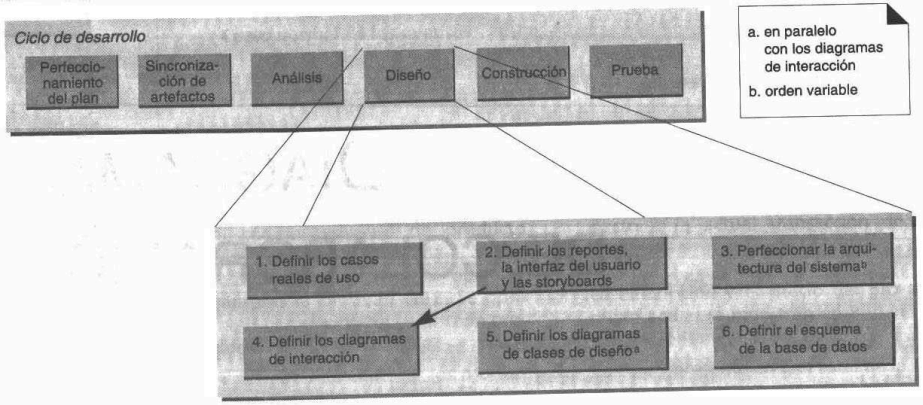
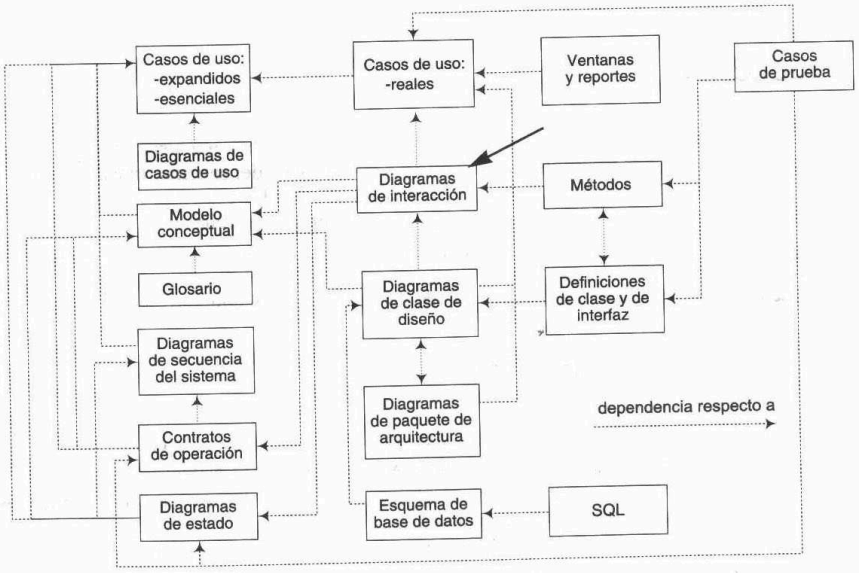


Actividades de la fase de diseño dentro de un ciclo de desarrollo.



Dependencias de los artefactos durante la fase de construcción.

- Casos de uso reales (o esenciales): a partir de ellos el diseñador recaba información sobre las tareas que realizan los diagramas de interacción, además de lo estipulado en los contratos.

17.3 Diagramas de interacción

Un **diagrama de interacción** explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de éstas. El punto de partida de las interacciones es el cumplimiento de las poscondiciones de los contratos de operación.

El UML define dos tipos de estos diagramas; ambos sirven para expresar interacciones semejantes o idénticas de mensaje:

1. diagramas de colaboración
2. diagramas de secuencia

Los **diagramas de colaboración** describen las interacciones entre los objetos en un formato de grafo o red, como se aprecia en la figura 17.1.

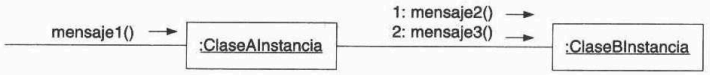


Figura 17.1 Diagrama de colaboración.

Los **diagramas de secuencia** describen las interacciones en una especie de formato de cerca o muro, como se observa en la figura 17.2.

