

Análisis de Sistemas. Modelado de Procesos de Negocio.

El negocio es quien define los requerimientos de un sistema de información. En consecuencia, es esencial construir un modelo del negocio donde operará el software.

Un negocio (una organización) es un sistema complejo que posee un objetivo o propósito específico. Todas las funciones del negocio interactúan para alcanzar ese objetivo.

Los procesos de negocio determinan como un conjunto de actividades pueden lograr los objetivos específicos de una organización. Describen la forma de operar, tomar decisiones, y establecer el flujo de la información necesario entre los participantes del proceso. Cada proceso es motivado por un evento interno o externo a la organización; se procesa la información de entrada, se manipulan los objetos necesarios, se toman las decisiones requeridas, y se genera la información y los eventos de salida. Además, los procesos están restringidos por un conjunto de reglas de negocio, que determinan las políticas y la estructura de la información del negocio.

Los procesos del negocio son la base de todo buen desarrollo de software y normalmente se identifican en la primera fase del proceso. Facilitan la abstracción del problema para construir la solución informática. Su análisis está orientado a la identificación de:

- los objetivos generales,
- los requisitos del software,
- la selección del tipo de sistema de información, y
- la arquitectura sobre la cual se debe construir el sistema.

Durante la construcción de un sistema de información, el negocio es modelado con el objetivo de comprender el proceso, y evaluar los sistemas de información actuales para determinar si brindan soporte a la manera de trabajar, si el sistema se adapta fácilmente a los cambios, si la información es empleada como un recurso estratégico, si es adecuada y correcta. A partir de la comprensión del proceso se persigue el diseño del sistema de información apropiado. En consecuencia, el modelado del negocio involucra responder:

- ¿Cómo interactúan los diferentes actores?
- ¿Qué actividades son parte de sus trabajos?
- ¿Qué otras personas, sistemas, o recursos están involucrados en el sistema?
- ¿Qué reglas gobiernan las actividades?
- ¿Hay manera de que los actores puedan realizar su trabajo de manera más eficiente?

Los modelos de proceso de negocio describen las distintas actividades que conforman un proceso de negocio. Los procesos de negocio usualmente son transversales a los departamentos de una organización, por ejemplo, la creación de un nuevo producto puede involucrar diversas actividades que combinarán el esfuerzo de varios empleados de múltiples departamentos. El modelado de estos procesos de negocio es una actividad importante para la obtención de los requerimientos, ya que son una herramienta de comunicación potente de los analistas con los usuarios (Dennis y colab., 2009). En este documento abordaremos los diagramas de actividades de UML 2.0 como medio de representación de modelos de procesos de negocio. El documento transcribe principalmente los capítulos 14 y 15 de Arlow y Neustadt (2006).

Los diagramas de actividad son diagramas de flujo orientados a objetos. Los diagramas de actividad permiten modelar procesos como una colección de actividades y transiciones entre estas actividades.

En UML 1 los diagramas de actividad eran casos especiales de máquinas de estado, donde cada estado tenía una acción de entrada que especificaba algún proceso o función que ocurrió cuando ingresó al estado. En UML 2 los diagramas de actividad tienen una semántica completamente nueva basada en Redes de Petri (Murata, 1989). Esto posee dos ventajas:

1. El formalismo de Redes de Petri proporciona mayor flexibilidad en el modelado de diferentes tipos de flujo.
2. Existe una clara distinción en UML entre diagramas de actividad y máquinas de estado.

Una actividad se puede anexar a cualquier elemento de modelado con la finalidad de modelar su comportamiento. El elemento proporciona el contexto para la actividad, y la actividad puede hacer referencia a características de su contexto. Las actividades se anexan normalmente a: Casos de uso, Clases, Interfaces, Componentes, Colaboraciones, y Operaciones.

También puede utilizar diagramas de actividad para modelar procesos de negocio y flujos de trabajo.

La esencia de un buen diagrama de actividad es que está centrado en comunicar un aspecto específico de un comportamiento dinámico de un sistema. Como tal, debe estar en el nivel correcto de abstracción para comunicar ese mensaje a su audiencia objetivo y debería contener la cantidad mínima de información necesaria para que tenga sentido. Usualmente es mejor mantener los diagramas de actividad lo más sencillo posible.

Diagramas de actividad y Proceso Unificado de Desarrollo

Los diagramas de actividad proporcionan un mecanismo de carácter general para modelar comportamientos y puede utilizarse donde añadan valor al proceso de desarrollo. Nosotros lo trataremos en el workflow de análisis.

Los diagramas de actividad permiten modelar un proceso sin tener que especificar la estructura estática de clases y objetos que realizan ese proceso. Claramente, esto es de gran utilidad cuando se encuentra en los primeros niveles de análisis y está tratando de descubrir qué es un proceso determinado.

Los diagramas de actividad se utilizan más comúnmente en las siguientes etapas:

- En el modelado de negocio:
 - Para modelar un proceso de negocio
- En el análisis
 - Para modelar el flujo en un caso de uso de una forma gráfica que es fácil de entender para los grupos de decisión.
 - Para modelar el flujo entre casos de uso. Esto utiliza una forma especial de diagrama de actividad, denominado diagrama de visión de interacción.
- En el diseño
 - Para modelar los detalles de una operación.
 - Para modelar los detalles de un algoritmo.

Los diagramas de actividad son fáciles de entender para los grupos de decisión. Esto es porque la mayoría de los grupos de decisión han tenido algún tipo de relación con los diagramas de flujo de alguna forma. Los diagramas de actividad pueden ser un estupendo mecanismo de comunicación siempre y cuando se mantengan sencillo.

Actividades

Las actividades son redes de nodos conectados por arcos. Existen tres categorías de nodo:

1. Nodos de acción, representan unidades de trabajo que son atómicas dentro de la actividad.
2. Nodos de control, controlan el flujo de la actividad.
3. Nodos de objeto, representan objetos utilizados en la actividad.

Los arcos representan el flujo en la actividad. Existen dos categorías de arcos:

1. Flujos de control, representan el flujo de control en la actividad.
2. Flujos de objeto, representan el flujo de objetos en la actividad.

En las siguientes secciones se analizan en detalle estos tipos de nodo y arcos.

La Figura 1 muestra un sencillo diagrama de actividad para el proceso de negocio Enviar carta. Observe que las actividades pueden tener precondiciones y postcondiciones. Las precondiciones deben ser ciertas antes de que la actividad pueda empezar y las postcondiciones serán ciertas después de que la actividad haya terminado. Las acciones dentro de la actividad pueden tener también sus propias precondiciones y postcondiciones locales.

Las actividades suelen empezar con un solo nodo de control, el nodo inicial. El nodo inicial indica el lugar donde empezará la ejecución cuando se invoca la actividad. Uno o más nodos finales indican los lugares donde termina la actividad.

En el ejemplo de la Figura 1, la actividad empieza en el nodo inicial y luego el control pasa al nodo de acción Escribir carta vía un arco. Este nodo indica un comportamiento que es atómico en lo que se refiere a la actividad que lo contiene. El flujo progresa hacia Escribir dirección y Enviar carta y luego al nodo final donde termina la actividad.

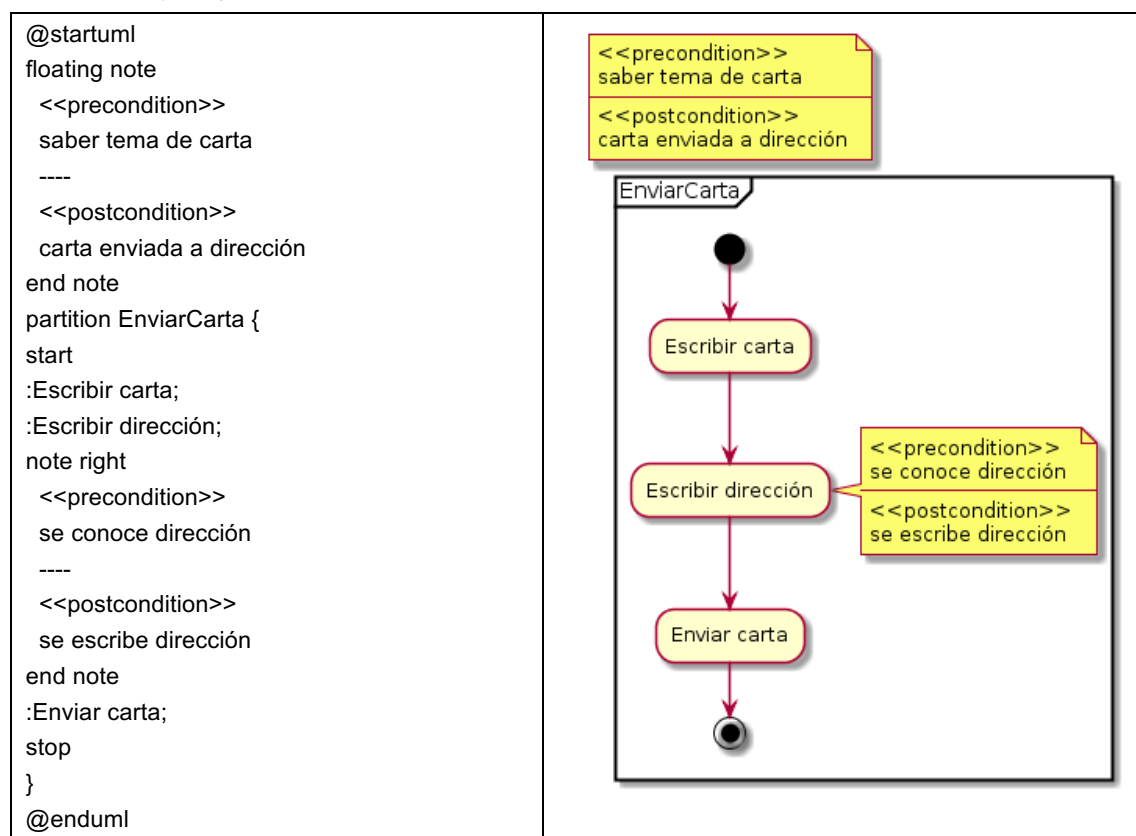


Figura 1. Diagrama de actividad para el proceso enviar carta

Un uso común de los diagramas de actividad es modelar un caso de uso como una serie de acciones. La Figura 2 muestra la descripción del caso de uso PagarImpuestoVentas. Este caso de

