4. DIAGRAMAS DE INTERACCIÓN

4.1. INTRODUCCIÓN

Todos los sistemas tienen una estructura estática y un comportamiento dinámico, y el UML proporciona diagramas para capturar y describir ambos aspectos. Los diagramas de clases se usan para documentar y expresar la estructura estática de un sistema, es decir, las clases y sus relaciones. Los diagramas de estado y los diagramas de interacción describen el comportamiento de un sistema, para demostrar cómo los objetos interactúan dinámicamente en diferentes momentos durante la ejecución del sistema.

Los objetos dentro de un sistema se comunican unos con otros, enviándose mensajes. Un mensaje es justo una operación donde un objeto llama a otro objeto. Así pues la dinámica de un sistema se refiere a cómo los objetos dentro del sistema cambian de estado durante el ciclo de vida del mismo y también a cómo dichos objetos colaboran a través de la comunicación. En el primer caso se utilizan los diagramas de estado y los diagramas de actividad. En el segundo caso, la comunicación entre los objetos se representa mediante los diagramas de interacción, que a su vez agrupan a dos tipos de diagramas: secuencia y colaboración.

4.2. DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia muestran la interacción de un conjunto de objetos a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes. Es decir, nos proporciona la interacción entre los objetos, que se sucede en el tiempo, para un escenario específico durante la ejecución del sistema (por ejemplo, cuando se utiliza un requisito funcional concreto).

Gráficamente, un diagrama de secuencia es una tabla con dos ejes: el eje horizontal muestra el conjunto de objetos y el eje vertical muestra el conjunto de mensajes, ordenados en el tiempo.

4.2.1. Objetos

Los objetos se representan en el eje horizontal, y cada uno de ellos mediante un rectángulo que contiene el nombre y la clase del objeto en el siguiente formato: nombre del objeto:nombre de la clase.

En los diagramas de secuencia hay dos características que los distinguen de los diagramas de colaboración: línea de vida y activación de un objeto.

- Línea de vida

La línea de vida de un objeto representa la existencia de un objeto durante un cierto período de tiempo. Se dibuja mediante una línea vertical discontinua desde el rectángulo que contiene al objeto hasta la parte final del diagrama.

- Activación

La activación de un objeto muestra el período de tiempo en el cual un objeto se encuentra desarrollando alguna acción. Un objeto activado está bien ejecutando su propio código o bien esperando la vuelta de otro objeto al cual ha enviado previamente un mensaje. Se dibuja como un rectángulo delgado sobre la línea de vida del objeto. La parte superior del rectángulo está alineada con el comienzo de la acción y la inferior alineada con su término (y puede estar marcada por un mensaje de vuelta).

Nota:
1. En un diagrama de colaboración no podemos mostrar de forma explícita la línea de la vida de un objeto, pero sí los mensajes *«create»* y *«destroy»*.
2. Tampoco podemos mostrar la activación de un objeto, aunque el número de secuencia de cada mensaje nos puede indicar si hay o no anidamiento.

4.2.2. Mensajes

Los mensajes representan la comunicación entre los objetos y se dibujan como líneas horizontales continuas, dirigidas desde el objeto que envía el mensaje hasta el objeto que lo ejecuta. La flecha especifica si el mensaje es simple, síncrono o asíncrono, tal como podemos ver en la Figura 4.1.