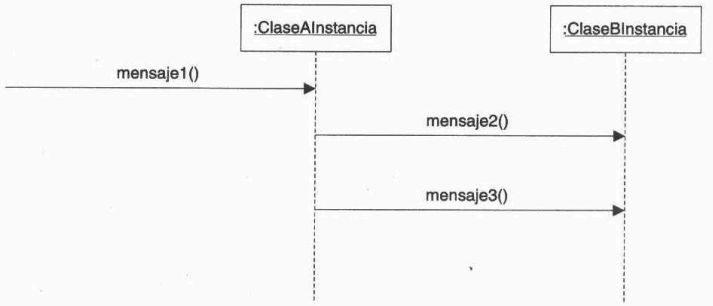


Figura 17.2 Diagrama de secuencia.

En este capítulo y a lo largo del libro nos centraremos en los diagramas de colaboración por su excepcional expresividad, su capacidad de comunicar más información

contextual y su economía de espacio.[1] Pero recuerde el lector que tanto una como otra notación pueden expresar conceptos parecidos.

17.4 Ejemplo de un diagrama de colaboración: efectuarPago

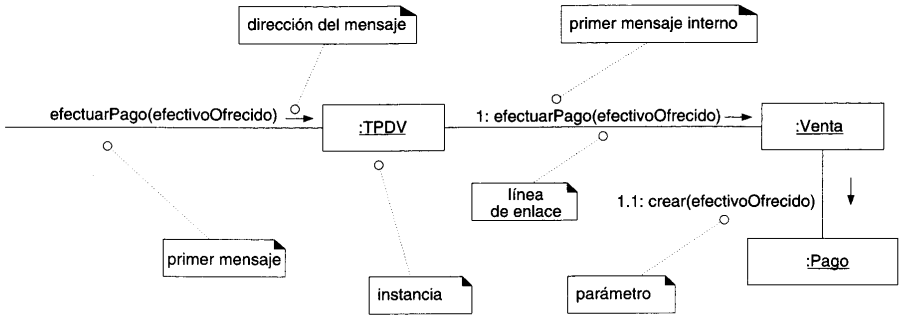


Figura 17.3 Diagrama de colaboración.

El diagrama de colaboración de la figura 17.3 se lee así:

1. El mensaje *efectuarPago* se envía a una instancia de *TPDV*. La instancia corresponde al mensaje *efectuarPago* de la operación del sistema.
2. El objeto *TPDV* envía el mensaje *efectuarPago* a la instancia *Venta*.
3. El objeto *Venta* crea una instancia de un *Pago*.

17.5 Los diagramas de interacción son un artefacto de gran utilidad

Un problema frecuente en los proyectos de la tecnología de objetos consiste en no comprender la utilidad de preparar los diagramas de interacción, la consideración cuidadosa de la asignación de responsabilidades y las habilidades que todo esto requiere. Son muy importantes la asignación de responsabilidades y el diseño de la colaboración entre los objetos.

Un porcentaje considerable de las actividades del proyecto ha de destinarse a esa fase. Más aún, es fundamentalmente durante ella cuando se exige aplicar las habilidades del diseño en los patrones, las expresiones y los principios.

[1] Los diagramas de colaboración nos permiten decir más en un espacio que los diagramas de secuencia y expresar además más información contextual; por ejemplo, el tipo de visibilidad entre los objetos. También resulta más fácil expresar la lógica condicional y la concurrencia.

En cierto modo, la creación de los casos de uso, de los modelos conceptuales y de otros artefactos resulta más fácil que asignar responsabilidades y elaborar diagramas bien diseñados de interacción. Ello se debe a lo siguiente: los sutiles principios del diseño en que se fundan son muchos más numerosos que cualquier análisis orientado a objetos o que cualquier artefacto de diseño.

> Los diagramas de interacción constituyen uno de los artefactos más importantes que se generan en el análisis y en el diseño orientados a objetos.
>
> El tiempo y el esfuerzo dedicados a su preparación deberían absorber un porcentaje considerable de la actividad total destinada al proyecto.
>
> Para mejorar la calidad de su diseño, es posible aplicar patrones, principios y expresiones codificados.

Los principios del diseño necesarios para construir eficazmente los diagramas de interacción *pueden* codificarse, explicarse y aplicarse de modo metódico. Esta forma de entender y utilizar los principios del diseño se basa en **patrones**: directrices y principios estructurados. Por tanto, luego de exponer la sintaxis de los diagramas de interacción, nos ocuparemos (en capítulos posteriores) de los patrones de diseño y de su aplicación a los diagramas de interacción.

