UML (II): el modelo dinámico

Benet Campderrich Falgueras
Recerca Informàtica, S.L.

PID_00198153

UOC
Universitat Oberta de Catalunya

Índice

Introducción

Hemos visto que el modelo estático de UML consiste esencialmente en un único diagrama, el de clases, que se tiene que elaborar en todo proyecto de *software*. El modelo dinámico y de implementación, en cambio, comprende diferentes diagramas, en parte relacionados entre sí y también con el diagrama estático, y algunos de estos diagramas sólo se utilizan en determinados casos.

Los aspectos dinámicos o de comportamiento del sistema se pueden modelar con los siguientes diagramas: el diagrama de casos de uso, el diagrama de estados y transiciones, el diagrama de actividad y los diagramas de interacción, que son el de secuencias y el de colaboración.

Objetivos

El objetivo principal de este módulo es conocer la notación y la semántica de las herramientas diagramáticas que UML ofrece para modelar los aspectos dinámicos y de implementación de un *software* que se quiere desarrollar de manera orientada a objetos, y saberlas aplicar. Este objetivo general se descompone en estos otros:

1. Aprender el diagrama de estados.

2. Adquirir los conceptos de caso de uso y de actor y saber identificar los diferentes casos de relaciones entre casos de uso.

3. Aprender a describir las interacciones mediante diagramas de comunicación y de secuencia y saberlos aplicar a la descripción de casos de uso y de operaciones complejas.

4. Aprender qué es un diagrama de actividades y entender las diferencias entre éste y el de estados.

1. El diagrama de estados

A veces hay objetos cuyo comportamiento puede variar a lo largo del tiempo; cuando esto sucede, se dice que el objeto tiene estados. Existen algunos tipos de aplicaciones, como las de tiempo real, para las cuales el modelado de estados es especialmente importante.

Ejemplos de estados

Hay objetos a los cuales no se puede pedir alguna de sus operaciones en cualquier momento, u objetos para los que alguno de sus atributos sólo puede tener un valor no nulo en circunstancias determinadas.

Información redundante, pero aclaractoria

La información que contiene el diagrama de estados es esencialmente redundante, ya que los cambios de estado son resultado de la dinámica del sistema y, por tanto, de la ejecución de las operaciones de la misma clase o de otras; pero aun así, representa otro punto de vista sobre la dinámica de una parte del sistema que puede contribuir decisivamente a comprenderla mejor.

En el **diagrama de estados** o **diagrama dinámico** tenemos que distingir los siguientes elementos:

- Las diferentes situaciones en que se puede encontrar un objeto (los estados).
- Qué cambios de estado son posibles (transiciones).
- Cuál es el hecho que los produce (acontecimientos).

En las especificaciones de UML se habla de *máquinas de estados* para distinguirlas de los *diagramas de estado clásicos* o *de Harel*, con los cuales existen diferencias. Sin embargo, nosotros utilizaremos el término *diagrama de estados* porque siempre representamos las máquinas de estado en forma de diagrama. No puede haber ninguna confusión, ya que en esta asignatura no se tratarán los diagramas de Harel.

El diagrama de estados se utiliza normalmente para describir objetos del dominio del usuario y se documenta normalmente en la etapa de análisis.

1.1. Conceptos básicos

A continuación explicaremos una serie de conceptos básicos:

1) Un **estado** es una situación determinada dentro de la vida de un objeto o la duración de una interacción durante la cual cumple alguna condición, lleva a cabo alguna acción o espera que se produzca un acontecimiento. Un estado no corresponde a un instante en el tiempo, sino que el objeto o interacción permanece en éste un tiempo finito.

Ved también

Consultad las interacciones en el apartado 3 de este módulo didáctico.

2) Una **transición simple** consiste en que el objeto o interacción pasa de un estado (**estado de origen**) a otro (**estado de destino**), que podría volver a ser el mismo. En el caso más general, este paso comienza cuando se produce un acontecimiento determinado y al mismo tiempo se cumple una condición especificada (**condición de guarda**); entonces se pueden ejecutar unas acciones y se pueden enviar mensajes a objetos individuales o a conjuntos de objetos.

Un estado puede tener **transiciones de llegada**, es decir, transiciones que tienen este estado como estado de destino; y **transiciones de salida**, es decir, transiciones que tienen este estado como estado de origen.

3) Las **transiciones internas** son pseudotransacciones en las que no hay cambio de estado. Sirven para especificar acciones que se deben ejecutar en respuesta a un acontecimiento que no provoca ningún cambio de estado en el objeto o interacción en cuestión.

Observación

Conviene no confundir una transición interna con una *autotransición*, que es una transición ordinaria en la que el estado de origen y el estado de destino son el mismo.

4) Una **acción** es la especificación de un proceso atómico, es decir, que o no se ejecuta o se ejecuta hasta el final. Como consecuencia de una transición, se pueden ejecutar una o más acciones.

5) Los objetos, por medio de mensajes, pueden recibir peticiones de operaciones o señales. Las **señales**, a diferencia de las operaciones, no realizan ningún proceso, y el único efecto directo que pueden tener es producir acontecimientos (que sí pueden provocar transiciones y la consiguiente ejecución de acciones).

6) Los acontecimientos[1] son hechos que, cuando se producen, pueden provocar transiciones de un estado a otro en objetos e interacciones y/o la ejecución de determinadas acciones. Un acontecimiento no va ligado a ningún objeto o clase en particular, sino que es un hecho que puede afectar en general a los elementos del paquete en cuyo interior está definido. Cada acontecimiento posee un nombre que lo identifica dentro del paquete, y puede tener parámetros.

(1)En inglés, *events*.

Normalmente, un acontecimiento se tiene que tratar[2] en el momento en que se produce; en caso contrario, se provoca la transición porque el objeto o interacción no está en el estado correspondiente, se pierde el acontecimiento a no ser que se declare como acontecimiento **diferido**.

(2)Por ejemplo, si se provoca.

Tipos de acontecimientos

Ahora veremos diferentes tipos de acontecimiento:

- **De llamada**: se producen cuando se llama una operación del objeto al que corresponde el diagrama.

- **De señal**: representan la recepción de una señal por el objeto a la que corresponde el diagrama.

- **De cambio**: representan una notificación de que una condición ha llegado a ser cierta. Esta condición no se tiene que confundir con condición de guarda, ya que una guarda se evalúa una vez se ha presentado el acontecimiento correspondiente a una transición para determinar si se provoca la transición o no, mientras que esta condición produce el acontecimiento cuando ha llegado a ser cierta.

- **De tiempo:** representan la notificación de que o ha pasado un periodo de tiempo desde que se ha producido un acontecimiento determinado (por ejemplo, que se ha entrado en el estado de origen de la transición), o de que es una hora determinada.

Acontecimientos internos

Son pseudoacontecimientos, ya que están ligados a un estado en lugar de a una transición, y sirven para poner en marcha acciones que no van asociadas a ningún cambio de estado concreto. Existen tres tipos de acontecimientos internos:

1) De entrada: se producen cuando el objeto entra en el estado correspondiente. No tienen ni guarda ni parámetros, porque se nombrarán implícitamente cuando haya una transición de entrada en el estado (incluida una autotransición), pero no cuando haya una transición interna, ya que entonces no se producirá un cambio de estado. Se especifican explícitamente y tienen como nombre la palabra clave *entry*.

2) De salida: son análogos a los de entrada, salvo que se producen a la salida del estado. Se especifican explícitamente y tienen como nombre la palabra clave *exit*.

3) De acción: especifican acciones que se ejecutan cuando se llega al estado en cuestión y acaban por sí mismas o cuando se sale del estado. Se especifican explícitamente y tienen como nombre la palabra clave *do*.

1.2. Notaciones básicas

La representación más sencilla de un estado es un retángulo con los vértices redondeados, tal y como podemos observar en la siguiente figura:

En la práctica, es imprescindible que cada estado tenga un nombre y que este nombre no se repita en ningún otro estado del mismo diagrama. Este hecho no impide que el objeto o interacción pueda llegar al mismo estado varias veces a lo largo de su vida.