uso se puede expresar como un diagrama de actividad según se ilustra en la Figura 3. El diagrama de actividad expresa el caso de uso como dos acciones, Calcular impuesto ventas (paso 2 del flujo principal), y Enviar pago electrónico (paso 3 del flujo principal).

Caso de uso: PagarImpuestoVentas
ID: 1
Breve descripción: Pagar impuesto ventas al organismo correspondiente al final del trimestre
Actores principales: Tiempo
Actores secundarios: Organismo
Precondiciones: 1. Es el final del trimestre.
Flujo principal: 1. El caso de uso empieza cuando es el final del trimestre. 2. El sistema determina la cantidad del impuesto ventas que se debe pagar al organismo correspondiente. 3. El sistema envía un pago electrónico al organismo correspondiente.
Postcondiciones: 1. El organismo recibe la cantidad correcta de impuesto ventas.
Flujos alternativos: Ninguno.

Figura 2. Descripción del caso de uso PagarImpuestoVentas

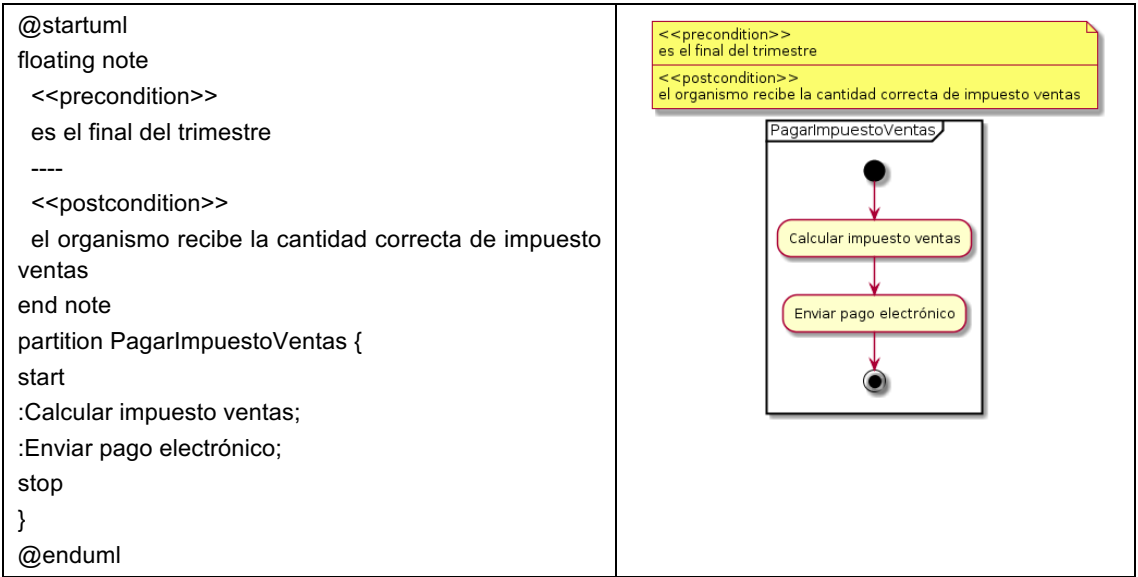


Figura 3. Diagrama de actividad para el caso de uso PagarImpuestoVentas

Cada una de estas acciones se podría expresar como un diagrama de actividad y esto probablemente ocurriría en el workflow de diseño cuando necesite descubrir cómo están

implementadas las acciones. No obstante, el actor y su interacción con el sistema están ausentes en el diagrama de la Figura 3. Los casos de uso expresan comportamiento del sistema como una interacción entre un actor y el sistema, mientras que los diagramas de actividad lo expresan como una serie de acciones. Son vistas complementarias del mismo comportamiento.

Semántica de actividad

Los diagramas de actividad de UML 2 están basados en redes de Petri. En el curso no se estudiarán las redes de Petri, pero pueden tener más información en <http://www.informatik.uni-hamburg.de/TGI/PetriNets/index.php>.

Los diagramas de actividad modelan comportamiento al utilizar el juego de token. Este juego describe el flujo de tokens en una red de nodos y arcos según ciertas reglas. Los tokens en los diagramas de actividad UML pueden representar: el flujo de control; un objeto; datos.

El estado del sistema en cualquier punto en el tiempo está determinado por la ubicación de sus tokens. En el ejemplo de la Figura 1, el token es el flujo de control ya que no existen objetos o datos que se pasan entre nodos en este caso en particular.

Los tokens se pasan de un nodo origen a un nodo destino a través de un arco. El movimiento de un token está sujeto a condiciones y solamente puede ocurrir cuando todas las condiciones se cumplen. Las condiciones varían dependiendo del tipo de nodo. Para los nodos de la Figura 1, estas condiciones son:

- Las postcondiciones del nodo origen.
- Condiciones de control en el arco.
- Las precondiciones del nodo destino.

Al igual que nodos de acción, existen nodos de control y nodos de objeto. Los nodos de control tienen semántica especial que controla cómo los tokens se pasan de sus arcos de entrada a los arcos de salida. Por ejemplo, el nodo inicial empieza una actividad, el nodo final termina una actividad y un nodo join ofrecerá un token en su arco de salida si, y sólo si, existen tokens en todos sus arcos de entrada. Los nodos de objeto representan objetos que fluyen por el sistema. Considere cómo este juego funciona para la actividad definida en la Figura 1. Cuando se ejecuta la actividad, un flujo de token de control empieza en el nodo inicial. No existen condiciones en este nodo, su arco de salida, o el nodo destino, y por lo tanto el token atraviesa automáticamente el arco de salida hasta el nodo destino, Escribir carta. Esto hace que la acción especificada por el nodo de acción Escribir carta se ejecute. Cuando Escribir carta ha terminado, el flujo del token de control se desplaza hasta el nodo de acción Escribir dirección si, y sólo si, su precondición, se conoce dirección, se cumple. Cuando Escribir dirección finaliza y la postcondición se escribe dirección se cumple, el control flujo de Escribir dirección a Enviar carta. Por último, puesto que no existen condiciones que impidan que el flujo salga de Enviar carta, el flujo de control se mueve al último nodo (nodo final de actividad) y la actividad termina.

En este sencillo ejemplo, el flujo de control pasa por cada nodo de acción en turno haciendo que se ejecute. Ésta es la semántica principal de la actividad.

Como hemos mencionado, el estado del sistema que se ejecuta se puede representar en cualquier punto en el tiempo por la disposición de sus tokens. Por ejemplo, cuando el token está en el nodo de acción Escribir carta, puede decir que el sistema está en el estado Escribir carta. Sin embargo, no toda ejecución de acción o flujo de token constituye un cambio notable en el estado del sistema desde el punto de vista de sus máquinas de estado. No obstante, la

disposición de los tokens proporciona un vínculo entre los diagramas de actividad y las máquinas de estado. Debe asegurarse de que los diagramas de actividad y las máquinas de estado para un elemento de modelo en particular son coherentes entre sí. Aunque la semántica de las actividades UML 2 se describen con el juego del token, casi nunca se implementan de esa forma. De hecho, una actividad es una especificación para la que puede haber muchas implementaciones posibles. Por ejemplo, en la Figura 1 estamos describiendo un sencillo proceso de negocio en lugar de un sistema de software, y las implementaciones de este proceso no implicarán pasar tokens.

Particiones de actividad

Para que los diagramas de actividad sean más fáciles de leer, es posible dividir las actividades en particiones al utilizar líneas verticales, horizontales o curvas. Cada partición de actividad representa una agrupación de alto nivel de acciones relacionadas. Las particiones de actividad a veces se denominan carriles o calles.

En UML 2, el modelador define la semántica de las particiones de actividad, no tienen una semántica predefinida. Por lo tanto, se pueden emplear para dividir diagramas de actividad de la forma que se quiera. Las particiones de actividad se utilizan comúnmente para representar: Casos de uso; Clases; Componentes; Unidades organizativas (en modelado de negocio); Roles (en modelados de workflow o flujo de trabajo).