17.6 Éste es un capítulo dedicado exclusivamente a la notación

En este capítulo nos proponemos describir y resumir la notación del UML para un tipo particular de diagrama de interacción: el diagrama de colaboración. No explicaremos los principios ni las directrices que rigen la elaboración de un diagrama de colaboración bien diseñado.

No es indispensable que el lector conozca todos los detalles de la notación que se ofrecen en este capítulo para proseguir el estudio del libro. No obstante, le recomendamos echar un vistazo a los ejemplos y familiarizarse un poco con ellos antes de pasar a los siguientes capítulos.

17.7 Lea las directrices de diseño en los siguientes capítulos

Hay varios principios de diseño que deben conocerse para poder crear buenos diagramas de interacción. Tras familiarizarse un poco con la notación de éstos, conviene leer los siguientes capítulos dedicados a esos principios y a la forma de aplicarlos.

17.8 Cómo preparar diagramas de colaboración

Aplique las siguientes normas cuando elabore un diagrama de colaboración.

Para preparar un diagrama de colaboración:

1. Elabore un diagrama por cada operación del sistema durante el ciclo actual de desarrollo.
   - En cada mensaje del sistema, dibuje un diagrama incluyéndolo como mensaje inicial.
2. Si el diagrama se torna complejo (por ejemplo, si no cabe holgadamente en una hoja de papel de 8.5 x 11), divídalo en diagramas más pequeños.
3. Diseñe un sistema de objetos interactivos que realicen las tareas, usando como punto de partida las responsabilidades del contrato de operación, las poscondiciones y la descripción de casos de uso. Aplique el GRASP y otros patrones para desarrollar un buen diseño.

17.8.1 Los diagramas de colaboración y otros artefactos

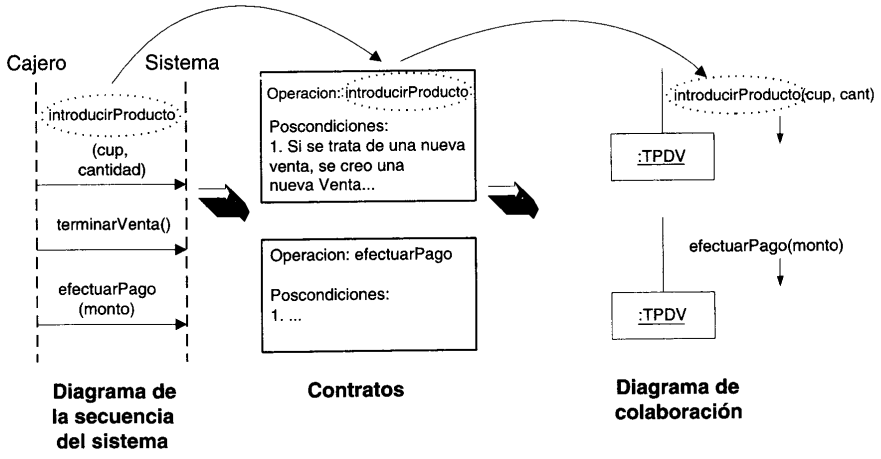


Figura 17.4 Relación entre los artefactos.

Como se observa en la figura 17.4, la relación entre los artefactos incluye lo siguiente:

- Los casos de uso indican los eventos del sistema que se muestran explícitamente en los diagramas de su secuencia.
- En los contratos se describe la mejor conjetura inicial sobre las operaciones del sistema.
- Las operaciones del sistema representan mensajes y éstos originan diagramas que explican gráficamente cómo los objetos interactúan para llevar a cabo las funciones requeridas.

17.9 Notación básica de los diagramas de colaboración

17.9.1 Representación gráfica de las clases y de las instancias

El UML ha adoptado un método simple y uniforme de describir visualmente las instancias para distinguirlas de los tipos (figura 17.5):

- Con cada tipo de elemento del UML (clase, actor, ...), una instancia utiliza el mismo símbolo gráfico usado para representar el tipo, pero se subraya el texto.

