencia de acciones para un evento de llamada donde cada acción está separada por un punto y coma. Estas acciones especifican la semántica de la operación y pueden utilizar atributos y operaciones de la clase de contexto. Si la operación tiene un tipo de retorno, el evento de llamada tiene un valor de retorno de ese tipo.

21.7.2. Eventos de señal

Una señal es un paquete de información que se envía asíncronamente entre objetos. Modele una señal como una clase estereotipada que alberga la información a comunicarse en sus atributos como lo muestra la figura 21.12. Una señal normalmente no tiene ninguna operación porque su única finalidad es transportar información.

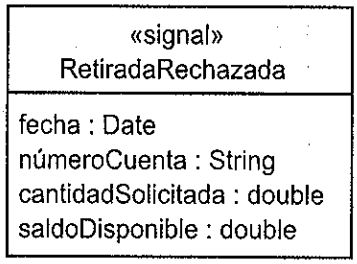


Figura 21.12.

En la figura 21.13 hemos actualizado la máquina de estado para `CuentaBancariaSencilla` de modo que envía una señal cuando se rechaza una retirada de efectivo. Un envío de señal se indica por medio de un pentágono convexo con el nombre de la señal dentro. Ésta es la misma sintaxis que se utiliza en diagramas de actividad.

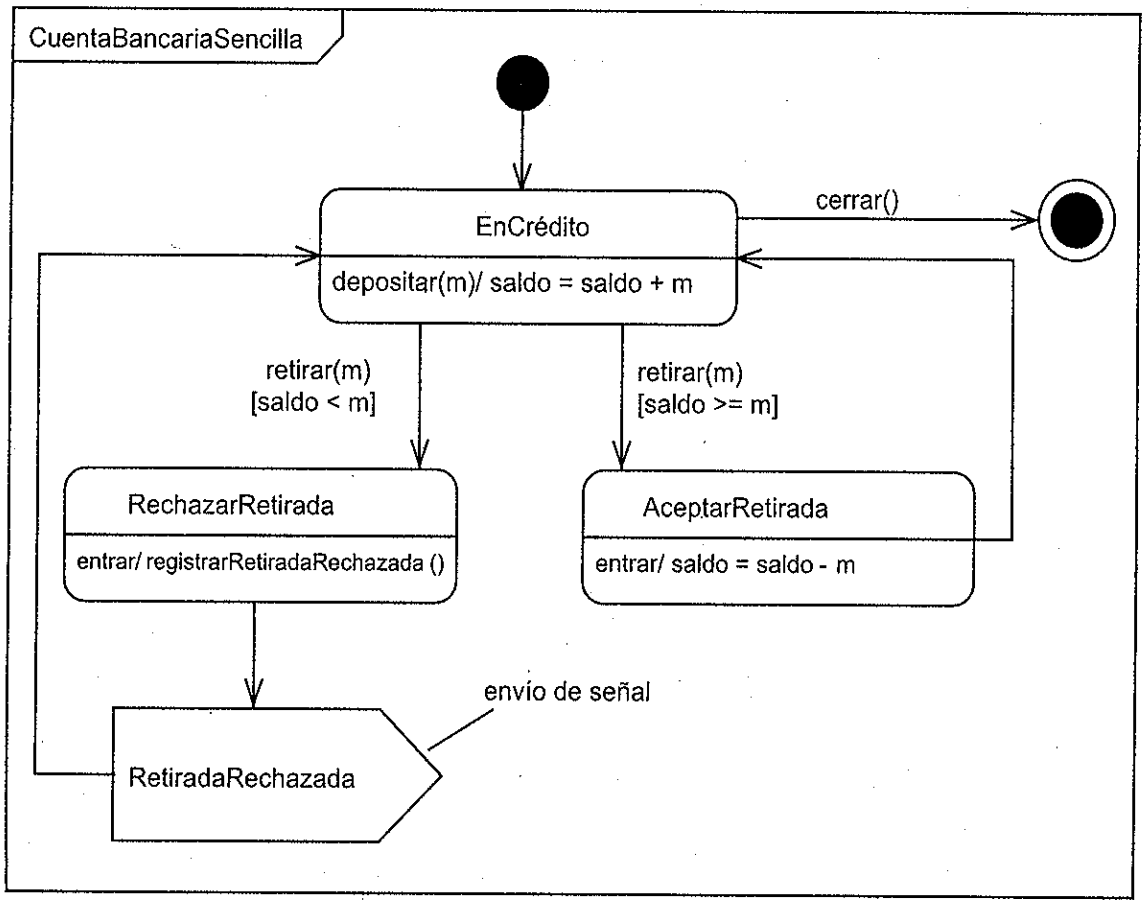


Figura 21.13.

Recibir una señal se indica por medio de un pentágono cóncavo, como se muestra en el fragmento de máquina de estado en la figura 21.14. Especifica una operación de la clase de contexto que afecta a la señal como un parámetro.

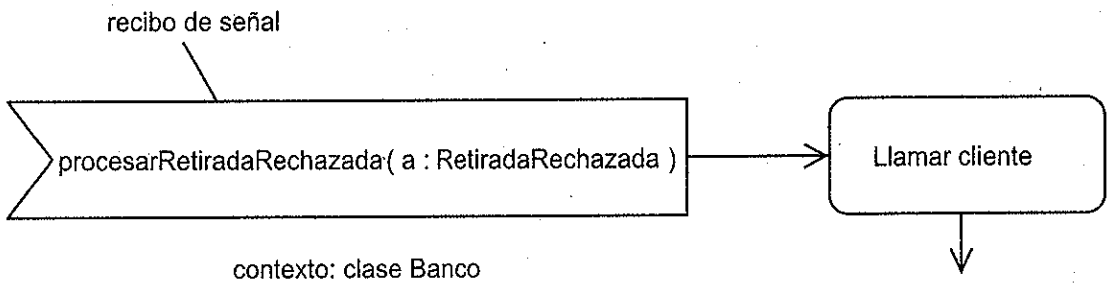


Figura 21.14.

También puede mostrar la recepción de señal en transiciones internas o externas al utilizar la notación estándar `nombreEvento/acción` que se ha tratado anteriormente.

21.7.3. Eventos de cambio