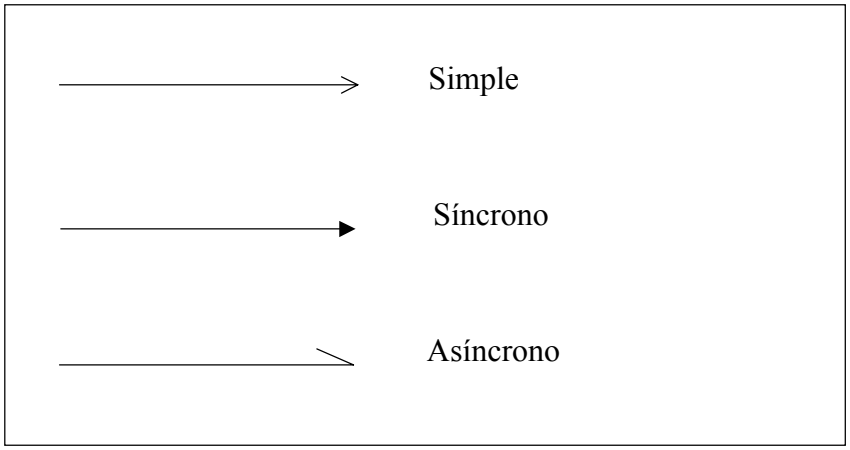

Figura 4.1

En UML se distinguen los siguientes tipos de mensajes:

- Simple: Representa un flujo de control simple. Muestra cómo el control se pasa de un objeto a otro sin describir ningún detalle sobre la comunicación. Este tipo de mensajes se utiliza cuando los detalles sobre la comunicación no son conocidos o no se consideran relevantes en el diagrama. También se usa para mostrar la vuelta de un mensaje síncrono.
- Síncrono: Representa un flujo de control anidado, implementado como una llamada a una operación. La operación que soporta el mensaje se termina (incluyendo otros

mensajes anidados) antes de que el objeto que envió el mensaje continúe con su ejecución. La vuelta se puede mostrar como un mensaje simple.

- Asíncrono: Representa un flujo de control asíncrono. No hay vuelta explícita al objeto que envió el mensaje, el cual continúa ejecutándose después de enviar el mensaje sin esperar ninguna respuesta. Este tipo de mensajes se utiliza en los sistemas de tiempo real donde los objetos se ejecutan concurrentemente.

Los mensajes tienen un nombre, el cual puede aparecer o no acompañado de parámetros. También pueden tener condiciones, expresadas entre corchetes, que se usan para modelar ramas o decidir si se envía o no un mensaje. Si las condiciones describen ramas, se pueden dar dos posibilidades. Una alternativa es que las condiciones sean excluyentes y entonces sólo se ejecute un mensaje cada vez. La otra es que no se excluyan, con lo cual los mensajes son enviados concurrentemente.

En el ejemplo de la Figura 4.2 podemos observar los componentes descritos anteriormente para un diagrama de secuencia.

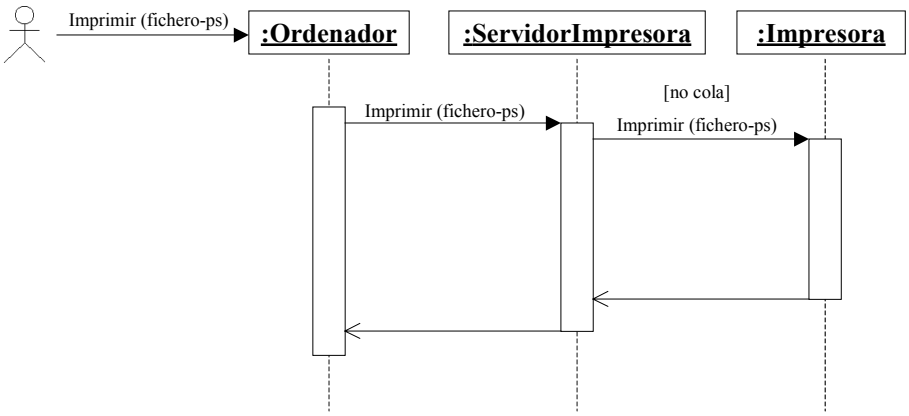

Figura 4.2

4.2.3. Creación y destrucción de un objeto

Los diagramas de secuencia pueden mostrar cómo los objetos son creados o destruidos como parte del escenario bien documentado. El mensaje que crea o destruye un objeto es normalmente un mensaje síncrono.