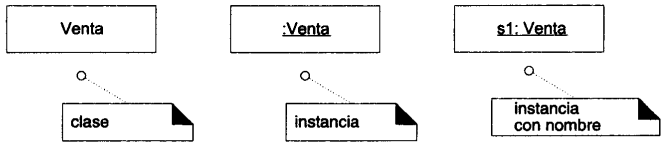


Figura 17.5 Clase e instancias.

Por tanto, para incluir la instancia de una clase en un diagrama de interacción, se recurre al símbolo gráfico usual de la casilla de la clase, sólo que el nombre se subraya. Además, en un diagrama de colaboración, al nombre de la clase siempre se le anteponen dos puntos.

Finalmente, un nombre de instancia sirve para identificarla de modo inequívoco.

17.9.2 Representación gráfica de los vínculos

El **vínculo** (o enlace) es una trayectoria de conexión entre dos instancias; indica alguna forma de navegación y visibilidad que es posible entre las instancias (figura 17.6). En un lenguaje más formal, el vínculo es una instancia de una asociación. Si vemos dos instancias en una relación de cliente/servidor, una trayectoria de navegación del cliente al servidor significa que los mensajes pueden enviarse del primero al segundo. Así, existe un vínculo —o trayectoria de navegación— entre *TPDV* y una *Venta*, a lo largo del cual pueden fluir los mensajes; por ejemplo, el mensaje *agregarPago*.

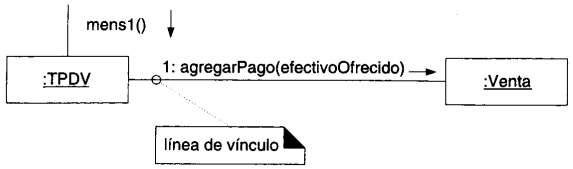


Figura 17.6 Líneas de vínculo (enlace).

17.9.3 Representación gráfica de los mensajes

Los mensajes entre objetos pueden representarse por medio de una flecha con un nombre y situada sobre una línea del vínculo. A través de éste puede fluir un número indefinido de mensajes (figura 17.7). Se agrega un número de secuencia que indique el orden consecutivo de los mensajes en la serie actual de control.

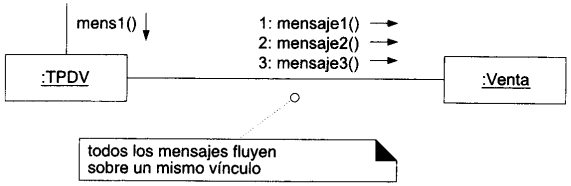


Figura 17.7 Mensajes.

17.9.4 Representación gráfica de los parámetros

Los parámetros de un mensaje pueden anotarse dentro de paréntesis después del nombre del mensaje (figura 17.8). Es opcional incluir o no el tipo de parámetro en cuestión.

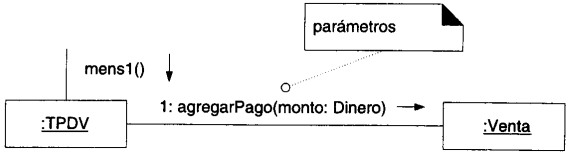


Figura 17.8 Parámetros.

17.9.5 Representación gráfica del mensaje de devolver valor

Puede incluirse un valor de retorno anteponiéndole al mensaje un nombre de variable de esa instrucción y un operador de asignación (':=') (figura 17.9). Es opcional mostrar el tipo del valor de retorno.

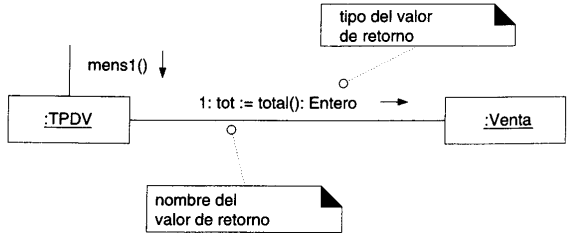


Figura 17.9 Devolver valores.

17.9.6 Sintaxis de los mensajes

El lenguaje UML cuenta con una sintaxis estándar para los mensajes:

```
retorno := mensaje(parametro : tipoParametro) : tipoRetorno
```

No obstante, como se aprecia en la figura 17.10, es legal servirse de otra sintaxis como la de Java o la de Smalltalk. Recomendamos emplear la sintaxis estándar de UML a fin de que los diagramas de colaboración sigan siendo un lenguaje relativamente independiente.