Un evento de cambio se especifica como una expresión booleana según ilustra la figura 21.15. La acción asociada con el evento se realiza cuando el valor de la expresión booleana pasa de falso a verdadero. Todos los valores en la expresión booleana deben ser constantes, globales o atributos u operaciones de la clase de contexto. Desde la perspectiva de implementación, un evento de cambio implica comprobar continuamente la condición booleana mientras se esté en el estado. En la figura 21.15 hemos modificado la máquina de estado `CuentaBancariaSencilla` de modo que el gestor sepa si el `saldo` de la cuenta supera o es igual a 5000. Esta notificación es para que el gestor pueda avisar al cliente sobre otras opciones de inversión.

Los eventos de cambio se activan desde el lado positivo. Es decir, se activan cada vez que el valor de la expresión booleana cambia de falso a verdadero. La expresión booleana debe regresar a falso y luego pasar a verdadero de nuevo para que se vuelva a activar el evento de cambio.

Éste es precisamente el comportamiento que queremos para nuestra `CuentaBancariaSencilla` La acción `notificarGestor()` se invocará solamente cuando el saldo de la cuenta pase de menos de 5000 a 5000 ó más. Claramente, si el saldo oscila rápidamente en torno a 5000, el gestor recibirá múltiples notificaciones. Asumimos que el gestor tiene algún proceso de negocio para gestionar esto.

21.7.4. Eventos de tiempo

Los eventos de tiempo se indican normalmente por medio de las palabras clave `cuando` y `después`. La palabra clave `cuando` especifica un momento particular en

el que se activa el evento; `después` especifica un momento después del que se activa el evento. Ejemplos son `cuando(fecha=07/10/2005)` y `después(3meses)`.