Las transiciones se representan mediante flechas de punta abierta que van del estado de salida al de llegada. Con la flecha se encuentra una expresión (denominada *cadena de la transición*) que presenta la siguiente sintaxis formal:

```
signatura '[' guarda ']' '/' acción
```

Puede haber varias acciones o no haber ninguna. El orden en que aparecen es en el que se deben ejecutar.

A continuación veremos una explicación de cada uno de los elementos de la cadena de transición:

- **Signatura**: tiene un formato que depende del tipo de acontecimiento. En el caso de un acontecimiento de llamada o de señal, se define así:

```
nombre_acontecimiento '(' nombre_parámetro ':' expresión_tipo ',' ... ')'
```

Si el acontecimiento es de tiempo, la signatura adopta una de estas formas:

```
'after(' expresión_de_tiempo ')'
```

donde *expresión_de_tiempo* consiste en una duración a partir de un origen, o:

```
'when(' hora o fecha ')'
```

Finalmente, si el acontecimiento es de cambio, tendremos lo siguiente:

```
'when(' expresión_booleana ')'
```

- **Guarda**: es una expresión que puede tomar el valor *verdadero* o *falso*, escrita en pseudocódigo o en el lenguaje de programación que se utiliza.

- **Acción**: es la especificación de una acción, en pseudocódigo o en el lenguaje de programación que se usa. En el caso de un acontecimiento diferido, debe aparecer la palabra clave *defer* como primera acción.

Nota terminológica

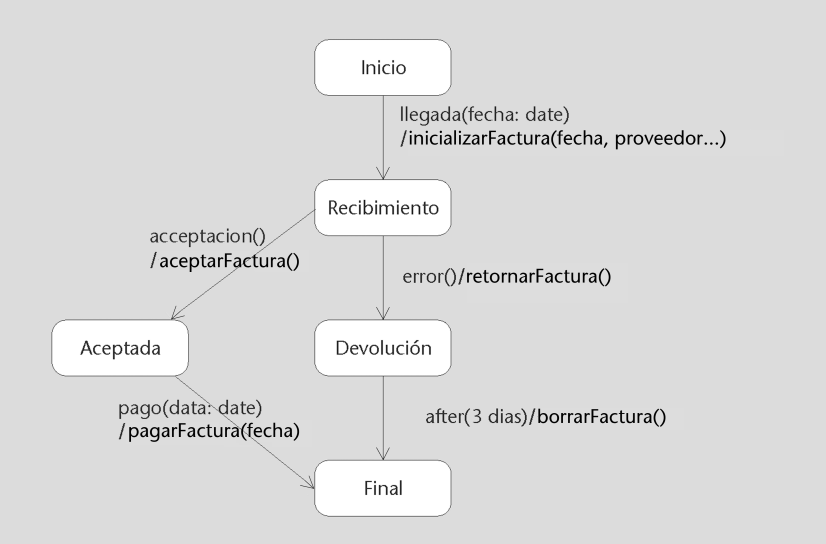
Respecto al uso de términos *parámetro* y *argumento* en relación con una llamada, hay que tener presente que *parámetro* corresponde al punto de vista de lo que se llama, y *argumento,* al punto de vista de lo que hace la llamada.

Ejemplos de diagramas de estado

En los ejemplos siguientes veremos diagramas de estados con transiciones, estados, acontecimientos, autotransición, etc.

1) Ejemplo de estados, transiciones y acontecimientos

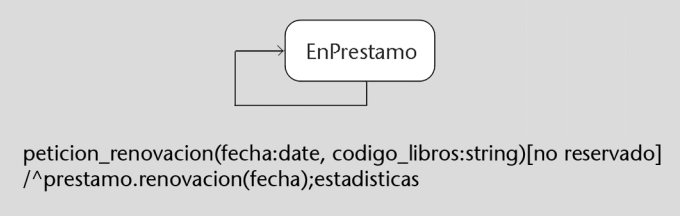
Este diagrama corresponde a una clase de facturas, cuyo nombre no aparece:



Todo empieza cuando llega una factura, hecho que describimos como el acontecimiento *llegada*, el cual tiene como parámetro la fecha; este acontecimiento tiene asociada la acción *inicializarFactura* y provoca la transición hacia el estado *Recibida*; *fecha*, y *proveedor* son parámetros de la acción. Después la factura es verificada (esto no aparece explícitamente), cosa que puede dar como resultado dos acontecimientos alternativos: *aceptación* –que ejecuta la acción *aceptarFactura* y provoca la transición hacia el estado *Aceptada*–, y error –que ejecuta la acción *devolverFactura* y provoca la transición al estado *Devuelta*. Si la factura está en el estado *Aceptada* y se produce el acontecimiento *pago*, la factura pasa al estado *Final* y se pide la ejecución de la acción *pagarFactura*. Si la factura está en estado *Devuelta*, la transición hacia el estado *Final* es provocada por un acontecimiento de tiempo consistente en el hecho de que hayan pasado tres días (como no se especifica desde cuándo, se entiende que es desde la entrada en el estado de origen), y entonces se pide la ejecución de la acción *borrarFactura*.

2) Ejemplo de autotransición, acción y guarda

El diagrama siguiente representa parte del diagrama de estados de los objetos de una clase de libros de una biblioteca pública:



Cuando un lector tiene un libro en préstamo, puede pedir renovaciones, que se le concederán si otro lector no ha reservado dicho libro. Los efectos del acontecimiento *peticion_renovacion* son la autotransición, la llamada a la acción *renovacion* y la ejecución de la acción *estadisticas*.

1.3. Transiciones complejas

Un objeto o interacción puede estar en más de un estado al mismo tiempo, y puede haber transiciones que salgan de más de uno o que vayan a parar a más de uno, o ambas cosas a la vez. Son las denomindas **transiciones complejas**.

Una transición compleja con varios estados de origen sólo tiene lugar si el objeto o interacción está en todos estos al mismo tiempo y, además, se produce el acontecimiento correspondiente a la transición (y se cumple la guarda, si la hay) y, por tanto, realiza una función de sincronización. Por analogía, cuando se produce una transición compleja con varios estados de destino, el objeto o interacción pasa a todos estos estados a la vez.

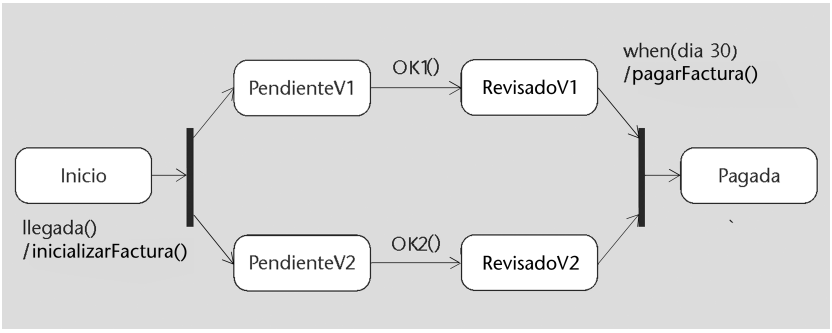
Una transición compleja se representa utilizando un pseudoestado intermedio, al que van a parar varias transiciones (pseudoestado de sincronización, *join pseudostate*) o del que salen varias (pseudoestado de bifurcación, *fork pseudostate*), o ambas circunstancias al mismo tiempo. Este tipo de pseudoestado se representa por medio de una barra vertical corta y gruesa que es origen y destino de transiciones en las que se descompone la transición compuesta.

Pseudoestado

Un pseudoestado es un símbolo que figura en una posición del diagrama donde normalmente habría un estado y no representa ningún concepto, sino que tiene una función puramente gráfica.

Ejemplo de transiciones complejas

Considerad las siguientes transiciones complejas:



Cuando llega una factura de una compra de material, pasa a estar pendiente de la verificación mediante la comparación con el pedido (V1) y con lo que se ha recibido (V2), por medio de una transición compleja de bifurcación; si de cada verificación resulta una conformidad (acontecimientos OK1 y OK2), se pasa al estado *Revisado1* y *Revisado2*, respectivamente. Cuando llega el día 30, se paga la factura sólo si estaba al mismo tiempo en estos dos estados, por medio de una transición compleja de sincronización.