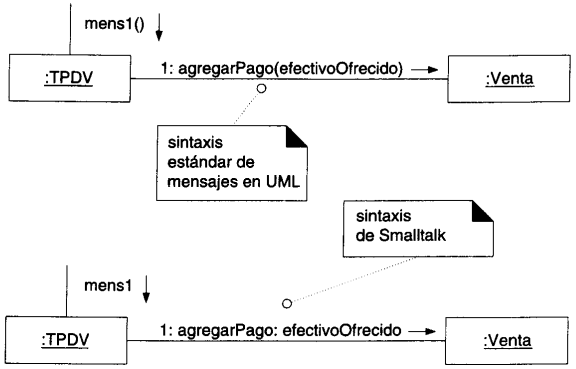


Figura 17.10 Los mensajes pueden expresarse en varias sintaxis.

17.9.7 Representación gráfica de los mensajes al "emisor" o a "esto"

Puede enviarse un mensaje de un objeto a sí mismo (figura 17.11).

Esto lo muestra gráficamente un vínculo consigo mismo, donde el mensaje fluye a lo largo del vínculo.

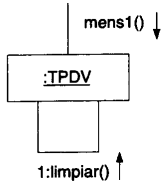


Figura 17.11 Mensajes a "esto".

17.9.8 Representación gráfica de la iteración

La iteración se indica posponiendo un asterisco (*) al número de secuencia.

Ese símbolo significa que, dentro de un ciclo, el mensaje va a ser enviado repetidamente al receptor (figura 17.12).

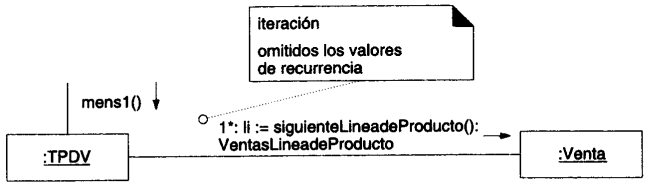


Figura 17.12 Iteración.

También es posible incluir una cláusula de iteración que indique los valores de recurrencia (figura 17.13).

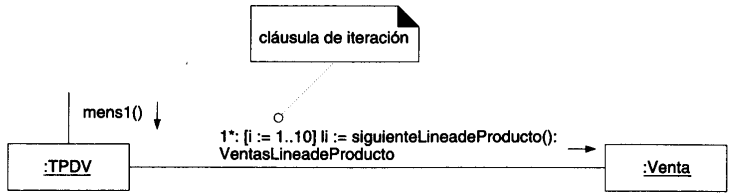


Figura 17.13 Cláusula de iteración.

Si se expresa más de un mensaje que ocurre dentro de la misma cláusula de iteración (por ejemplo, una serie de mensajes en un ciclo *for*), se repetirá la cláusula con cada mensaje (figura 17.14).

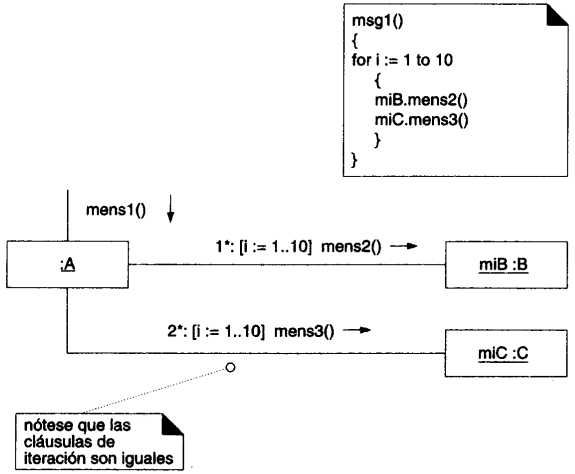


Figura 17.14 Mensajes múltiples dentro de la misma cláusula de iteración.

17.9.9 Representación gráfica de la creación de instancias

El mensaje de creación independiente del lenguaje es *crear*, que se muestra en el momento de ser enviado a la instancia que vamos a generar (figura 17.15).