Un objeto puede crear otro objeto por medio de un mensaje, estereotipado como *«create»*. El objeto creado se dibuja con el símbolo que representa al objeto situándolo donde se crea (en el eje vertical). Asimismo se puede eliminar un objeto vía un mensaje, estereotipado como *«destroy»*. Cuando se destruye un mensaje, se marca con una X larga y además la línea de vida del objeto sólo se dibuja hasta el punto en el que se ha eliminado.

En el ejemplo de la Figura 4.3 podemos ver tanto la creación como la destrucción de un objeto de la clase BilleteAgente, por lo que a este tipo de objeto se le cataloga como transitorio. También se puede observar que una instancia de la clase BilleteAgente se envía un mensaje a sí mismo, llamando a la operación HallarRuta.

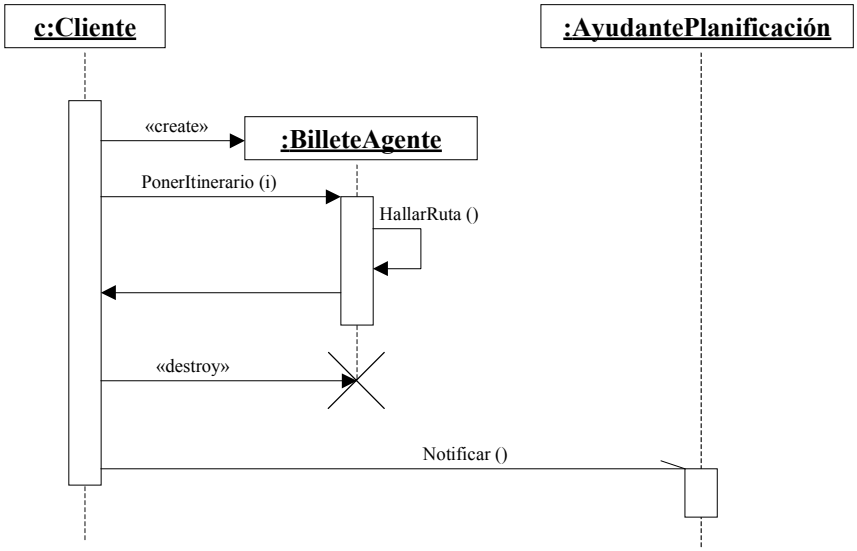

Figura 4.3

El tipo de mensaje más común es la llamada, en la cual un objeto solicita una operación de otro objeto (o del mismo). Si un objeto, tal como c de la clase Cliente, llama a la operación PonerItinerario en una instancia de la clase BilleteAgente, entonces dicha operación no sólo tiene que estar definida en la clase BilleteAgente (es decir, tiene que estar declarada en la clase BilleteAgente o en uno de sus padres), sino que también debe ser visible al objeto c que le ha llamado. Cuando el objeto de la clase BilleteAgente devuelve el control al objeto c, se puede modelar como un mensaje síncrono o simple y de forma optativa incluir el valor devuelto.

4.3. DIAGRAMAS DE COLABORACIÓN

Los diagramas de colaboración muestran la interacción de un conjunto de objetos en el espacio. Esta descripción se centra en la organización estructural de los objetos que envían y reciben mensajes.

Gráficamente, un diagrama de colaboración es un grafo formado por conjunto de vértices, los objetos, y de arcos, los enlaces, que conectan dichos vértices.

4.3.1. Objetos

Un objeto se representa mediante un rectángulo que contiene el nombre y la clase del objeto en el siguiente formato: nombre del objeto:nombre de la clase.

4.3.2. Enlaces

Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una línea continua que une a dos objetos.

En los diagramas de colaboración hay dos características que los distinguen de los diagramas de secuencia: path y número de secuencia, que pueden acompañar a un enlace.

