3.2.3 Diagramas de secuencia

Los diagramas de interacción muestran interacciones, es decir, un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos. Cubren la vista dinámica del sistema. La clase a la que pertenece cada objeto aparece tras el nombre del objeto, separada por dos puntos.

Tanto los diagramas de secuencia como los de colaboración son casos particulares de diagramas de interacción. Los diagramas de secuencia resaltan la ordenación temporal de los mensajes, mientras que el diagrama de colaboración resalta la organización estructural de los objetos que envían y reciben mensajes.

Normalmente se usa un diagrama de secuencia para especificar el flujo principal de un caso de uso, y se usan variaciones de ese diagrama para especificar los flujos excepcionales del caso de uso.

El diagrama de secuencia consta de una serie de objetos (instancias de clases, normalmente) dispuestos a lo largo del eje X y los mensajes se representan ordenadamente en el tiempo en el eje Y. El diagrama de colaboración es una colección de nodos y arcos.

En los diagramas de secuencia los objetos tienen lo que se conoce por línea de vida. Es una línea discontinua vertical que representa la existencia de un objeto a lo largo de un período de tiempo.

A continuación se adjuntan algunos ejemplos de diagramas de secuencia para ilustrar como definen la implementación de un caso de uso.

En realidad uno de los diagramas de secuencia no define la implementación de ningún caso de uso, sino que es una secuencia que debe seguirse en todos los casos de uso llamados *Crear interfaz gráfico* que aparecen en los interfaces de nodos, de tramos y de líneas de bus. En concreto este diagrama define la secuencia de acciones necesarias para crear las dos imágenes que aparecen en la parte derecha de cada uno de estos interfaces.

El primer diagrama de secuencia que se muestra (figura 3.2.3.1) es el que define la implementación del caso de uso *Cargar datos de la red vial para representación gráfica*, que aparece en el diagrama de casos de uso del interfaz para representación gráfica.

Lo primero que puede apreciarse es qué objetos (y no clases) intervienen en el caso de uso.

Siguiendo la ordenación temporal del diagrama de secuencia, a lo largo del eje Y, se muestran los métodos que deben ejecutarse para implementar el caso de uso.

En primer lugar, el objeto *contexto* (de la clase *Contexto*) crea los objetos *imagen* e *imagen_aux*. Puede observarse como el objeto *contexto* envía un mensaje a cada uno de los objetos que crea.

A continuación, es el formulario *Form1*, el que se crea inicialmente, el que llama al método *Leer_nodos* de la propiedad *mapa* del objeto *contexto.* Este método sirve para leer los nodos de disco, para ello lee los datos del fichero y añade cada nodo llamando al método *Add_elemento* del objeto *Lista_nodos*, que es la lista que contiene objetos de la clase *Punto* donde se almacena la información de todos los nodos.