Sin embargo, éstas no son las únicas posibilidades. Por ejemplo, en los modelos de diseño para sistemas distribuidos puedo incluso utilizar particiones de actividades para modelar la distribución de procesos en máquinas físicas. Cada conjunto de particiones debería tener una única dimensión que describa la semántica base del conjunto. Dentro de esta dimensión, las particiones pueden estar jerárquicamente anidadas. La Figura 4 muestra una actividad que tiene un conjunto jerárquicamente anidado de particiones de actividad.

La dimensión es Ubicación y dentro de esta dimensión existe una jerarquía de particiones según se muestra en la Figura 5. Este diagrama modela el proceso de negocio de generación de curso en una empresa con departamentos en Zurich y London.

A menudo existe una conexión entre las particiones de actividad y los flujos concurrentes de control. Por ejemplo, es común que departamentos diferentes o unidades de negocio realicen líneas concurrentes de trabajo y luego se sincronicen en algún punto. Los diagramas de actividad con particiones de actividad son una buena herramienta para modelar esto.

Algunas veces no es posible organizar los nodos en particiones verticales u horizontales sin dificultar la lectura del diagrama. En este caso podría utilizar líneas curvas para crear particiones irregulares, o podría indicar particiones al utilizar texto. UML tiene una notación textual para particiones de actividad, en la acción se antecede el nombre de la misma con el nombre de la partición entre paréntesis. En caso de que el nombre de partición sea una jerarquía, se especifica el nombre de la misma y cada parte se separa con dos puntos dobles (::). Por ejemplo, la acción Desarrollar curso en Ubicación::London:Desarrollo de la Figura 4 se denominaría (Ubicación::London::Desarrollo) Desarrollar curso. Para el caso de particiones múltiples entre paréntesis se indica la lista de particiones separado por comas.

```
@startuml
title Producción curso;
|Ubicación::Zurich::Marketing|
start
:Crear caso de
negocio de curso;
|Ubicación::Zurich::Programación|
|Ubicación::London::Desarrollo|
:Desarrollar curso;
|Ubicación::Zurich::Programación|
:Programar curso;
:Avisar formadores;
:Reservar salas;
|Ubicación::Zurich::Marketing|
:Comercializar curso;
stop
@enduml
```

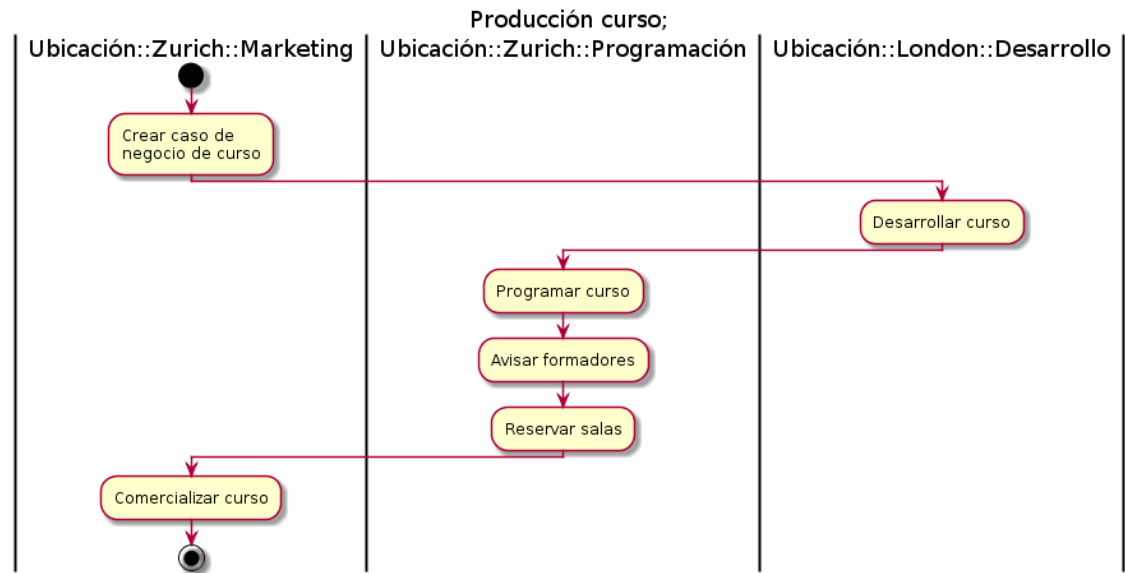
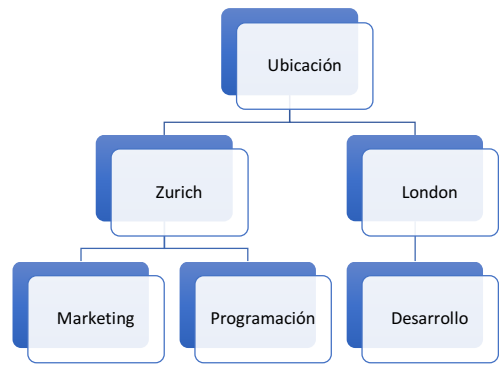


Figura 4. Diagrama de actividad con un conjunto jerárquicamente anidado de particiones de actividad



Representación textual de las particiones de actividades:
(Ubicación, (Zurich, Marketing, Programación), (London, Desarrollo))

Figura 5. Dimensión Ubicación

En raras ocasiones, puede necesitar comportamiento en un diagrama de actividad que está fuera del ámbito del sistema. Esto podría ser para mostrar cómo el sistema interactúa con algún otro sistema externo. Los diagramas de actividad posibilitarían esto añadiendo el estereotipo <<external>> al nombre de la partición que está fuera del ámbito del sistema. No obstante, tenga en cuenta que la partición externa no es una parte del sistema y por lo tanto, no se puede anidar dentro de ninguna jerarquía de partición del modelo.

Nodos de acción

Los nodos de acción se ejecutan cuando:

- Existe un token simultáneamente en cada uno de sus arcos de entrada, aplica un AND.
- Los tokens de entrada satisfacen todas las precondiciones locales del nodo de acción.

Los nodos de acción realizan un AND lógico en sus tokens de entrada; el nodo no está listo para ejecutarse hasta que los tokens están presentes en todos sus arcos de entrada. Esto es un join implícito. Incluso cuando los tokens necesarios están presentes, el nodo solamente se ejecutará cuando su precondición local se cumple.

Cuando el nodo de acción ha terminado de ejecutarse, se comprueba la postcondición local. Si se cumple, el nodo ofrece simultáneamente tokens en todos sus arcos de salida. Esto es un fork implícito ya que un nodo de acción puede dar lugar a muchos flujos. A diferencia de los diagramas de flujo convencionales, los diagramas de actividad son concurrentes intrínsecamente.

Dado que los nodos de acción realizan algo, normalmente se nombran con un verbo o frase verbal. La especificación UML no proporciona ninguna directriz sobre el nombrado de nodos de acción. La convención que utilizamos es nombrar el nodo empezando con una letra en mayúscula y continuando en minúscula, utilizando espacios donde sea apropiado. La única excepción a esta regla ocurre cuando un nodo de acción contiene una referencia a otro elemento de modelo. En este caso, siempre utilizamos el nombre del elemento de modelo como está.

Los detalles de la acción se capturan en la especificación del nodo de acción. A menudo es simplemente una descripción de texto como “Escribir una carta”, pero en diseño, podría ser texto estructurado, pseudocódigo o código. Si el diagrama de actividad está modelando un caso de uso, entonces podría ser uno o más pasos del flujo del caso de uso.

Existen cuatro tipos de nodos de acción, los cuales se resumen en la Tabla 1.

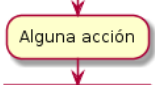
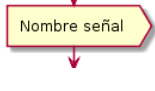


Nodos de acción de llamada

El tipo más común de nodo de acción es el nodo de acción de llamada. Este tipo de nodo puede invocar:

- Una actividad.
- Un comportamiento.
- Una operación

En la Figura 6 se ilustran algunos ejemplos de la sintaxis de nodo de acción de llamada.

Tabla 1. Nodos de acción

Sintaxis	Sintaxis en PlantUML	Nombre	Semántica
	:Alguna acción;	Nodo de acción de llamada.	Invoca una actividad, comportamiento u operación.
	:Nombre señal >	Enviar señal.	Envía acción de señal. Envía una señal asíncronamente, el emisor no espera a la confirmación de la recepción de la señal. Puede aceptar parámetros de entrada para crear la señal.
	:Aceptar evento <	Nodo de acción de aceptar evento.	Acepta un evento. Espera eventos detectados por el objeto que los posee y ofrece el evento en su arco de salida. Se activa cuando obtiene un token en su arco de entrada. Si no existe arco de entrada, empieza cuando la actividad que lo contiene se inicia y siempre está activo.
	No definido, pero podemos definir: :<<Aceptar evento temporal>> expresión de tiempo <	Nodo de acción de aceptar evento de tiempo.	Acepta un evento de tiempo, responde a tiempo. Genera eventos de tiempo según su expresión de tiempo.