Aunque en la mayoría de los lenguajes orientados a objetos, las instancias suelen generarse utilizando un mensaje *nuevo* (u operador) con la clase y no con una instancia, esta notación ocupa menos espacio aunque no sea completamente exacta.

Es opcional que la nueva instancia contenga o no un símbolo «*nuevo*».[1]

El mensaje *crear* puede contener parámetros, lo cual indica la transferencia de los valores iniciales. En Java, de ese modo se indican —por ejemplo— los parámetros de constructores.

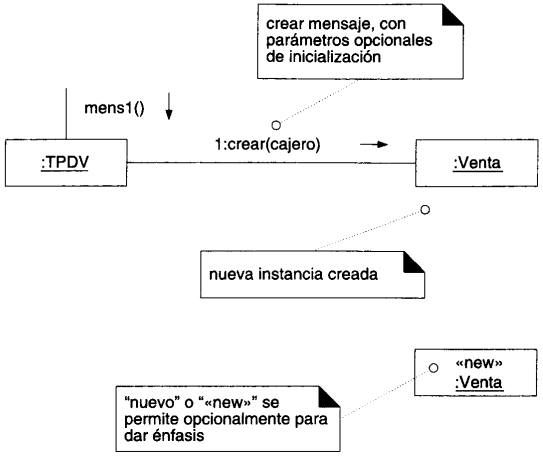


Figura 17.15 Creación de instancias.

En varios lenguajes, el mensaje *crear* se traduce así:

Lenguaje	Significado de crear ()
C++	Asignación automática u operador *new* seguido de una llamada a un constructor.
Java	operador *new* seguido de una llamada a un constructor.
Smalltalk	mensaje *new* o una variación de *new:* seguido del mensaje *initialize*.

[1] Un **estereotipo** del UML que estudiaremos más adelante.

17.9.10 Representación gráfica de la secuencia de número de los mensajes

El orden de los mensajes se indica con un **número de secuencia**, como se aprecia en la figura 17.16. El esquema de la numeración es:

1. El primer mensaje no se numera. Así, *mens1()* no lleva número.
2. El orden y el anidamiento de los mensajes siguientes se indican con un esquema legal de numeración, donde a los mensajes anidados se les ha antepuesto un número. La anidación se denota anteponiendo el número del mensaje de entrada al del mensaje de salida.

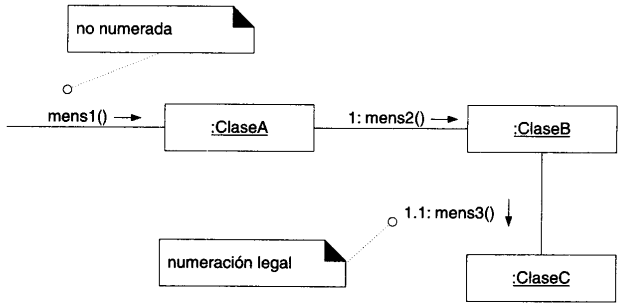


Figura 17.16 Numeración de secuencias.

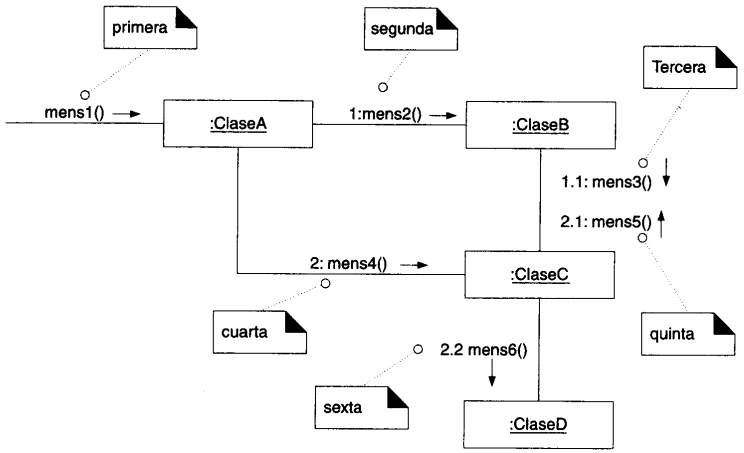


Figura 17.17 Numeración compleja de secuencias.