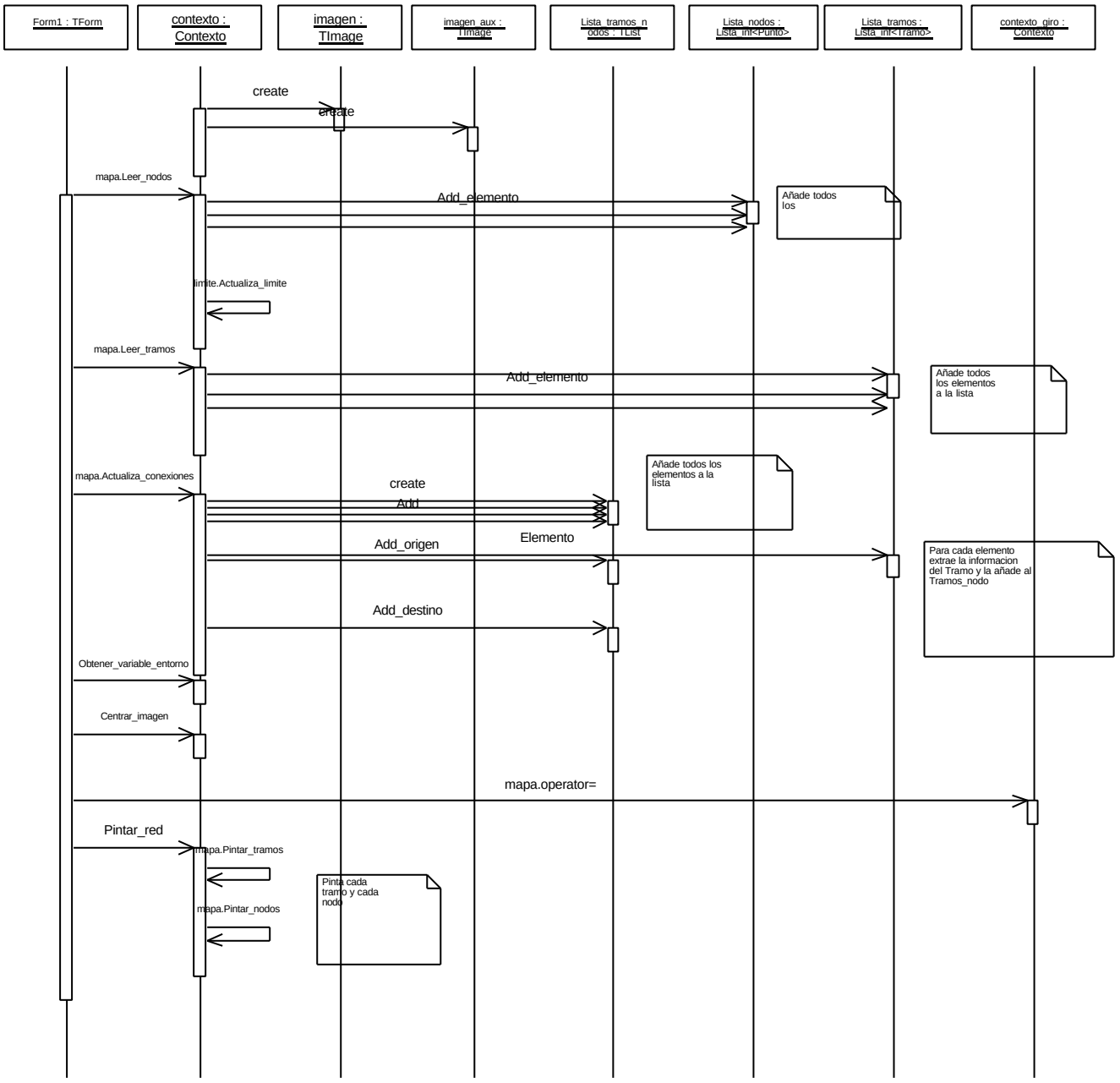

Figura 3.2.3.1: Diagrama de secuencia de Cargar red vial para representación gráfica

Tras leer todos los nodos y añadirlos a la lista se está en disposición de calcular las coordenadas del rectángulo que incluirá a todos los nodos. Para ello *Form1* llama al

método *mapa.Actualiza_limite* del objeto *contexto.* Esta notación indica que se llama al método *Actualiza_limite* de la propiedad *mapa* del objeto *contexto.*

A continuación deben leerse los tramos. Para ello el objeto *Form1* llama al método *mapa.Leer_tramos* del objeto contexto. Se sigue un proceso similar: se lee cada uno de los nodos y se añaden a la lista *Lista_tramos*, la lista que contiene objetos de la clase *Tramo* y almacena la información de los tramos, llamando al método de dicha lista *Add_elemento.*

A continuación *Form1* llama al método *mapa.Actualiza_conexiones*. Por supuesto, al método *Actualiza_conexiones* de la propiedad *mapa* del objeto *contexto*: puede apreciarse en el diagrama ya que el mensaje es enviado al objeto *contexto.*

A continuación se crea la lista *Lista_tramos_nodo*. Seguidamente, para cada elemento de la lista *Lista_tramos* se lee el nodo origen y destino y se añade la información a la lista *Lista_tramos_nodo*, usando los métodos *Add_origen* y *Add_destino*.