1.4. Notación ampliada del estado

Igual que la clase, el estado tiene dos representaciones gráficas: una que sólo tiene el nombre, que es la que hemos utilizado hasta el momento, y otra con varios compartimentos. Los compartimentos son los siguientes:

- **Compartimento del nombre**, que contiene sólo el nombre del estado.

- **Compartimento de las transiciones internas y los acontecimientos internos**: contiene una lista de las cadenas correspondientes a los acontecimientos y pseudoacontecimientos correspondientes. En el caso de los pseudoacontecimientos *entry* y *exit*, no puede haber ni condiciones de guarda ni parámetros.

Ved también

En este punto, conviene que realicéis los ejercicios de autoevaluación del 1 al 5 y después, la actividad 1 de este módulo didáctico.

2. El diagrama de casos de uso

Los diagramas de casos de uso[3] sirven para mostrar las funciones de un sistema de *software* desde el punto de vista de sus interacciones con el exterior y sin entrar ni en la descripción detallada ni en la implementación de estas funciones.

(3)En inglés, *use case*.

Ved también

Podéis encontrar más información sobre los casos de uso y su especificación textual en el módulo "Requisitos".

El diagrama de casos de uso complementa, pero en ningún caso sustituye, la descripción textual de los casos de uso, puesto que no incluye información sobre cuál es el comportamiento del sistema. ¿Cuál es, pues, la utilidad de este diagrama? Por un lado, nos permite relacionar visualmente a los actores y los casos de uso y, por otro, nos da una visión rápida de cuál es la funcionalidad que el sistema ofrece a los diferentes actores.

2.1. Actores

La finalidad de un software[4] es proporcionar información a personas, máquinas y dispositivos o *softwares* del exterior, cuyo conjunto denominaremos *entitades exteriores*; también hay entidades exteriores –las mismas u otras– que piden funciones al *software* o le suministran información para que la trate.

(4)Finalidad compartida conla infraestructura de *hardware, software* básico y red sobre la cual se ejecuta.

Un **actor** es un conjunto de papeles de una entidad exterior en relación con el sistema de *software* considerado.

Por tanto, un actor no es la entidad exterior en sí, sino sólo los aspectos que tienen que ver con su interrelación con el sistema de *software*; podríamos decir que un actor es la visión que el *software* tiene de una entidad exterior. Desde este punto de vista, un actor es un conjunto de papeles, ya que se considera que el actor desempeña un papel diferente en cada interacción (cada caso de uso, en UML) que tiene con el *software*. Además, si una entidad exterior forma dos conjuntos de papeles con poca relación entre sí, le corresponderán dos actores diferentes (se podría decir que, en este caso, el *software* no puede saber si los dos actores son o no la misma entidad exterior).

A menudo hay discusiones sobre si una entidad exterior en particular es o no un actor en relación con un *software*; nosotros consideraremos que para ser actor, una entidad exterior tiene que cumplir estas dos condiciones:

- Ser autónoma con respeto al *software*, es decir, que la actividad en que utiliza la información suministrada por el *software* no esté subordinada a la de éste.

- Mantener relación directa con el *software* o con entidades exteriores que desempeñan tareas subordinadas a la actividad del *software*.

Ejemplos de actores

Algunos ejemplos nos ayudarán a aclarar el concepto de actor aplicando las reglas que acabamos de indicar.

Un motor eléctrico que es puesto en marcha o parado por un *software* de tiempo real es un actor, ya que su actividad es autónoma (su rotación sirve para accionar algún dispositivo mecánico), mientras que una impresora no tiene actividad autónoma porque se limita a imprimir la información que le envía un sistema de *software* (o varios) y, por tanto, su actividad está subordinada a la generación de listados por parte del *software*.

Si la persona A recoge un listado de la impresora y lo entrega a una persona B que hace un resumen manual de la información y lo da a otra persona C, sólo B será actor, ya que A no tiene una actividad autónoma y C no recibe la información directamente del *software* y, por tanto, el *software* no "conoce" su existencia. Naturalmente, si A tuviera alguna actividad autónoma que no tuviera nada que ver con el *software* –por ejemplo, barrer la oficina– no por ello sería actor del *software*.

Un actor es un clasificador (pero a pesar de esto, no es ningún estereotipo estándar del clasificador). Si hay varias entidades exteriores que desempeñan el mismo conjunto de papeles en relación con el *software*, un único actor las representa a todas, de igual manera que una clase representa todos sus objetos posibles (o, dicho de otra forma, estas entidades exteriores serían instancias del actor).

El ejemplo de los bibliotecarios (I)

Si en una biblioteca pública todos los bibliotecarios pueden utilizar las mismas funciones del *software* de apoyo a la gestión de la biblioteca, definiremos un único actor para todos; pero si hay un bibliotecario que puede realizar algunas funciones que los demás no pueden (como variar la duración de los préstamos), estas funciones se asignarían a otro actor. Hay dos maneras de hacerlo; leed, unas líneas más abajo, la segunda parte de este ejemplo.

Entre actores, se pueden establecer relaciones de especialización/generalización con herencia, igual que entre clases. Un actor A que es una especialización de otro B realiza como mínimo todos los papeles de B.

El ejemplo de los bibliotecarios (II)

En el ejemplo de los bibliotecarios, el hecho de que haya algunos bibliotecarios (digamos, los jefes de biblioteca) que pueden hacer que todas las funciones (casos de uso, recordémoslo) que desempeñan los demás, más otras propias, hace que puedan definir un actor *Bibliotecario* y otro *JefeDeBiblioteca*, y este último heredará de aquél. Pero esto mismo se podría expresar de otra forma: haciendo que el actor *JefeDeBiblioteca* sólo comprendiera los papeles que tienen los jefes de biblioteca y no tienen los demás bibliotecarios; en este caso, evidentemente, no habría herencia, y a los jefes de biblioteca corresponderían dos actores, *Bibliotecario* y *JefeDeBiblioteca*.

2.2. Concepto de caso de uso

Un caso de uso documenta una interacción entre el *software* y un actor o más. Dicha interacción tiene que ser, en principio, una función autónoma dentro del *software*.

Entre los actores que participan en un caso de uso, conviene distinguir el **actor primario** del caso de uso, que es el que lo pone en funcionamiento pidiendo la función correspondiente del *software*.

Caso especial

Si un proceso se tiene que poner en funcionamiento, automáticamente o no, en una fecha y/o hora determinada, se considera que hay un actor ficticio, por ejemplo "Reloj", que desempeña este papel.

Los casos de uso son un caso particular de los clasificadores; una instancia de un caso de uso es una ejecución de éste con intervención de casos particulares de los actores involucrados. Los casos de uso pueden tener atributos y operaciones que pueden servir para describir el proceso (que también se puede describir de otras maneras, como texto ordinario y diagramas de estados y de actividad).

Ved también

Se pueden hacer descripciones más formales y detalladas de los casos de uso durante el análisis, así como de su implementación "suprimir por medio de diagramas de comunicación o de secuencias". Consultad los subapartados 3.2 y 3.3 de este módulo didáctico.

Entre los casos de uso y los actores que intervienen se establecen asociaciones (lo cual no tiene nada de especial, ya que unos y otros son clasificadores); el significado de estas asociaciones es el papel del actor en relación con el caso de uso, pero no representan ni la dirección ni el contenido de un eventual flujo de datos entre el *software* y el actor.

2.3. Relaciones entre casos de uso

Entre los casos de uso se pueden establecer tres tipos de relación:

1) Relaciones de extensión: se dice que el caso de uso A extiende el B si dentro de B se ejecuta A cuando se cumple una condición determinada. A tiene que ser un caso de uso que también se pueda ejecutar de forma separada de B, y debe de tener el mismo actor primario que éste.