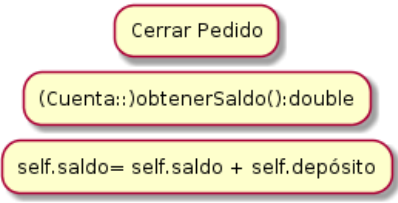
:Cerrar Pedido; :(Cuenta::)obtenerSaldo():double; :self.saldo= self.saldo + self.depósito;	
--	--

Figura 6. Ejemplos de nodos de acción de llamada

En UML es posible indicar que la acción invoca a otra actividad anexando el símbolo especial de rastrillo en la esquina inferior derecha del nodo. Esta característica no se visualiza en la Figura 6, dado que no está soportada por PlantText.

La acción Cerrar Pedido en Figura 6 ilustra la invocación de comportamiento, es una invocación directa de un comportamiento del contexto de la actividad. La siguiente acción de la Figura 6, obtenerSaldo(), ejemplifica la invocación de una operación. Cuenta está entre paréntesis porque es opcional el nombre de la Clase donde la operación es definida. El último ejemplo ilustra los detalles de la operación en un lenguaje determinado de programación. Esto tiene sentido si se utiliza alguna herramienta UML que permita generar código a partir de diagramas de actividad. Mediante la palabra clase self se hace referencia a características del contexto de la actividad. Cuando se emplea nodos de acción de llamada en diagramas de actividad a nivel de análisis, normalmente se invoca un comportamiento, tal como en el primer ejemplo: Cerrar Pedido.

Nodos de acción de aceptar evento de tiempo

Uno nodo de acción de aceptar evento de tiempo responde a tiempo. Este tipo de nodo tiene una expresión de tiempo y genera un evento de tiempo cuando esta expresión es verdadera. Este nodo se comporta de forma diferente dependiendo de si tiene o no un arco de entrada.

Por ejemplo, la Figura 7 muestra un nodo de acción de aceptar evento de tiempo que no tiene arco de entrada. Cuando se activa la actividad que lo posee, este nodo se activará y generará un evento de tiempo siempre que su expresión de tiempo sea verdadera. En el ejemplo mostrado, se genera un evento de tiempo al final de cada año comercial y esto hace que se ejecute la acción Devolver impuesto a empresa.

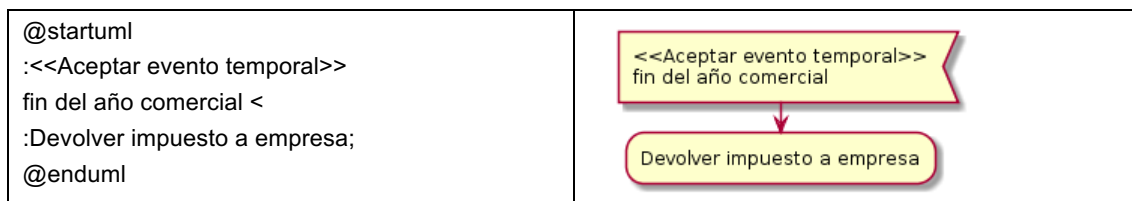


Figura 7. Acción de aceptar evento de tiempo sin arco de entrada

Sin embargo, en el ejemplo en la Figura 8, la acción tiene un arco de entrada y solamente estará activa cuando se reciba un token en ese arco. Este ejemplo es un fragmento de un sistema de ascensor. La primera acción abre la puerta del ascensor y activa la acción de aceptar evento de tiempo. Esta acción espera diez segundos y luego ofrece un token a la acción Cerrar puerta.

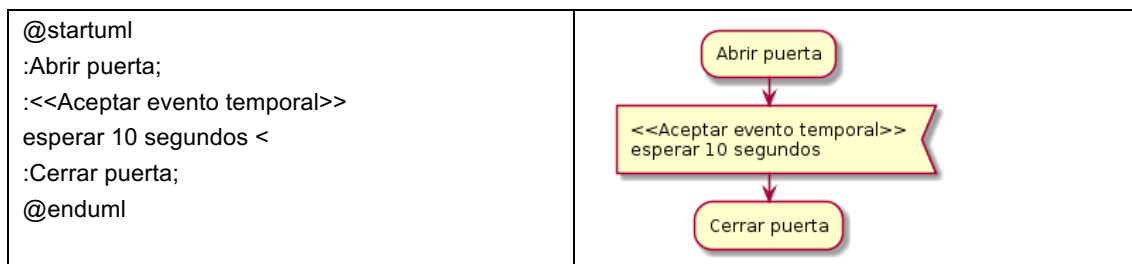


Figura 8. Acción de aceptar evento de tiempo con arco de entrada

La expresión de tiempo puede hacer referencia a:

- Un evento en el tiempo, por ejemplo, final del año.
- Un punto en el tiempo, por ejemplo, el 21/07/1990.
- Una duración, por ejemplo, esperar 10 segundos.

Nodos de control

Los nodos de control gestionan el flujo de control dentro de una actividad. La Tabla 2 resume los nodos de control de UML 2.

Nodo inicial y nodos finales




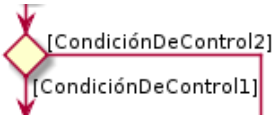



El nodo inicial es el punto en el que empieza el flujo cuando se invoca una actividad. Una actividad puede tener más de un nodo inicial. En este caso, el flujo empieza en todos los nodos iniciales simultáneamente y se ejecuta concurrentemente.

Una actividad también se puede iniciar por una acción de aceptar evento, o por un nodo de parámetro de actividad, por lo que los nodos iniciales no son obligatorios dado que existe otra forma de iniciar una actividad.

El nodo final de actividad detiene todos los flujos dentro de una actividad. Una actividad puede tener muchos nodos finales de actividad y el primero en activarse termina el resto de flujos y la propia actividad.

El nodo final de flujo simplemente detiene uno de los flujos dentro de la actividad, los otros flujos continúan.

Tabla 2. Nodos de control

Sintaxis	Sintaxis en PlantUML	Nombre	Semántica
	start	Nodo inicial.	Indica dónde empieza el flujo cuando se invoca una actividad.
	stop	Nodo final de actividad.	Termina una actividad.
	end	Nodo final de flujo.	Termina un flujo específico dentro de una actividad, los otros flujos no se ven afectados.
	if (Condición) then ([CondiciónDeControl1]) ... else ([CondiciónDeControl2]) ... endif repeat ... repeat while () Condición en el if o repeat while es opcional.	Nodo de decisión.	Se sigue el arco de salida cuya condición de control es verdadera.
	if ... endif repeat ... repeat while ()	Nodo de fusión (merge).	Copia tokens de entrada en su arco de salida.
	fork ... fork again ... end fork	Nodo fork.	Divide el flujo en múltiples flujos concurrentes.
	end fork	Nodo join (sincronización).	Sincroniza múltiples flujos concurrentes.

Nodos de decisión y nodos de fusión

Un nodo de decisión tiene un arco de entrada y dos o más arcos de salida. Un token que llega al arco de entrada se ofrecerá a todos los arcos de salida, pero atravesará al menos uno de ellos. El nodo de decisión actúa como un cruce de caminos en el flujo donde el token debe tomar solamente una dirección.

Cada uno de los arcos de salida está protegido por una condición de control de modo que el arco aceptará un token si, y sólo si, la condición de control evalúa como verdadero. Es importante asegurarse de que las condiciones de control son mutuamente excluyentes de modo que solamente una de ellas pueda ser cierta en cualquier momento en el tiempo. Si no es así, el comportamiento del nodo de decisión está sin definir según la especificación de UML 2.

La palabra clave `else` se puede utilizar para especificar el arco atravesado si ninguna de las condiciones de control es verdadera.

La Figura 9 muestra un sencillo ejemplo de un nodo de decisión. Después de la acción `Comprobar correo`, el flujo de control llega a un nodo de decisión. Si la condición es `basura` es verdadera, entonces el correo es borrado (acción `Borrar correo`), de lo contrario el correo se abre (acción `Abrir correo`).

En la Figura 10 se ilustra una condición anexada al nodo de decisión. Cabe destacar que el nodo de decisión no es una acción, no representa comportamiento. En la figura se lo ilustra dentro del símbolo del nodo de decisión, sin embargo, en UML no existe tal representación y se lo puede anexar como una nota vinculada al nodo. En la condición se compara la cantidad a retirar con el saldo en la cuenta. Si el saldo es mayor que o igual a la cantidad solicitada, entonces la condición evalúa como verdadero y el flujo pasa a `Retirar cantidad`. De lo contrario, se registra un fallo.

