UML: Lenguaje de Modelado Unificado

Tema 4. Diagramas de actividad

UML: Lenguaje de Modelado Unificado

Tema 4. Diagramas de actividad

por: Javier Martín Juan

Revisión: `daa571c72927`

Última actualización: 28 de marzo de 2013

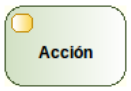
Índice

1. Diagrama de actividad

Los diagramas de actividad son un tipo de **diagramas de comportamiento** que representan un flujo de acciones paso a paso y permiten expresar **condiciones**, **iteraciones** y **concurrencia**.

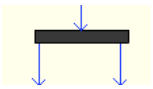
Elementos del diagrama

Los diagramas de actividad están formados por una serie de figuras conectadas por flechas. Las figuras más importantes son:



Rectángulos redondeados: representan una acción.

Rombos: representan puntos donde el flujo de control se ramifica en base a alguna condición *(branch)*. También se usan para "des-ramificar" o reagrupar *(merge)* flujos previamente ramificados.

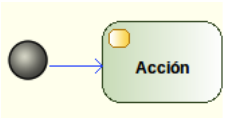


Barras horizontales: sirven para sincronizar el inicio *(fork)* o el fin *(join)* de acciones concurrentes.

Círculo negro: representa el inicio del flujo.

Un círculo negro rodeado de una circunferencia: indica el final de la actividad, incluyendo todos los posibles flujos concurrentes que pudiera haber.

Un círculo con un aspa: final de un flujo (pero el resto de flujos concurrentes continúa).



Flechas: las flechas se trazan desde unas figuras a otras para indicar el orden en que se suceden las actividades.

Los diagramas de actividad son parecidos a los tradicionales diagramas de flujo, con la ventaja de que permiten expresar concurrencia (con las barras horizontales) mientras que los diagramas de flujo no.

Ejemplo: máquina de café

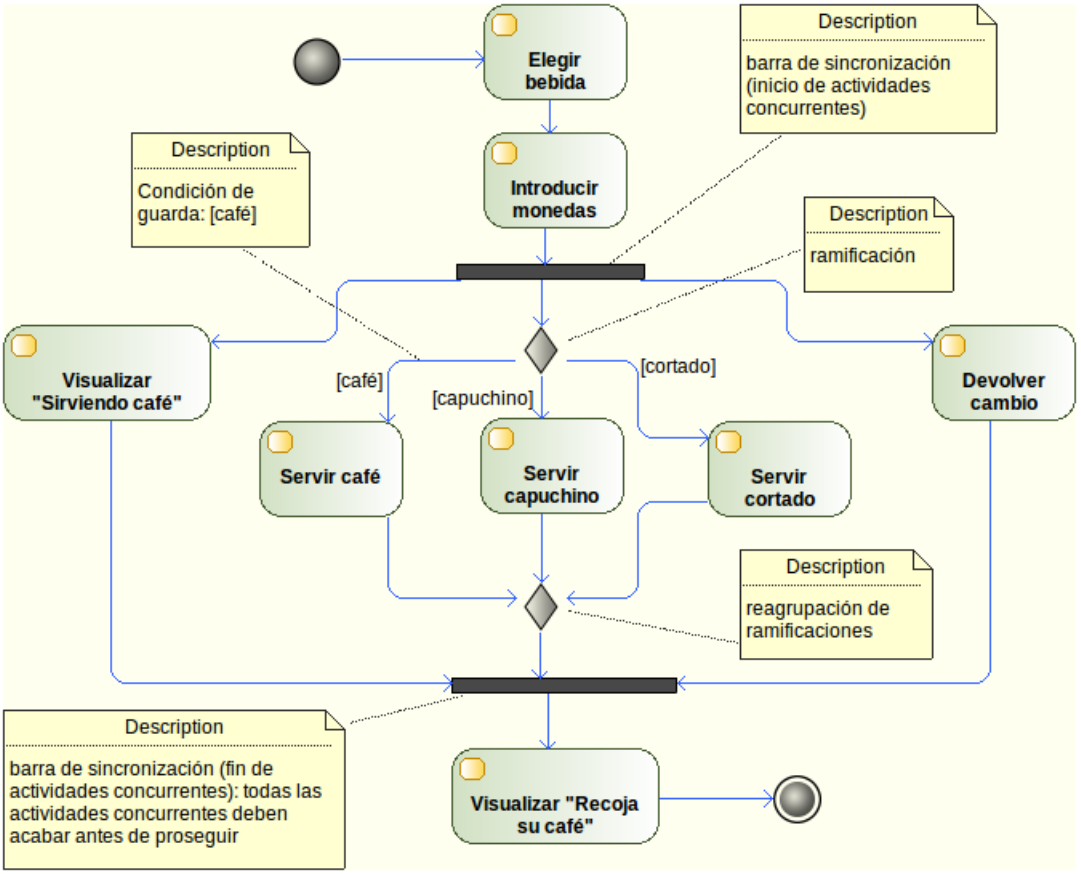
El siguiente diagrama modela las acciones básicas de una máquina capaz de servir cafés, capuchinos y cortados, e ilustra los elementos principales del diagrama de actividad: acciones, ramificaciones y sincronización:

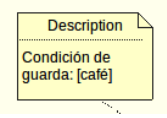
Primero se selecciona la bebida y se recibe el pago. Entonces se inician 3 acciones concurrentemente (barra de sincronización):

- Visualizar "Sirviendo café" en la pantalla de la máquina.

- Servir café, capuchino o cortado en función de la selección (rombo de ramificación). Las condiciones se representan en las flechas que salen del rombo, se ponen entre corchetes [ ] y en UML se denominan **condiciones de guarda**. El rombo de más abajo sirve para reagrupar las ramificaciones.
- Devolver el cambio.

La barra de sincronización donde concurren las acciones de visualizar, servir bebida y devolver cambio significa que todas las acciones deben acabar antes de continuar con la siguiente (visualizar "Recoja su café").