2) Relaciones de inclusión: un caso de uso A está incluido dentro de los casos de uso B, C, etc., si es una parte de proceso común a todos éstos. A no es un caso de uso autónomo, en el sentido de que no tendrá actor primario, sino que siempre será puesto en funcionamiento por uno u otro de los casos de uso que lo incluyen. No obstante, su implementación no puede depender de éstos[5]. Por tanto, la inclusión de casos de uso es esencialmente una forma de reutilización.

(5)Por ejemplo, no puede utilizar sus variables.

3) Relaciones de generalización/especialización: un caso de uso A es una especialización de otro caso de uso B si A realiza todo el proceso de B, más algún proceso específico.

Además, hay una forma de relación no tipificada (parecida a las agregaciones entre clases) entre casos de uso que consiste en definir, por ejemplo, un caso de uso que corresponde a todo el sistema, otros que corresponden a todos sus subsistemas, etc.

2.4. Notación

Tanto para los casos de uso como para los actores, no se utiliza el símbolo de los clasificadores, sino símbolos especiales como los siguientes:

- Los casos de uso se representan mediante elipses de trazo continuo. A veces se agrupan todas las elipses dentro de un rectángulo que representa todo el *software*.

- Los actores se representan mediante una figura humana esquemática.

- Las relaciones de especialización entre actores y entre casos de uso se representan mediante el mismo tipo de flecha que en el caso de clases.

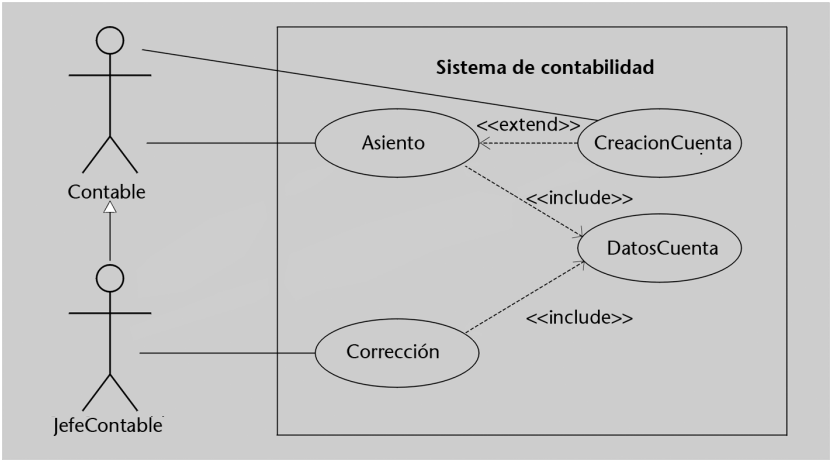
- Las relaciones de extensión y de inclusión entre casos de uso son estereotipos de la dependencia entre clasificadores, y se representan mediante las palabras clave *extend* e *include*, respectivamente.

Ved también

Consultad las relaciones de dependencia entre clasificadores en el subapartado 6.3 del módulo "UML(I): el modelo estático" de esta asignatura.

Ejemplo de casos de uso y actores con diferentes relaciones

Consideremos el ejemplo siguiente:



Los usuarios correspondientes al actor *Contable* sólo pueden intervenir en los casos de uso *Asiento* y *CreacionCuenta*, mientras que el actor *JefeContable* puede hacer también el caso de uso *Correccion*. Por tanto, *JefeContable* es una especialización de *Contable*. El caso de uso *CreacionCuenta* extiende el caso de uso *Asiento* porque cuando se intenta hacer un asiento con una cuenta inexistente es necesario crear esta cuenta. El caso de uso *DatosCuenta* representa el acceso a los datos de una cuenta desde dentro de los casos de uso *Asiento* y *Correccion*; a diferencia de *CreacionCuenta*, no es un caso de uso que pueda ser llevado a cabo de manera directa e independiente por parte un actor y, por tanto, la relación con los casos que utilizan es de inclusión y no de extensión.

Ved también

Ahora conviene que realicéis los ejercicios de autoavaluación 6 y 7 y la actividad 2.

3. Los diagramas de interacción

La ejecución de un *software* orientado a objetos consiste en un encadenamiento de operaciones y de cambios de estado de objetos, el cual, a su vez, consiste en que durante la ejecución de una operación o durante una transición se llaman operaciones sobre otros objetos (o sobre el mismo) y se envían señales que provocan otras transiciones. Así, se puede describir el funcionamiento de los casos de uso y de operaciones complejas. En UML, esta acción se lleva a cabo mediante los denominados **diagramas de interacción**.

3.1. Interacciones y colaboraciones

Para empezar, tenemos que definir los conceptos de *interacción* y *colaboración*.

3.1.1. Interacciones

Una interacción es la especificación del comportamiento de un caso de uso u operación en términos de secuencias de intercambio de mensajes entre objetos (o, más exactamente, entre instancias de clasificadores). Estos mensajes contienen estímulos, que pueden ser peticiones de ejecución de operaciones o señales.

Un hilo de ejecución[6] es una secuencia de mensajes tal que el primero contiene un estímulo que provoca el envío del segundo, etc. Si el mensaje A ha sido la causa de que se emitiera el mensaje B, se dice que A es el predecesor de B y B, el sucesor de A. Que un mensaje tenga varios sucesores o predecesores quiere decir que hay una bifurcación o confluencia de hilos, respectivamente.

(6)En inglés, *thread*.

3.1.2. Colaboraciones

De la misma forma que para que puedan circular mensajes entre ordenadores es necesario que los ordenadores estén unidos por enlaces de comunicaciones, también debe haber una cierta "infraestructura" para la circulación de mensajes entre objetos (a diferencia del caso de las redes, aquí no se trata de una necesidad física, sino de una necesidad de coherencia entre los diferentes diagramas de UML que describen un *software*); esta infraestrutura está formada por las clases o clasificadores y las asociaciones entre éstas, definidas en el modelo estático, así como las asociaciones entre actores y objetos que se obtienen cuando se describen casos de uso mediante interacciones.

De acuerdo con esto, para cada interacción se tiene que indicar qué parte del modelo estático utiliza. Si un objeto de la clase A tiene que pedir una operación de un objeto de la clase B, es necesario que estén estas dos clases y que B tenga definida la operación que se pide. Sin embargo, debe haber también una asociación entre estas dos clases y que esté especificada la posibilidad de navegar a través de ella por lo menos desde la clase A hacia la B.

Observación

Conviene no confundir estos papeles con los que desempeñan los clasificadores con respecto a una asociación que los relaciona.

Por tanto, en una colaboración sobre la cual debe tener lugar una interacción determinada, es necesario que figuren todos los clasificadores y asociaciones entre éstos que se utilizarán en la interacción. Sin embargo, de la misma manera que en un caso de uso de las entidades exteriores no nos interesan todos los aspectos, sino sólo el papel que ejecutan en relación con aquel caso de uso, también en una colaboración, más que clases u objetos, lo que se tendrá en cuenta serán sus papeles respectivos en relación con dicha interacción, y lo mismo es válido para las asociaciones.

Resumiendo, una **colaboración** es un conjunto de papeles de clasificadores o instancias y de papeles de asociaciones entre aquellos que intervienen en una interacción. Una colaboración se puede definir por lo que respecta a clasificadores o por lo que respecta a instancias.

La sintaxis de los nombres de los papeles de clasificadores es la que presentamos a continuación:

‘/' nombre_papel ‘:' nombre_clasificador

UML ofrece dos diagramas para representar una interacción y la colaboración en que se basa: el diagrama de comunicación, que pone énfasis en la descripción de la comunicación/colaboración entre las clases, y el de secuencias, que pone énfasis en la sucesión temporal de los mensajes de la interacción. Hay una tercera manera de describir las interacciones, el diagrama de actividades, que tiene otra orientación.

Ved también

Consultad el diagrama de actividades en el apartado 4 de este módulo didáctico.

3.2. Diagrama de comunicación

El diagrama de comunicación es la representación de una interacción mediante un diagrama estático de la colaboración correspondiente sobre la cual se representan los mensajes de la interacción.