A continuación *Form1* llama al método *Obtener_variable_entorno* para calcular el valor de la variable *entorno*. Seguidamente llama al método *Centrar_imagen* que calcula el desplazamiento necesario para que la red vial se represente en el centro de la imagen.

Seguidamente se copian los valores de la propiedad *mapa* del objeto *contexto* en la propiedad *mapa* del objeto *contexto_giro*.

Finalmente llama el método *Pintar_red* del objeto *contexto*, que a su vez llama a los métodos *mapa.Pintar_tramos* y *mapa.Pintar_nodos*. Como indican las flechas, estos métodos pertenecen a la propiedad *mapa* del objeto *contexto.*

Otro caso de uso cuya implementación se ha definido completamente es Realizar zoom y windowing, que aparece en el mismo diagrama de casos de uso que el anterior, en el interfaz para representación gráfica.

El diagrama de secuencia que define dicha implementación se muestra en el figura 3.2.3.2.

En primer lugar, hay que indicar que este diagrama de secuencia vale para tanto para el caso de zoom como para el windowing, sin más que cambiar la llamada al método *Realizar_zoom* por *Realizar_windowing.*

En este diagrama aparece un objeto llamado *boton_zoom*. Este objeto ejecuta tres métodos que tienen como fin asignar los métodos *Caja_zoom_desplazamiento, Caja_zoom_final* y *Caja_zoom_inicio* a los eventos *onMouseMove, onMouseUp* y *onMouseDown* sobre la propiedad *imagen* del objeto *contexto*, respectivamente.

Una vez hecho esto, sólo queda esperar a que se produzcan los eventos.

Lo normal es que en primer lugar se produzca el evento *onMouseDown* sobre la propiedad *imagen* del objeto *contexto.*

Esto provoca que el objeto *contexto* llame a los métodos *Activar_caja* y *Actualizar_caja* del objeto *zoom*. El primero de ellos activa el zoom, y el segundo almacena en el objeto *zoom* las dimensiones del rectángulo que ha seleccionado el usuario, que inicialmente será un solo punto.

A continuación se producirán consecutivamente varios eventos *onMouseMove* sobre la imagen. La secuencia de métodos que provoca la llegada de este evento, como se muestra en el diagrama, consigue que el rectángulo que va seleccionando el usuario arrastrando el ratón sobre la imagen aparezca en color.