Siempre debería asegurarse de que las unidades de tiempo (horas, días, meses, etc.) se graban en el diagrama para cada evento de tiempo. Cualquier símbolo en la expresión deben ser constantes, globales o atributos u operaciones de la clase de contexto.

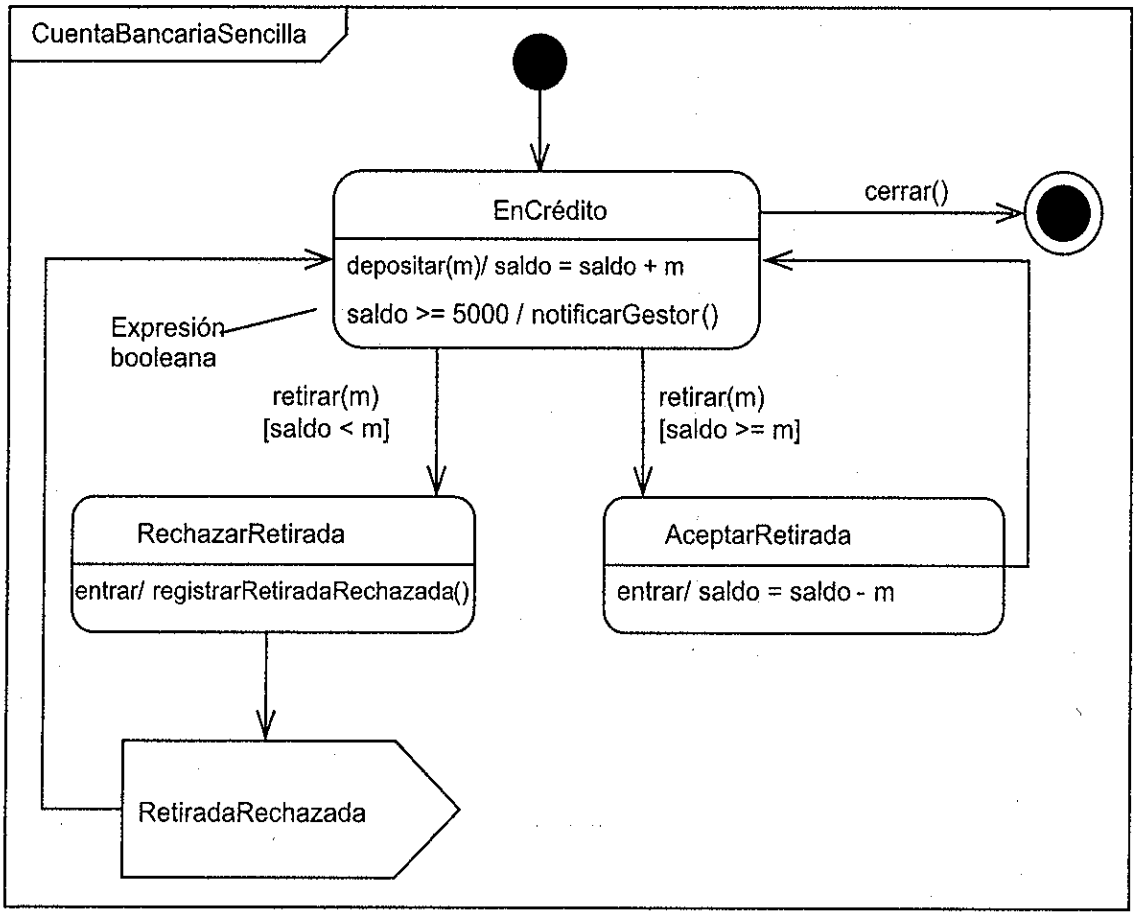
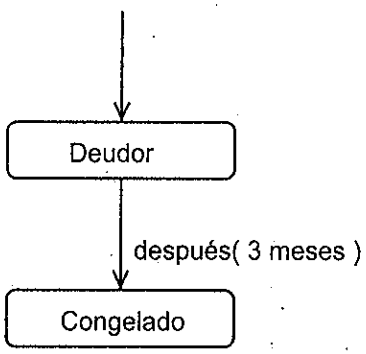


Figura 21.15.