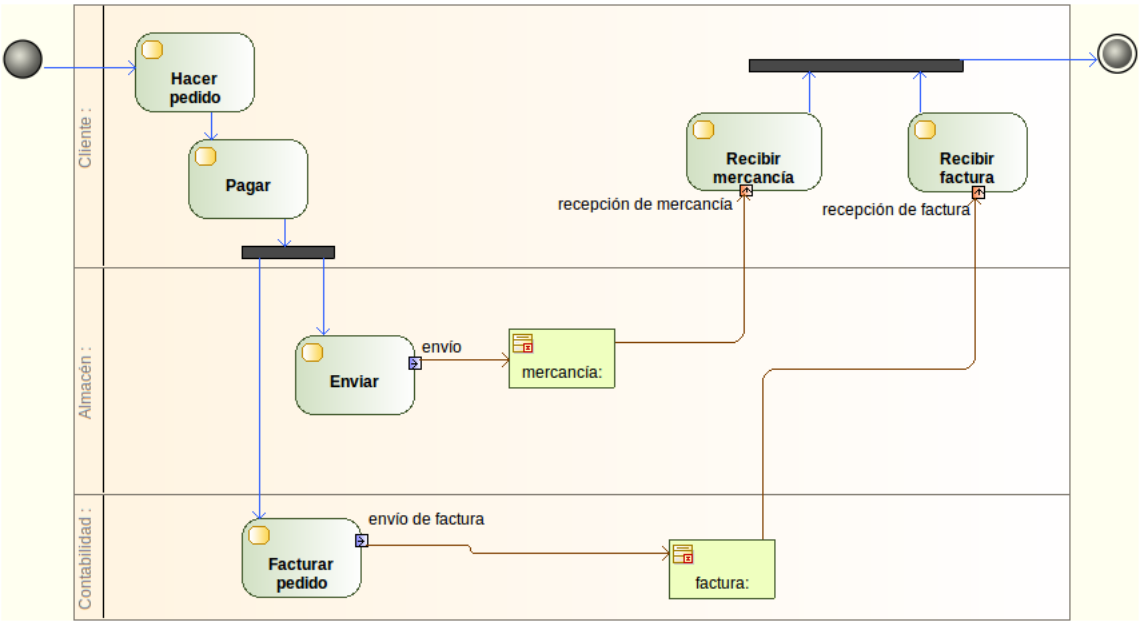
En este diagrama se han utilizado recuadros como el de la izquierda para describir los elementos del diagrama. En realidad estos recuadros son también elementos válidos en un diagrama UML y su utilidad es precisamente clarificar o añadir comentarios al resto de elementos. Tienen el aspecto de una hoja de papel con la esquina superior derecha doblada.

Particiones

A veces puede ser útil organizar las acciones del modelo según la responsabilidad ("quién las realiza"). Para ello dividimos el diagrama en **particiones** mediante líneas horizontales o verticales. Por el aspecto del diagrama a las particiones también se les puede llamar **calles** (como las de una piscina) o ***swimlanes***.

Ejemplo: pedidos

El siguiente ejemplo muestra el flujo de actividad en una empresa desde que llega un pedido hasta que se recoge. Los responsables involucrados son el cliente, el departamento de contabilidad y el almacén:



En este diagrama aparece una nueva figura: los *objetos*. Los objetos se dibujan como rectángulos (sin redondear) y representan resultados o valores de entrada a alguna acción. Cuando representan el resultado se dibuja una flecha desde la acción al objeto, mientras que si representan los datos de entrada para alguna acción, la flecha va desde el objeto a la acción.

Ejemplo: adivina

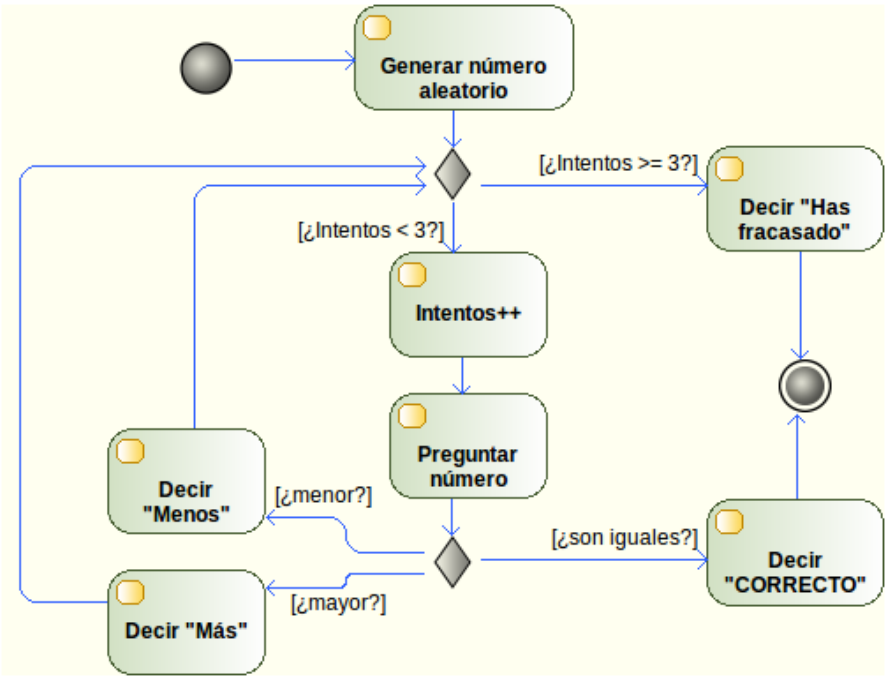
Los diagramas de actividad pueden expresar estructuras de control (condiciones y bucles), lo que los hace idóneos para representar gráficamente algoritmos de programación.