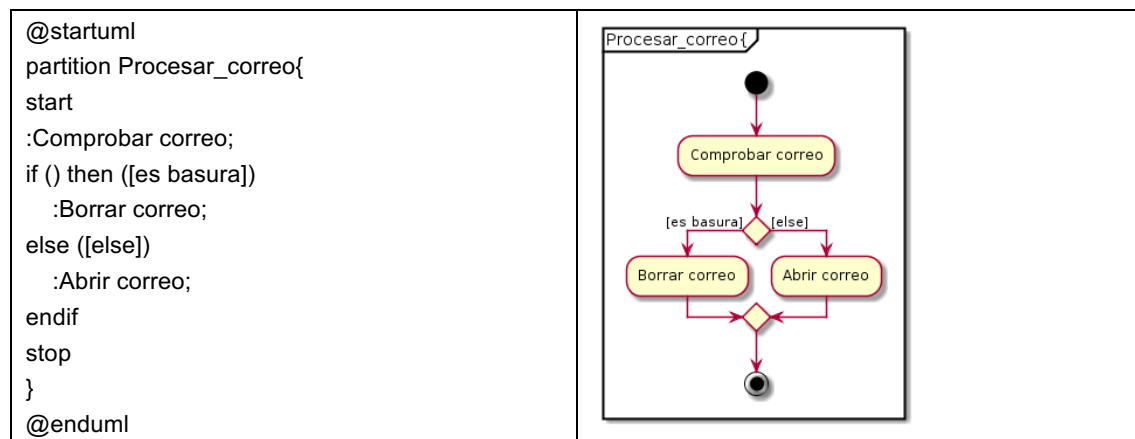


Figura 9. Un ejemplo de nodo de decisión y de fusión.

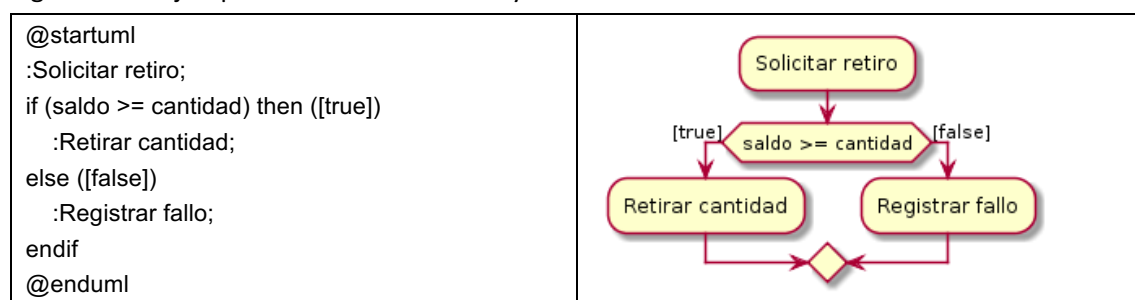


Figura 10. Fragmento de actividad ilustrando un nodo de decisión

La Figura 9 muestra un nodo de fusión (merge). Los nodos de fusión tienen dos o más arcos de entrada y un solo arco de salida. Fusionan todos sus flujos entrantes en un solo flujo de salida. La semántica consiste en que todos los tokens ofrecidos en los arcos entrantes se ofrecen en el arco saliente y no existe modificación del flujo o los tokens.

Un nodo de fusión y un nodo de decisión se pueden combinar en un solo símbolo, sin embargo diversos autores, tales como Arlow y Neustadt (2006), no recomiendan esta notación abreviada ya que normalmente es más claro mostrar nodos separados de fusión y decisión.

Nodos fork y join, concurrencia

Puede crear flujos concurrentes dentro de una actividad al utilizar un nodo fork para dividir un simple flujo en múltiples flujos concurrentes. Mientras que la concurrencia es normalmente una decisión de diseño, a menudo es necesario mostrar actividades concurrentes cuando se está modelando procesos de negocio. Por lo tanto, normalmente utilizará nodos fork y join en el workflow de análisis al igual que en el de diseño.

Un nodo fork tiene un arco de entrada y dos o más arcos salientes. Los token que llegan al arco entrante se duplican y se ofrecen en todos los arcos salientes simultáneamente. Esto divide el único flujo entrante en múltiples flujos salientes paralelos.

Los nodos join tienen múltiples arcos entrantes y un solo arco saliente. Sincronizan flujos al ofrecer un token en su único arco de salida cuando existe un token en todos sus arcos de entrada. Realiza un AND lógico en sus arcos de entrada.

La Figura 11 muestra un ejemplo de Procesar producto que utiliza nodos fork y join. En este ejemplo:

- El producto se diseña primero.
- El producto se pone en el mercado y manufactura concurrentemente.
- El producto se vende solamente después de que esos procesos están completos.

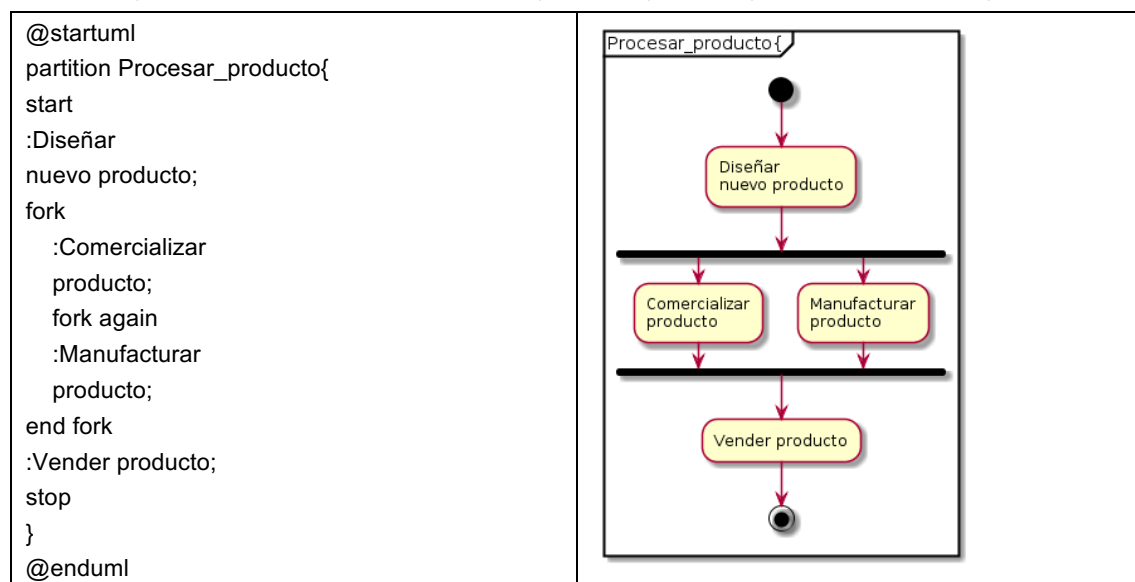


Figura 11. Ejemplo de actividad con nodos fork y join

En la Figura 11 la actividad empieza con la acción Diseñar nuevo producto. Después de esta acción, un nodo fork divide el flujo único en dos flujos concurrentes. En uno de estos flujos el producto se pone en el mercado (acción Comercializar producto) y en el otro se manufactura (acción Manufacturar producto). El nodo join sincroniza los flujos concurrentes porque espera

un token de cada una de las acciones concurrentes. Cuando tiene un token de cada acción, ofrece un token a su arco de salida y el flujo pasa a la acción Vender producto.

Cuando modela nodos join, es importante asegurarse de que todos los arcos de entrada recibirán un token. Por ejemplo, en la Figura 11, si existieran un nodo de decisión en vez del fork, la comercialización del producto y la manufactura serían mutuamente excluyentes y el join nunca recibiría suficientes tokens para su activación, esto produciría que la actividad se colgara.

Nodos de objeto

Los nodos de objeto son nodos especiales que indican que las instancias de un cierto concepto, objetos, están disponibles en un punto específico en la actividad. Se los nombra con el nombre del concepto (Clase) y representan instancias de esa Clase o sus subclases. El fragmento de actividad en la Figura 12 muestra un nodo de objeto que representa instancias de la clase Pedido (o subclases de Pedido). PlantUML no soporta el flujo de objeto, por lo que su representación es limitada, se lo va a representar como una especie de acción con el símbolo] en vez de ;.

Los arcos de entrada y salida de los nodos de objeto son flujos de objeto. Estos son tipos especiales de flujos que representan el movimiento de objetos alrededor de la actividad. Los propios objetos se crean y consumen por nodos de acción. La Figura 13 muestra la secuencia de Diseñar nuevo producto y Manufacturar producto vinculada por el producto EspecificaciónProducto. Diseñar nuevo producto crea el Objeto EspecificaciónProducto. El Objeto EspecificaciónProducto se consume por la acción Manufacturar producto que lo utiliza para definir el proceso de manufacturación.