17.9.11 Representación gráfica de los mensajes condicionales

Un mensaje condicional (figura 17.18) se indica posponiendo al número de la secuencia una cláusula condicional entre corchetes, en forma parecida a como se hace con una cláusula de iteración. El mensaje se envía sólo si la cláusula se evalúa como *verdadera*.

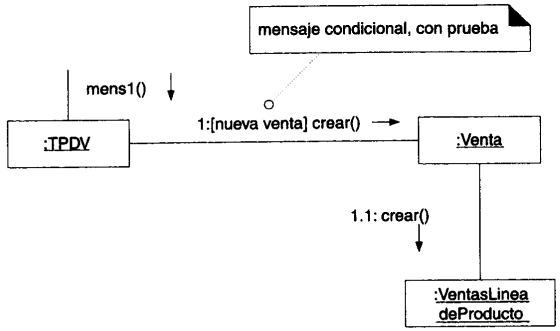


Figura 17.18 Mensaje condicional.

17.9.12 Representación gráfica de trayectorias condicionales mutuamente excluyentes

El ejemplo de la figura 17.19 contiene los números de secuencia con trayectorias condicionales que se excluyen mutuamente.

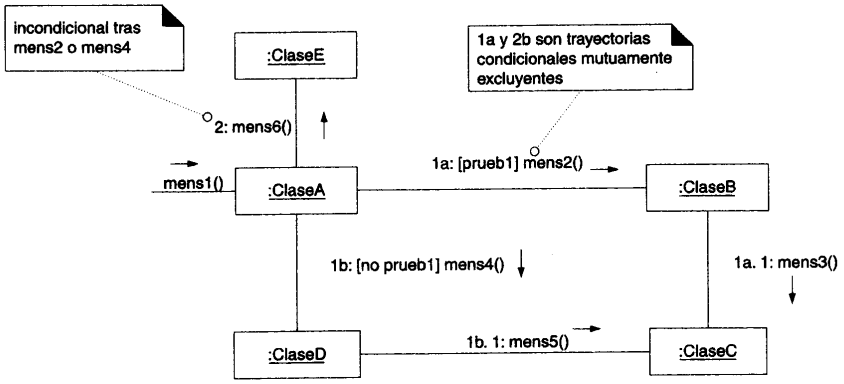


Figura 17.19 Mensajes mutuamente excluyentes.

En este caso es necesario modificar las expresiones de la secuencia con una letra de la trayectoria condicional. Por convención, la primera letra en usarse es **a**. La figura 17.19 establece qué tanto *1a* como *1b* podrían ejecutarse después de *mens1()*. Ambas son el número de la secuencia 1 porque pueden ser el primer mensaje interno.

Nótese que a los subsecuentes mensajes anidados todavía se les sigue anteponiendo la secuencia de sus mensajes externos. Así *1b.1* es un mensaje anidado dentro de *1b*.

17.9.13 Representación gráfica de las colecciones

Un **multiobjeto**, o conjunto de instancias, puede dibujarse como un icono de pila según se observa en la figura 17.20.

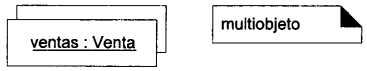


Figura 17.20 Un multiobjeto.

Un multiobjeto suele implementarse como un grupo de instancias guardadas en un contenedor u objeto colección; por ejemplo, un *vector* de la STL de C++, un *Vector* de Java o una *ColeccionOrdenada* (*OrderedCollection*) de Smalltalk. Pero no necesariamente se implementa así; representa tan sólo un conjunto lógico de instancias.

17.9.14 Representación gráfica de los mensajes dirigidos a multiobjetos

Un mensaje dirigido a un icono de multiobjeto indica que se envía al objeto colección. Así, en la figura 17.21 el mensaje *tamaño* está siendo enviado a la instancia *java.util.Vector* para solicitar el número de elementos del *Vector*.

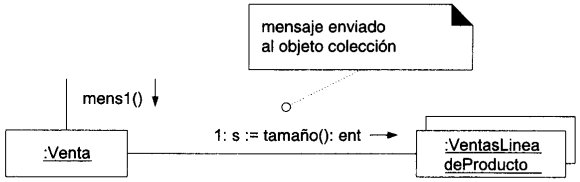


Figura 17.21 Mensaje dirigido a un multiobjeto.