- Path

El path indica qué tipo de objeto recibe el mensaje. Hay varios estereotipos que indican el papel del objeto en el enlace, es decir, indican si el objeto que recibe el mensaje es un objeto local (*«local»*), global (*«global»*), un parámetro de un mensaje anterior (*«parameter»*), un objeto que se envía mensajes a sí mismo (*«self»*). Si es un atributo (*«association»*), no se indica pues se asume por defecto.

- Número de secuencia

El número de secuencia indica el orden de un mensaje dentro de la interacción. Se fija el mensaje con el número 1 y se incrementa en una unidad por cada nuevo mensaje en el flujo de control. Pueden darse varios niveles de subíndices para indicar anidamiento de operaciones (1 es el primer mensaje; 1.1 es el primer mensaje anidado en el mensaje 1; y así sucesivamente). En un mismo enlace se pueden mostrar muchos mensajes, pero cada uno de ellos con un número de secuencia único.

Nota:

1. En un diagrama de secuencia no podemos mostrar de forma explícita los enlaces entre los objetos.
2. Tampoco podemos mostrar el número de secuencia de un mensaje, aunque está implícito el orden físico de los mensajes desde la parte superior hasta la inferior en el diagrama de secuencia.
3. Tanto en el diagrama de secuencia como en el de colaboración se pueden mostrar la iteración y la ramificación. Sin embargo, en la práctica la ramificación simple sólo se puede mostrar en el diagrama de secuencia y la compleja en el diagrama de colaboración.

4.3.3. Flujo de mensajes

El flujo de mensajes, también llamado flujo de control, expresa el envío de los mensajes. Se representa mediante una flecha dirigida cercana a un enlace. Dependiendo del tipo de flecha se especifica si el mensaje es síncrono, asíncrono o simple.

El diagrama de colaboración de la Figura 4.4 se corresponde con el diagrama de secuencia de la Figura 4.2. Un actor envía el mensaje Imprimir al objeto de la clase Ordenador para imprimir un fichero postscript. Dicho objeto envía el mismo mensaje al objeto de la clase ServidorImpresora, que a su vez envía el mensaje Imprimir a la Impresora, con la condición de que no haya ficheros en la cola para ser imprimidos. Es decir, la operación Imprimir tiene un nivel de anidamiento, observando el número de secuencia 1.1, y está sujeta a la condición de que la impresora esté libre, la cual viene expresada entre corchetes como [no cola].

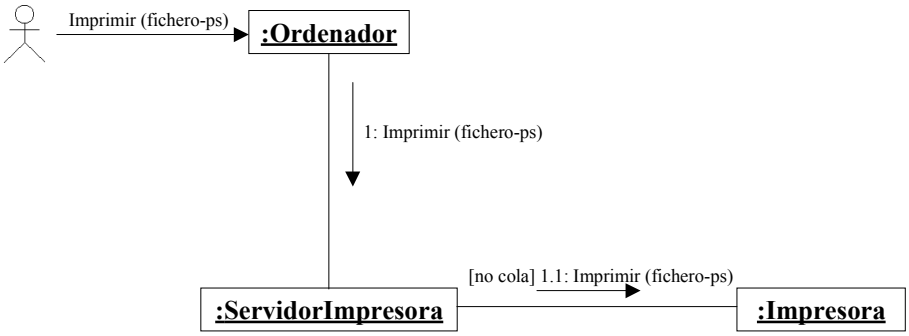

Figura 4.4

La mayoría de las veces el flujo de control es secuencial. Sin embargo, se pueden modelar flujos más complejos, donde esté implicada una iteración o una ramificación.

La Figura 4.5 nos muestra un diagrama de colaboración con un flujo de control no secuencial, en este caso, iterativo.