Para cada mensaje hay una especificación con la siguiente sintaxis:

[número_de_secuencia [guarda]:] nombre_operación(lista_parámetros) [: regreso]

A continuación, daremos una explicación de cada elemento del mensaje:

- *número_de_secuencia* es un número que identifica el mensaje dentro de la secuencia de mensajes del diagrama. El mensaje que pone en marcha toda la interacción es el 1. Este campo es opcional aunque es muy recomendable para poder visualizar un orden cronológico de los mensajes.

- *guarda* es una condición booleana que se tiene que cumplir para que se produzca la comunicación. Es opcional.

- *nombre_operación (lista_parámetros)* es el nombre de la operación que se llama junto con la lista de parámetros a enviar en la llamada entre paréntesis.

- *devolución* es el resultado obtenido como respuesta a la llamada. Es opcional.

Tipos de mensajes

A continuación, veremos los diferentes tipos de mensajes:

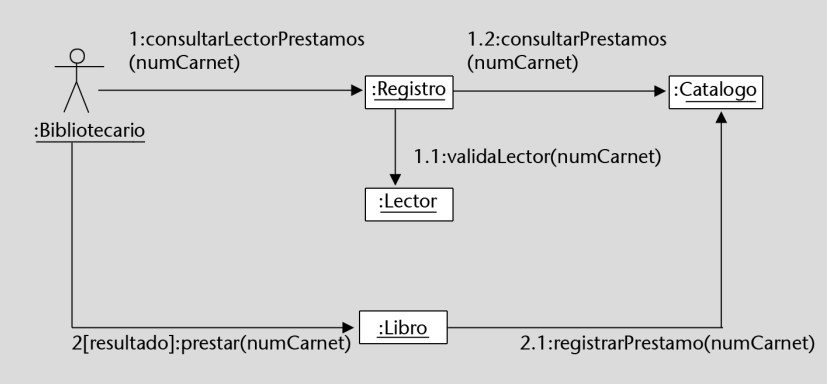
a) Mensajes asíncronos: la comunicación asíncrona se produce cuando la clase emisora envía un mensaje al suministrador y se continúa ejecutando sin esperar a que llegue el resultado; la clase receptora, por su parte, no ejecuta la operación immediatamente, sino que deja la petición en una cola. Se representa mediante una flecha con la punta abierta y cortada horizontalmente.

b) Mensajes síncronos: sólo se dan cuando el cliente envía un mensaje al suministrador y éste acepta el mensaje. La clase emisora ejecuta el código hasta que envía el mensaje y, después, se espera a recibir el resultado de la operación asociada al mensaje. Se representan mediante una flecha con la punta coloreada que indica el sentido.

Ejemplo de diagrama de comunicación

El siguiente diagrama de comunicación muestra el proceso seguido cuando un bibliotecario pide información al sistema sobre si un lector (identificado por su número de carné)

puede coger en préstamo un libro, y a continuación, si el resultado de la confirmación es positivo, registra el préstamo en el sistema.



3.3. El diagrama de secuencias

A diferencia del diagrama de comunicación, en el diagrama de secuencias no se representan explícitamente los papeles de asociaciones (quedan implícitos en los mensajes) y se representa explícitamente el orden en el tiempo, e incluso la duración, de los mensajes y de las operaciones que ponen en marcha.

El diagrama de secuencias está estructurado según dos dimensiones. El tiempo se representa verticalmente y corre hacia abajo, y no está representado a escala necesariamente. En dirección horizontal, hay franjas verticales sucesivas que corresponden a los diferentes papeles de clasificadores que participan en la interacción; cada papel de clasificador está representado por el símbolo habitual, que encabeza su línea de vida. El orden de los clasificadores de izquierda a derecha no es significativo, aunque la tendencia debe ser que los mensajes circulen de izquierda a derecha y los resultados y respuestas, de derecha a izquierda.

La **línea de vida** simboliza la existencia del papel en un cierto periodo de tiempo. Se representa mediante una línea discontinua vertical que va desde la creación del objeto hasta su destrucción.

Si la destrucción no está prevista en el diagrama, entonces la línea de vida irá hasta la parte inferior del diagrama, es decir, hasta más allá del último mensaje que se muestra. Si se quiere indicar la destrucción del objeto, entonces el final de la línea de vida irá marcado con una X. Puede ser que durante una parte de la línea de vida haya dos activaciones del objeto a la vez, que no serán concurrentes sino alternativas según una condición que determina que se emita el el mensaje que pone en marcha una u otra.

Una **activación** es una parte de la línea de vida durante la cual dicho papel ejecuta una acción, u otros papeles ejecutan otras acciones, como consecuencia de una acción ejecutada por el papel.

Las activaciones sirven para modelar relaciones de control entre clases y se representan mediante rectángulos alargados verticalmente insertados en las líneas de vida. El inicio y el final del rectángulo coinciden con la llegada del mensaje que pone en marcha la acción y el envío del mensaje de respuesta, respectivamente. Pueden tener etiquetas, que especifican la acción que corresponde al mensaje. Para especificar llamadas recurrentes del mismo papel, se representará una nueva activación desplazada un poco más a la derecha.

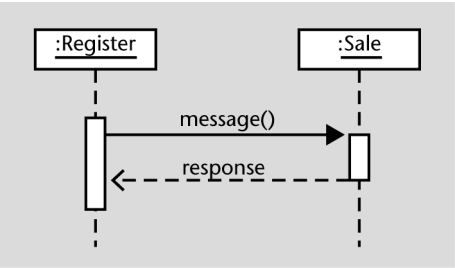
Los mensajes se indican con flechas iguales a las del diagrama de colaboración, que comienzan en una activación (al principio de ésta o por el medio) y acaban en otra. Su orden de arriba abajo expresa el orden en que se producen en el tiempo. Además, una flecha inclinada (hacia abajo, necesariamente) indica un mensaje de duración no ignorado, o de otro modo la flecha sería horizontal.

La especificación de los mensajes es muy parecida a la que hemos visto en el diagrama de comunicación. Las principales diferencias son las siguientes:

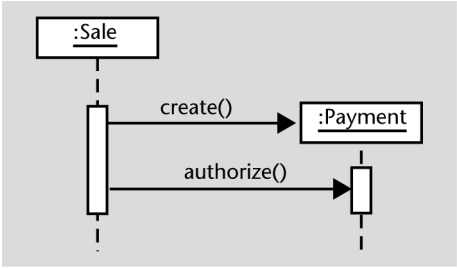
a) No se indican los números de secuencia, ya que quedan implícitos en el orden temporal de los mensajes y de las activaciones que provocan.

b) Hay tres tipologías específicas de mensajes:

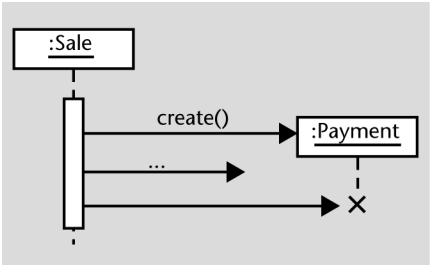
- Mensaje de respuesta: corresponde al mensaje de respuesta a una llamada. Opcionalmente se puede indicar cuál es la respuesta. Se representa con una flecha con la punta abierta y la línea del cuerpo discontinua.



- Mensaje de creación: corresponde a un mensaje para definir una nueva instancia o línea de vida. Se representa igual que un mensaje síncrono pero el destino es la caja de la instancia y no la línea de vida o zona de activación.



- Mensaje de destrucción: corresponde a un mensaje que se envía para mostrar de manera explícita la destrucción de una instancia. Se representa igual que un mensaje de creación pero con estereotipo <<destroy>>. Este mensaje siempre va acompañado del símbolo X al final de la línea de vida de la instancia destruida.