En el lenguaje UML, los mensajes dirigidos a un multiobjeto *no* se transmite a todos los elementos (como sucedía en las versiones anteriores del UML).

En la figura 17.22 se explican gráficamente los mensajes a un multiobjeto y a un elemento.

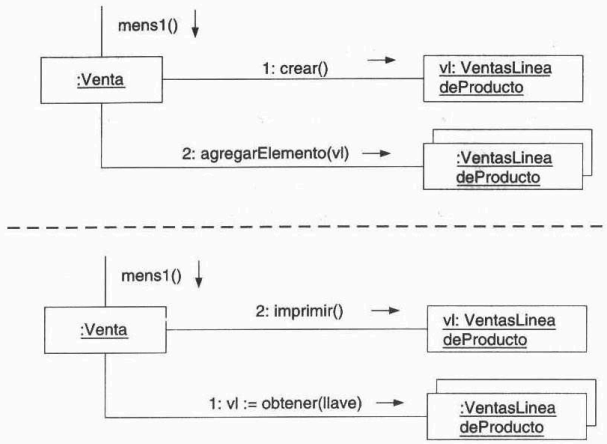


Figura 17.22 Mensajes dirigidos a un multiobjeto y a un elemento.

17.9.15 Representación gráfica de los mensajes dirigidos a un objeto clase

Los mensajes pueden ser dirigidos a la propia clase y no a una instancia, con el fin de llamar los métodos de la clase. Por ejemplo, en Java éstos se implementan como métodos estáticos; en Smalltalk son métodos de clases.

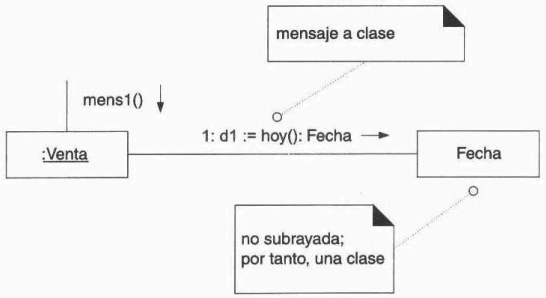


Figura 17.23 Mensajes a un objeto clase (llamada a método estático).

Los mensajes se representan de un modo muy simple: se incluyen dentro de una casilla de clases cuyo nombre no esté subrayado; se indica con ello que el mensaje va a ser enviado a una clase y no a una instancia (figura 17.23).

> En consecuencia, es importante ser consistente en subrayar los nombres de las instancias cuando se desea denotar una instancia; pues de lo contrario pueden interpretarse erróneamente los mensajes dirigidos a las instancias y los dirigidos a las clases.

17.10 Modelos muestra

Los diagramas de interacción son miembros del modelo del comportamiento de objetos, ya que describen el comportamiento de los objetos del software.

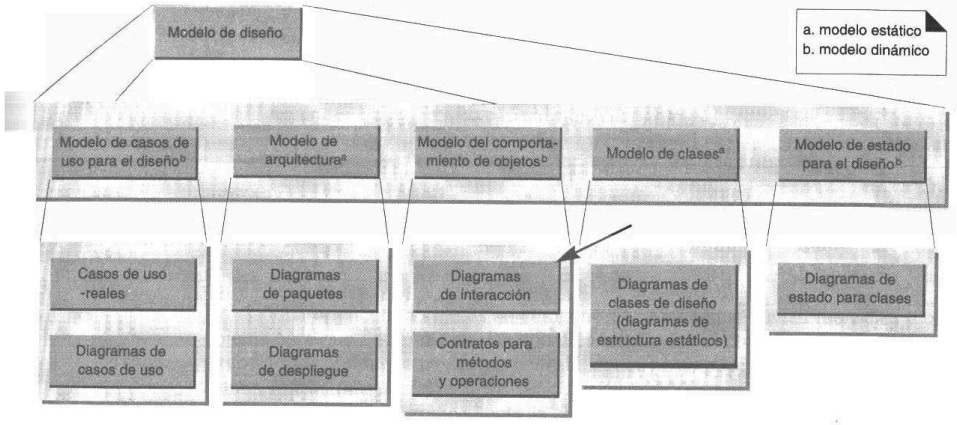


Figura 17.24 El modelo de diseño.