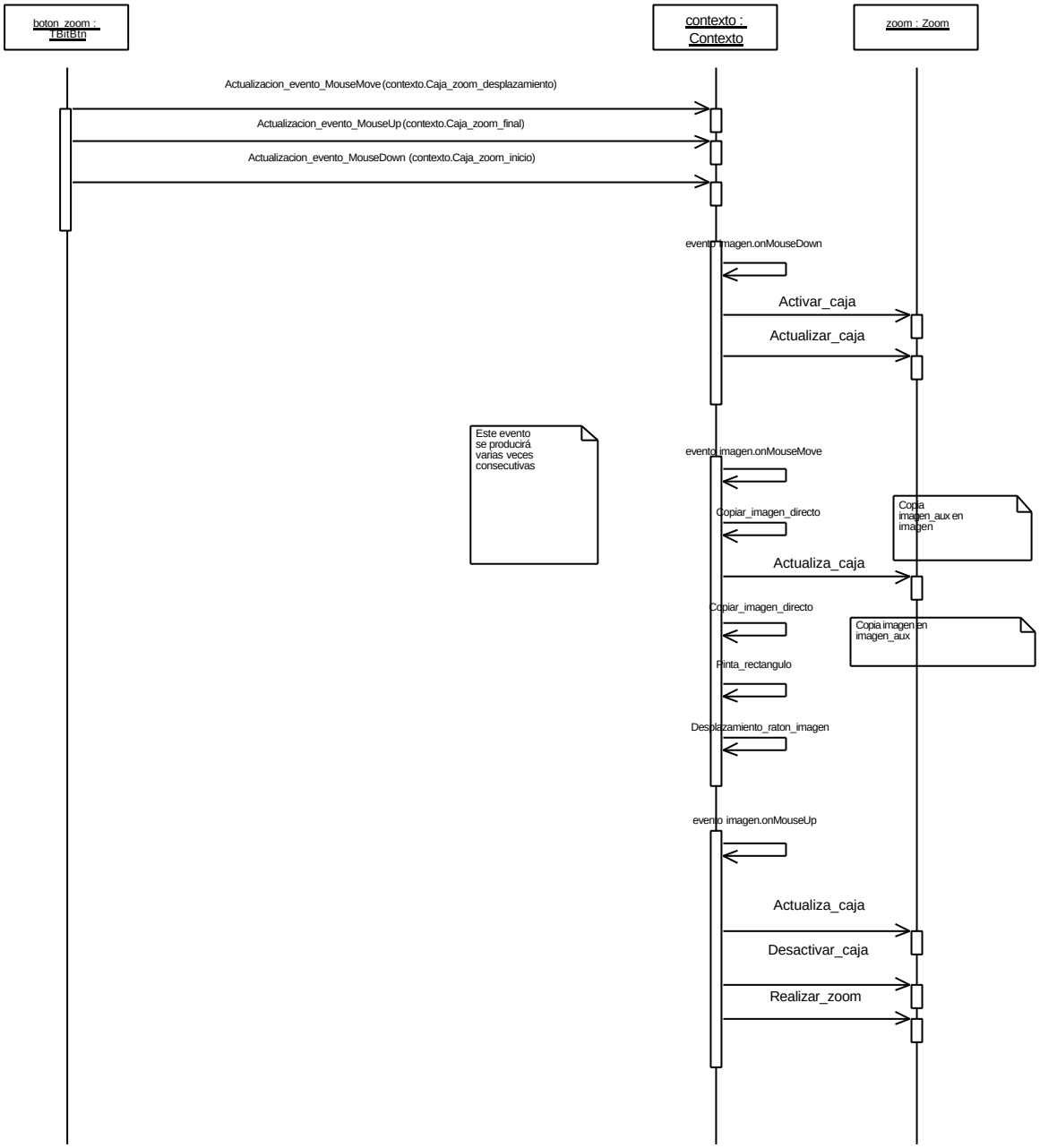

Figura 3.2.3.2: Diagrama de secuencia de Realizar zoom y windowing

Finalmente debe producirse un evento *onMouseUp* sobre la imagen. Entonces se llama a *Actualiza_caja*, para obtener el valor final del rectángulo que ha seleccionado el usuario. Seguidamente se llama a *Desactivar_caja*, para desactivar el zoom, y por último se llama al método *Realizar_zoom*.

En el caso de la operación de windowing se llamaría al método *Realizar_windowing*.

El último diagrama de secuencia, como ya se ha dicho, no describe un caso de uso completo, sino sólo una parte de los casos de uso Crear interfaz gráfico, en concreto este diagrama muestra la creación de las dos imágenes que aparecen en el interfaz. Este diagrama se muestra en la figura 3.2.3.3.