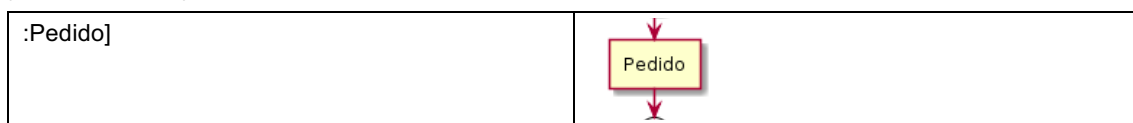


Figura 12. Nodo de objeto, instancia de Pedido

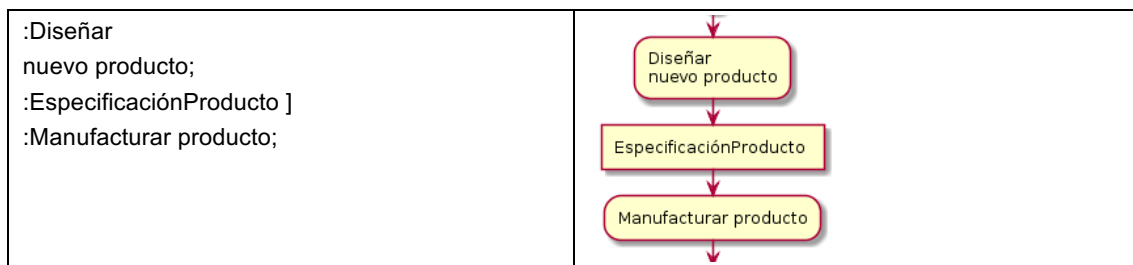


Figura 14. Flujo de objeto

Cuando un nodo de objeto recibe un token de objeto en uno de sus arcos de entrada, ofrece este token en todos sus arcos de salida simultáneamente, pero sigue habiendo un solo token. El token no se replica en los arcos. El primer arco que acepta el token, lo consume.

Los nodos de objeto actúan como buffers, es decir, lugares en la actividad donde pueden residir los tokens de objeto mientras esperan a ser aceptados por otros nodos. Los nodos de objeto pueden albergar un número infinito de tokens de objeto.

Representar objetos en estado

Los nodos de objeto pueden representar objetos en un estado determinado. Por ejemplo, la Figura 15 muestra un fragmento de una actividad de procesamiento de pedido que acepta objetos Pedido que están en el estado Abierto y los entrega (estado Enviado). Los estados de objeto referenciados por nodos de objeto se pueden modelar con máquinas de estado.

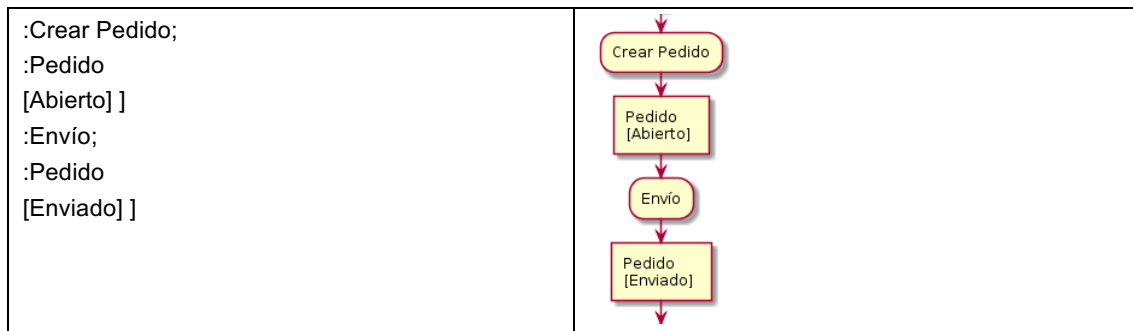


Figura 15. Flujo de objeto, representación de estados

Parámetros de actividad

Puede utilizar nodos de objeto para proporcionar entradas y salidas de actividades, según se ilustra en la Figura 16. Los nodos de objeto de entrada y salida se deberían dibujar solapando el marco de actividad. Los nodos de objeto de entrada tienen uno o más arcos de salida en la actividad; los nodos de objeto de salida tienen uno o más arcos de entrada fuera de la actividad. En la Figura 16 la actividad Proceso de producto encargado tiene SolicitudoCliente como parámetro de entrada y Pedido como parámetro de salida.

En UML es posible tener varios parámetros de entrada y salida. La herramienta empleada para ilustrar los ejemplos, PlantUML, no soporta los parámetros, sólo se los representa como nodos que están fuera de la actividad (partition) y a lo sumo es posible definir uno solo de entrada y otro de salida.

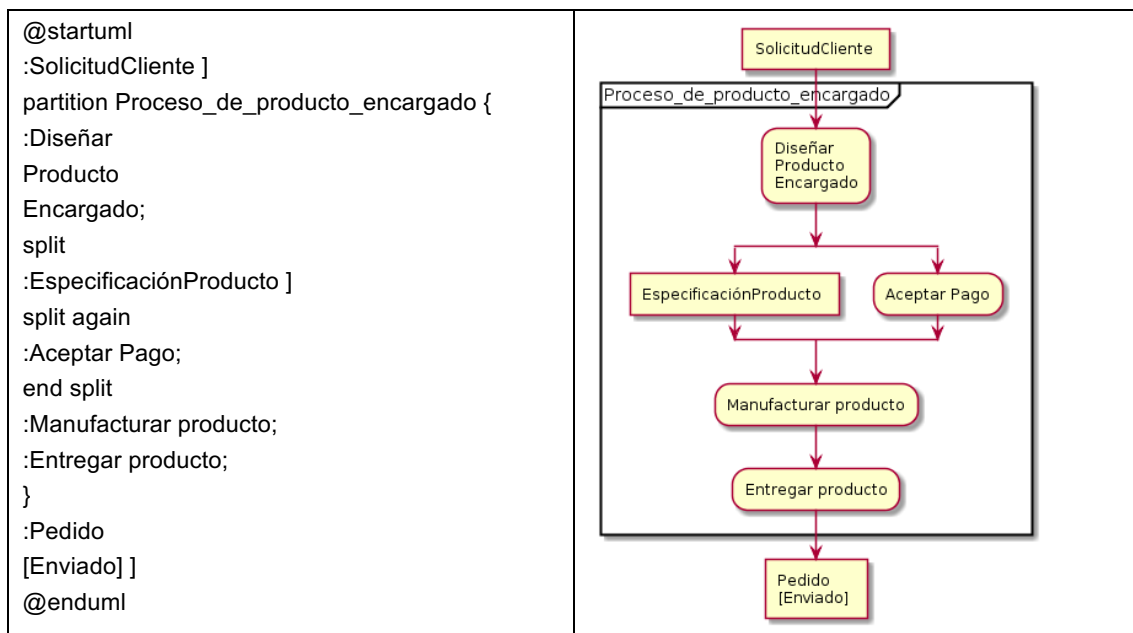


Figura 16. Parámetros de actividades

En la Figura 16 se especifica que:

1. La actividad comienza cuando existe una SolicitudoCliente en el flujo de objeto de entrada de la acción Diseñar producto encargado. Esta acción consume su objeto de entrada y muestra como salida un objeto EspecificaciónProducto.
2. Cuando la acción Aceptar pago recibe un token de control de Diseñar producto encargado se ejecuta.

3. El flujo de control pasa entonces a la acción Manufacturar producto. Esto consume la salida de objeto EspecificaciónProducto que había sido generado por Diseñar producto encargado y ofrece un token de control a Entregar producto.
4. Entregar producto se ejecuta cuando un token de control está disponible desde Manufacturar producto. Esta acción tiene como salida el objeto Pedido en el estado Enviado. Este objeto Pedido es el parámetro de salida de la actividad.

Pins

Una actividad puede llegar a tener un gran número de flujos de objeto, esto puede hacer muy complicada su representación. UML propone el uso de pins para representar estos objetos. Un pin es un nodo de objeto que representa una entrada o salida de una acción. Los pins de entrada tienen exactamente un arco de entrada y los pins de salida tienen exactamente un arco de salida. Aparte de esto, tienen la misma semántica y sintaxis que los nodos de objeto. Sin embargo, puesto que son tan pequeños, se debe escribir toda la información, como el nombre de la clase, fuera del pin, pero tan cerca de éste como se pueda.

La Figura 17 muestra una actividad Conectar que tiene dos flujos de objeto. La actividad comienza con la acción Obtener NombreUsuario. Esto tiene como resultado un objeto NombreUsuario en estado Válido. La siguiente acción es Obtener Contraseña que tiene como resultado un objeto Contraseña en estado Válida. La acción Autenticar usuario se ejecuta cuando recibe un NombreUsuario válido y una Contraseña válida en sus flujos de objeto de entrada.