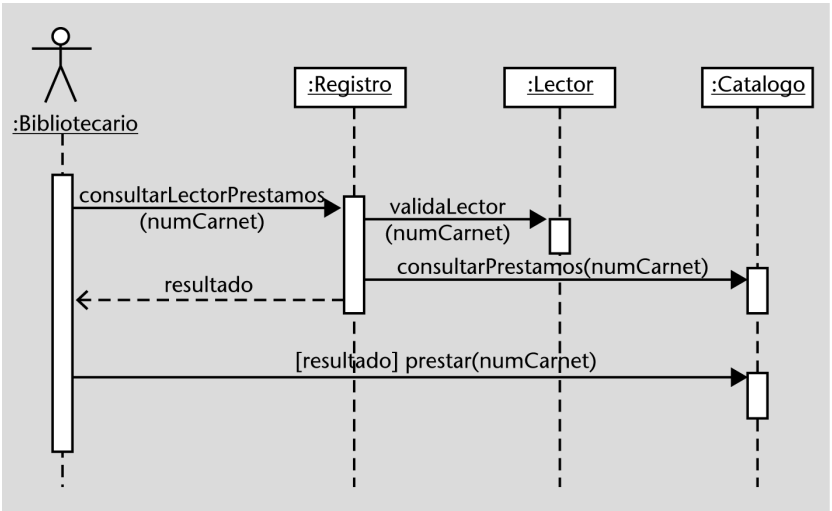


A diferencia del diagrama de colaboración, se pueden indicar los mensajes de retorno al final de una activación, en forma de flechas de línea discontinua y punta abierta.

Ejemplo de diagrama de secuencias

Diagrama de secuencias equivalente a diagrama de comunicación

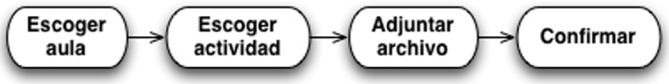
Este ejemplo corresponde al ejemplo de diagrama de comunicación del final del subapartado 3.2:



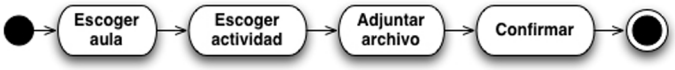
4. El diagrama de actividades

El diagrama de actividades combina diferentes ideas del mundo de la modelización de procesos y, por lo tanto, nos puede ayudar a documentar el comportamiento del sistema si lo vemos como un proceso. La modelización de actividades enfatiza más la secuencia y la coordinación de las posibles acciones por documentar que quién es el responsable de ejecutarlas.

El elemento básico de este diagrama es la actividad, que representa el hecho de que alguien hace a alguien. El otro elemento básico son las transiciones o flujos, que representan el hecho de que se pasa de una actividad a la siguiente. A continuación, tenemos un diagrama de actividades muy sencillo para el caso de uso "Entregar una actividad".



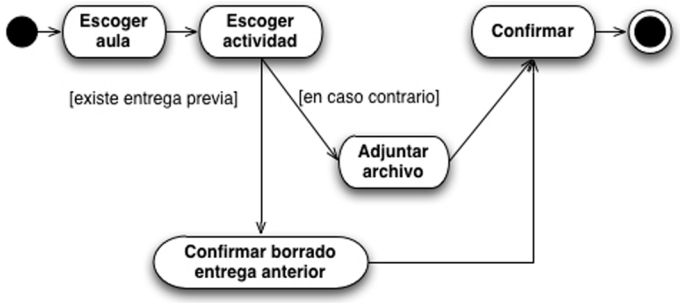
En este caso, cada paso del caso de uso lo hemos representado como una actividad y las diferentes transiciones nos dan una idea sobre cuál es la secuencia en la que tienen lugar las diferentes actividades. Se suele facilitar la lectura de la secuencia, indicando el inicio y el final:



En este ejemplo podemos ver fácilmente que el proceso empieza por "Elegir aula", tal como indica el punto negro, y el final se produce después de "Confirmar", tal como indica la transición hacia el punto negro dentro de un círculo. A pesar de que, en este caso, sólo hay uno, podemos poner en nuestro diagrama tantos símbolos de final como queramos.

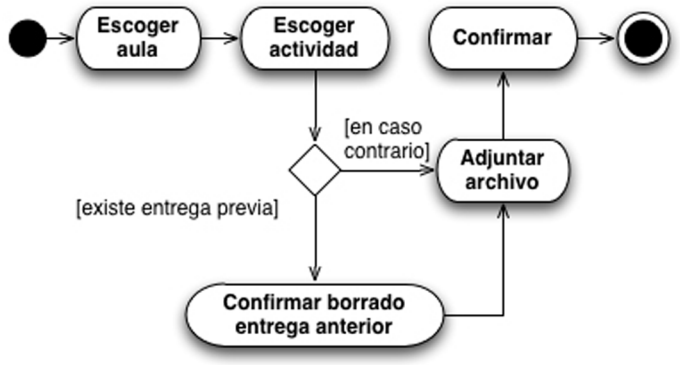
4.1. Lógica condicional

Una de las ventajas del diagrama de actividades es que podemos representar la lógica condicional que nos lleva a la ejecución de los diferentes escenarios. Así pues, continuando con el caso anterior, podríamos hacer el diagrama siguiente:



En este diagrama hemos introducido un nuevo elemento de notación, la condición, también llamada *condición de guarda*, que se representa mediante el texto entre corchetes en la transición. La condición *existe entrega previa* nos indica que esta transición sólo se produce cuando la condición es cierta. Por lo tanto, "Elegir actividad" tiene dos transiciones posibles: una se produce cuando hay una entrega previa y la otra cuando no es así.

Si queremos hacer más explícito el punto en el que hay una toma de decisión, lo podemos representar mediante el elemento gráfico de decisión, que se representa mediante un rombo:



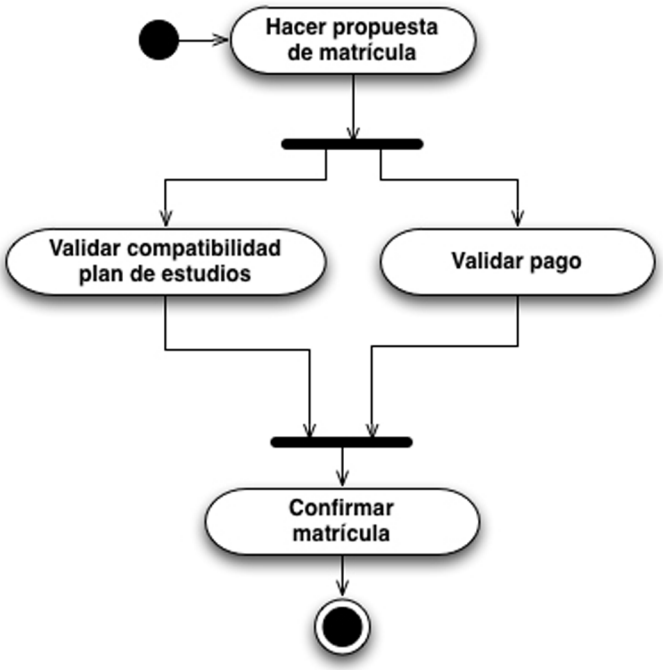
La decisión tiene un flujo de entrada y varios flujos de salida, cada uno con una condición. Sólo uno de los flujos de salida (aquel para el que la condición se cumpla) será el que se tome en un escenario concreto. Por ello, las diferentes condiciones deberían ser mutuamente excluyentes.

Otro elemento condicional que podemos utilizar es la fusión (*Merge*), que se representa como un rombo (igual que la decisión) pero que tiene varios flujos de entrada y sólo uno de salida, que se tomará cuando se llegue por alguno de los flujos de entrada:

Como hemos visto, las decisiones y fusiones son elementos opcionales que nos permiten enfatizar la lógica condicional. Habitualmente sólo usaremos, con este propósito, las decisiones.