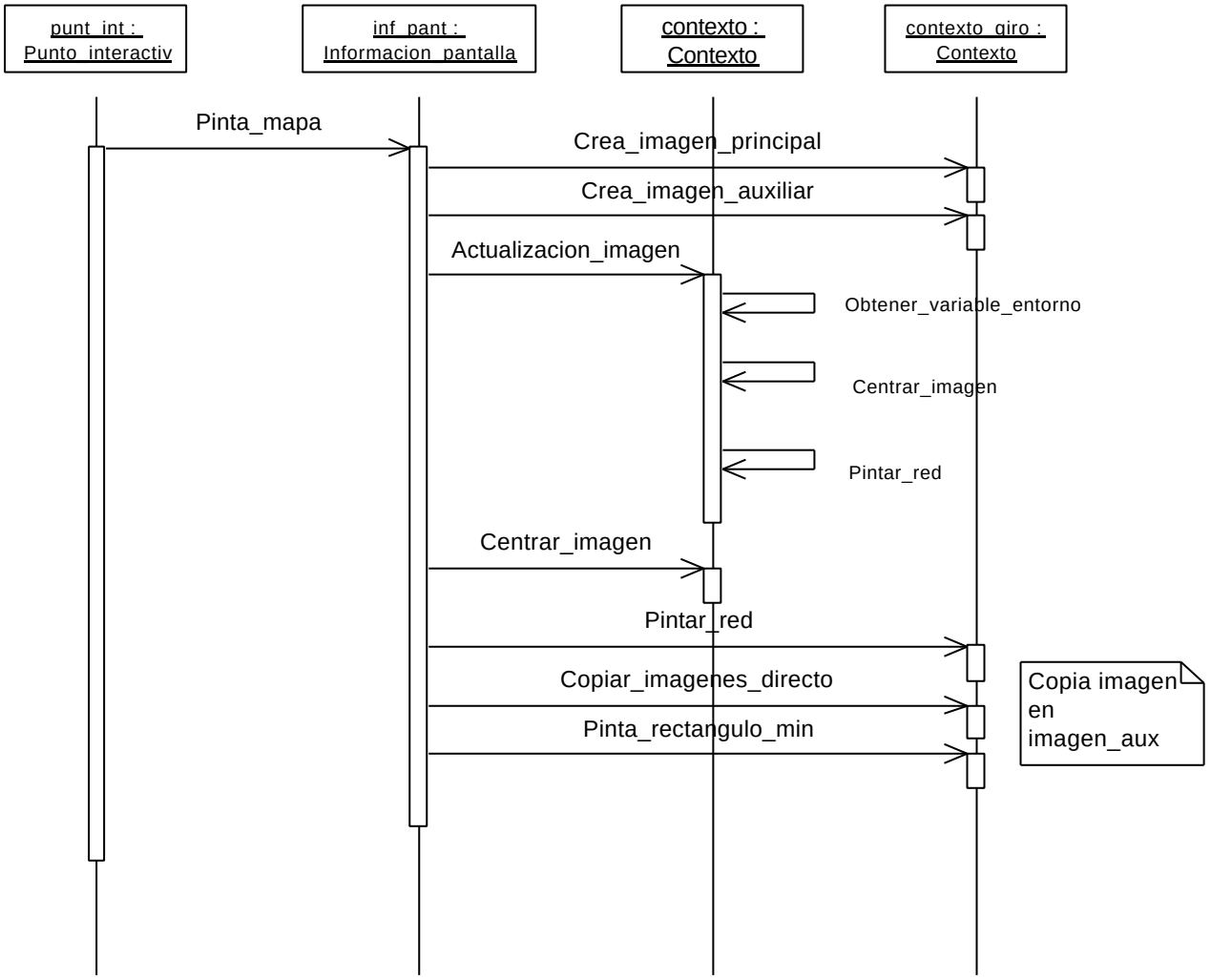

Figura 3.2.3.3: Diagrama de secuencia para crear imágenes

En este diagrama aparece el objeto *punt_int*, suponiendo que se está en el interfaz de tramos. Sin embargo, es totalmente aplicable al interfaz de nodos y el de líneas de bus.

Para crear las imágenes, el objeto *punt_int* tan solo tiene que llamar al método *Pinta_mapa* del objeto *inf_pant*.

Lo primero que hace este método es llamar a los métodos *Crear_imagen_principal* y *Crear_imagen_auxiliar* del objeto *contexto_giro*. Estos métodos crean las propiedades *imagen* e *imagen_aux* del objeto *contexto_giro*.

A continuación se llama al método *Actualizacion_imagen*, que tiene como fin modificar las dimensiones de la imagen y el entorno para adaptarla al nuevo espacio donde tendrá que representarse y posteriormente representarla. Para ello se usan los métodos *Obtener_variable_entorno, Centrar_imagen* y *Pintar_red*. Ya se tiene dibujada la imagen inferior.

A continuación se pinta la imagen superior, que representa la red vial completa. Para ello se llama al método *Pintar_red* del objeto *contexto_giro*.

Seguidamente se llama a *Copiar_imagen_directo* para copiar *imagen* en *imagen_aux* del objeto *contexto_giro*.

Finalmente se llama a *Pinta_rectangulo_min* para que en la imagen superior se represente la porción de red vial mostrada en la imagen inferior en color inverso.