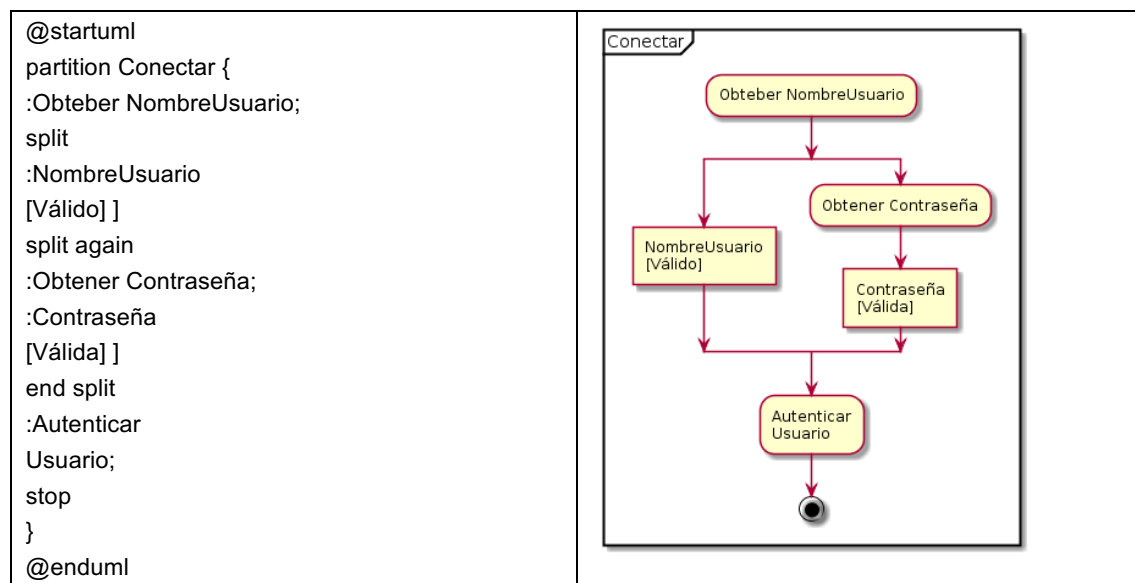


Figura 17. Actividad Conectar

La Figura 18 muestra exactamente la misma actividad, pero dibujada utilizando pins. Como PlantUML no soporta esta notación, se incluye la figura como es presentada en Arlow y Neustadt (2006).

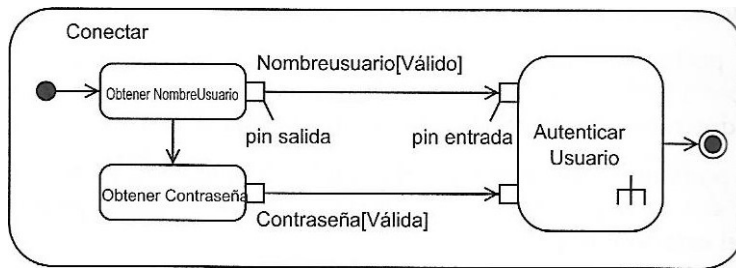


Figura 18. Actividad Conectar con notación de pin.

Enviar señales y aceptar eventos

Una señal representa información que se pasa asincrónicamente entre objetos. Una señal se modela como una clase estereotipada <<signal>>. La información a pasar se alberga en los atributos de la señal. Puede utilizar señales en análisis para mostrar el envío y recepción de eventos asíncronos de negocio (como PedidoRecibido) y puede utilizarlos en diseño para ilustrar comunicación asíncrona entre diferentes sistemas, subsistemas o piezas de hardware. La Figura 19 muestra dos señales que se utilizan en la actividad que se muestra en la Figura 20. Estas dos señales son tipos de EventoSeguridad. La señal EventoPeticiónAutorización posee un PIN y detalles de tarjeta. Estos están probablemente cifrados. El EventoAutorización posee un indicador booleano para indicar si la tarjeta y el PIN se han autorizado.

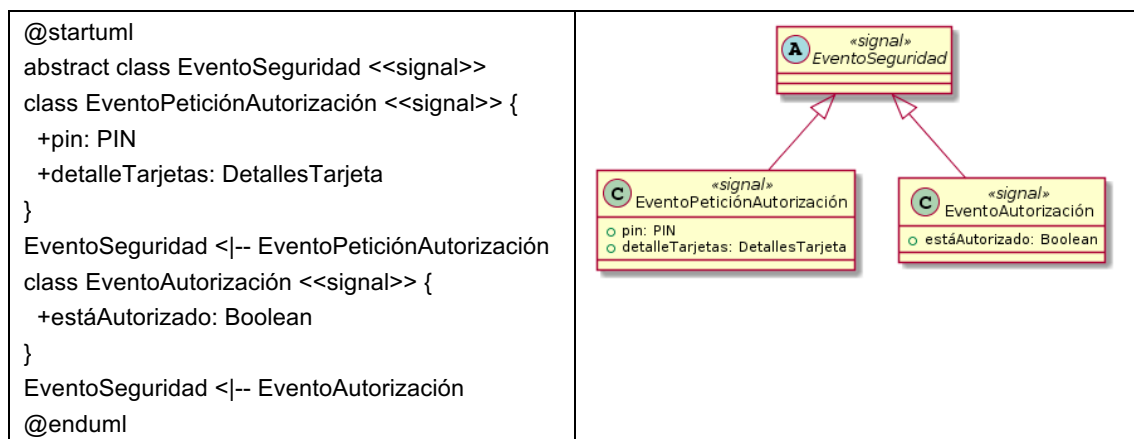


Figura 19. Señales

Puede enviar una señal al utilizar un nodo de acción de enviar señal. Esto envía la señal asincrónicamente; la actividad de envío no espera a la confirmación del recibo de la señal. La semántica de la acción de enviar señal es la siguiente:

- La acción de enviar señal se inicia cuando existe un token simultáneamente en todos los arcos de entrada. Si la señal tiene pins de entrada, debe recibir un objeto de entrada del tipo correcto para cada uno de sus atributos. En la Figura 20 se debería incluir un arco de flujo de objeto entre DetallesTarjeta y EventoPeticiónAutorización.
- Cuando la acción se ejecuta, se construye y envía un objeto de señal. El objeto destino normalmente no se especifica.
- La acción de envío no espera por la confirmación del recibo de señal; es asíncrono.
- La acción termina y los token de control se ofrecen en sus arcos de salida.

Un nodo de acción de aceptar evento tiene cero o un arco de entrada. Espera eventos asíncronos detectados por el contexto y los ofrece en su único arco de salida. Tiene la siguiente semántica:

- La acción de aceptar evento se inicia por un arco de control entrante, y si no tiene arco entrante, se inicia cuando se inicia la actividad que lo posee.
- La acción espera a recibir un evento del tipo especificado. Este evento se conoce como el activador.
- Cuando la acción recibe un activador de evento del tipo correcto, muestra como salida un token que describe el evento. Si el evento era un evento de señal, el token es una señal.
- Si la acción no posee arcos de entrada, esta misma acción continúa para aceptar eventos mientras se ejecuta la actividad.

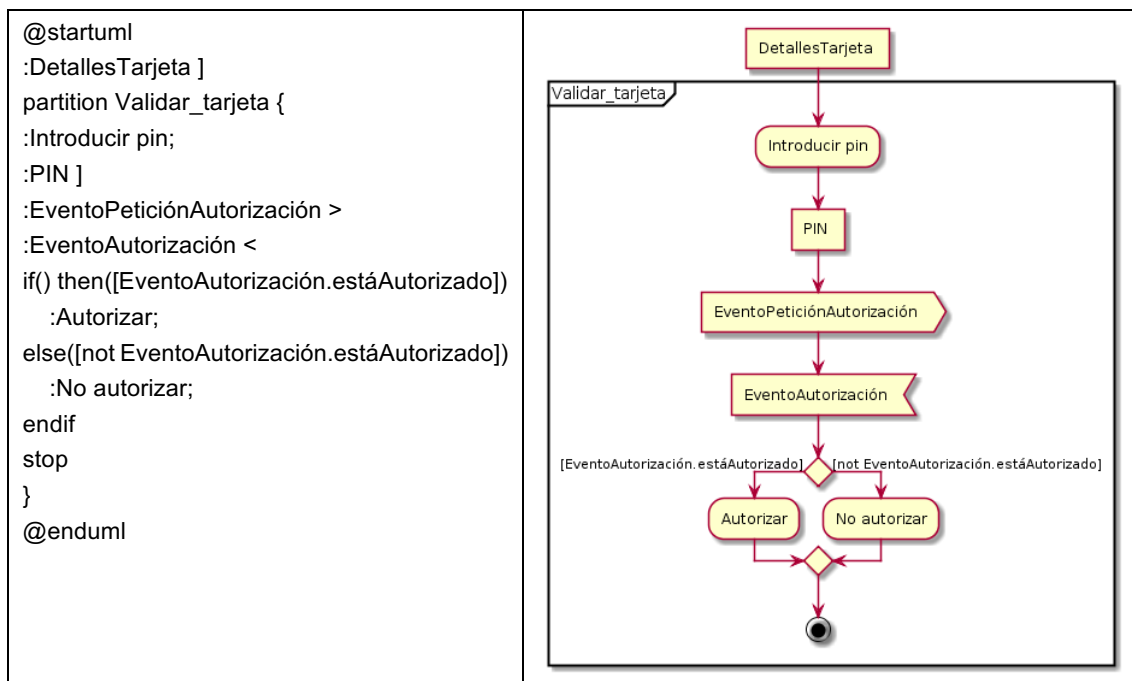


Figura 20. Actividad Validar tarjeta

La Figura 20 muestra una actividad Validar tarjeta que envía EventosPeticiónAutorización y recibe EventosAutorización:

La actividad Validar tarjeta empieza cuando recibe DetallesTarjeta como un parámetro de entrada. Luego le pide al usuario Introducir PIN.

La acción EventoPeticiónAutorización se ejecuta una vez que obtiene el objeto PIN y un objeto DetallesTarjeta en sus arcos de entrada (este último arco no ilustrado en la Figura 20). Construye una señal EventoPeticiónAutorización, utilizando estos parámetros de entrada, y lo envía. Las acciones de enviar señal están representadas por pentágonos convexos según se muestra en la figura.