4.2. Paralelización

Con los elementos de lógica condicional, los diferentes caminos son excluyentes: o bien se toma uno o bien se toma el otro, pero siempre hay un único camino (y, por lo tanto, una única actividad) ejecutándose en un momento concreto. A veces, no obstante, nos puede interesar tener la posibilidad de indicar que dos caminos se ejecutan de manera paralela:



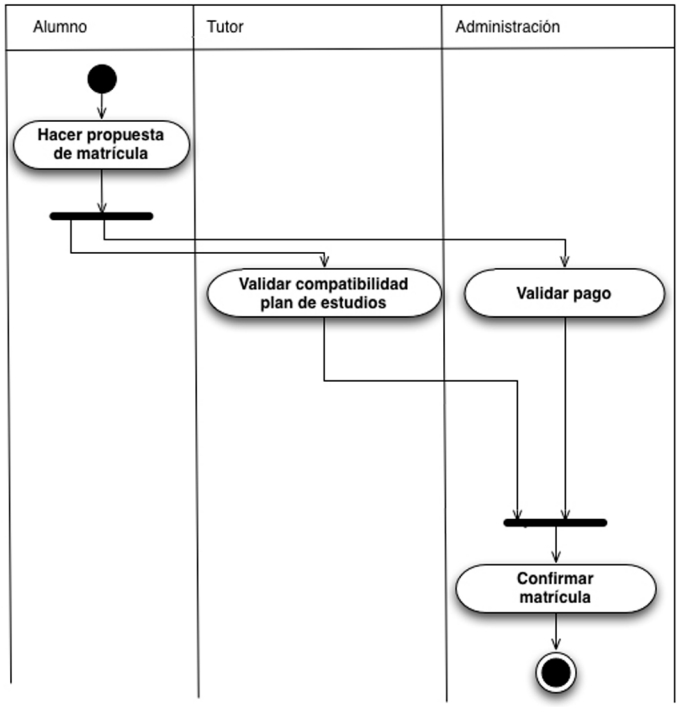
En este ejemplo, una vez se ha hecho la propuesta de matrícula, se inician dos flujos paralelos: uno que comprueba la compatibilidad con el plan de estudios y otro que comprueba que el pago se haga correctamente. Los dos se ejecutan en paralelo, puesto que no hay que esperar a que uno acabe para empezar el otro. La primera línea gruesa del diagrama es un ejemplo de bifurcación (*fork*); cada bifurcación tiene un flujo de entrada y varios flujos de salida.

El elemento contrario a la bifurcación es la unión (*join*). Este elemento (la otra barra gruesa) tiene varios flujos de entrada y uno de salida, que, a diferencia de la fusión, no se ejecuta hasta que no se han ejecutado todos los flujos de entrada.

Así pues, en nuestro ejemplo, no se confirma la matrícula hasta que no se ha validado la compatibilidad con el plan de estudios y se ha validado el pago.

4.3. Organización: carriles

Finalmente, hay un elemento de notación que nos puede ayudar a organizar las actividades. Los carriles (*swimlanes*) se usan, típicamente, para indicar qué actividades hace cada uno de los actores del proceso (o, en nuestro caso, del caso de uso) que estamos documentando. Se indican, gráficamente, de la manera siguiente:



En este caso, vemos que el alumno es el responsable de hacer la propuesta de matrícula, el tutor de validar la compatibilidad con el plan de estudios y el departamento de administración de validar el pago y confirmar la matrícula.

4.4. Otros elementos de notación

El diagrama de actividades soporta otros elementos de notación (como, por ejemplo, las señales o las subactividades) que no veremos en este módulo, puesto que sirven para modelizar comportamientos más complejos que los que encontramos típicamente en un caso de uso.

Resumen

En este módulo hemos visto el resto de diagramas que considera el estándar UML. Dichos diagramas son los siguientes:

- diagrama de casos de uso
- diagrama de estados
- diagramas de interacción: diagrama de secuencias y diagrama de comunicación
- diagrama de actividad

Hemos visto detalladamente los conceptos y el uso de todos éstos.

También hemos tratado las conexiones que se establecen entre estos diagramas, así como entre éstos y el diagrama estático. Así, por ejemplo, los clasificadores figuran en el diagrama estático y en el de interacción. Los estados, en el diagrama de estados y en el de actividad; los actores, en el diagrama de casos de uso y en los de interacción que describen la implementación de los casos de uso, etc.

Actividades

1. Supongamos que el acceso a la universidad desde el bachillerato se desarrolla según el procedimiento siguiente (no se pretende que sea fiel a la realidad):

Cuando un alumno ha aprobado el bachillerato, se preinscribe en ocho opciones de estudios universitarios y se puede presentar a las pruebas de acceso; si no las aprueba, se tiene que volver a examinar al año siguiente, pero conserva la preinscripción. Si las aprueba, entra en el proceso de asignación de plazas y, dependiendo de su nota, se le asignará plaza en la 1.ª opción, en una opción de la 2.ª a la 8.ª, o en ninguna.

Si se le ha asignado plaza de la 1.ª opción, entra en el proceso de matrícula de julio; si se le ha asignado plaza de una opción de la 2.ª a la 8.ª, entra en el proceso de matrícula de septiembre, y si no se le ha asignado plaza, queda en la situación de no admitido.

Los procesos de matrícula de julio y de septiembre tienen unos plazos fijados para llevar a cabo el proceso administrativo. Un alumno que se tenga que matricular en julio, y el día 1 de agosto no lo haya hecho, pasa a la situación de no admitido, igual que un alumno que se tenga que matricular en septiembre y el 1 de octubre tampoco lo haya hecho.

Confeccionad el diagrama de estados del alumno.

2. Confeccionad un diagrama de casos de uso para esta situación:

En un hotel, el encargado de las reservas recibe peticiones de reserva de los clientes por teléfono, momento en el que hace una consulta de las habitaciones disponibles los días comprendidos en la reserva. Si hay alguna disponible, la reserva y se lo comunica al cliente; si un cliente indica que llegará después de las 8 de la tarde, se pondrá una marca en la reserva. Cuando llega un cliente, un recepcionista comprueba la reserva del cliente o le hace la reserva después, si no la tenía, y en los dos casos introduce en la reserva la marca de la llegada del cliente. Cuando un cliente se va, se introduce la fecha del día. A las ocho de la tarde se pone en marcha un proceso que borra todas las reservas que no lleven la marca que indica que el cliente llegará más tarde.

3. Haced los diagramas de comunicación y de secuencia correspondientes al enunciado de la actividad 2.

4. El profesor de un grupo quiere preparar el inicio de curso. Primero escoge el grupo al cual quiere preparar el inicio de curso y, después, tiene que escribir un mensaje de bienvenida en el tablero y, si su asignatura tiene fórum, organizar las carpetas del fórum. El mensaje de bienvenida en el tablero y la organización de las carpetas se pueden hacer en cualquier orden, y no se puede dar el caso de uso por finalizado hasta que el profesor no haya hecho las dos cosas (a menos que el grupo no tenga fórum, en cuyo caso solo hay que escribir el mensaje de bienvenida).

Ejercicios de autoevaluación

1. ¿Cómo se representan el punto de entrada y el de salida de un diagrama de estados?

2. ¿Qué diferencia hay entre una transición interna y una autotransición?

3. ¿Qué diferencia hay entre una condición de guarda relativa a un acontecimiento y la condición ligada a un acontecimiento de cambio?

4. ¿De qué tipo son los acontecimientos del ejemplo de estados, transiciones y acontecimiento (excepto el acontecimiento de tiempo)?

5. ¿Puede haber más de una transición con el mismo estado de origen y el mismo acontecimiento?

6. ¿Es correcta la afirmación "los actores son clases"?

7. En una agencia de viajes, un cliente pide a un empleado que averigüe con el ordenador si hay plazas en un vuelo determinado. ¿Qué actor o actores participan?

8. ¿Qué relación existe entre *interacción* y *colaboración*?

Ved también

Consultad el "Ejemplo de diferentes diagramas de estados" en el subapartado 1.2. de este módulo didáctico.

9. En un diagrama de comunicación, dos mensajes tienen, respectivamente, los números 1.1 y 1.2. ¿Qué significa esto? ¿Cómo se representaría la misma información en el diagrama de secuencia de la misma interacción?

10. ¿Qué diferencia existe entre *línea de vida* y *activación*? ¿En qué casos coinciden?