El ejemplo en la figura 21.16 es un fragmento de la máquina de estado de la clase `CuentaCrédito` que tiene un crédito limitado. Puede ver que después de que el objeto `CuentaCrédito` ha estado en el estado `Deudor` durante tres meses pasa al estado `Congelado`.



contexto: clase CuentaCrédito

Figura 21.16.

21.8. ¿Qué hemos aprendido?

En este capítulo ha visto cómo construir máquinas de estado básicas al utilizar estados, acciones, actividades, eventos y transiciones.

Ha aprendido lo siguiente:

- Las máquinas de estado están basadas en el trabajo de Harel.
  - Las máquinas de estado modelan el comportamiento dinámico de un objeto reactivo.
  - Los diagramas de máquina de estado contienen una sola máquina de estado para un solo objeto reactivo.
- Un objeto reactivo proporciona el contexto para una máquina de estado.
  - Objetos reactivos:
    - Responden a eventos externos.
    - Pueden generar eventos internos.
    - Tienen un ciclo de vida claro que se puede modelar como una progresión de estados, transiciones y eventos.
    - Tienen comportamiento actual que puede dependen de comportamiento pasado.
  - Ejemplos de objetos reactivos:
    - Clases (lo más común).
    - Casos de uso.
    - Subsistemas.
    - Sistemas enteros.
- Existen dos tipos de máquina de estado:
  - Máquinas de estado de comportamiento:
    - Modelan el comportamiento de un clasificador de contexto.
    - Los estados en las máquinas de estado de comportamiento pueden contener cero o más acciones y actividades.
  - Máquinas de estado de protocolo:
    - Modelan el protocolo de un clasificador de contexto.
    - Los estados en máquinas de estado de protocolo no tienen acciones o actividades.
- Acciones: Trabajo que es instantáneo y sin interrupción.

- Pueden ocurrir dentro de un estado, asociadas con una transición interna.
- Pueden ocurrir fuera de un estado, asociadas con una transición externa.

- Actividades: Trabajo que adopta un tiempo finito y se pueden interrumpir.
  - Pueden ocurrir solamente dentro de un estado.
- Estado: Una condición semánticamente importante de un objeto.
  - El estado del objeto está determinado por:
    - Valores del atributo del objeto.
    - Relaciones con otros objetos.
    - Actividades que realiza el objeto.
  - Sintaxis de estado:
    - Acción de entrada: Realizada inmediatamente al entrar en el estado.
    - Acción de salida: Realizada inmediatamente al salir del estado.
    - Transiciones internas: Están causadas por eventos que no son suficientemente importantes para garantizar una transición a un nuevo estado: el evento se procesa por una transición interna dentro del estado.
    - Actividad interna: Un trabajo que adopta una cantidad finita de tiempo y que se puede interrumpir.
- Transición: Un movimiento entre dos estados.
  - Sintaxis de transición:
    - Evento: El evento que activa la transición.
    - Condición de protección: Una expresión booleana que debe ser cierta antes de que ocurra la transición.
    - Acción: Una acción que ocurre instantáneamente con la transición.
    - Pseudo estado de unión: Une o ramifica transiciones.
    - Pseudo estado de opción: Dirige el flujo por la máquina de estado según las condiciones.
- Evento: algo que ocurre a un objeto reactivo. Los tipos de eventos son:
  - Evento de llamada:
    - Una llamada para que ocurran un conjunto de acciones.
    - Una invocación sobre el objeto.
  - Evento de señal:
    - Recibir una señal: Una señal es una comunicación asíncrona de una sola dirección entre objetos.

- Evento de cambio:
  - Ocurre cuando alguna condición booleana cambia de falso a verdadero.
- Evento de tiempo:
  - Palabra clave `después`: Ocurre después de un período de tiempo.
  - Palabra clave `cuando`: Ocurre cuando alguna condición de tiempo se convierte en verdadera.