El siguiente programa Java es un juego en el que el debemos adivinar un número entre 1 y 100 elegido por el ordenador en 3 intentos como máximo:

```
import java.util.Random;
import java.util.Scanner;

public class Adivina {
    public static void main(String[] args) throws java.io.IOException {
        Random generador = new Random();
        Scanner scanner = new Scanner(System.in);

        int numero_ordenador = generador.nextInt(100) + 1;
        int numero_humano;
        int intentos = 0;

        System.out.println("Adivina el número del 1 al 100");
        while (intentos < 3) {
```

```
            // Pregunta número
            intentos++;
            numero_humano = scanner.nextInt();

            // Da pistas
            if (numero_ordenador < numero_humano) {
                System.out.println("Menos");
            } else if (numero_ordenador > numero_humano) {
                System.out.println("Mas");
            } else {
                System.out.println("¡¡CORRECTO!!");
                return;
            }
        }
        System.out.println("NOOOOO, ¡HAS FRACASADO!");
    }
}
```

El siguiente diagrama modela el algoritmo del juego:



2. Nuevo en UML 2

Los diagramas de actividad son los que más cambios han experimentado con el paso de UML 1 a UML 2. Hasta ahora todos los ejemplos presentados están hechos siguiendo la notación UML 1. Éstas son algunas de las mejoras introducidas por UML 2:

Condiciones y bucles

Los **nodos condicionales** sirven para expresar un flujo de control condicional. Las condiciones se llaman **cláusulas** y se representan **dentro** del nodo separados por líneas horizontales. Equivaldrían a las instrucciones *if* y *switch/case* en Java y otros lenguajes.

Los **nodos bucle** sirven para expresar estructuras repetitivas del tipo *while .. do*, *do .. while*, *repeat .. until*, etc. Constan de 3 sub-elementos:

- *Setup*: inicialización previa a la entrada al bucle
- *Test*: condición (el bucle se repite mientras sea verdadera)
- *Sub-acciones*: las acciones que se llevan a cabo en cada repetición.

En Java:

```
Setup;
while (Test) {
    Sub-acciones;
}
```

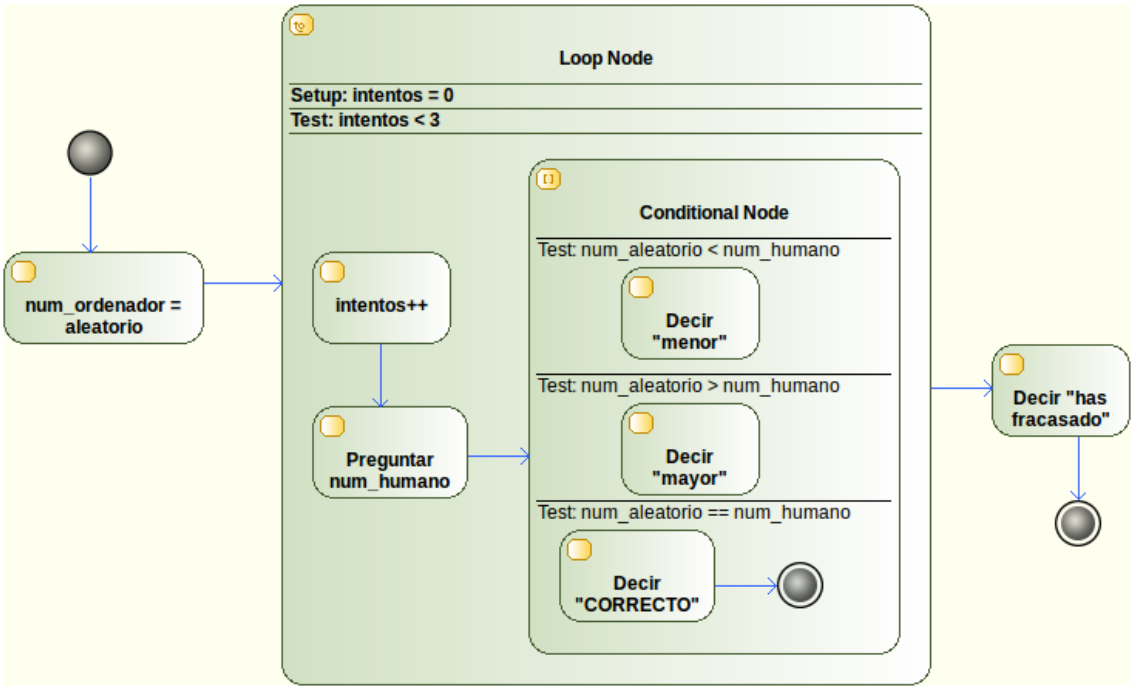
O bien:

```
Setup;
do {
    Sub-acciones;
} while (Test);
```

Ejemplo: juego “adivina” en UML 2

Veamos cómo se modelaría el diagrama de actividad del juego de adivinar de la página 6.

El diagrama ilustra el uso de los **nodos bucle** y **condicional**:



Otros elementos del diagrama

- **Actividades estructuradas**: sirven para hacer agrupaciones lógicas de nodos. Se representan mediante un rectángulo con borde discontínuo y esquinas redondeados.
- **Regiones de expansión**: es un tipo de actividad estructurada que se ejecuta múltiples veces, actuando sobre un conjunto de datos de entrada y produciendo una conjunto de datos de salida.

  Se representan igual que una **actividad estructurada**, pero en los bordes se añaden las **entradas** y **salidas** de la expansión. Las expansiones se pueden etiquetar con las palabras *iterative, parallel* o *stream*, según si el proceso de las entradas se hace de forma **secuencial**, **simultánea** o **contínua**, respectivamente.
- **Entradas y salidas de la expansión**: son el conjunto de datos que entran o salen a una región de expansión. Se representan como un rectángulo alargado y dividido en 4 "cajas" sobre el borde de una región de expansión.
- **Regiones interrumpibles**: se puede marcar una región del diagrama para indicar que la actividad se puede interrumpir mediante alguna acción externa, por ejemplo apretar un botón de cancelación.