Los envíos de señal son asíncronos y el flujo de control progresa inmediatamente hacia la acción de aceptar evento EventoAutorización que se representa como un pentágono cóncavo. Esta acción espera hasta recibir una señal EventoAutorización.

Al recibir la señal, el flujo se mueve hasta el nodo de decisión. Si EventoAutorización.estáAutorizado es verdadero, se ejecuta la acción Autorizar, de lo contrario se ejecuta la acción No autorizar.

Aquí tiene otro ejemplo de acciones de aceptar evento. La Figura 21 modela una actividad Mostrar noticias que tiene dos acciones aceptar evento que empiezan automáticamente cuando

se inicia la actividad. Cuando la acción aceptar evento EventoNoticias recibe un EventoNoticias, este evento se pasa a la acción Mostrar noticias.

El control luego fluye hasta un nodo final de flujo y este flujo en particular se termina. Sin embargo, la actividad continúa ejecutándose y ambas acciones de aceptar evento continúan esperando eventos.

Cuando la actividad obtiene un EventoTerminar, el control se mueve a un nodo final de actividad y toda la actividad, incluidas las acciones de aceptar evento, terminan inmediatamente.

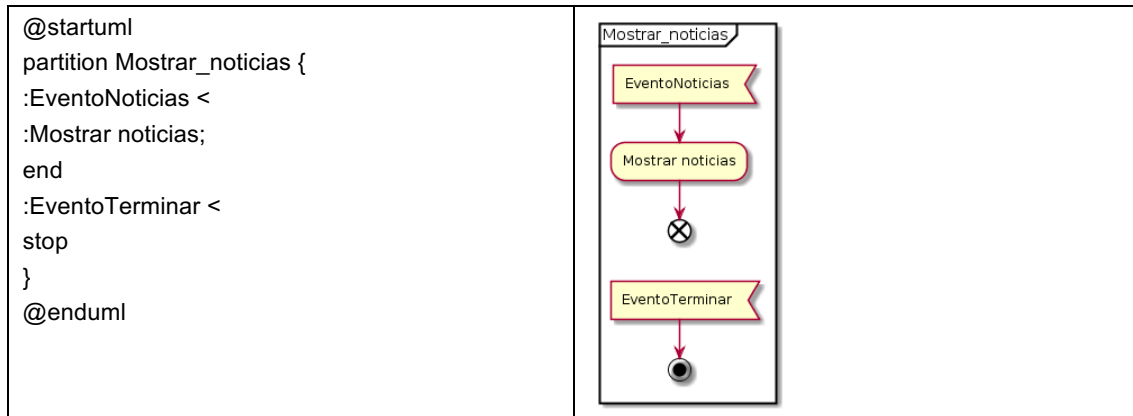


Figura 21. Actividad Mostrar noticias

En caso de que necesitara mostrar cómo las señales se mueven alrededor de diagramas de actividad, puede representarlas como nodos de objeto.

Conectores

Como principio general debería evitar utilizar conectores. Sin embargo, si encontrara una complejidad irreducible, puede utilizar conectores para romper arcos largos que son difíciles de seguir. Esto puede simplificar un diagrama de actividad y hacerlo más sencillo de leer.

La sintaxis del conector se muestra en la Figura 22, donde se los representa por un círculo con su etiqueta en el interior. Para una actividad dada, cada conector de salida debe tener exactamente un conector entrante con la misma etiqueta. Las etiquetas son identificadores para el conector y no tienen otra semántica. A menudo simplemente son letras del alfabeto.

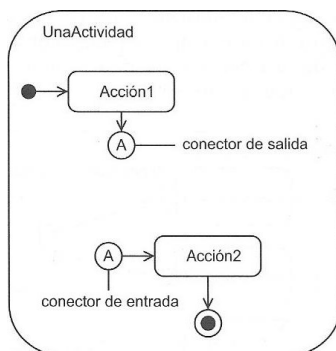


Figura 22. Conectores

Regiones interrumpibles de actividad

Las regiones interrumpibles de actividad son regiones de una actividad que se interrumpen cuando un token alcanza un arco de interrupción. Cuando la región se interrumpe, todo el flujo dentro de la región se aborta inmediatamente. Las regiones interrumpibles de actividad le

proporcionan un medio de utilidad de modelar interrupciones y eventos asíncronos. Se utilizan más a menudo en diseño, pero también se pueden utilizar para mostrar la gestión de eventos asíncronos de negocio. La Figura 23 muestra una sencilla actividad Conectarse que tiene una región interrumpible. La región se muestra como un rectángulo redondeado de puntos que engloba las acciones Obtener NombreUsuario, Obtener Contraseña y Cancelar. Si la acción de aceptar evento Cancelar obtiene el evento Cancelar mientras el control está en la región, muestra un token en el arco de interrupción e interrumpe la región. Las acciones Obtener NombreUsuario, Obtener Contraseña y Cancelar, todas se terminan.

Los arcos de interrupción se dibujan como una flecha de zigzag según se muestra en la Figura 23. Cabe destacar que la mayoría de las herramientas de modelado no soportan la representación de la región interrumpible ni al arco de interrupción.

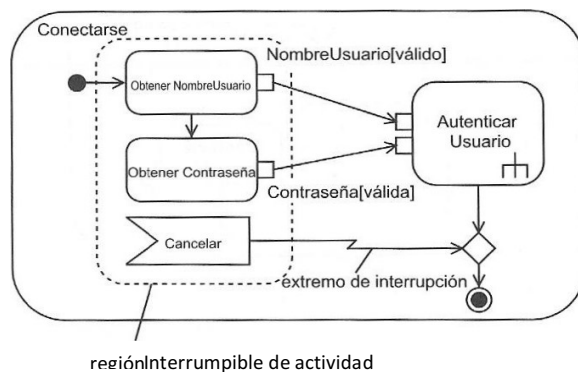


Figura 23. Región interrumpible de actividad

Gestión de excepciones

Los lenguajes informáticos modernos a menudo gestionan los errores por medio de un mecanismo denominado gestión de excepciones. Si se detecta un error en una parte protegida de código, se crea un objeto de excepción y el flujo de control salta hasta un manejador de excepción que procesa el objeto de excepción de alguna forma. El objeto de excepción contiene información sobre el error que se puede utilizar por el manejador de excepción. El manejador de excepción puede terminar la aplicación o tratar de recuperarse. La información en el objeto de excepción a menudo se guarda en un registro de error.

Puede modelar esta gestión de excepción en diagramas de actividad al utilizar pins de excepción (tampoco soportado por las herramientas actuales), nodos protegidos y manejadores de excepción. En la Figura 24 hemos actualizado nuestra actividad Conectarse para hacer que la actividad Autenticar usuario tenga como resultado un objeto RegistrarExcepción si el usuario no se puede autenticar. Este objeto se consume por la acción Registrar error que escribe la información de error en un registro de error. Puede mostrar que un pin representa el resultado de un objeto de excepción al anotarlo con un pequeño triángulo equilátero. El nodo Registrar error actúa como un manejador de excepción que procesa las excepciones generadas por Autenticar usuario. Cuando un nodo tiene un manejador de excepción asociado, se conoce como un nodo protegido. Puesto que la gestión de excepción es normalmente una cuestión de diseño en lugar de análisis, tiende a utilizar nodos protegidos más en diseño que en análisis. Sin embargo, algunas veces puede ser de utilidad modelar un nodo protegido en análisis si tiene semántica importante de negocio.

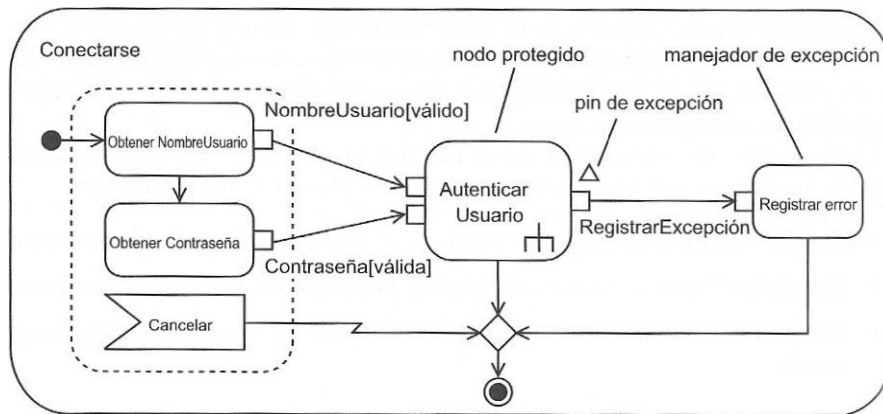


Figura 24. Pin excepción y manejador de excepción

Bibliografía

Arlow, J., I. Neustadt. UML 2.0. Anaya, 2006.

Dennis, A., B. Wixom, D. Tegarden. System Analysis Design. UML Version 2.0. An object oriented approach, third edition, Wiley, 2009.

Murata, T. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 77: 4, 1989.