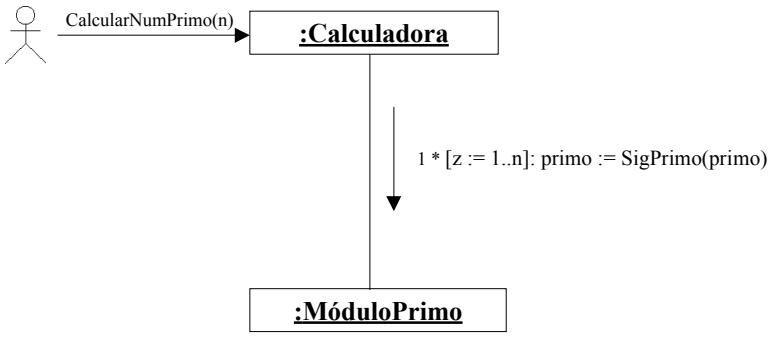

Figura 4.5

Una iteración representa una secuencia repetida de mensajes. Para modelar una iteración, se puede prefijar el número de secuencia de un mensaje con una expresión de iteración, tal como * [i := 1..n] (o simplemente * para no especificar más detalles). De forma similar, una condición representa un mensaje cuya ejecución depende de la evaluación de una expresión booleana. Para modelar una condición, se prefija el número de secuencia de un mensaje con una cláusula de condición, tal como [x > 0]. La otra alternativa de una ramificación tiene el mismo número de secuencia, pero las dos cláusulas deben ser excluyentes. Las expresiones correspondientes tanto a la iteración como a la ramificación deben ir después del número de secuencia (y antes de :).

Para la iteración y la ramificación, el UML no obliga que el formato de la expresión encerrada entre corchetes sea uno concreto; se puede utilizar el pseudocódigo o la sintaxis de un lenguaje de programación específico.

4.3.4. Creación y destrucción de un objeto

Normalmente, los objetos que participan en una interacción existen durante toda la interacción. Pero, en otras interacciones, los objetos se pueden crear y destruir. Para especificar qué objetos son creados y destruidos en un diagrama de colaboración, se puede añadir una de las siguientes restricciones (debe ir encerrada entre llaves) cerca del rectángulo del objeto:

- new: indica que se crea un nuevo objeto.
- destroyed: indica que se elimina un objeto ya existente.
- transient: indica que se crea un objeto durante la ejecución de la interacción, pero se destruye antes de que se termine la misma.

El diagrama de colaboración de la Figura 4.6 se corresponde con el diagrama de secuencia de la Figura 4.3.

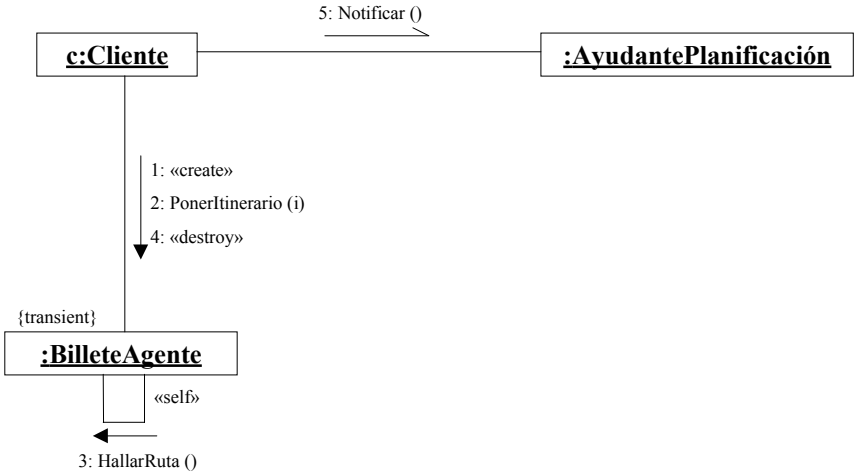

Figura 4.6

El objeto de la clase BilleteAgente es de tipo transitorio, ya que relacionados con él aparecen los mensajes *«create»* y *«destroy»*. Cuando un objeto se envía un mensaje a sí mismo, como es el caso del objeto de la clase BilleteAgente, el papel de dicho objeto en el enlace viene estereotipado como *«self»*.