  Las regiones interrumpibles son un tipo de actividad estructurada, así que se representan igual que éstas (rectángulo discontínuo, esquinas redondeadas).

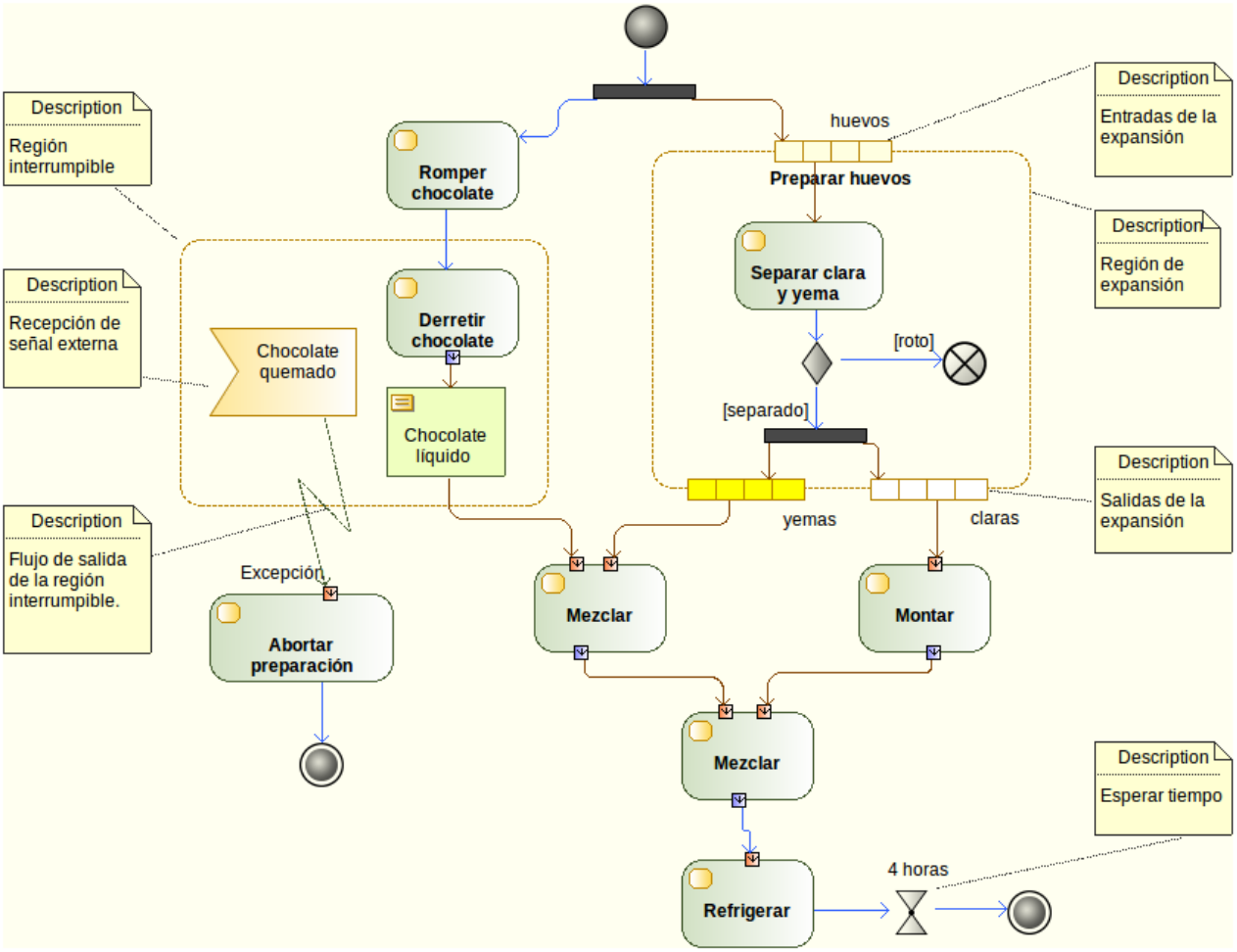
  Para indicar una interrupción se usa una flecha con forma de "relámpago" que conecta con la acción que asume la ejecución.

  Típicamente el origen de una interrupción es la recepción de una **señal** externa.
- **Señales**: las señales son eventos asíncronos que pueden alterar el flujo normal de la actividad, como por ejemplo las interrupciones o excepciones. Se representan como un pentágono cóncavo.

- **Esperar tiempo**: indica una espera. Se simboliza con dos triángulos opuestos por sus vértices, similar a un reloj de arena.

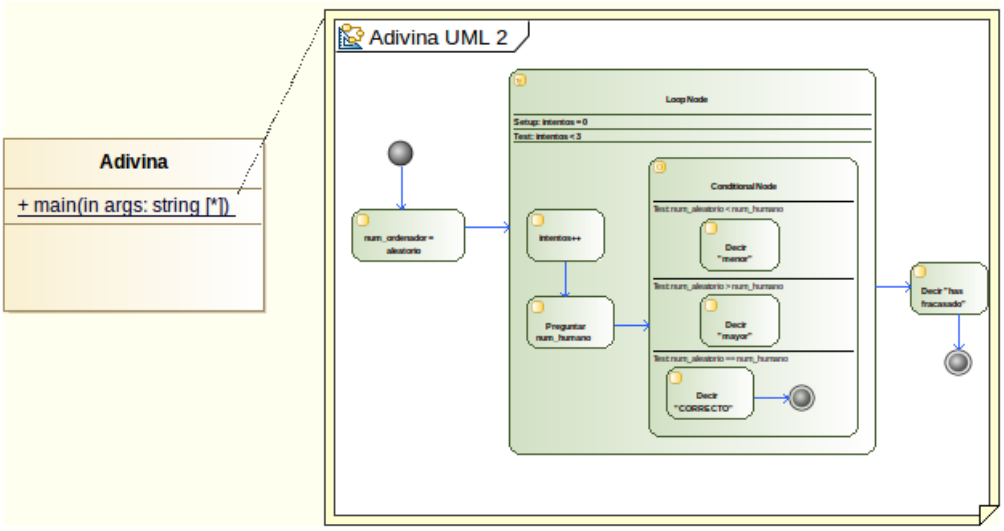
Ejemplo: mousse de chocolate

El siguiente diagrama representa la receta para la preparación de un mousse de chocolate. Los comentarios describen los nuevos elementos introducidos en UML 2:



3. Enlaces entre diagramas

En UML se pueden relacionar unos diagramas con otros. Esta característica es particularmente útil en el caso de los diagramas de clases y actividades vistos hasta ahora, ya que podemos hacer referencia desde una operación de una clase al diagrama de actividad que lo implementa:



Para enlazar diagramas en Modelio podemos utilizar la herramienta *Related diagram link* en la sección *Common* de la paleta de elementos.

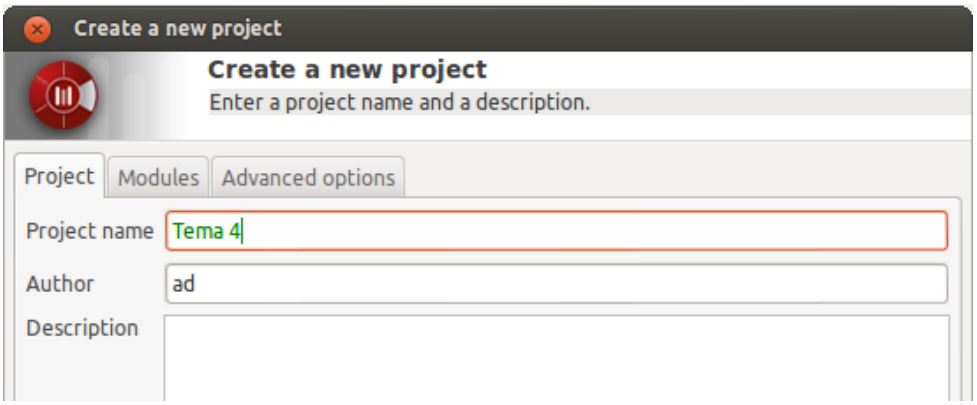
4. Ejercicio guiado

Realizar un diagrama de actividad que muestre el proceso de devolución de un artículo por parte de un cliente. Cuando el cliente solicita una devolución, el departamento de ventas por Internet se encargará de darle un código de devolución. Una vez obtenido, el cliente envía el artículo defectuoso por mensajería, adjuntando dicho código. El almacén es entonces el encargado de la recepción de la mercancía y de la recolocación del artículo. Al mismo tiempo que se recoloca, el departamento de contabilidad actualiza las cuentas. Después de realizarse ambas tareas, el departamento de contabilidad realiza el ingreso a favor del cliente.

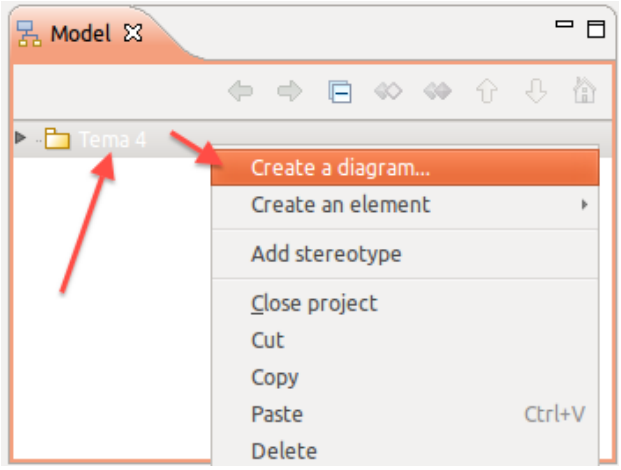
(nota: usar particiones para los distintos responsables)

Solución

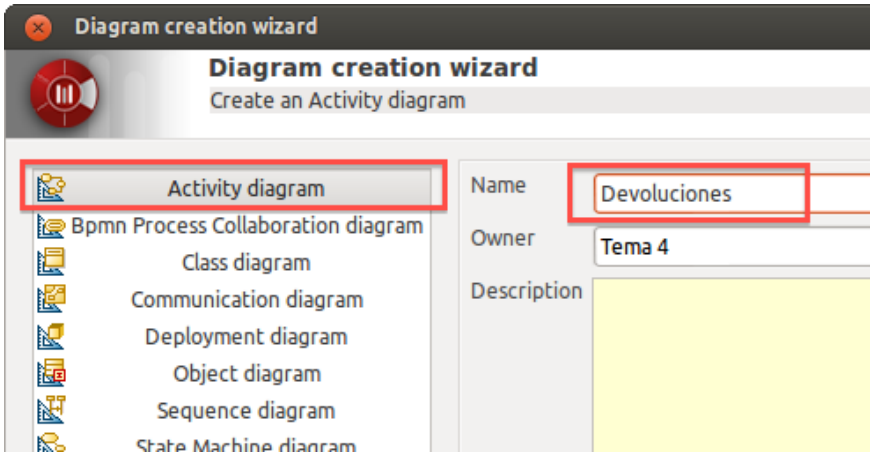
Lo primero que haremos será crear un proyecto nuevo llamado "Tema 4":