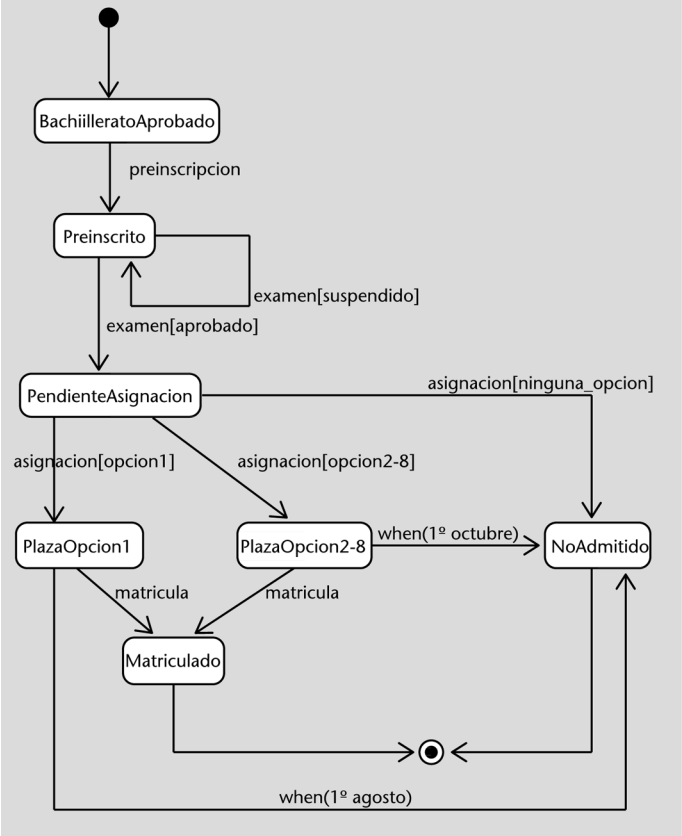
11. ¿Qué diferencia hay entre un estado y un estado de acción?

12. ¿Cómo podemos indicar la lógica condicional en un diagrama de actividades?

Solucionario

Actividades

1.



Hay otras soluciones.

Por razones de claridad del diagrama se duplican algunos estados; los estados que tienen el nombre seguido de un asterisco son duplicados del que tiene el mismo nombre sin asterisco.

2. Hay tres actores, *EncargadoReservas*, *Recepcionista* y *Reloj*. Los casos de uso son *Reserva* (del primer actor), *Llegada* y *Partida* (del segundo) y *Cancelacion*; *Reserva* se extiende a *Llegada*.

3. Hay muchas soluciones posibles. Como orientaciones generales, podríamos decir que probablemente sea necesaria una clase *Habitacion* y otra *Reserva*, con una asociación de uno a varios entre éstas, y que el proceso de una reserva precisaría por lo menos dos operaciones diferentes sobre la clase *Habitacion*: una consulta sobre un multiobjeto que serían las habitaciones disponibles en una fecha determinada, y otra que crearía una reserva de aquella habitación.

Ejercicios de autoevaluación

1. La entrada y salida en un diagrama de estados se representan mediante un par de símbolos: un círculo completamente negro para indicar el inicio del diagrama, y un círculo completamente negro rodeado de una circunferencia para indicar su salida.

2. En una autotransición, figura que hay cambio de estado y, por tanto, se ejecutan las eventuales acciones de entrada y salida. En cambio, en una transición interna no se cambia de estado y, por lo tanto, aquellas acciones no se ejecutan.

3. Una condición de guarda se evalúa una vez se ha producido un acontecimiento para comprobar si debe tener lugar una determinada transición. Va ligada a ésta más que al acontecimiento, y se evalúa después de que éste se ha producido.

Un acontecimiento de cambio se produce cuando la condición ligada a éste pasa de no cumplirse a cumplirse y, por tanto, la evaluación de dicha condición es anterior al acontecimiento.

4. Son acontecimientos de llamada, porque llaman operaciones o señales.

5. Sí. Por una parte, puede haber varias transiciones ligadas a guardas diferentes (por ejemplo, una y su contraria) y, por la otra, puede haber una transición compleja con varios estados de destino.

6. La afirmación no es correcta. Los actores son clasificadores al igual que las clases, pero no son clases.

7. Hay un único actor que es el empleado, porque es el único que tiene acceso directo al *software*.

8. Una colaboración es un conjunto de papeles de clasificadores conectados por asociaciones. Una interacción es un intercambio de mensajes entre instancias que tiene lugar dentro de una colaboración siguiendo estas asociaciones.

9. Los dos mensajes son consecuencia del mensaje 1 y se emiten primero uno y después el otro. En el diagrama de secuencias, saldrían los dos de la activación puesta en marcha por el mensaje 1, primero 1.1 y después (es decir, más abajo) 1.2.

10. La línea de vida representa toda la duración de un papel de un clasificador, desde que se crea hasta que se destruye, mientras que una activación representa el tiempo en el que se están ejecutando acciones relativas a dicho papel durante una interacción.

La activación coincidiría con la línea de vida si el papel fuera creado y destruido durante la interacción y, en medio, se ejecutasen sólo acciones relativas al papel en cuestión.

11. A un estado se llega por medio de una transición provocada por un acontecimiento, a un estado de acción se llega cuando acaba la acción asociada a éste.

12. Lo podemos hacer mediante la condición de guarda y, opcionalmente, una decisión (que se representa mediante un rombo).

Glosario

acción Proceso que o no se ejecuta o se ejecuta hasta el final.

acontecimiento Hecho que se produce en un instante en el tiempo y que puede provocar una transición.

activación Parte de la línea de vida de un papel de clasificador durante la cual se ejecutan acciones ligadas a éste.

actor Conjunto de papeles que desempeña una entidad exterior en relación con un *software*. Cada papel corresponde a un caso de uso.

caso de uso Interacción entre el *software* y un actor o más que comporta una o más acciones.

estado Situación durante la vida de un objeto o la duración de una interacción en la que cumple alguna condición, lleva a cabo alguna actividad o espera algún acontecimiento.

estado de acción Estado que corresponde al hecho de que se encuentre en curso una acción determinada.

estímulo Petición de una operación o comunicación de una señal que llega a un objeto.

interacción Especificación del funcionamiento de una operación o caso de uso en términos de secuencias de mensajes entre instancias de clasificadores que piden operaciones o envían señales.

línea de vida Intervalo de tiempo durante el cual existe un papel.

señal Estímulo entre dos instancias de clasificadores que puede provocar un acontecimiento.

transición compuesta Transición con varios estados de origen y/o de destino.

transición interna Recurso para especificar que cuando se produce un determinado acontecimiento que tiene lugar mientras que el objeto está en un cierto estado, se tienen que ejecutar determinadas acciones sin que se dé cambio de estado.

transición simple Paso de un estado de origen a otro de destino provocado por un acontecimiento. Puede poner en funcionamiento acciones.

Bibliografía

Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley.

Esta obra es una muy buena introducción corta a la UML. Tiene un enfoque pragmático más que normativo y es muy aclaratoria. Muestra el uso de todos los diagramas, con ejemplos, y da consejos personales del autor sobre cómo y cuándo hay que utilizar cada parte de la especificación. Pero evidentemente no entra en detalles.

Larman, C. (2005). *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development*. Prentice Hall.

Larman cubre, en esta obra, el análisis y el diseño de sistemas informáticos usando el lenguaje UML. Da un enfoque ágil y otro de UP, y el libro está escrito de manera iterativa e incremental. Cubre todo lo que se explica en este módulo y bastante más, ya que entra en detalles de diseño e implementación.

Bibliografía complementaria

Booch, G.; Rumbaugh, J.; Jacobson, I. (2005). *The unified modeling language user guide*. Addison-Wesley.

En esta obra los autores de UML escriben una guía de uso del lenguaje de carácter didáctico, más que de referencia. La obra se centra en el lenguaje UML, pero nos da consejos y sugerencias sobre cuándo hay que usar una parte del estándar u otra.

Varios autores (2010). *Unified Modeling Language™ (UML®)*. Object Management Group.

Esta es la especificación de UML. A pesar de su lenguaje formal y normativo, es, evidentemente, la fuente más fiable en caso de duda sobre el lenguaje UML.