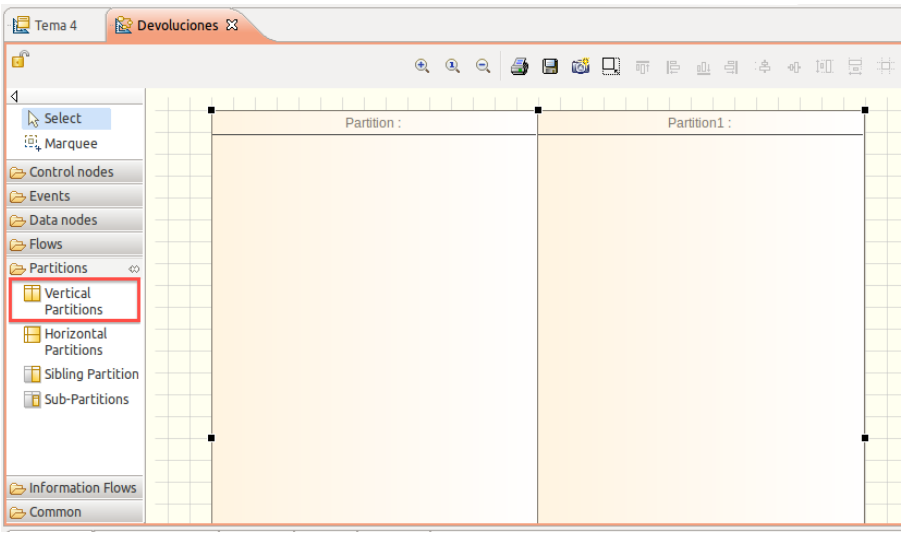
Para crear un diagrama de actividades, botón derecho sobre "Tema 4" en la vista *Model* →*Create a diagram...*:



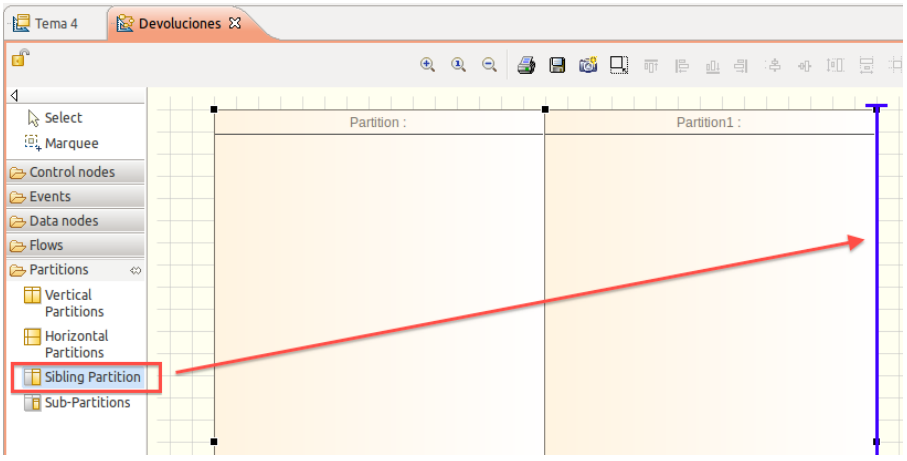
Elegimos *Activity diagram* y le ponemos nombre: "Devoluciones":



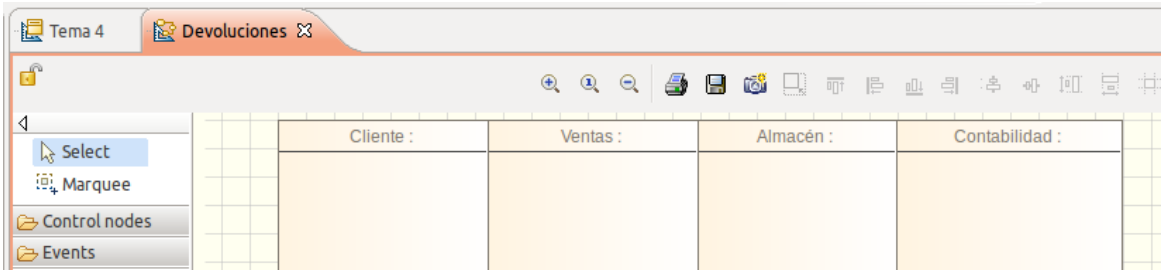
Hay cuatro responsables que intervienen en las actividades: el cliente, el departamento de ventas, el almacén y el departamento de contabilidad.



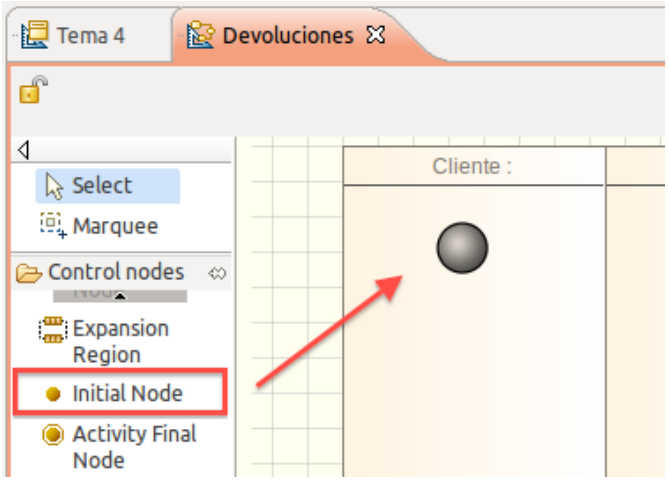
Por defecto nos aparecen sólo dos particiones. Como necesitamos 4, usamos el elemento *Sibling partition* (partición hermana) para añadirlas:



Ponemos nombres a las particiones:

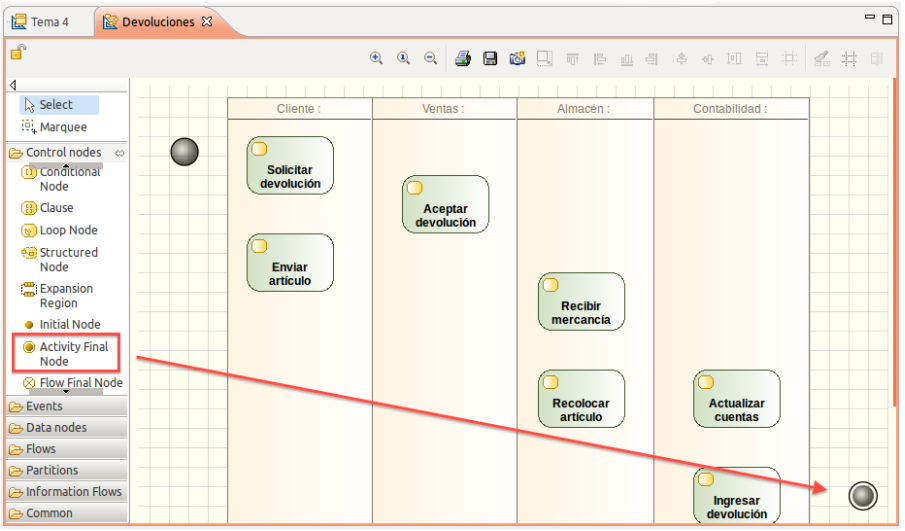


Simbolizamos el inicio de actvidades con el elemento *Initial node*:

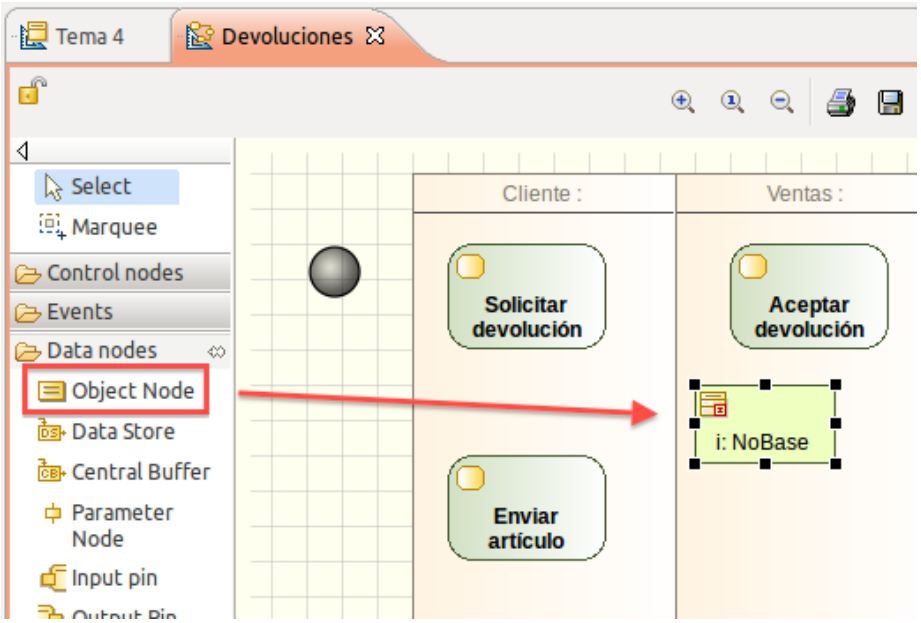


Colocamos las actividades:

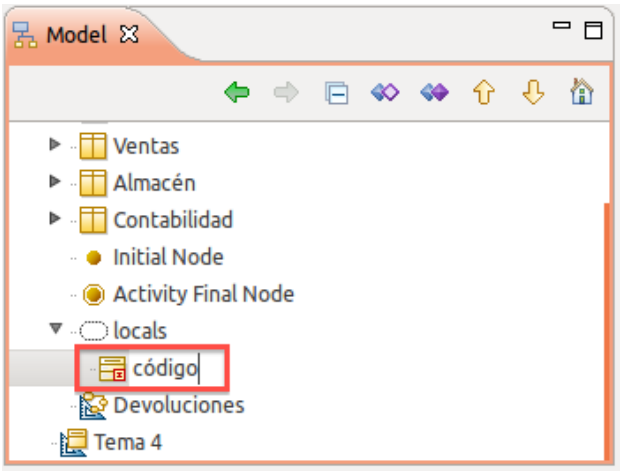
El fin del proceso se señala con el símbolo *Activity Final Node*:



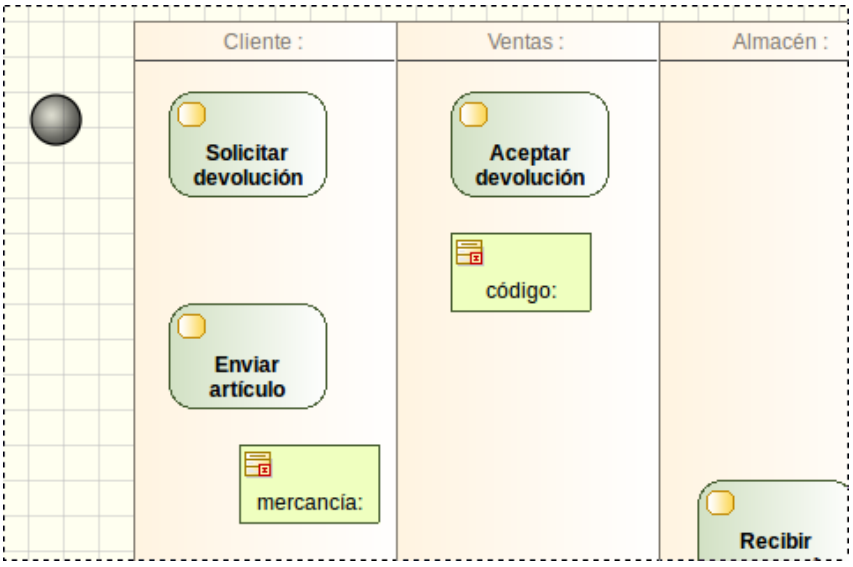
El departamento de ventas envía un código de devolución al cliente. Vamos a crear un nuevo objeto para destacar este aspecto del enunciado:



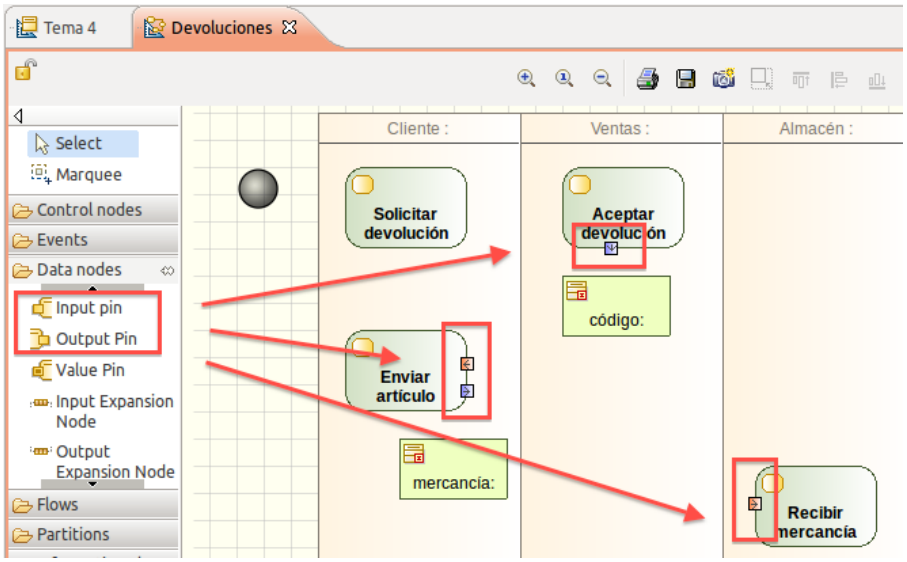
Para cambiar el nombre a los objetos hay que ir a la vista *Model* →*locals*:



Para darle más expresividad al diagrama vamos a añadir también un objeto "Mercancía":

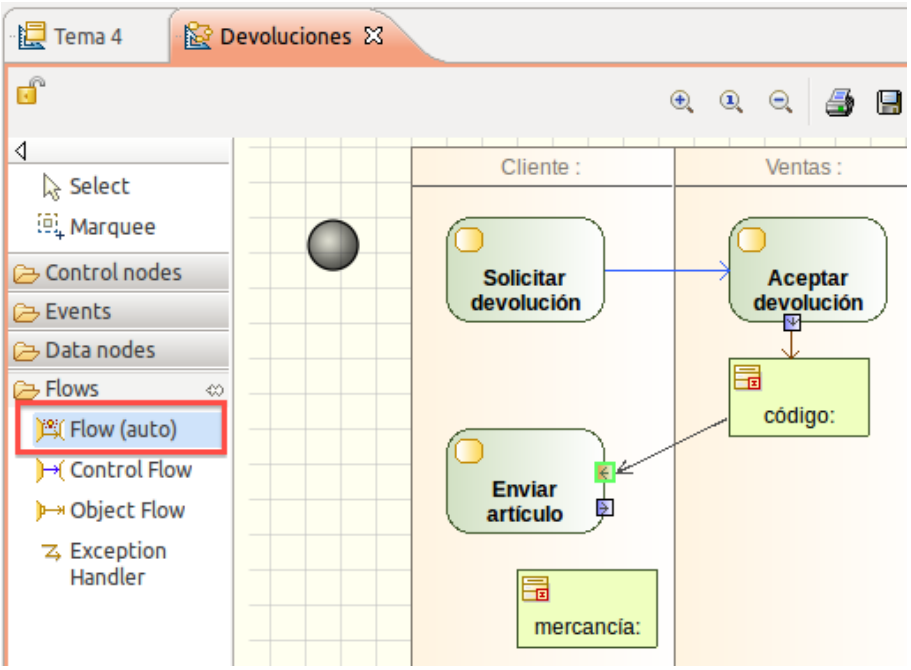


Ahora vamos a conectar las actividades y los objetos. Las actividades se pueden conectar entre ellas sin problemas, pero para conectar con un objeto necesitan *pines* de entrada y de salida:

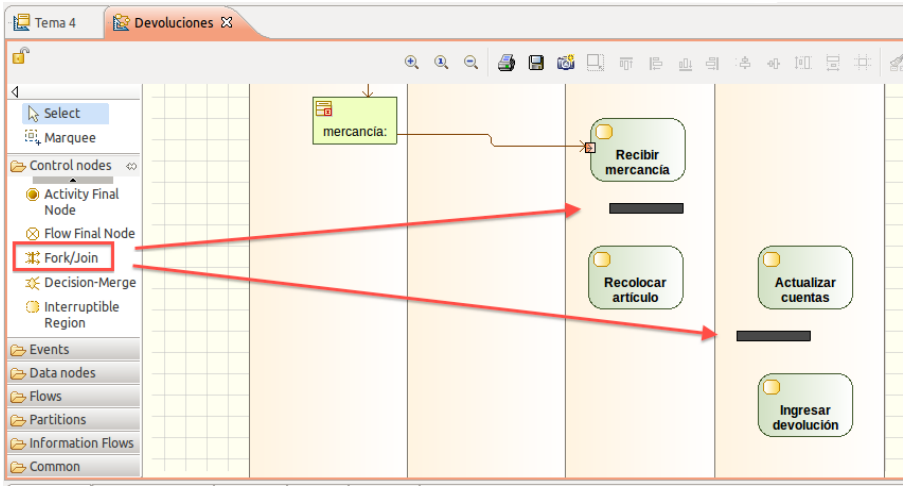


Hay dos tipos de conectores: el flujo de control y el flujo de objetos. El primero conecta dos actividades, mientras que el segundo conecta un objeto con el pin de entrada o salida de una actividad.

Si utilizamos el elemento *Flow (auto)*, el programa elegirá el adecuado automáticamente:



Ya sólo queda expresar la concurrencia. El enunciado nos dice que la recolocación del artículo y la actualización de cuentas ocurren simultáneamente, y que ambas deben completarse antes de ingresar la devolución:



El diagrama finalizado: