Capítulo 6. Introducción a la Referencia de BPMN

Esta Sección provee una amplia referencia autosuficiente para modeladores en BPMN. Nuestra hipótesis es que el lector se referirá a esta sección de vez en cuando, por lo tanto, la sección de referencia se organiza conceptualmente, pasando por todos los aspectos de comportamiento de un tipo particular de elemento de BPMN de un modo lógico.

A medida que introducimos cada capítulo, hemos tratado de resaltar los aspectos que interesarán al Analista de Negocio y al Usuario Final (denominados como "Básicos"). Estos se diferencian de los aspectos más complejos que atraerán a aquellos que están buscando ejecutar o simular los procesos (denominados como "Avanzados"). También hemos tratado de destacar las mejores prácticas que ayudarán al modelador evitar modelos incorrectos o confusos.

Recuerden que en este libro nos hemos referido a los elementos gráficos de BPMN con Iníciales Mayúsculas. Cuando se referencia un concepto importante (que no es un elemento gráfico de BPMN), hemos utilizado *cursiva* en la frase.

A lo largo de este libro, utilizamos el concepto de un *"token"* para explicar algunos de los comportamientos subyacentes de un modelo de BPMN. Piense en *tokens* como si estuvieran en movimiento a lo largo del Flujo de Secuencia y pasando por los otros objetos del proceso. Como grupo, estos otros objetos (Eventos, Actividades, y Gateways) se llaman *objetos de flujo*.

Un *token* es un objeto "teórico" que utilizamos para crear una "simulación" descriptiva del comportamiento (no es actualmente una parte formal de la especificación de BPMN). Mediante este mecanismo, la ejecución del proceso (y de sus elementos) se representa mediante la descripción de cómo *tokens* teóricos viajan (o no viajan) por los caminos de Flujo de Secuencia y a través de los *objetos de flujo*.

Un *token* atraviesa Flujos de Secuencia, desde el principio hasta el final (hasta la punta de flecha), de forma instantánea (ver Figura 6-1). Es decir, no hay un tiempo asociado con el *token* viajando por los Flujos de Secuencia.

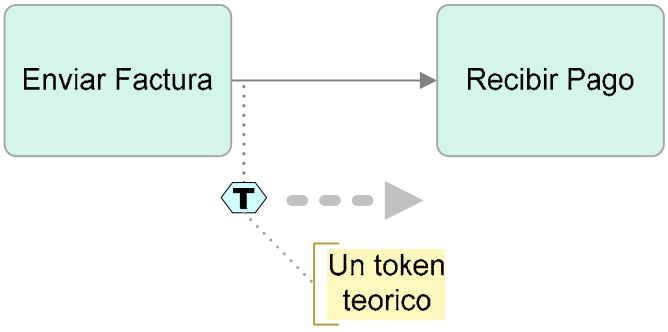


Figura 6-1—Un token viajando por un Flujo de Secuencia

Cuando un *token* llega a un *objeto de flujo*, puede continuar instantáneamente o retrasarse, dependiendo de la naturaleza del *objeto de flujo*. Discutiremos como nuestros *tokens* teóricos interactúan con cada tipo de *objeto de flujo*. El flujo de un *token* entre *objetos de flujo*, mientras operan normalmente, se conoce como *flujo normal*. Sin embargo, ocasionalmente una Actividad no operará normalmente. Puede ser interrumpida por un error u otro Evento, y el flujo resultante se conoce como *flujo de excepción* (ver sección “Interrupción de Actividades mediante Eventos” en la página 96 para más detalle en el *flujo de excepción*).

Capítulo 7. Actividades

Una Actividad representa algo realizado en un Proceso de Negocio. Tiene una forma rectangular con esquinas redondeadas (ver Figura 7-1 and Figura 7-2). Una Actividad tomará normalmente cierto tiempo para ejecutarse, involucrará uno o más recursos de la organización, por lo general requerirá de algún tipo de *entrada*, y producirá algún tipo de *salida*.

Las Actividades son *atómicas* (es decir, son el nivel más bajo de detalle presentado en el diagrama) o son *compuestas* (es decir, no son *atómicas*, en el sentido de que se pueden expandir para ver otro nivel inferior de proceso. El tipo de Actividad *atómica* se conoce como Tarea y puede ser visto en la Figura 7-1.

Figura 7-1—Una Tarea

El tipo compuesto de Actividad se llama Sub-Proceso (ver Figura 7-2). La diferencia gráfica entre una Tarea y un Sub-Proceso es que el Sub-Proceso tiene un "signo de más" colocado en la parte inferior central de la forma, lo que indica se puede abrir para más detalles.

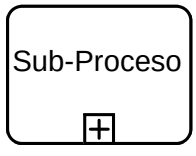


Figura 7-2—Un Sub-Proceso

Dependiendo de la herramienta de modelado de proceso, hacer clic o doble clic puede expandir el diagrama del Sub-Proceso en el lugar o abrir otra ventana. Hacer doble clic en una Tarea podría también traer más información, como la asignación de roles u otros atributos de la Actividad.[16]

Las Actividades pueden ejecutarse una vez, o pueden tener definidos bucles internos. Una Tarea individual con un ícono de bucle (ver Figura 7-3) puede definir condiciones adicionales para que la Tarea se ejecute correctamente, por ejemplo, que se complete una *salida* en un formato adecuado. Más sobre las Actividades Bucle en la página 42.

[16] BPMN uses attributes to store information about the Process (not shown graphically).

Figura 7-3—Una Tarea con un bucle interno

Tareas

Utilice una tarea cuando el detalle del proceso no se descompone aún más, aunque eso no significa que el comportamiento de la tarea no es complejo. En teoría, una tarea siempre puede dividirse en un mayor nivel de detalle. Sin embargo, a los efectos del modelo no se define con mayor detalle.

Hay siete tipos de Tareas especializadas:

- **Simple**—A Una Tarea genérica o indefinida, de uso frecuente durante las primeras etapas del desarrollo del proceso.
- **Manual**—Una Tarea no automatizada que un intérprete humano realiza fuera del control de un motor de workflow o BPM.
- **Recibo**—Espera que le llegue un *mensaje* de un *participante* externo (relacionado al Proceso de Negocio). Una vez recibida la Tarea es completada. Estas son similares en naturaleza a los Eventos de tipo Mensaje de captura (página 103).
- **Script**—Ejecuta un script definido por el modelador.
- **Envío**—Envía un *mensaje* a un *participante* externo. Estas son similares en naturaleza a los Eventos de tipo Mensaje de *lanzamiento* (página 103).
- **Servicio**—Enlaza a algún tipo de servicio, que puede ser un servicio Web o una aplicación automatizada.
- **Usuario**—Una Tarea típica de "flujo de trabajo" donde un intérprete humano lleva a cabo una tarea con la ayuda de una aplicación de software (generalmente programadas mediante un administrador de lista o una Bandeja de Entrada de cierto tipo)

Desde el punto de vista del Analista de Negocio, la única tarea básica es la tipo de Tarea Común. Todos los otros tipos de Tareas son para usos más avanzados de BPMN.

Dependiendo de la herramienta utilizada, expandir una Tarea puede revelar información detallada como la asignación de roles u otros atributos (presentados en un cuadro de dialogo).

Además, diferentes herramientas de modelado de proceso pueden extender BPMN adicionando marcadores gráficos a la Tarea para ayudar a distinguir entre los diferentes tipos de Tareas. Cualquier marcador agregado a una Tarea no debe cambiar la huella de la Tarea (su forma en general) o entrar en conflicto con ningún otro elemento estándar de BPMN. Está es una regla general para extender BPMN.

Mejor Práctica:

***Enviando y Recibiendo Mensajes**—El modelador puede elegir utilizar solo Tareas de Envío y Recibo, o utilizar los Eventos Intermedios de Mensaje de lanzar y capturar. La Mejor Práctica es la de evitar mezclar ambos enfoques en el mismo modelo.*

Ambos enfoques tienen ventajas y desventajas. Los Eventos Intermedios de tipo Mensaje dan el mismo resultado y tienen la ventaja de ser distinguibles gráficamente (mientras que las tareas no lo son). Por otra parte, utilizar Tareas en lugar de Eventos puede permitir al modelador asignar recursos y simular costos.

Sub-Procesos

Un Sub-Proceso representa una Actividad *compuesta*. En este sentido, "compuesta" significa que su trabajo puede dividirse en un nivel más fino de detalle (por ejemplo, otro Proceso). De esta forma, es posible obtener un modelo de Proceso "jerárquico" con diferentes niveles de detalle en cada nivel.

Hay dos representaciones gráficas de los Sub-Procesos:

- **Colapsada**—Esta versión del Sub-Proceso se ve como una Tarea (un rectángulo con esquinas redondeadas) con la adición de un signo de más en la parte central inferior (ver Figura 7-4). Los detalles del Sub-Proceso no son visible en el diagrama.

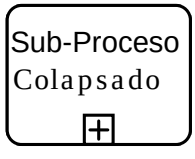


Figura 7-4—Un Sub-Proceso colapsado (los detalles escondidos)

- **Expandida**—Esta versión de la forma del Sub-Proceso es "estirada" y abierta para que los detalles del Sub-Proceso sean visibles dentro de los límites de la forma (ver Figura 7-5). En este caso no hay ningún marcador en la parte inferior central de la forma. Sin embargo, algunas herramientas de modelado de procesos colocan un pequeño signo de menos en la parte inferior central de la forma para indicar que el Sub-Proceso puede ser colapsado. Esto no es parte del estándar BPMN, pero es una extensión válida.

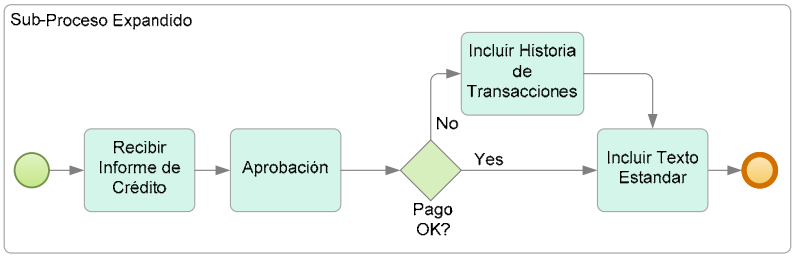


Figura 7-5—Un Sub-Proceso *expandido* (los detalles visibles)

Tipos de Sub-Procesos

Hay dos tipos de Sub-Procesos:

- **Embebidos**—Un Proceso modelado que en realidad es parte del Proceso *padre*. Los Sub-Procesos Embebidos no son reutilizables por otros procesos. Todos los "datos relevantes del proceso" utilizados en el Proceso *padre* son directamente accesibles por el Sub-Proceso *embebido* (porque es parte del *padre*).
- **Reutilizable**—Un Proceso modelado por separado que puede ser utilizado en múltiples contextos (por ejemplo, la comprobación del crédito de un cliente). Los "datos relevantes del proceso", del Proceso padre (que está llamando) no están disponibles automáticamente al Sub-Proceso. Todos los datos deben ser transferidos específicamente, algunas veces reformateados, entre el *padre* y el Sub-Proceso. Tenga en cuenta que el nombre "Reutilizable" se cambió de "Independiente" en BPMN 1.1.

Dependiendo de la herramienta o de la preferencia del modelador, los diagramas de Sub-Procesos pueden expandirse en el lugar o abrir otro diagrama.

La distinción entre embebido y *reutilizable* no es importante para la mayoría de los modeladores. Sin embargo, a medida que las organizaciones desarrollan un gran número modelos de procesos, algunos de los cuales quieren reutilizar, la diferencia se hace más importante—particularmente al considerar como los datos se utilizan a través de los niveles del Proceso. Estas diferencias se vuelven aún más importantes para las organizaciones que buscan utilizar sus procesos en una suite de BPM, o en una Arquitectura Orientada a Servicios.

Sub-Procesos Embebidos

Un Sub-Proceso *embebido* es "parte de" el proceso. Es decir, pertenece solo al Proceso *padre* y no está disponible para ningún otro Proceso. Además, el 'alcance' de un Sub-Proceso *embebido* es 'local' al Proceso *padre* ya que puede utilizar datos que están almacenados con el Proceso *padre* sin la necesidad de mapeo o traducción. Por ejemplo, si datos como "Nombre del Cliente" o "Número de Orden" son parte del Proceso *padre*, entonces actividades dentro del Sub-Proceso *embebido* tendrán ac-

ceso a estos datos directamente (sin ningún mapeo especial entre los elementos).

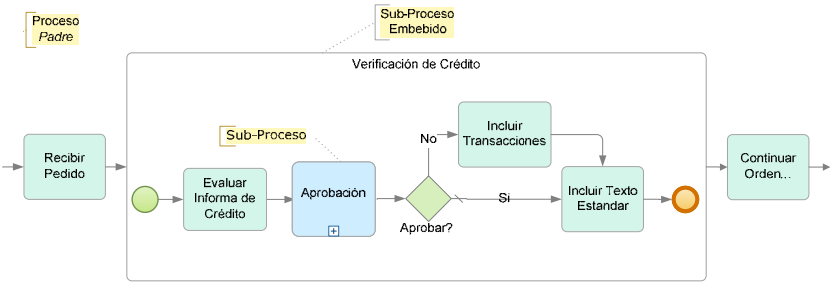


Figura 7-6—Un Sub-Proceso *embebido*

Una característica especial de un Sub-Proceso *embebido* es que solo puede empezar con un Evento de Inicio de tipo Simple—por ejemplo, sin un *disparador* explícito como un mensaje (ver Figura 7-6 arriba). Solo los Procesos de Mayor Nivel pueden hacer uso de los Eventos de Inicio basados en *disparadores*. La razón es que el token que llega del Proceso *padre* actúa como el *disparador* del Sub-Proceso.

Sub-Procesos Reutilizables

Como lo indica su nombre, un Sub-Proceso *reutilizable* puede aparecer en múltiples Procesos *padre*—no es una "parte de" el Proceso cuando es instanciado. El Sub-Proceso *reutilizable* es un conjunto auto contenido de Actividades. Proporciona un mecanismo de referencia de tal manera de que un único Proceso o servicio (en un Arquitectura Orientada a Servicio) esté disponible para cualquier número de procesos que puedan invocarlo.

Estos Sub-Procesos *reutilizables* son "semiindependientes" del Proceso *padre* y pueden aparecer, sin cambios, en varios diagramas. Por ejemplo, la Figura 7-6 arriba, muestra un Sub-Proceso de "Verificación de Crédito" que podría figurar en muchos Procesos en los que el crédito de un cliente necesite ser verificado.

No hay diferencia gráfica entre los Sub-Procesos *embebidos* y *reutilizables*. La diferencia es puramente técnica—las herramientas los manejan de diferentes formas. Sin embargo, esperamos que BPMN 2.0 provea una distinción gráfica entre estos dos tipos de Sub-Procesos.

Al igual que un Sub-Proceso embebido, un Sub-Proceso reutilizable debe tener un Evento de Inicio de tipo Simple. Del mismo modo, el *token* del Proceso *padre* es el disparador para el inicio de cualquier Sub-Proceso.

Como un Sub-Proceso *reutilizable* típicamente proporciona una capacidad bien definida (por ejemplo la verificación de crédito), también podría actuar como un Sub-Proceso de alto nivel. En cuyo caso, podría tener un Evento de Inicio como disparador cargado para este propósito (ver Figura 7-7). Cada vez que se inicia un Proceso con un Evento de Inicio *basado en un disparador* (por ejemplo, por un mensaje), creará un nuevo contex-

to—es decir, es decir un nuevo proceso de alto nivel, no un Sub-Proceso. En esta situación, el Sub-Proceso tendrá por lo menos dos Eventos de Inicio (como un Evento de Inicio de tipo Simple siempre se requiere en un Sub-Proceso)

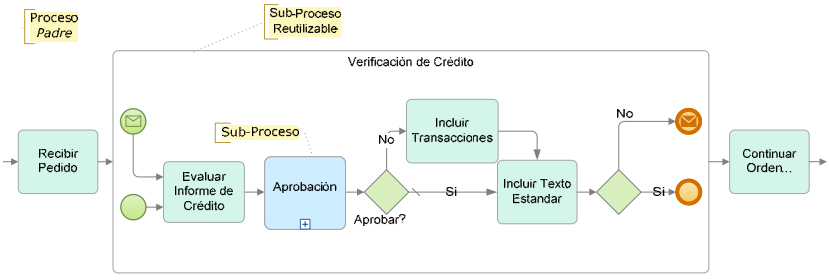


Figura 7-7—Un Sub-Proceso *Reutilizable* también puede ser un Proceso de *Alto Nivel*

Otra característica de un Sub-Proceso *reutilizable* es que los datos utilizados por el Sub-Proceso son totalmente independientes de los datos del Proceso *padre* que lo invoca. La capacidad de reutilización se basa en el hecho de que sus datos son totalmente auto contenidos. Por ejemplo, si el Proceso *padre* almacena y utiliza datos como "nombre del cliente" o "número de orden" y el Sub-Proceso *reutilizable* necesita acceder a estos datos, no puede referenciarlos directamente. Por lo tanto un mapeo es necesario para que el Sub-Proceso *reutilizable* pueda tener sus propias copias.

Esto se debe a que los Sub-Procesos *reutilizables* existen independientemente de cualquier Proceso *padre* en particular, y entonces, los nombres exactos de los elementos de datos pueden no corresponder exactamente. Por ejemplo, un Proceso *padre* puede haber nombrado un elemento de datos "nombre de cliente", y el Sub-Proceso *reutilizable* puede haber utilizado una convención para acortar el nombre a "nom_cli". Transferir datos desde el Proceso *padre* al Proceso *reutilizable* se basará en un "mapeo" entre los elementos de datos de los dos niveles. El mapeo es necesario tanto para la *entrada* como para la *salida* del Sub-Proceso *reutilizable*. El atributo *asignación* de las Actividades se encarga del mapeo de datos de entrada y de salida del Sub-Proceso (y de cualquier Actividad que lo necesite)

Este mapeo no es un elemento gráfico de BPMN, pero es importante cuando se trabaja en ambientes de ejecución o simulación. Si los elementos de datos tienen exactamente los mismos nombres, la herramienta de modelado de procesos puede tener funcionalidades para realizar este mapeo automáticamente. Sin embargo, si los elementos de datos tienen nombres diferentes, entonces el modelador tendrá que elegir los elementos de datos que se mapean (tal vez asistido por la herramienta de modelado)

Conectando Actividades

Como se describe en más detalle en la sección “Flujo de Secuencia” en la página 161, el Flujo de Secuencia conecta *objetos de flujo*, incluyendo Actividades. Cada Actividad puede tener uno o más Flujos de Secuencia *de entrada* y uno o más Flujos de Secuencia *de salida*.

Por lo general, una Actividad tendrá un único Flujo de Secuencia *de entrada* y *de salida* (ver Figura 7-8).

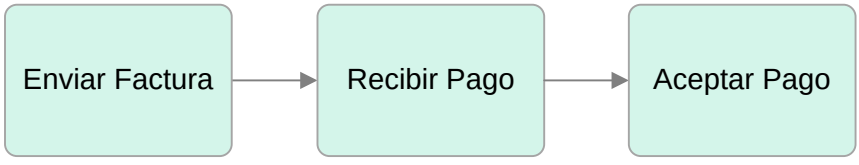


Figura 7-8—El Flujo de Secuencia conecta Tareas

Comportamiento de la Actividad

Como se discutió anteriormente, el flujo del Proceso está determinado por los caminos del Flujo de Secuencia. Cuando un *token* llega a una Actividad, esa Actividad está *lista* para comenzar (ver Figura 7-9). Nos referiremos al comienzo y la ejecución de una Actividad como la *instancia* de una Actividad. Una nueva *instancia* es creada cada vez que un *token* llega a una Actividad.

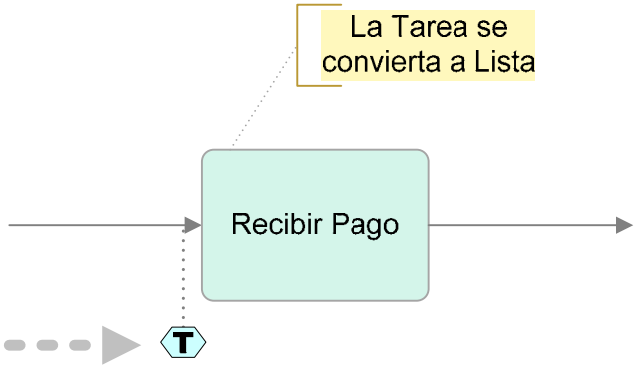


Figura 7-9—Un *token* llegando a una Tarea

La actividad seguirá *lista* (para ejecutarse) hasta que todos los requisitos definidos, como datos *de entrada*, se satisfagan. Luego la Actividad se ejecuta—es decir, se vuelve “*activa*” (ver Figura 7-10).

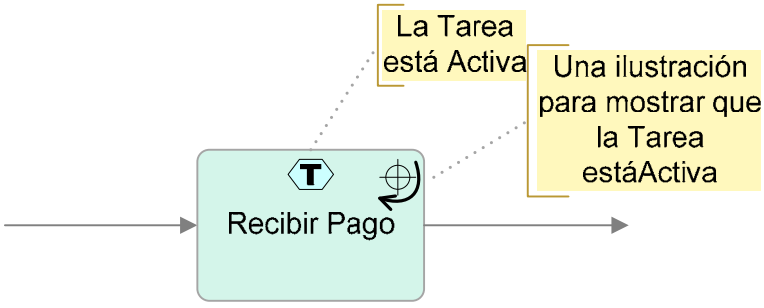


Figura 7-10—Un token dentro de una Tarea Activa (la cruz y la flecha curva dentro de la Actividad indican que está procesando)

Luego de que la instancia de la Actividad finalice, el *token* se mueve hacia el Flujo de Secuencia *de salida*, siguiendo el camino del Proceso (ver Figura 7-11). Para más detalles del ciclo de vida de una Actividad ver la página 173.

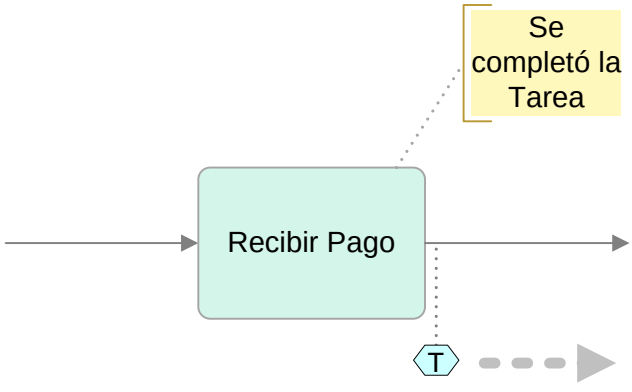


Figura 7-11—Un *token* dejando una Tarea completada

Con Flujos de Secuencia de Entrada Múltiples

Una Actividad puede tener múltiples Flujos de Secuencia *de entrada*. Cada Flujo de Secuencia *de entrada* es independiente de los otros Flujos de Secuencia *de entrada*. Esto significa que cuando llega un *token* de un Flujo de Secuencia de entrada, la Actividad está *lista* para comenzar (ver Figura 7-12). La Actividad no necesita esperar *tokens* de ningún otro Flujo de Secuencia *de entrada*.

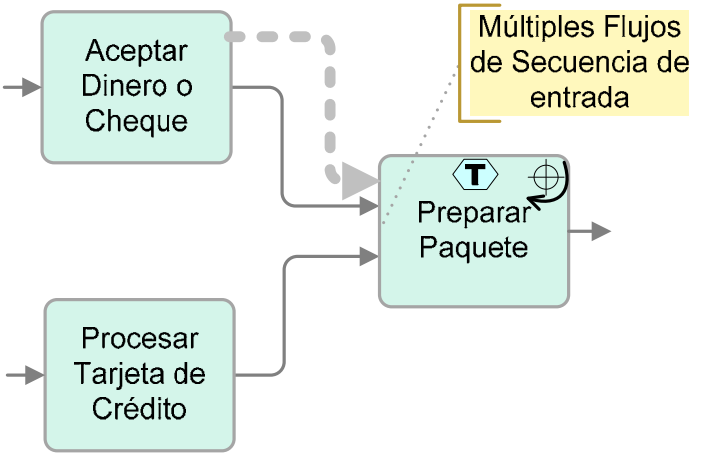


Figura 7-12—Un ejemplo de una Tarea con múltiples Flujos de Secuencia *de entrada*

Si otro *token* llega desde cualquier otro Flujo de Secuencia *de entrada* (ver Figura 7-13), entonces la Actividad queda *lista* para comenzar otra vez. *Instancias* independientes de la Actividad Preparar Paquete se ejecutan para cada *token* que llega a la Actividad. Técnicamente es posible tener dos o más *instancias* de la Actividad ejecutándose al mismo tiempo en el mismo Proceso.

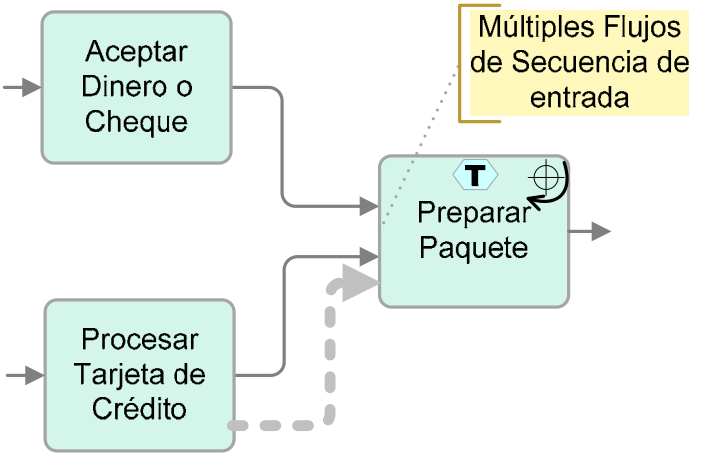


Figura 7-13—Un segundo Flujo de Secuencia *de entrada* generará otra *instancia* de la Actividad Preparar Paquete

Nota: Flujos de Secuencia *de entrada* múltiples para una Actividad representan un flujo incontrolado. Para controlar el flujo (por ejemplo, para que la actividad espere a los otros Flujos de Secuencia *de entrada*), utilice un Gateway Paralelo (ver sección “Gateway Paralelo Unificando” en la página 139 para más información).

Con Flujos de Secuencia de Salida Múltiples

Una Actividad puede tener múltiples Flujos de Secuencia *de salida*. Cada Flujo de Secuencia *de salida* es independiente de los otros Flujos de Secuencia *de salida*. Esto significa que cuando se completa la Actividad un *token* se mueve por cada Flujo de Secuencia *de salida* (ver Figura 7-14). Esto crea un conjunto de *tokens* paralelos. Esto mimetiza el comportamiento que resultaría de utilizar un Gateway Paralelo después de la Actividad (ver la sección titulada "Gateway Paralelo Dividiendo en la página 138).

Con el fin de seleccionar el Flujo de Secuencia *de salida* que obtendrá el *token(s)*, utilice un Gateway para controlar el flujo de los *tokens* (ver la sección "Gateways" en la página 128 para más información). También es posible filtrar la salida de los *tokens* colocando condiciones directamente en los Flujos de Secuencia *de salida* (ver la sección Flujo de Secuencia Condicional en la página 162 para más información).

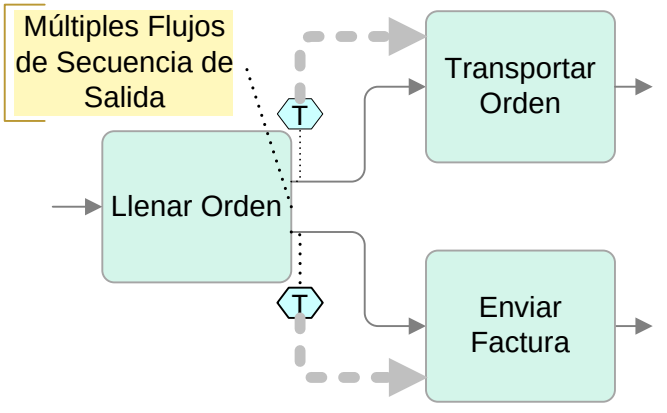


Figura 7-14—Un ejemplo de una Tarea con Flujos de Secuencia *de salida* múltiples

Bucles

Los bucles representan otro tipo de comportamiento de las Actividades. Hay tres formas diferentes de crear bucles en BPMN. Actividades Individuales pueden tener características de bucle como en la Figura 7-15, Flujos de Secuencia pueden modelar el bucle explícitamente como en la Figura 7-16.

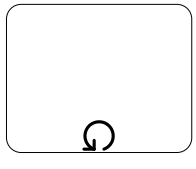


Figura 7-15—Una Actividad con un bucle interno

En una Actividad es posible definir una *condición* de iteración que determina la cantidad de veces que se tiene que ejecutar esa Actividad. Hay dos variaciones para la iteración de las Actividades:

- **Bucle *Mientras*** (o Mientras-Hacer) —se muestra con un símbolo de bucle en la Actividad. La *condición* de iteración se verifica antes de que se ejecute la Actividad. Si la *condición* de iteración resulta ser verdadera, entonces se ejecuta la Actividad. Si no, la Actividad finaliza y el Proceso continua (un *token* se mueve por el Flujo de Secuencia *de salida*), incluso si la Actividad no se ejecuto nunca. Luego de que se ejecute la Actividad, la Actividad itera hacia atrás a verificar la *condición* de iteración nuevamente. El ciclo de verificar la *condición* de iteración y de ejecutar la Actividad continua hasta que la *condición* de iteración es Falsa.
- **Bucle *Hasta*** (o Hacer-Mientras) —también se muestra con el mismo símbolo de bucle en la Actividad. La *condición* de iteración se verifica luego de que la Actividad se ejecuta. Si la *condición* resulta ser verdadera, entonces la Actividad se ejecuta de nuevo. Si no, la Actividad finaliza y el Proceso continúa (un *token* se mueve por el Flujo de Secuencia *de salida*). El ciclo de verificar la *condición* de iteración y de ejecutar la Actividad continua hasta que la *condición* de iteración es Falsa.

Utilizando los atributos de las Actividades es posible establecer la máxima cantidad de iteraciones (*máximo de iteraciones*) para los bucles *hasta* y *mientras*. Luego de que la Actividad alcanzó el *máximo de iteraciones*, esta se detendrá (incluso si la *condición* de iteración aún es verdadera). La cantidad de veces que se ejecuta la Actividad se almacena en un atributo de *conteo de iteraciones* que se incrementa automáticamente en cada iteración. Algunas herramientas de modelado no soportan actualmente este tipo de funcionalidad.

La Figura 7-16 muestra una forma visualmente evidente de crear bucles utilizando Flujos de Secuencia para conectarse a un *objeto de flujo anterior*. Este tipo de bucle debe incluir un Gateway u ocurrirá un bucle infinito. El Gateway verifica la *condición* en la Flujo de Secuencia *de salida* para determinar si repetir la iteración.

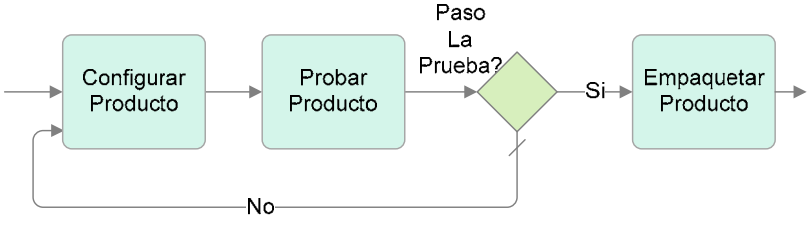


Figura 7-16—Un bucle mediante Flujo de Secuencia

Actividades Multi-Instancia

Un mecanismo sutilmente diferente se necesita para representar el comportamiento donde una Actividad debe ejecutarse varias veces con diferentes conjuntos de datos. Por ejemplo, cuando una gran empresa está comprobando los resultados financieros de todas las subsidiarias, es necesario llevar esto a cabo muchas veces.

Representada gráficamente con tres líneas verticales en la Actividad, la Actividad Multi-Instancia (o ParaCada) soporta este comportamiento. El punto clave a comprender aquí es que la Actividad no itera sobre sí misma; cada Ejecución de la Actividad es distinta de las otras (a pesar de que son parte del mismo Proceso).

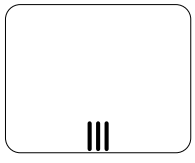


Figura 7-17—Una Actividad Multi-Instancia

El valor del atributo de *condición* de iteración determina la cantidad de veces que la Actividad se ejecuta. Se verifica al inicio de la Actividad y luego la Actividad se "clona" esa cantidad de veces. La *condición* debe resolverse a un entero.

Las *instancias* individuales de una Actividad Multi-Instancia deben ocurrir en *secuencia* o en *paralelo*. Se dispone de atributos para controlar este comportamiento. Cuando este atributo se establece en "Paralelo", se dispone de un atributo adicional para controlar como estas *instancias* se combinan nuevamente. Las opciones (Ninguno, Uno, Todos o Complejo) son equivalentes a utilizar Gateways para controlar los hilos de ejecución paralelos de un Proceso.

En BPMN 1,1, el marcador de Multi-Instancia cambió de dos líneas verticales a tres líneas verticales. Las dos líneas verticales eran usualmente malinterpretadas como una condición de espera (se parecía demasiado a un símbolo de "pausa" en un reproductor de casete o CD)

Niveles de Proceso

En BPMN, es posible desarrollar estructuras jerárquicas para los Procesos a través del uso de uno o más niveles de Sub-Procesos. En este libro, nos referimos a un Proceso que contiene a un Sub-Proceso como el Proceso *padre* del Sub-Proceso. A la inversa, el Sub-Proceso es el *hijo* del Proceso que lo contiene.

Por supuesto, los modeladores pueden desear incluir un Sub-Proceso dentro de otro Sub-Proceso, creando tantos niveles como se necesite. Cada nivel es un Proceso completo. La Figura 7-18 proporciona un ejemplo de un Sub-Proceso incluido dentro de un Proceso *padre*.

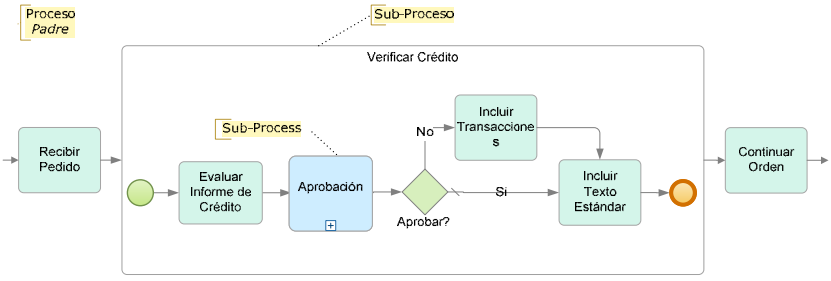


Figura 7-18—Un Proceso con Sub-Procesos: Mostrando tres niveles de Proceso

El Proceso que se muestra en la Figura 7-18 arriba tiene tres niveles de Proceso—la Actividad "Aprobación" en el Sub-Proceso "Verificar Crédito" es un Sub-Proceso en sí (*colapsado*)

Proceso de Alto Nivel

Cualquier Proceso que no tiene un Proceso *padre* se considera un Proceso de *alto nivel*—es decir, un Proceso que no es un Sub-Proceso es un Proceso de *alto nivel* (ver Figura 7-19).

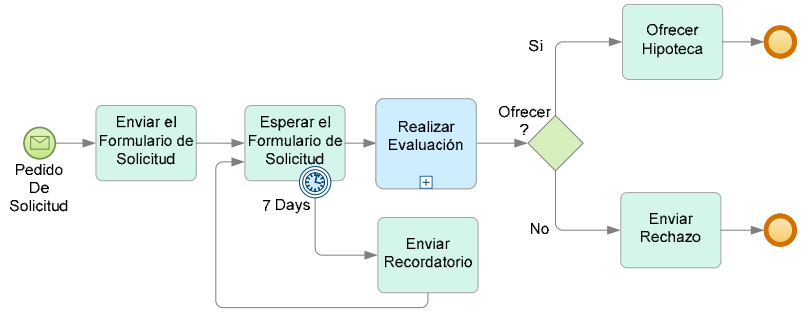


Figura 7-19—Un Proceso de *Alto Nivel*

La suposición es que un Proceso de alto nivel se dispara por cierto estímulo externo—es decir por fuera del alcance del Proceso. El disparador no siempre se modela y se utiliza un Evento de Inicio de tipo Simple. En otros casos, el Evento de Inicio mostrará el tipo de disparador utilizado para iniciar el Proceso. Por ejemplo, un Evento de Inicio de tipo Mensaje o un Evento de Inicio de tipo Temporizador (esta en la Figura 7-19, arriba).

Comportamiento Entre Niveles de Proceso

Para entender la forma en que interactúan los niveles de Proceso (entre los niveles) utilizamos *tokens* nuevamente. Cuando un token del Proceso *padre* llega a un Sub-Proceso (ver Figura 7-20), se invoca el Evento de Inicio de ese Sub-Proceso. Sin embargo, tenga en cuenta que el Flujo de Secuencia del Proceso *padre* no se extiende al Sub-Proceso. El Flujo de Secuencia del Proceso *padre* se conecta a los límites del Sub-Proceso y

un conjunto diferente de Flujos de Secuencia y Actividades (un Proceso diferente) está contenido en el Sub-Proceso.

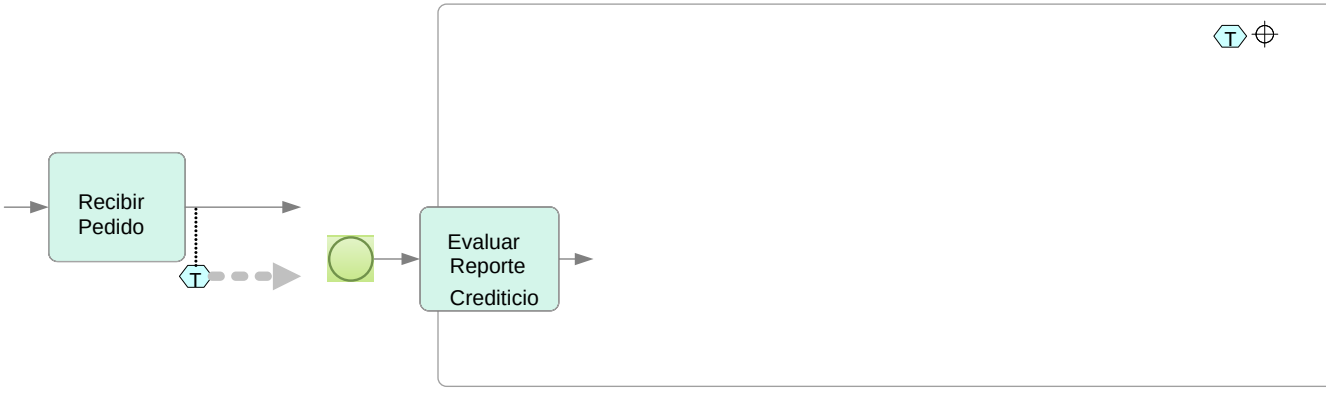


Figura 7-20—Un *token* de un Proceso *padre* llega a un Sub-Proceso (el resto del Sub-Proceso no está visible)

El del *Proceso padre* ahora reside en el Sub-Proceso (ver Figura 7-21). La presencia del *token* del *nivel del padre* en el Sub-Proceso dispara el Evento de Inicio, con lo que se inicia el trabajo del Sub-Proceso (el Sub-Proceso está ahora ejecutándose—esto se representa en la figura con el símbolo al lado del *token*). Un *token* del Proceso hijo es creado por el Evento de Inicio y este *token* se mueve hacia el Flujo de Secuencia *de salida* del Evento de Inicio del Sub-Proceso.

Figura 7-21—El Evento de Inicio del Sub-Proceso se dispara creando un *token* de menor nivel

Cuando el token del nivel del hijo alcanza al Evento de Fin del Sub-Proceso, el Evento de Fin se dispara (ver Figura 7-22). Esto significa que

el Sub-Proceso culmino su trabajo (es decir, terminaron todas las Actividades del Sub-Proceso) y el *token* de *menor nivel* se consume.

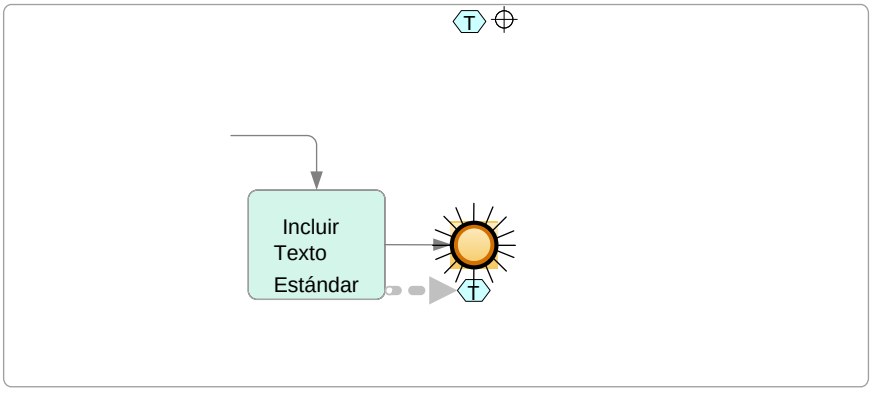


Figura 7-22—El *token* del Sub-Proceso llega al Evento de Fin

Sin embargo, un Sub-Proceso sigue estando activo hasta que todos los *hilos de ejecución* finalicen. En un ambiente de ejecución (Suite de BPM), el sistema puede rastrear cualquier trabajo pendiente de un par de maneras—ya sea rastreando los *tokens* o los estados de las Actividades dentro del sistema.

De cualquier manera, cuando el Sub-Proceso finaliza, el *token del nivel del padre* se mueve hacia el Flujo de Secuencia *de salida* del Sub-Proceso (ver Figura 7-23), continuando el flujo del Proceso *padre*.

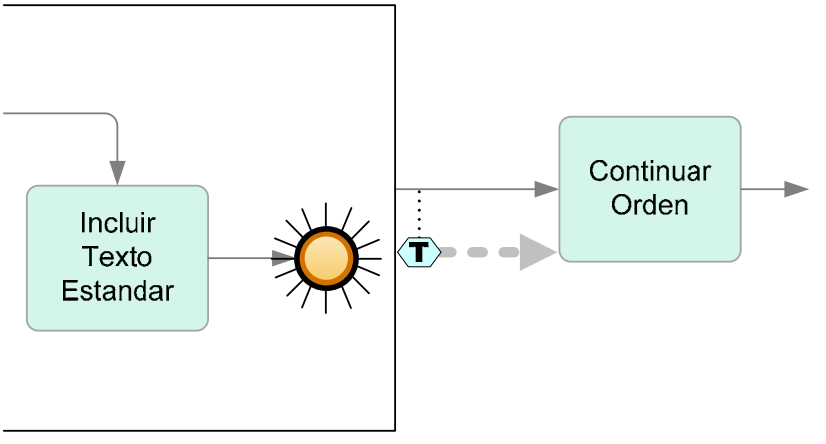


Figura 7-23—El Proceso *padre* continua luego de que el Sub-Proceso finaliza

Recuerde que aunque el Flujo de Secuencia no cruza los límites del Proceso (Pools o Sub-Procesos), los Flujos de Mensaje y las Asociaciones si pueden cruzar estos límites.

Conectando Sub-Procesos

Ya hemos hablado de como se conectan las Actividades (ver "Conectando Actividades" en la página 70). Como los Sub-Procesos son un tipo de actividad, entonces se aplican las mismas reglas de conexión. Sin embargo, existen más opciones cuando se conecta un Sub-Proceso *expandido* que cuando se conecta una Tarea.

Hasta ahora, hemos estudiado el mecanismo típico—conectando el Flujo de Secuencia a los límites del Sub-Proceso (ver Figura 7-24). El Evento de Inicio del Sub-Proceso está dentro de los límites y no está conectado al Flujo de Secuencia del Proceso *del nivel del padre*.

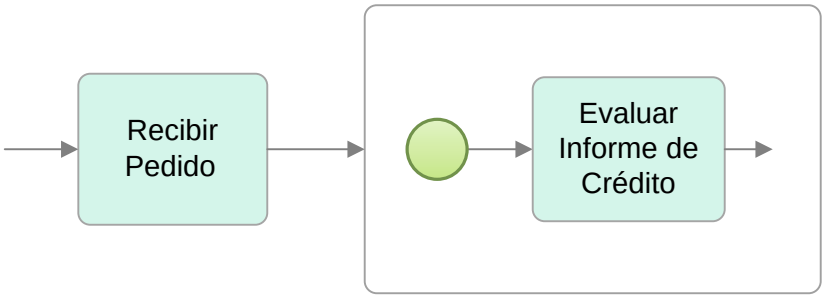


Figura 7-24—El Evento de Inicio está dentro del Sub-Proceso

Sin embargo, para proporcionar más claridad en la interacción del Proceso entre niveles, es posible ubicar el Evento de Inicio del Sub-Proceso en su límite y conectar el Flujo de Secuencia *de entrada* al Evento de Inicio (ver Figura 7-25). Esta capacidad es solo una conveniencia gráfica; de otra forma un Evento de Inicio no se ubicaría en los límites del Sub-Proceso. Teniendo en cuenta esto, el comportamieto de las dos versiones (Figura 7-24 y Figura 7-25) es exactamente el mismo.

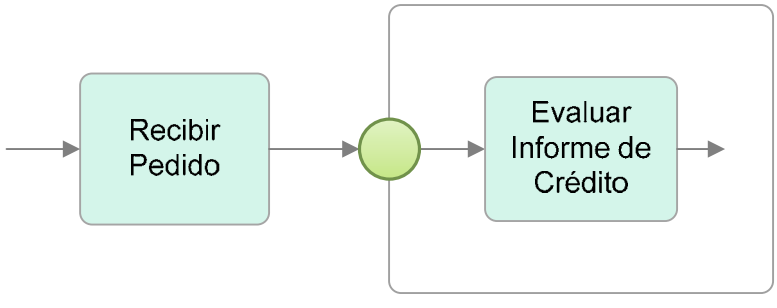


Figura 7-25—El Evento de Inicio en el límite

Sin embargo, si la relación entre el Proceso *padre* y el Sub-Proceso es más compleja, entonces la ubicación del Evento de Inicio en el límite del Sub-Proceso puede ayudar a clarificar el flujo entre el Flujo de Secuencia *de entrada* (al Sub-Proceso) y los Eventos de Inicio de ese Sub-Proceso.

En la Figura 7-26, el camino de un Flujo de Secuencia *de entrada* está dirigido a uno de los Eventos de Inicio del Sub-Proceso y el otro Flujo de Secuencia *de entrada* esta dirigido al otro Evento de Inicio.

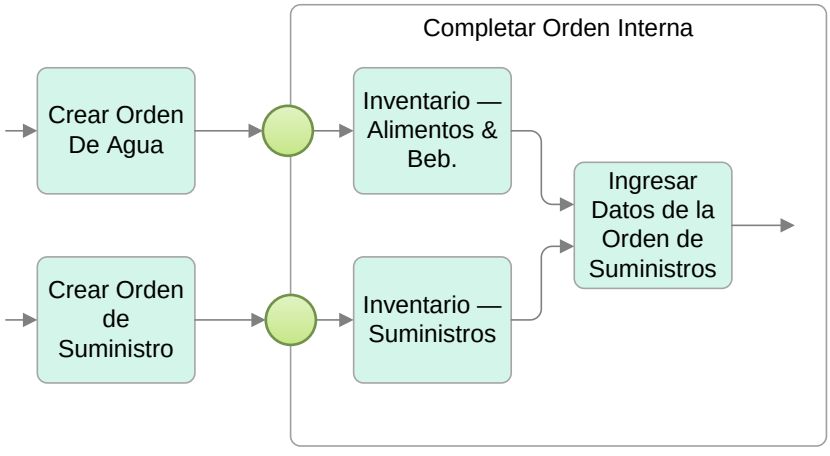


Figura 7-26—Múltiples caminos entran a un Sub-Proceso guiados por Eventos de Inicio en los límites

Esto significa que cuando un *token* llega desde la Tarea “Crear Orden de Agua” el Evento de arriba del Sub-Proceso se disparará (ver Figura 7-27).

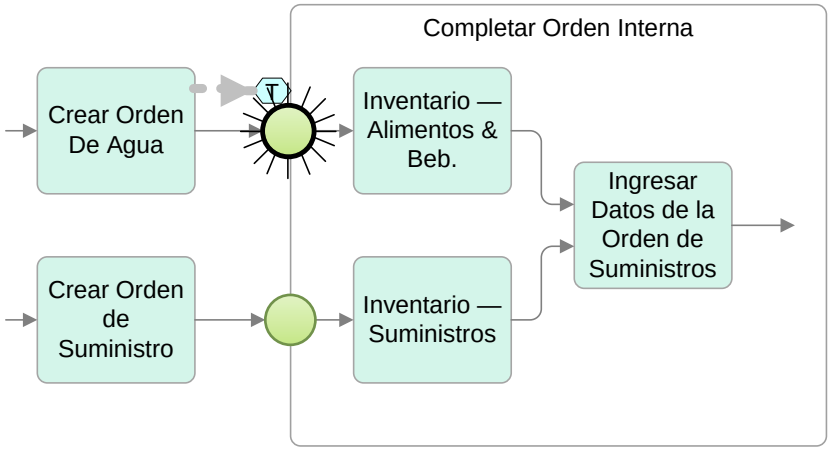


Figura 7-27—Múltiples caminos entran a un Sub-Proceso guiados por Eventos de Inicio en los límites

Sin embargo, cuando llega un *token* desde la Tarea “Crear Orden de Suministro” el Evento de abajo del Sub-Proceso se disparará (ver Figura 7-28).

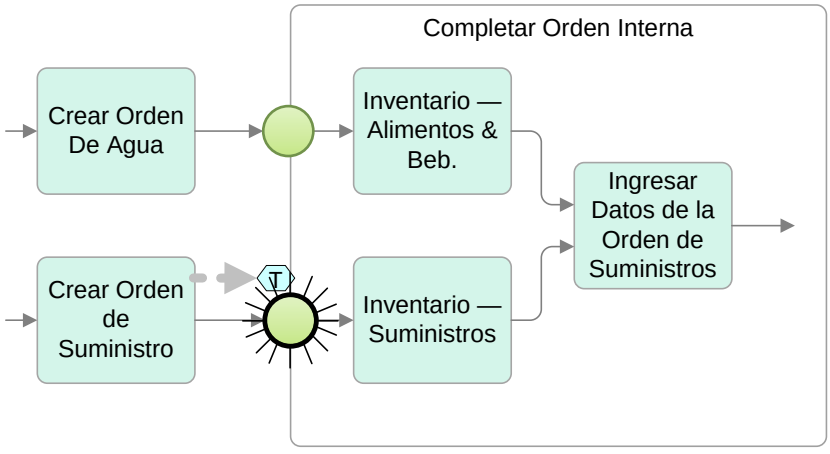


Figura 7-28—Múltiples caminos entran a un Sub-Proceso guiados por Eventos de Inicio en los límites

Tenga en cuenta que cada vez que se dispara un Evento de Inicio, se crea una nueva *instancia* del Sub-Proceso. Es decir, si existen una "Orden de Agua" y una "Orden de Suministro", entonces habrá dos *instancias* independientes del Sub-Proceso para cada orden. Dependiendo de la estructura *anterior* del Proceso, las dos *instancias* del Sub-Proceso podrían operar en paralelo como parte de una única *instancia* del Proceso padre. Alternativamente, podrían ser parte de dos *instancias* independientes del Proceso *padre* (si se hubiese utilizado un Gateway Exclusivo *anteriormente*)

En BPMN 1.1, existen algunos cambios técnicos para permitir que una herramienta de modelado soporte mejor estas conexiones lógicas (entre el Flujo de Secuencia del *padre* y los Eventos de Inicio del Sub-Proceso).

Capítulo 8. Eventos

Un Evento es algo que "sucede" durante el curso de un Proceso. Estos Eventos afectan el flujo del Proceso y usualmente tienen un *disparador* o un *resultado*. Pueden iniciar, retrasar, interrumpir, o finalizar el flujo del Proceso.

Representado por círculos, el estilo del borde (línea única, línea doble, línea gruesa) indica el tipo. Los tres tipos de Eventos son:

- Evento de Inicio (línea fina única)
- Evento Intermedio (línea fina doble)
- Evento de Fin (línea gruesa única)

Las próximas tres secciones exploran estos tipos de Eventos, introduciendo las diferentes opciones disponibles.

Eventos de Inicio

Un Evento de Inicio muestra donde empieza un proceso. Un Evento de Inicio es un pequeño círculo abierto, con una única línea fina como límite (ver Figura 8-1).

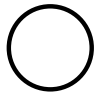


Figura 8-1—Un Evento de Inicio

Hay diferentes tipos de Eventos de Inicio para indicar las diferentes circunstancias que pueden disparar el inicio de un Proceso. De hecho, estas circunstancias, como la llegada de un *mensaje* o un temporizador que se consumió, se llaman *disparadores*. Todos los Eventos de BPMN fueron diseñados para tener el centro abierto de tal forma que los marcadores para los diferentes tipos de *disparadores* de Eventos pudieran aparecer dentro de la forma del Evento. Un *disparador* no es necesario—esos detalles se pueden ocultar o añadir más tarde.

Hay seis tipos de Eventos de Inicio, cada uno con su propia representación gráfica. Los Eventos están divididos entre *básicos* y *avanzados*:

Eventos de Inicio Básicos (ver Figura 8-2):

- **Simple**—No se define ningún disparador.
- **Temporizador**—El *disparador* son una fecha y hora específicos, o un intervalo de tiempo regular (por ejemplo, el primer Viernes de cada mes a las 8am).
- **Mensaje**—El disparador es un *mensaje* que llega desde otra entidad de negocio o rol (*participante*). Por ejemplo, un cliente pide una verificación de su cuenta.
- **Señal**—El *disparador* es una *señal* difundida desde otro proceso. Por ejemplo, un Proceso difunde un cambio en la Tasa de Interés,

disparando cierta cantidad de procesos a iniciarse como resultado.[17]

Figura 8-2—Eventos de Inicio de tipo Básico

Eventos de Inicio de tipo *Avanzado* (ver Figura 8-3):

- **Condicional**—El *disparador* es una *expresión de condición* que debe ser satisfecha para que empiece el Proceso.
- **Múltiple**—Define uno o más disparadores que puede ser cualquier combinación de *mensajes, temporizadores, condiciones o señales* (cualquiera de los cuales inicia un Proceso).

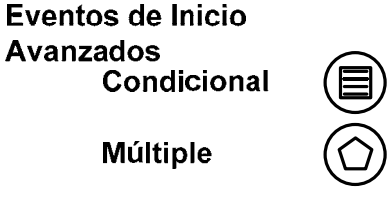


Figura 8-3—Eventos de Inicio de tipo *Avanzado*

Las herramientas de modelado también utilizan diferentes tipos de atributos estándar para registrar los detalles de cada tipo de Evento (no siempre visibles a nivel de diagrama).

Conectando Eventos de Inicio

Un punto clave a recordar sobre los Eventos de Inicio es que solo tienen Flujos de Secuencia *de salida*. No está permitido que los Flujos de Secuencia se conecten a un Evento de Inicio (dado que un Evento de Inicio representa el inicio de un Proceso). La Figura 8-4 muestra un uso inco-

[17] Hemos incluido el uso de *señales* en el conjunto *básico* de Eventos debido a su utilidad. Comprender como modelar usando estas características es considerado esencial para que un Analista de Negocios represente muchos de los comportamientos deseados.

rrecto de un Evento de Inicio (es decir, tiene un Flujo de Secuencia *de entrada*)

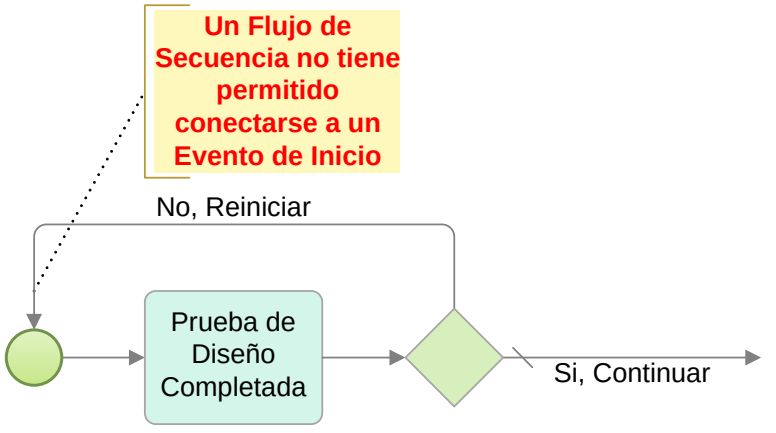


Figura 8-4—Un uso incorrecto de un Evento de Inicio

La versión correcta del fragmento de Proceso que se muestra en la Figura 8-4 tendría el Flujo de Secuencia conectándose de nuevo a la primer Tarea (ver Figura 8-5)

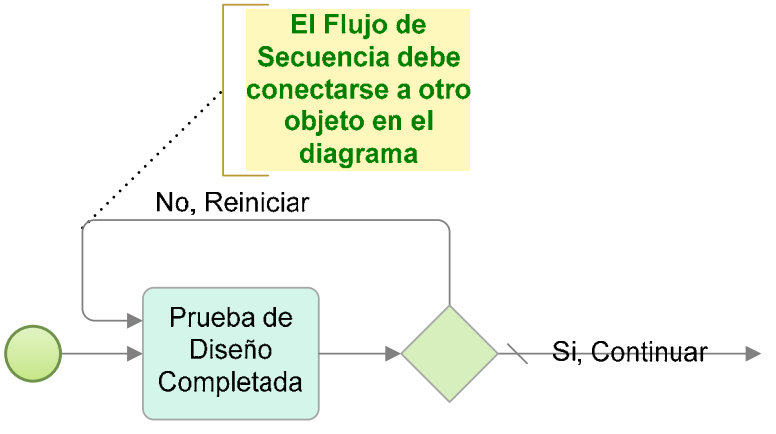


Figura 8-5—Una versión correcta de la Figura 8-4

En este caso, a pesar de la etiqueta del Flujo de Secuencia que itera hacia atrás, no se reinicia todo el proceso; este itera hacia la primer Actividad.

La única manera de reiniciar un Proceso (en el Evento de Inicio) es finalizar el Proceso y luego disparar nuevamente el Evento de Inicio. Esto crearía una nueva instancia del Proceso.

Comportamiento del Evento de Inicio

Los Eventos de Inicio son donde el flujo de un Proceso comienza, y por lo tanto, donde se crean los *tokens*. Cuando se dispara un Evento de Inicio, se genera un *token* (ver Figura 8-6).[18]

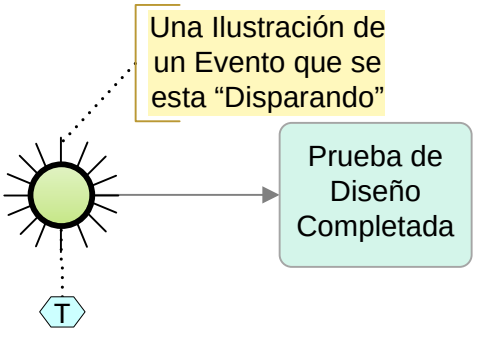


Figura 8-6—Un *token* es generado cuando se dispara un Evento de Inicio

Inmediatamente después de que se dispara el Evento de Inicio y se genera el *token*, el *token* sale del Evento de Inicio y viaja a través del Flujo de Secuencia *de salida* (ver Figura 8-7).

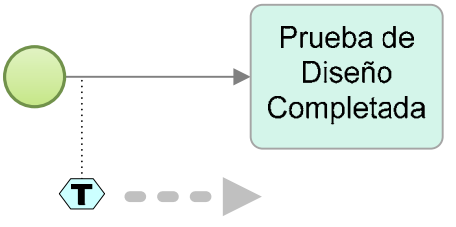


Figura 8-7—El *token* sale del Evento de Inicio a través del Flujo de Secuencia

El Evento de Inicio Básico

Un Evento de Inicio sin un disparador se conoce como un Evento de Inicio de tipo Simple. Se utiliza donde el inicio de un Proceso no está definido. Dado que no hay un disparador definido, no hay un marcador en el centro de la forma (ver Figura 8-8). Además, un Evento de Inicio de tipo Simple siempre se utiliza para marcar el inicio de un Sub-Proceso (pasando de un nivel al siguiente)

[18] Las líneas que muestran radios salientes desde el Evento de Inicio son a efectos ilustrativos y no son parte de la especificación BPMN.

Figura 8-8—Un Evento de Inicio Básico

Evento de Inicio Temporizador

Gráficamente, el Evento de Inicio de tipo Temporizador utiliza un reloj como marcador dentro de la forma del Evento (ver Figura 8-9). Este indica que el Proceso inicia (es decir, se dispara) cuando una condición específica de tiempo ocurre. Esto puede ser una fecha y una hora específica (por ejemplo, El 1 de Enero de 2009 a las 8am) o un intervalo de tiempo recurrente (por ejemplo, cada Lunes a las 8am).

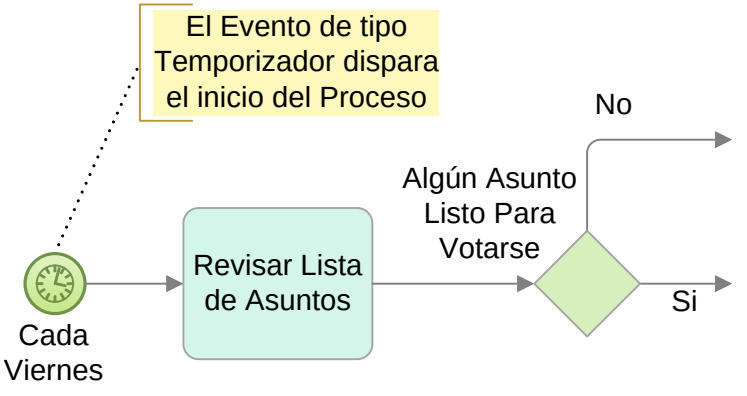


Figura 8-9—Un Evento de Inicio de tipo Mensaje utilizado para Iniciar un Proceso

La Figura 8-9, arriba, provee un ejemplo de un Proceso que se inicia todos los Viernes para revisar un conjunto de asuntos que pueden ser votados por un grupo.

Evento de Inicio Mensaje

El Evento de Inicio de tipo Mensaje representa una situación donde se inicia un Proceso (es decir, se dispara) por la recepción de un *mensaje*. El tipo de Evento tiene un marcador en forma de sobre (ver Figura 8-10).

Un *mensaje* es una comunicación directa entre dos *participantes* del negocio. Estos *participantes* deben estar en Pools independientes (es decir, no pueden ser enviados desde otro Carril en un único Pool)

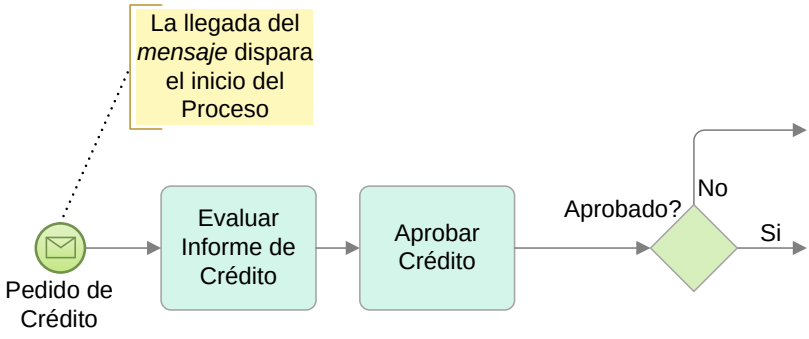


Figura 8-10—Un Evento de Inicio de tipo Mensaje utilizado para iniciar un Proceso

La Figura 8-10, arriba, provee un ejemplo donde un *mensaje* "Pedido de Crédito" dispara el inicio de un Proceso que evalúa y aprueba (o no) el crédito del solicitante.

Eventos de Inicio Señal

Los Eventos de Inicio de tipo señal utilizan un triangulo como marcador dentro de la forma del Evento (ver Figura 8-11). Este indica que el proceso se inicia (es decir, se dispara) cuando se detecta una *señal*. Esta *señal* fue una comunicación difundida desde un *participante* de negocio u otro Proceso. Las *señales* no tienen un objetivo o destinatario específico—es decir, todos los Procesos y *participantes* pueden ver la *señal* y es decisión de cada uno si reaccionar o no. Una señal es análoga a una bengala o una sirena; cualquiera que vea la bengala o la sirena puede, o no, reaccionar.

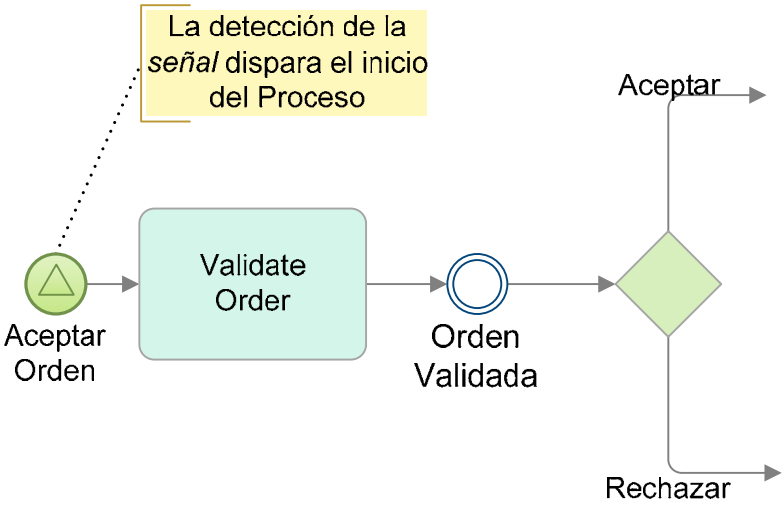


Figura 8-11—Un Evento de Inicio de tipo Señal utilizado para Iniciar un Proceso

A diferencia de los *mensajes*, las *señales* pueden operar dentro de un Proceso (tal vez entre un Sub-Proceso y su *padre* que lo llama), o entre Procesos de diferentes *participantes*. La Figura 8-11 arriba provee un ejemplo donde una *señal* "Aceptar Orden" dispara el inicio de un Proceso que va a evaluar y aceptar (o no) la orden para procesar.

BPMN 1.1 adiciona el Evento de Inicio de tipo Señal y elimina el "Evento de Inicio de tipo Vínculo". Las señales proporcionan una forma más general de comunicación entre procesos.

Eventos de Inicio Condicional

Los Eventos de Inicio de tipo Condicional representan una situación donde un Proceso se inicia (es decir, se dispara) cuando una *condición* predefinida se vuelve *verdadera*. Este tipo de Evento tiene como marcador un papel con reglones (ver Figura 8-12). Este tipo de Evento suele dispararse por algún cambio en los "datos relevantes del proceso", como en un escenario bancario, cuando el saldo de los clientes cae por debajo de determinado umbral. Una *condición* se utiliza para definir los detalles del cambio en los datos que se espera.

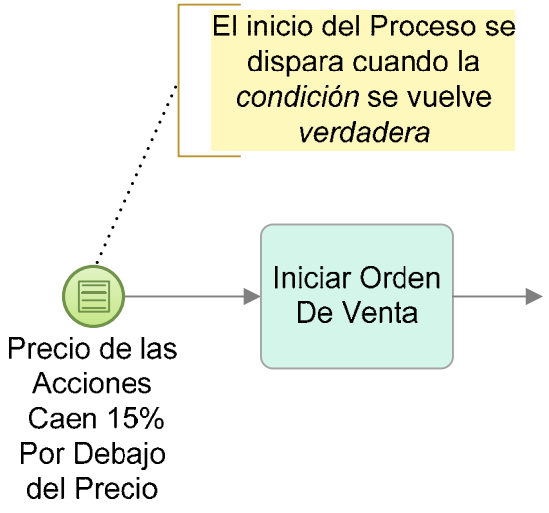


Figura 8-12—Un Evento de Inicio de tipo Condicional utilizado para Iniciar un Proceso

Una *condición* es una expresión en lenguaje natural o en lenguaje computacional que prueba ciertos datos. La prueba resultará en una respuesta de *verdadero* o *falso*. El cambio esperado ocurre cuando el resultado de la *condición* que se prueba es *verdadero*. La *condición* del Evento debe convertirse en *falso* y luego en *verdadero* nuevamente para que el Evento pueda dispararse de nuevo.

Cuando se está creando un modelo de alto nivel (para documentar un Proceso), una *condición* en lenguaje natural es generalmente suficiente. Por ejemplo una *condición* para un Proceso que vende una acción:

“El Precio Actual de la Acción Cae 15 por ciento por Debajo del Precio de Compra”

Si bien este tipo de *condición* es suficiente en un nivel de documentación, un modelo diseñado para ejecución (por una Suite de BPM) requerirá un lenguaje más formal que el sistema pueda entender.

Por ejemplo, una definición formal de la misma condición para el precio de la acción podría leerse como:

“(dataObject[name="infoAccion"]/precioActual) < ((dataObject[name="infoAccion"]/precioCompra) * 0.85)”

La Figura 8-12, arriba, provee un ejemplo de un Proceso que se inició por la condición anterior (se muestra la versión en lenguaje natural).

En BPMN 1.1, el Evento de Inicio de tipo Regla se renombró a Evento de Inicio de tipo Condicional ya que representaba una descripción más apropiada de su comportamiento.

Eventos de Inicio Múltiple

Los Eventos de Inicio de tipo Múltiple utilizan un pentágono como marcador dentro de la forma del Evento (ver Figura 8-13). Este representa una colección de dos o más *disparadores* de Eventos de Inicio. Los *disparadores* pueden ser cualquier combinación de *mensajes, temporizadores, condiciones, y/o señales*. Cualquiera de esos *disparadores* instanciará el Proceso—es decir, ni bien se desencadena el *disparador*, se genera una nueva *instancia* del Proceso y el flujo continúa desde ese Evento de Inicio (ignorando otras *instancias* que ya puedan existir). Si uno de los otros *disparadores* se desencadena, o el mismo *disparador* se desencadena nuevamente, entonces se genera otra *instancia* de Proceso.

Figura 8-13—Un Evento de Inicio de tipo Múltiple

El ícono del Evento de Inicio de tipo Múltiple a cambiado de una estrella de seis puntas a un pentágono en BPMN 1.1 (como se muestra en la Figura 8-3 anterior).

Modelando con Más de un Evento de Inicio

La mayor parte de los Procesos tienen un único Evento de Inicio. Sin embargo, es posible incluir más de un Evento de Inicio en un único Proceso (ver Figura 8-14). Esto se necesita usualmente cuando hay varias formas de que un Proceso se inicie, cada uno iniciando en un punto diferente del Proceso. A veces, como en el siguiente ejemplo, los Eventos de Inicio pueden dar lugar a diferentes secuencias de Actividades, sin embargo, esto no es un requerimiento.

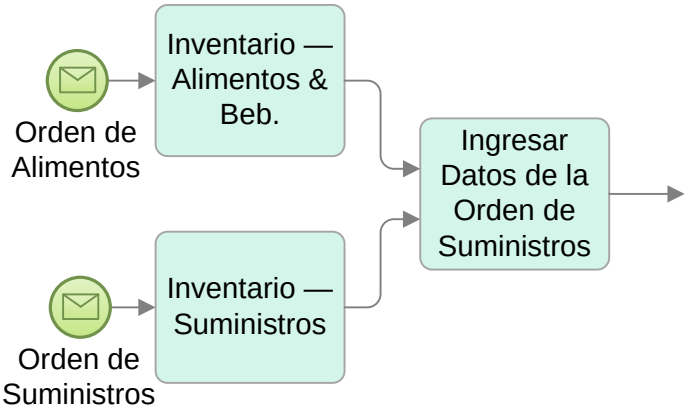


Figura 8-14—Un Proceso con Eventos de Inicio múltiples

Cada Evento de Inicio es independiente de los otros Eventos de Inicio en el Proceso. Esto significa que el Proceso empezará cuando cualquiera de los Eventos de Inicio se dispare. El Proceso no esperará por todos los Eventos de Inicio. Si otro Evento de Inicio se dispara luego de que el Proceso empieza, entonces se crea una *instancia* (ejecución) independiente del Proceso.

Cualquier Proceso, tenga o no múltiples Eventos de Inicio, puede tener múltiples *instancias* del Proceso activas al mismo tiempo. Cada vez que se dispara un Evento de Inicio (cualquier Evento de Inicio), se crea una nueva *instancia* del Proceso.

Eventos de Inicio y Sub-Procesos

Tenga en cuenta que los Eventos de Inicio basados en *disparadores* solo pueden aparecer en Procesos de "Alto Nivel". Los Eventos de Inicio basados en *disparadores* no se utilizan nunca en Sub-Procesos porque es el Proceso *padre* el que invoca el inicio de un Sub-Proceso (cuando un *token* llega desde el Proceso *padre* al Sub-Proceso.)

Los Eventos de Inicio Son Opcionales

La mayor parte de los Procesos tienen un Evento de Inicio para mostrarla posición del inicio del Proceso o para indicar el *disparador* que iniciará el Proceso. Sin embargo, un Proceso, ya sea de *alto nivel* o un Sub-Proceso, no requiere tener un Evento de Inicio (ver Figura 8-15).

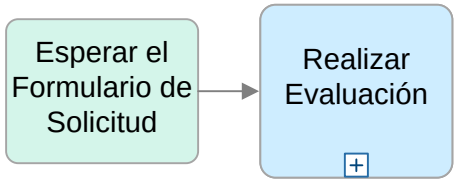


Figura 8-15—El Inicio de un Proceso Sin un Evento de Inicio

Si no se utiliza el elemento Evento de Inicio, entonces el primer *objeto de flujo* que no tiene un Flujo de Secuencia *de entrada* representa la posición donde empieza el Proceso. Esta representación, se comportará como si existiera un Evento de Inicio invisible conectado al Primer elemento del Proceso (ver Figura 8-16). Si hay varios elementos que no tienen Flujos de Secuencia *de entrada*, entonces todos esos elementos se iniciarán al inicio del Proceso.

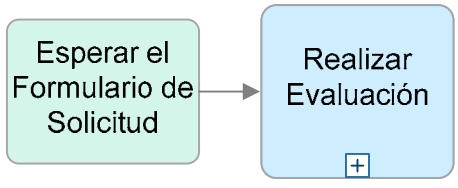


Figura 8-16—El Proceso Actúa como si hubiese un Evento de Inicio Virtual

El uso de un Evento de Inicio está atado al uso de un Evento de Fin. Si no se utiliza un Evento de Inicio, entonces tampoco se utiliza un Evento de Fin. Si se utiliza un Evento de Fin, entonces también se utiliza un Evento de Inicio.

Mejor Práctica: ***Utilización de Eventos de Inicio****—En general, recomendamos que los modeladores utilicen Eventos de de Inicio y de Fin.*

Sin embargo, hay situaciones en que es más adecuado crear un Proceso sin ellos. Por lo general, un modelador hará esto para Sub-Procesos pequeños y simples donde el comienzo y el fin del flujo se entienden claramente. Un ejemplo de esto es el concepto de "caja paralela" —un Sub-Proceso con un conjunto de Actividades que deben correr en paralelo (ver Figura 8-17). Al modelar este comportamiento con un Sub-Proceso sin Eventos de Inicio y Fin, el diagrama resultante es más compacto y menos confuso.

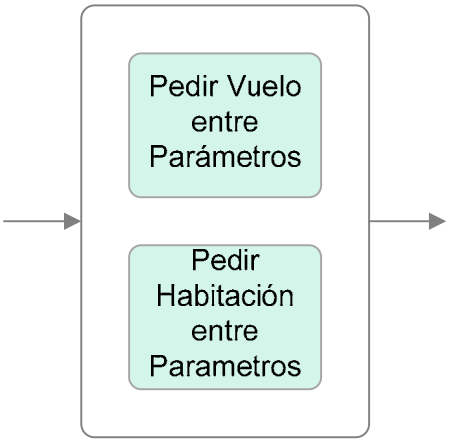


Figura 8-17—Un Ejemplo de una "Caja Paralela" donde no se utilizan Eventos de Inicio y de Fin

La figura siguiente muestra el método alternativo de modelado de este mismo comportamiento. Si bien este método comunica el comportamiento apropiado, es menos atractivo visualmente.

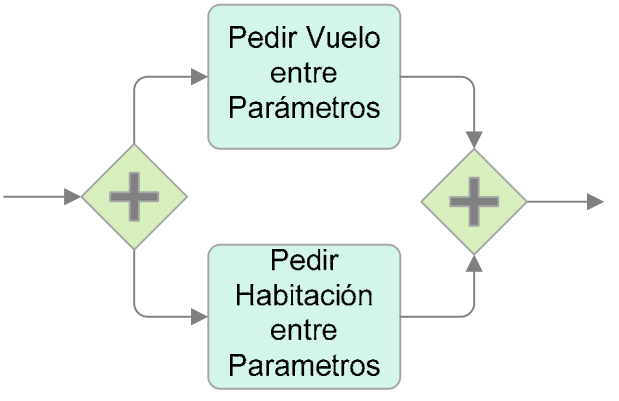


Figura 8-18—Un método alternativo de modelado del comportamiento de "Caja Paralela"

Para proporcionar la capacidad de modelar cosas como las "cajas paralelas" (como se vio en la Figura 8-17 anterior), se hicieron opcionales los Eventos de Inicio y Fin en BPMN

Eventos Intermedios

Un Evento Intermedio indica cuando algo sucede/ocurre después de que un Proceso ha comenzado y antes de que haya finalizado. Un Evento Intermedio se representa mediante un pequeño círculo abierto, con una doble línea fina marcando su límite (ver Figura 8-19).

Figura 8-19—Un Evento Intermedio.

Los Eventos Intermedios se colocan dentro del flujo del Proceso para representar cosas que suceden durante las operaciones normales del Proceso, y que generalmente ocurren entre las Actividades. Además, estos también pueden interrumpir el procesamiento normal de una Actividad.

Hay nueve tipos diferentes de Eventos Intermedios, cada uno con su propia representación gráfica (véase Figura 8-21). Cada tipo de Evento Intermedio puede *lanzar* o *capturar* el evento.

Una vez más, se han dividido los mismos de acuerdo a su tipo en *básicos* y *avanzados*:

Eventos Intermedios Básicos (véase Figura 8-20):

Simple

- **Básico**—No se define ningún *disparador*.
- **Temporizador**—El *disparador* se basa en una fecha y hora específica, o en un ciclo regular de fecha-hora (por ejemplo, el primer Viernes del mes a las 8 am.)
- **Mensaje**—El *disparador* es un *mensaje*. El mensaje debe ser enviado a otra entidad de negocio en el Proceso, o debe ser recibido de una de estas. Estas entidades de negocio (*participantes*), si se muestran en el diagrama, son representadas por Pools.
- **Señal**—El *disparador* es una *señal* que se emite o recibe.

Figura 8-20—Tipos de Eventos Intermedios Básicos.

Eventos Intermedios Avanzados (véase Figura 8-21):

- **Error**—Define un Evento que normalmente interrumpirá el Proceso o requerirá corrección (véase Interrupción de Actividades mediante Eventos en la página 96).
- **Cancelación**—Es utilizado para cancelar un Sub-Proceso de Transacción (véase "Transacciones y Compensación" en la página 174).

- **Compensación**—Es utilizado para establecer el comportamiento necesario para "deshacer" Actividades en caso de que un Sub-Proceso de Transacción sea cancelado o necesite ser deshecho (véase "Transacciones y Compensación" en la página 174).
- **Condicional**—Define una regla que debe cumplirse para que el Proceso continúe.
- **Vínculo**—Es utilizado para crear un mecanismo visual "Go To", ocultando un Flujo de Secuencia largo, o para establecer conectores "off-page", para imprimir.
- **Múltiple**—Define dos o más *disparadores* que pueden ser cualquier combinación de *mensajes*, *temporizadores*, *errores*, *condiciones*, o *señales*.

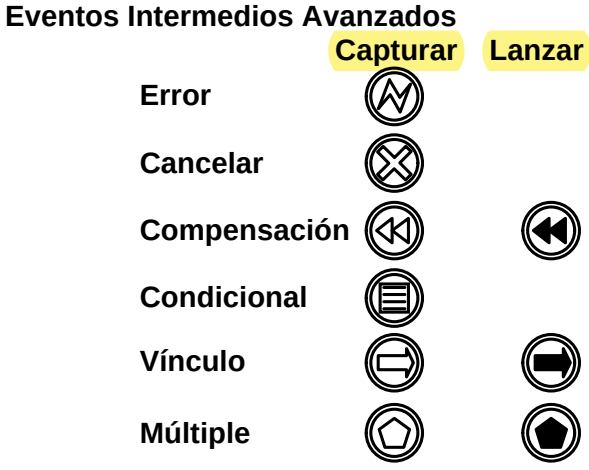


Figura 8-21—Tipos de Eventos Intermedios Avanzados (*lanzados* y *capturados*).

Esta diferencia gráfica entre *capturar* y *lanzar* es nueva en BPMN 1.1. Otros cambios relacionados con los Eventos Intermedios en BPMN 1.1 consisten en la inclusión del Evento Intermedio de Señal, el cambio del símbolo para representar el Evento Intermedio Múltiple, y el cambio de nombre del Evento Intermedio de Regla, que es ahora el Evento Intermedio Condicional.

Comportamiento de un Evento Intermedio

Cuando llega un *token* desde un Flujo de Secuencia entrante a un Evento Intermedio, este hará una de dos cosas:

- Esperará a que algo suceda (por ejemplo, esperar por la condición definida en el *disparador* – véase "Evento de Inicio Mensaje" en la página 86 para un ejemplo). Este tipo de Evento se conoce como Evento Intermedio *capturador*. El interior de los símbolos para todos los Eventos Intermedios *capturador* tiene un fondo de color

blanco. Esto también incluye los Eventos de Inicio, ya que este tipo de Eventos también espera para ser lanzado.

- Se lanzará inmediatamente (creando las condiciones definidas en el *disparador* – véase "Evento de Inicio Mensaje" en la página 86 para un ejemplo). Este tipo de Evento se conoce como Evento Intermedio *lanzador*. El interior de los símbolos para todos los Eventos Intermedios *lanzadores* tiene un fondo de color negro. Esto también incluye Eventos de Fin, ya que este tipo de Eventos también se lanza inmediatamente.

Los Eventos que pueden *capturar* son:

- Mensaje
- Temporizado
- Error
- Cancelación
- Compensación
- Condicional
- Vínculo
- Señal
- Múltiple (captura cualquier evento entrante en la lista)

Un *token* arribando a un Evento Intermedio *capturador* deberá esperar hasta que el *disparador* se active. A continuación el *token* saldrá inmediatamente, moviéndose hacia el Flujo de Secuencia *saliente*.

Los Eventos que pueden *lanzar* son:

- Mensaje
- Compensación
- Vínculo
- Señal
- Múltiple (lanza todos los eventos en la lista)

Un *token* arribando a un Evento Intermedio *lanzador* activará al instante el *disparador*. A continuación saldrá inmediatamente, moviéndose hacia el Flujo de Secuencia *saliente*.

Conexión de Eventos Intermedios

Los Eventos Intermedios son colocados en el *flujo normal* de un Proceso (por ejemplo, entre las Actividades), o son adjuntados al límite de una Actividad para *disparar* una interrupción de la misma—esto es discutido con mayor profundidad más adelante—véase "Interrupción de Actividades mediante Eventos."

Dado que los Eventos Intermedios no son Gateways (discutidos más adelante en "Gateways", en la página Capítulo 9128), no se utilizan para dividir o fusionar Flujos de Secuencia. Por lo tanto, sólo se permite un Flujo de Secuencia *entrante* y uno *saliente* para un Evento Intermedio dentro de un *flujo normal* (véase Figura 8-22).

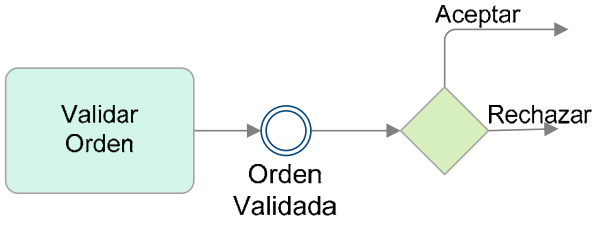


Figura 8-22—Un Evento Intermedio dentro del *flujo normal*.

Interrupción de Actividades mediante Eventos

BPMN utiliza Eventos adjuntos al límite de una Actividad como una manera de modelar *excepciones* al flujo normal del Proceso. El Evento adjunto indica que la Actividad debe ser interrumpida cuando el Evento es disparado (véase Figura 8-23). Este tipo de Eventos siempre *captura* el *disparador* adecuado, por lo que el símbolo del *disparador* tendrá siempre el fondo de color blanco (véase Figura 8-19, citada anteriormente).

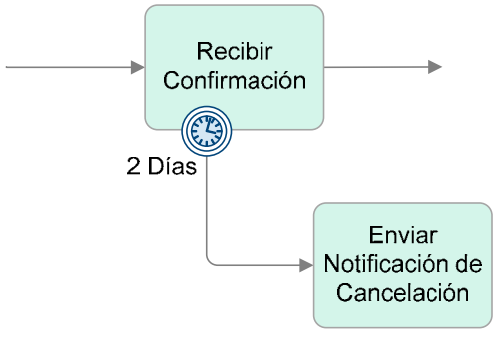


Figura 8-23—Un Evento Intermedio adjunto al límite de una Actividad.

La forma en que BPMN maneja las *excepciones* es una innovación para las notaciones de modelado de procesos. La manera normal de salir de una Actividad es, completar el trabajo en la Actividad, y para el conector *saliente* (un Flujo de Secuencia) mostrar el camino a la siguiente Actividad (u otro *objeto del flujo*). Como una forma de acentuar los caminos de *excepción* dentro de un Proceso, el enfoque de BPMN acerca de adjuntar Eventos Intermedios al límite de una Actividad, provee una manera natural y evidente para mostrar la forma anormal de salir de una Actividad. Tanto las Tareas como los Sub-Procesos son interrumpidos de la misma manera.

Los tipos de Eventos que pueden interrumpir una Actividad son:

- Temporizado
- Mensaje
- Error
- Cancelación
- Condicional
- Señal

- Múltiple

Un Evento Intermedio de Compensación también puede ser adjuntado al límite de una Actividad. Sin embargo, estos Eventos no interrumpen la Actividad, ya que estos sólo son operativos después que una Actividad se ha completado. Véase “Transacciones y Compensación” en la página 174 para más detalles sobre Eventos de Compensación.

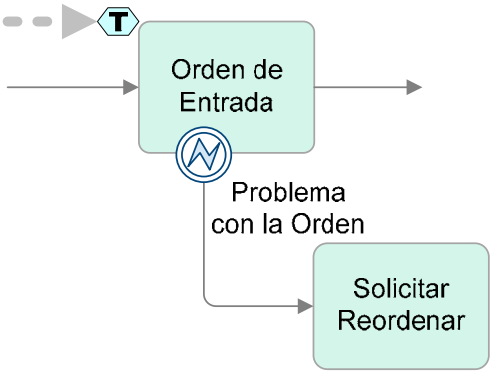
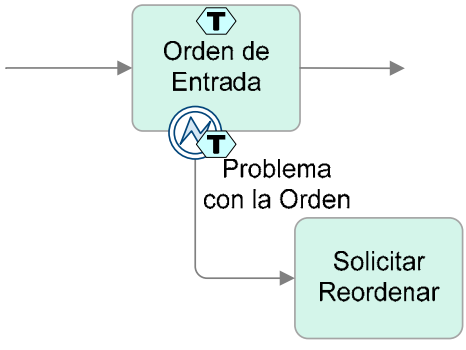


Figura 8-24—Un *token* llega a una Tarea con un Evento Intermedio de Error adjunto.

Rastrear un *token* permite observar cómo funciona el *manejo de excepciones*. El *token* abandona el *objeto de flujo* anterior y llega a la Actividad que contiene un Evento Intermedio adjunto (véase Figura 8-24, citada anteriormente). [19]

El *token* ingresa a la Actividad y se inicia el trabajo y el ciclo de vida de la Actividad (para más información sobre este tema, véase la sección “El Ciclo de Vida de una Actividad” en la página 173). Al mismo tiempo, otro *token* se crea y reside en el Evento Intermedio, más específicamente en su límite (véase Figura 8-25). Esto prepara al Evento Intermedio que potencialmente puede interrumpir, para ser lanzado. Si el Evento adjunto se tratara de un *temporizador*, entonces el reloj es iniciado.



[19] Se utiliza un Evento Intermedio de Error para demostrar el comportamiento, pero puede ser utilizado cualquiera de los Eventos Intermedios capaces de interrumpir, listados anteriormente.

Figura 8-25—Un *token* que reside tanto en la Tarea como en el Evento Intermedio de Error.

La Actividad y el Evento Intermedio adjunto participan en una *condición de carrera*. Cualquiera que termine primero ganará la *carrera* y tomará el control del Proceso junto con su *token*. Si la Actividad termina antes que el *disparador* ocurra (en este caso de error), entonces el *token* de la Actividad se mueve hacia el Flujo de Secuencia *saliente* normal de la Actividad (véase Figura 8-26) y el *token* adicional se consume.

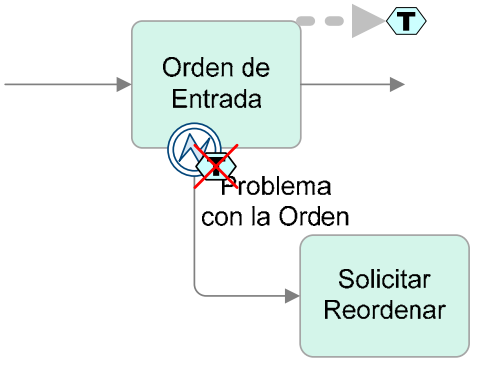


Figura 8-26—El *token* abandona la Tarea antes que el Evento Intermedio de Error se dispare

Sin embargo, si el Evento Intermedio adjunto se *dispara* después de que la Actividad finalice, entonces la Actividad es interrumpida (se detiene todo el trabajo). En este caso, el *token* del Evento se mueve hacia el Flujo de Secuencia *saliente*, pero no el Flujo de Secuencia *saliente* normal de la Actividad (véase Figura 8-27). El *token* que estaba en la Actividad es consumido.

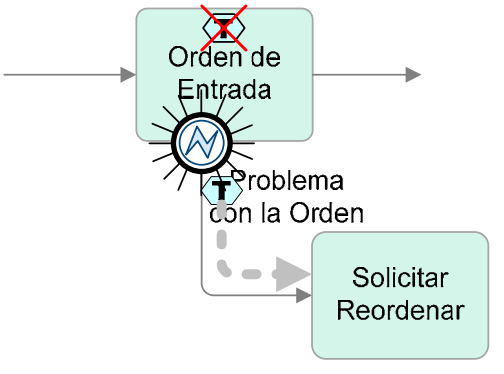


Figura 8-27—El Evento Intermedio de Error es *disparado* interrumpiendo la Tarea

El flujo de los Eventos Intermedios puede ir en cualquier dirección. Puede ir por un nuevo camino, puede reincorporarse a la ruta normal, o puede volver *hacia atrás*.

Los Eventos capaces de interrumpir, pueden ser adjuntados tanto a Tareas como a Sub-Procesos. La excepción a esto es el Evento Intermedio de Cancelación, que sólo se utiliza en los Sub-Proceso de Transacción (véase “Transacciones y Compensación” en la página 174).

Eventos Intermedios Básicos

Al igual que con un Evento de Inicio, un *disparador* no es siempre necesario para un Evento Intermedio. Un Evento Intermedio sin un *disparador* se conoce como Evento Intermedio Básico (cómo en la Figura 8-19, citada anteriormente).

Los Eventos Intermedios Básicos se utilizan principalmente para documentar aquellas Actividades que se han completado, o aquellas en las cuales el Proceso ha alcanzado un estado definido, como un hito. El nombre del Evento a menudo puede proporcionar información suficiente para estos fines.

Para un Evento Intermedio Básico, no se define ninguna condición (es decir, no hay *disparador*). Por lo tanto, este se lanza de inmediato (véase Figura 8-28).

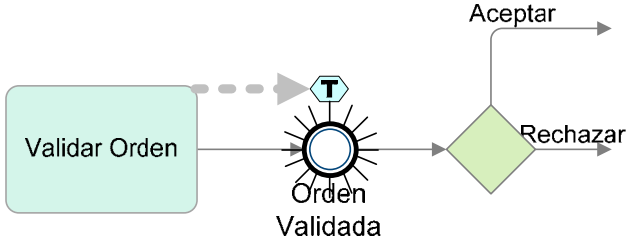


Figura 8-28—Un *token* llega al Evento Intermedio

Inmediatamente después del lanzamiento, el *token* se mueve hacia el Flujo de Secuencia *saliente*, continuando el Proceso (véase Figura 8-29).

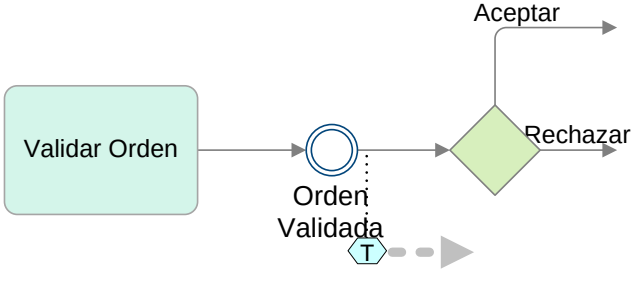


Figura 8-29—El *token* abandona el Evento Intermedio Básico

Eventos Intermedios Temporizador

El Evento Intermedio de tipo *Temporizador* se representa mediante el símbolo de un reloj, dentro de la silueta del Evento (véase Figura 8-30, abajo). Los Eventos Intermedios Temporizador sólo pueden *capturar*. Es-

tos añaden dependencias basadas en el tiempo dentro de un Proceso y son introducidos en el Flujo de Secuencia para crear un delay, o se adjuntan a los límites de una Actividad para crear una condición de Deadline o Time-Out (Fecha Límite y Tiempo Expirado respectivamente). Los Eventos Intermedios Temporizador también pueden utilizarse como parte de un Gateway Basado en Eventos (véase página 134).

Cuando un *token* llega a un Evento Intermedio Temporizado, el "reloj comienza", y el *token* aguarda a que suceda la condición específica relacionada al tiempo (véase Figura 8-30).

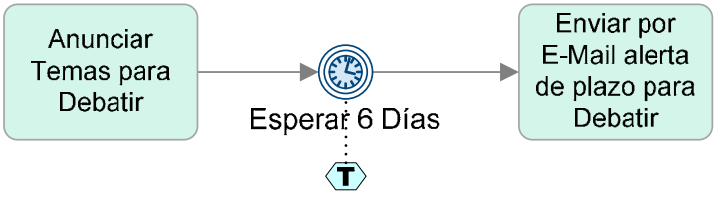


Figura 8-30—Un *token* abandonando un Evento Intermedio *Temporizado*.

Delays

Cuando se inserta entre las Actividades de un Proceso, el Evento Intermedio Temporizado representa un *delay* en el flujo del Proceso. Los Eventos Intermedios Temporizador pueden representar una Fecha y Hora específicas (por ejemplo, esperar hasta el 15 de abril, a las 5 pm.), un tiempo relativo (por ejemplo, esperar 6 días), o una fecha relativa repetitiva (por ejemplo, esperar hasta el próximo lunes a las 8 am.)

Figura 8-31—Un Evento Intermedio Temporizado creando un *delay* en el Proceso

Cuando el temporizador "se apaga"—es decir, se produce la condición específica de tiempo—en este caso, 6 días después de iniciado el temporizador, entonces el *token* se mueve hacia el Flujo de Secuencia *saliente* del Evento (véase Figura 8-30) y el Proceso continúa.

Para mostrar cómo se comporta un delay *Temporizado*, se seguirá el rastro del *token* a través del ejemplo de la figura citada anteriormente. El *token* abandona el *objeto de flujo* previo y llega al Evento Intermedio Temporizado (véase Figura 8-32). El *objeto de flujo* que precede al Temporizador puede ser una Actividad, un Gateway, u otro Evento Intermedio.

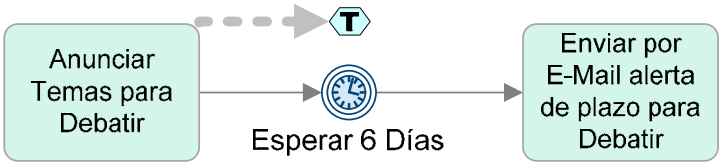


Figura 8-32—Un *token* llega al Evento Intermedio Temporizado

El *token* se mantendrá en el Evento Intermedio Temporizado (véase Figura 8-33) hasta que el reloj alcance la configuración de tiempo del Evento (es decir, hasta que la "alarma se apague").

El reloj para *delays* de tiempo relativos (por ejemplo, 6 días) se inicia cuando el *token* llega al Evento. El reloj para tiempos/fechas específicos y recurrentes, se compara siempre contra un calendario existente (o uno simulado).

Es necesario tener en cuenta que si se fija una fecha y hora específica (por ejemplo, Enero 1 de 2007 a las 8 am.) y esta ocurre antes de que el Evento Intermedio Temporizado se active (es decir, antes de que el *token* llegue), entonces este Evento nunca ocurrirá.

Mejor Práctica: ***Configurar Temporizadores****—evitar condiciones específicas de fecha y hora, ya que estas impiden la reutilización del proceso.*

Figura 8-33—El *token* se *retrasará* hasta que el Temporizador sea activado

Cuando el Evento es finalmente activado, el *token* abandona el Evento inmediatamente y se mueve hacia el Flujo de Secuencia *saliente* (véase Figura 8-34).

Figura 8-34—Cuando el Temporizador es disparado, el *token* continúa

Deadlines y Time-Outs

Como se mencionó anteriormente, los Eventos Intermedios Temporizador también pueden interrumpir una Actividad. Cuando la Actividad se inicia, también lo hace el *temporizador*. Si la Actividad finaliza en primer lugar,

entonces el *temporizador* se completa normalmente y el Proceso continúa con normalidad. Si el *temporizador* se apaga antes que la Actividad se complete, la Actividad se interrumpe de inmediato y el Proceso continúa en el Flujo de Secuencia del Evento Intermedio Temporizado (véase Figura 8-35).

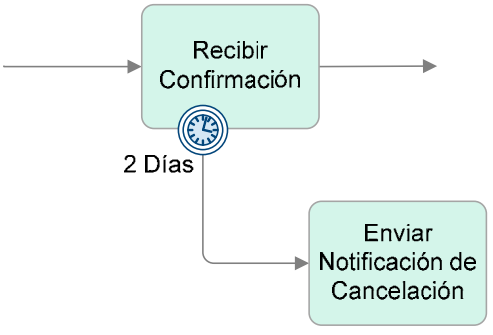


Figura 8-35—Una Actividad con un *time-out*

El comportamiento del Evento Intermedio Temporizado adjunto, y otros Eventos de interrupción, se describen en “Interrupción de Actividades mediante Eventos” en la página 96.

Time-Outs No Interrumpibles

Como se describió en la sección anterior, un Evento Intermedio Temporizado adjunto interrumpe una Actividad cuando se dispara. Sin embargo, hay momentos en los cuales un *temporizador* puede ser necesario para *disparar* Actividades adicionales, sin interrumpir la Actividad. Por ejemplo, si el trabajo de una Actividad no ha terminado en el tiempo esperado, entonces el comportamiento deseado es a menudo enviar un email al superior del Actor, para agilizar la situación. Se hace referencia a este tipo de escenarios como comportamiento Evento No Interrumpible.

En BPMN 1.1, es imposible utilizar un Evento Intermedio adjunto para crear el comportamiento de un Evento No Interrumpible. Sin embargo, hay “Patrones de Procesos” que resuelven fácilmente esta necesidad.

La Figura 8-36 proporciona un ejemplo de un Patrón de Procesos que resuelve este problema. El Patrón se basa en un Sub-Proceso con el Evento Intermedio Temporizado configurado en un flujo paralelo a la Actividad, que termina con un Evento de Fin Terminador. Tanto la Actividad como el Evento Intermedio Temporizado siguen un Evento de Inicio.

El *temporizador* se inicia al mismo tiempo que la Actividad. Si el *temporizador* se dispara antes de la finalización de la Actividad “Recibir Confirmación”, entonces se envía un recordatorio sin interrumpir la Actividad. El *temporizador* se encuentra en un *loop* de modo que se envían recordatorios cada dos días. Cuando la Actividad eventualmente finaliza, el trabajo se traslada a un Evento de Fin Terminador, el cual detendrá toda la Actividad del Sub-Proceso, incluyendo el *loop* del *Temporizador*. El flujo

luego continúa a nivel de los Procesos *padres*. Véase también Evento de Fin Terminador en la página 123.

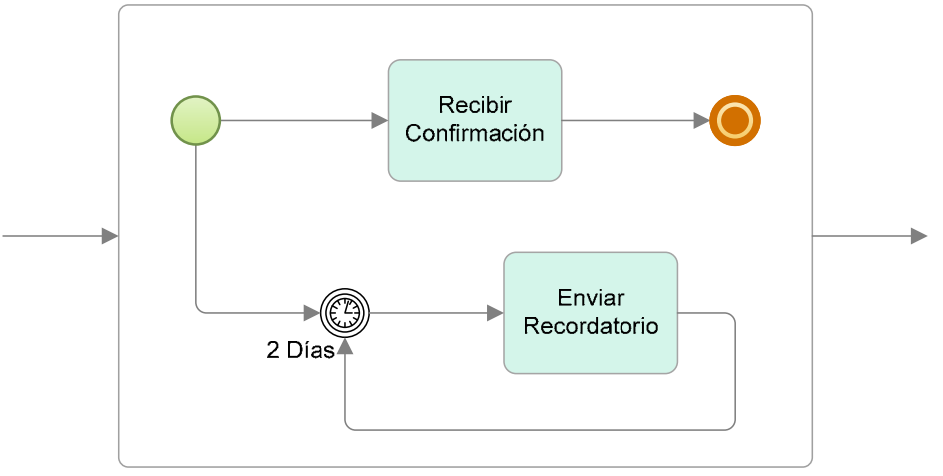


Figura 8-36—Un delay Evento Intermedio Temporizado utilizado para escalación.

Patrón de Proceso: El ejemplo mostrado en la Figura 8-36, citada anteriormente, corresponde a un "Patrón de Proceso". Este patrón puede ser utilizado con diferentes Eventos Intermedios capturadores (reemplazando el Evento Intermedio Temporizado) para crear diversos comportamientos del tipo no interrumpible, basados en mensajes, señales, u otros tipos de disparadores.

Otra serie de ejemplos se presentan en la introducción a los escenarios de apertura, en la sección "Estableciendo Temporizadores" en la página 39, y también en "Otro Enfoque para la Escalada" en la página 55.

En la próxima versión de BPMN (BPMN 2.0), se está construyendo una capacidad del tipo *no interrumpible* en los Eventos Intermedios que son adjuntados a los límites de una Actividad. Esto permitirá el uso de Eventos adjuntos tanto para *interrumpir* una Actividad, como para *disparar* un Evento No Interrumpible. Patrones de Procesos como el de la Figura 8-36 seguirán siendo válidos, pero ya no serán necesarios. La notación para los Eventos No Interrumpibles no está finalizada aún, por lo que no puede proveerse un ejemplo de momento.

Eventos Intermedios Mensaje

Los Eventos Intermedios Mensaje se distinguen de otros tipos de Eventos Intermedios por el símbolo de un sobre puesto dentro de la forma del Evento (véase Figura 8-37). Los *Mensajes* son diferentes de las *señales* en el sentido de que son dirigidos entre los *participantes* del Proceso—es decir, siempre operan a través de Pools. Además, no pueden ser utilizados para comunicarse entre Carriles dentro de un Pool.

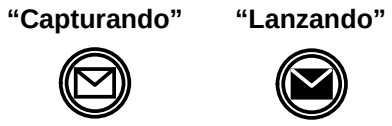


Figura 8-20—Eventos Intermedios Mensaje

Como se muestra en la figura anterior, hay dos tipos de Eventos Intermedios Mensaje: *lanzadores* y *capturadores*—es decir, de envío y recepción. [20]

Enviando *Mensajes*

Un tipo de Evento Intermedio Mensaje es un Evento *lanzador* que envía un *mensaje*. El interior del símbolo que lo representa es un sobre con relleno color negro (véase Figura 8.21, citada anteriormente). Este tipo de Eventos indica que el Proceso enviará un *mensaje* en ese momento del Proceso.

Cuando un *token* llega a un Evento Intermedio Mensaje *lanzador*, inmediatamente dispara el Evento, el cual envía el *mensaje* a un *participante* específico (véase Figura 8-38).

Figura 8-21—Un *token* arribando a un Evento Intermedio Mensaje *lanzador*

Inmediatamente después de que el *mensaje* es enviado, el *token* se mueve hacia el Flujo de Secuencia *saliente* (véase Figura 8-39), continuando el Proceso.

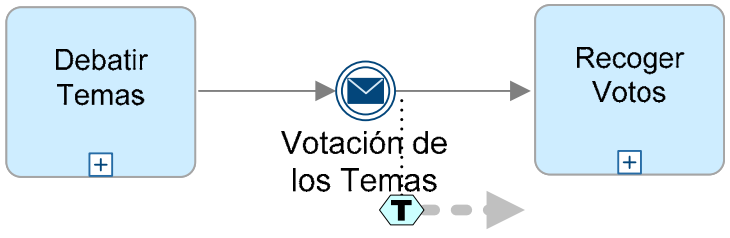


Figura 8-22—El *token* abandona el Evento Intermedio Mensaje

Recibiendo *Mensajes*

[20] En el ejemplo provisto, se muestran Evento Intermedio Mensaje dentro del *flujo normal* del Proceso. No se **muestra** el Flujo de Mensajes que emana de estos Eventos y que se conecta a un *participante* externo. Para más información acerca de Flujo de Mensajes consulte la página 190.

El otro tipo de Evento Intermedio Mensaje *captura*—es decir, espera que arribe un *mensaje*. El interior del símbolo que lo representa es un sobre con relleno color blanco (véase Figura 8-21, citada anteriormente).

Cuando un *token* llega a un Evento Intermedio Mensaje *capturador* en el *flujo normal*, el Proceso se detiene hasta que llega el *mensaje* (véase Figura 8-40). Nótese que si una Suite BPMN está ejecutando el modelo, y el *mensaje* llega antes que el Evento Intermedio Mensaje este "activo" (es decir, antes que llegue el *token*), entonces el *mensaje* es ignorado. En tal situación, el Evento Intermedio Mensaje esperará indefinidamente a menos que el *mensaje* sea enviado nuevamente.

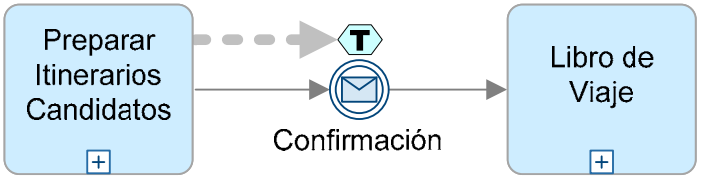


Figura 8-23—El *token* llega a un Evento Intermedio Mensaje *capturador*

Si el *token* está esperando en el Evento Intermedio y llega el *mensaje*, entonces el Evento se dispara. El *token* inmediatamente se mueve hacia el Flujo de Secuencia *saliente*, continuando el Proceso (véase Figura 8-41).

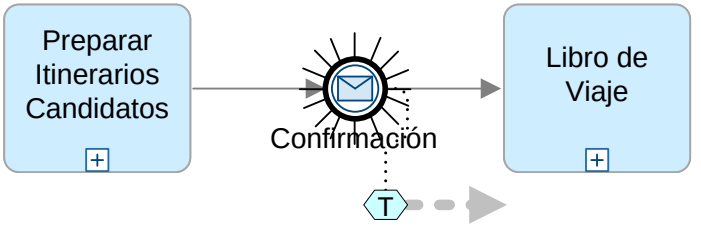


Figura 8-24—El *token* abandona el Evento Intermedio Mensaje *capturador*

Los Eventos Intermedios de Mensaje también pueden aparecer adjuntos al límite de una Actividad como un Evento *capturador*. Si el Evento ocurre, entonces la Actividad es interrumpida y el *token* la abandona a través del Flujo de Secuencia adjunto al Evento Intermedio Mensaje. Para más información acerca de la naturaleza exacta de este asunto, consulte la sección "Deadlines y Time-Outs" a partir de la página 99.

Los Eventos Intermedios Mensaje pueden además formar parte de un Gateway Basado en Eventos (véase la página 134).

Eventos Intermedios Señal

Al igual que los Eventos de Inicio de Señal, el Evento Intermedio de Señal utiliza el símbolo de un triángulo dentro de la forma del Evento (véase Figura 8-42). Al igual que el Evento Intermedio Mensaje, hay dos tipos de Eventos Intermedios de Señal—*lanzadores* y *capturadores*.

Figura 8-25—Eventos Intermedios Señal

Ejemplos acerca de cómo son utilizados los Eventos de Señal, pueden ser encontrados en la sección “Eventos de Señal” en la página 105.

Los Eventos de Señal son una nueva característica en BPMN 1.1. Estos reemplazan parte de la funcionalidad de los Eventos de Vínculo, agregan nuevas capacidades y ofrecen una amplia gama de patrones de Procesos.

Transmitiendo una Señal

El Evento Intermedio de Señal *lanzador* transmite. Cuando un *token* llega, se dispara inmediatamente el Evento, el cual transmite la *señal* a cualquier otro Evento que pueda estar esperando por ella (véase Figura 8-43); no sabe nada acerca de Eventos que podrían esperar para capturar la Señal. El interior de su símbolo es un triángulo con relleno de color negro.

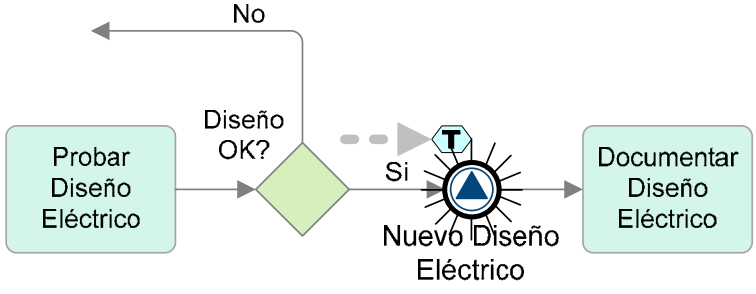


Figura 8-26—Un *token* llega a un Evento Intermedio de Señal *lanzador*

Inmediatamente después de que la *señal* es emitida, el *token* continúa hacia el Flujo de Secuencia *saliente* (véase Figura 8-44), continuando el Proceso.

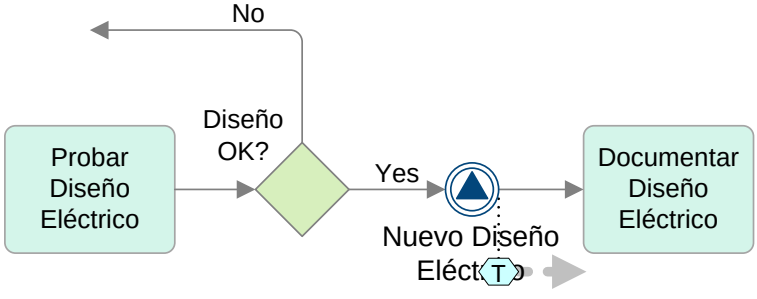


Figura 8-27—El *token* abandona el Evento Intermedio de Señal *lanzador*

Recibiendo una Señal

El Evento Intermedio de Señal *capturador* espera a que llegue una *señal*; el Proceso se detiene hasta que se detecta la *señal*. El interior del símbolo que lo representa es un triángulo con relleno de color blanco.

Cuando un *token* llega a un Evento Intermedio de Señal, el Proceso espera a detectar la *señal* (véase Figura 8-45). Nótese que si la *señal* llega antes que el Evento Intermedio de Señal esté listo—es decir, antes de que el *token* llegue, entonces la *señal* es ignorada. A menos que la misma *señal* sea enviada de nuevo, el Proceso esperará indefinidamente.

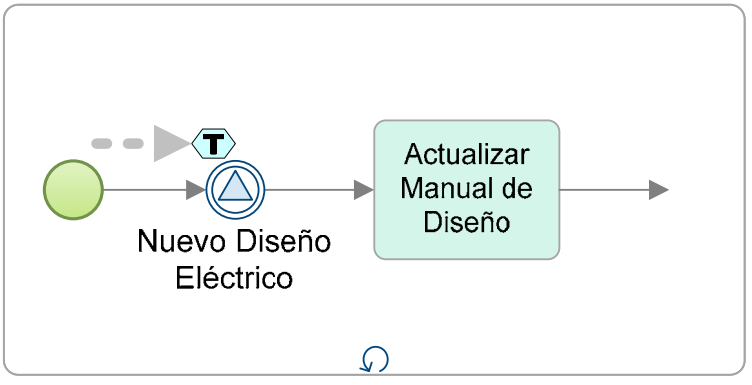


Figura 8-28—Un *token* llegando a un Evento Intermedio de Señal *capturador*

Cuando el Evento Intermedio de Señal es identificado (es decir, el Evento se dispara), entonces el *token* se mueve hacia el Flujo de Secuencia *saliente*, y el Proceso continúa (véase Figura 8-46).

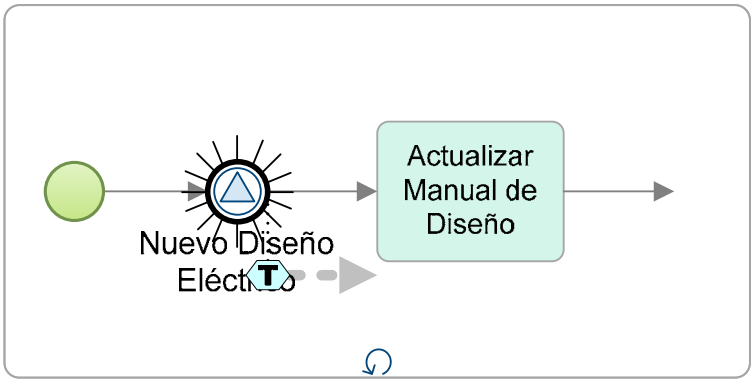


Figura 8-29—El *token* abandona el Evento Intermedio de Señal *capturador*

Uso de Eventos de Señal

Los Eventos de Señal proporcionan una capacidad general de comunicación dentro de y entre los Procesos. Tienen algunas similitudes con los Eventos de Mensaje. Al igual que los Eventos de Mensaje, hay dos tipos de Eventos de Señal: un tipo envía o *lanza* la *señal*, y el otro tipo recibe o *captura* la *señal*. A diferencia de los Eventos de Vínculo, los Eventos de

Señal no deben ser utilizados en pares. Un Proceso dado puede tener solo un Evento de Señal *lanzador* o *capturador*.

Por un lado, un *mensaje* es dirigido a un objetivo específico (es decir, otro *participante* en una relación business-to-business), mientras que las *señales* se emiten de forma general. Piense en las *señales* como señales de bengala. Estas se disparan y cualquier número de personas mirando pueden optar por reaccionar (o no). Sin embargo, las *señales* también tienen un nombre—por lo tanto, *capturar* Eventos puede filtrar *señales* que no tienen el nombre correcto (o pueden ser configuradas para reaccionar ante cualquier *señal*).

Hay varias maneras de utilizar Eventos de Señal, incluyendo:

- Manejo de excepciones
- Encadenar Procesos; esto es, marcando el inicio de un Proceso después de la realización de otro.
- Destacando que un *hito* determinado ha ocurrido.
- Comunicación Inter-Proceso—especialmente útil cuando hilos de ejecución paralelos de una actividad requieren coordinación.
- Como parte de un Gateway Basado en Eventos (véase página 134).

Los Eventos de Señal pueden operar a través de los niveles del Proceso (a través de Sub-Procesos hacia los *padres*, viceversa, o entre Sub-Procesos) o incluso a través de Pools. Por ejemplo, un Evento de Señal podría enviar reportes de estado a un cliente, indicando que el Proceso ha alcanzado un hito acordado (el cliente y la organización son *participantes* operando dentro de sus propios Pools).

La Figura 8-47 presenta un ejemplo de un *hito* en el cual hay dos Sub-Procesos. Los dos Sub-Procesos envían *señales* hacia el Proceso *padre*. El primer Proceso envía además una *señal* al segundo Sub-Proceso. Los Sub-Procesos están diseñados para ser reutilizados (en otros Procesos). Por lo tanto, para funcionar bien entre sí y con sus Procesos *padres*, deben enviar *señales* en los momentos adecuados.

La Actividad “Diseñar Tapa del Libro” en el Sub-Proceso intermedio, espera hasta que la Actividad “Desarrollar Texto y Conceptos Principales” del Sub-Proceso ubicado en la parte superior haya concluido. Ya que el Flujo de Secuencia no puede cruzar los límites del Sub-Proceso, los Eventos de Señal se encargan de la comunicación y sincronizan los dos Sub-Procesos.

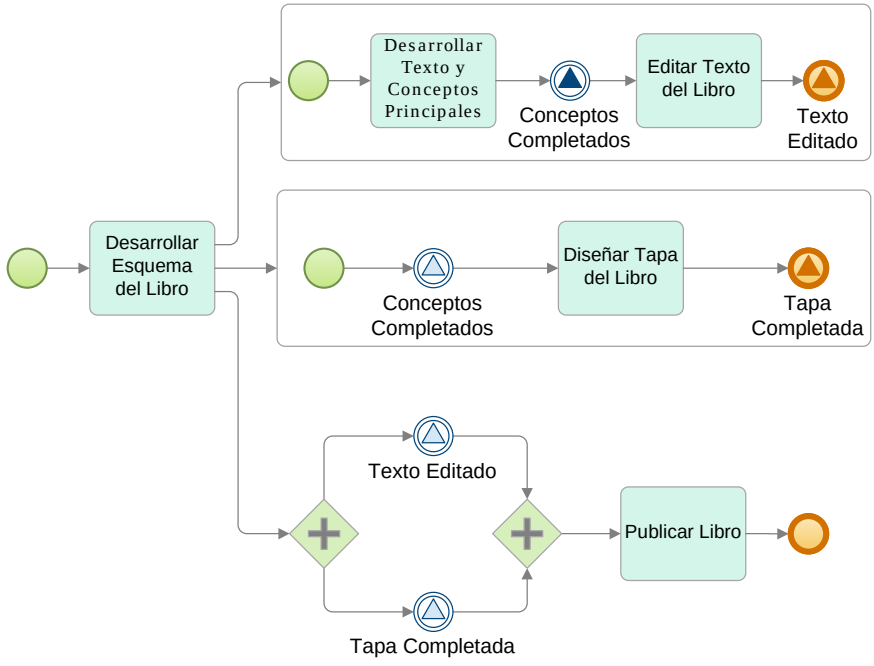


Figura 8-30—Los Eventos de Señal pueden comunicarse a través de los niveles del Proceso.

Además de la restricción de *hito* anterior, la Actividad "Publicar Libro" en el Proceso *padre* debe esperar hasta que la Actividad "Editar Texto del Libro" en el Sub-Proceso ubicado en la parte superior y la Actividad "Diseñar Tapa del Libro" en el Sub-Proceso intermedio se hayan completado. Para hacer esto, el Proceso *padre* detecta la *señal* de cada uno de los Sub-Procesos.

El rastreo de un *token* a través del ejemplo anterior, comienza en el punto en el cual ambos Sub-Procesos han comenzado su trabajo (véase Figura 8-48). Cada Evento de Inicio generará un *token* y lo enviará hacia el Flujo de Secuencia. El *token* en el Sub-Proceso superior irá a la Actividad "Desarrollar Texto y Conceptos Principales". El *token* en el Sub-Proceso inferior irá al Evento de Señal *capturador*, lo que hará que el Evento quede a la espera de una *señal*.[21]

[21] Los *tokens* también se envían a los Evento Intermedio de Señal capturador en el Proceso padre, pero no se rastrearán los mismos en este ejemplo.

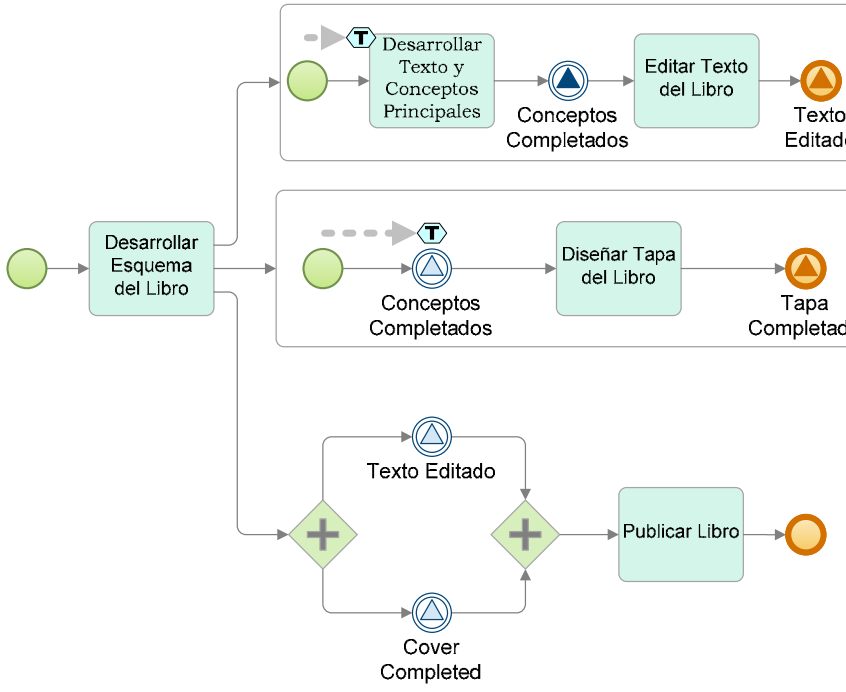


Figura 8-31—El Evento de Señal en el Sub-Proceso inferior está a la espera de una *señal*

Eventualmente la Actividad “Desarrollar Texto y Conceptos Principales” en el Sub-Proceso superior terminará y enviará el *token* al Evento de Señal *lanzador* (véase Figura 8-49).

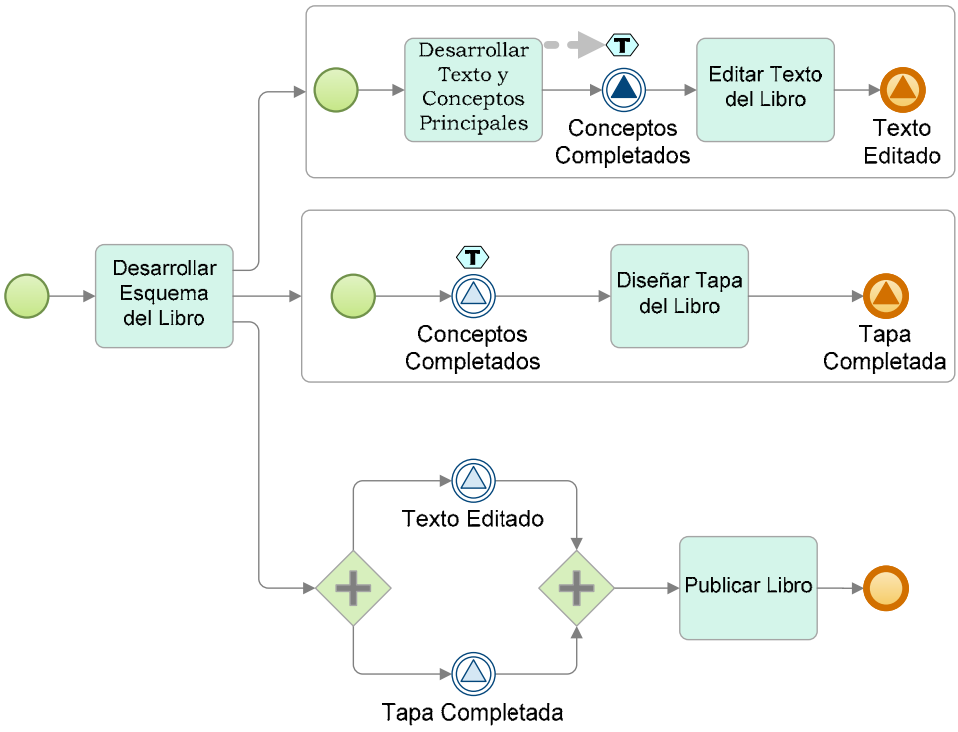


Figura 8-32—Eventualmente el *token* llegará al Evento de Señal *lanzador* en el Sub-Proceso superior

Cuando el *token* llega al Evento de Señal *lanzador* en el Sub-Proceso superior, este dispara el Evento que causa la emisión de la *señal* (véase Figura 8-50). Después de que la *señal* se dispara, el *token* continúa en el Flujo de Secuencia *saliente*, hacia la Tarea “Editar Texto del Libro”.

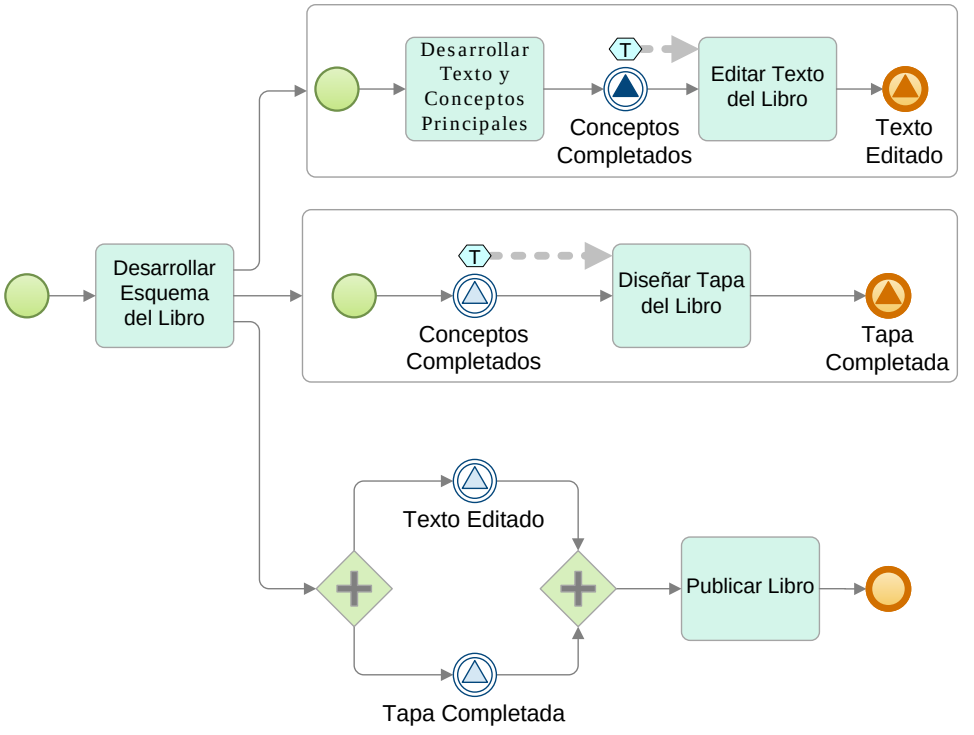


Figura 8-33—El Evento de Señal *lanzador* en el Sub-Proceso superior se dispara, lo cual es detectado por el Evento de Señal *capturador* en el Sub-Proceso inferior

En el Sub-Proceso inferior, el Evento de Señal *capturador* detecta la *señal* emitida por el Sub-Proceso superior. Esto disparará el Evento, lo cual significa que el *token* se mueve hacia la Actividad "Diseñar Tapa del Libro" a fin de que pueda comenzar su labor. [22]

Eventos Intermedios de Error

El Evento Intermedio de Error se utiliza para manejar la ocurrencia de un *error* que puede requerir la interrupción de una Actividad (a la cual está adjunto). Un *error* es generado cuando hay un problema crítico en el procesamiento de una Actividad. Los *Errores* pueden ser generados por aplicaciones o sistemas involucrados en el trabajo (que son transparentes para el Proceso) o por Eventos de Fin (véase página 119). El Evento Intermedio de Error utiliza el símbolo de un rayo dentro de la forma del Evento (véase Figura 8-51).

[22] Nótese que la especificación de BPMN no define el mecanismo preciso utilizado para soportar *señales*—es decir, la especificación es independiente de la implementación. En su nivel más simple, las personas pueden interpretar las señales enviadas entre sí, o una infraestructura de software sofisticada como una Cola de Mensajes puede garantizar que un Motor BPMN (motor de workflow) detecta los cambios pertinentes en el estado.

Figura 8-34—Un Evento Intermedio de Error

Este Evento puede ser utilizado solo cuando es adjuntado al límite de una Actividad, por lo que solo puede utilizarse para *capturar* un *error*, nunca para *lanzar* un *error*. El Evento de Fin de Error es utilizado para *lanzar* un *error* (véase "Evento de Fin de Error" en la página 124). Cuando ocurre un *error* todo el trabajo se detiene para ese Proceso; por lo tanto, no tiene sentido utilizar un Evento Intermedio para *lanzar* un *error*, ya que no se lleva a cabo el trabajo.

Cuando se activa este Evento, entonces todo el trabajo dentro de la Actividad se detiene. La Actividad puede ser una Tarea o un Sub-Proceso. Consulte "Interrupción de Actividades mediante Eventos" en la página 96 para observar un ejemplo del comportamiento de este Evento.

Eventos Intermedios de Cancelación

El Evento Intermedio de Cancelación está diseñado para manejar una situación en la cual una *transacción* es *cancelada*.[23] El Evento Intermedio de Cancelación usa un símbolo "X" con relleno color blanco dentro de la forma del Evento (véase Figura 8-52).

Figura 8-35—Un Evento Intermedio de Cancelación

Los Eventos Intermedios de Cancelación pueden solamente *capturar* una *cancelación de transacción*; no son capaces de *lanzarlos*. El Evento de Fin de Cancelación lanza la cancelación (véase "Evento de Fin de Cancelación" en la página 125).

Además, el Evento Intermedio de Cancelación puede ser adjuntado solamente al límite de un Sub-Proceso de Transacción. Puede ser activado por un Evento de Fin de Cancelación dentro de un Sub-Proceso, o a través de una cancelación recibida vía el *protocolo de transacción* asignado al Sub-Proceso de Transacción. Cuando se activa, el Sub-Proceso de Transacción es interrumpido (se detiene todo el trabajo) y el Sub-Proceso es *deshecho*, lo que puede dar lugar a la *compensación* de algunas Actividades dentro del Sub-Proceso. Consulte "Transacciones y Compensación" en la página 174 para obtener más detalles sobre la forma en que los Sub-Procesos de Transacción son cancelados.

[23] Una *transacción* es representada por un Sub-Proceso de Transacción.

Eventos Intermedios de Compensación

El Evento Intermedio de Compensación se distingue de otros tipos de Evento Intermedios por el símbolo de "rebobinar" que se ubica dentro de la forma del Evento (véase Figura 8-53).

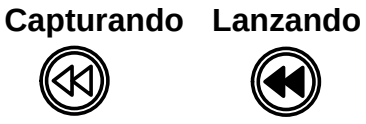


Figura 8-36—Eventos Intermedios de Compensación

Como se muestra en la figura anterior, existen dos tipos de Eventos Intermedios de Compensación: *lanzador* y *capturador*—es decir, de envío y recepción. El Evento Intermedio de Compensación *capturador* solo puede ser utilizado si es adjuntado a los límites de una Actividad. El *flujo normal* no puede utilizarse por el Evento de Compensación *capturador*. Sin embargo, el Evento Intermedio de Compensación *lanzador* sí es utilizado en el *flujo normal*.

El uso de ambas versiones, tanto *lanzador* como *capturador*, de estos Eventos se detallan en "Transacciones y Compensación" en la página 174.

Eventos Intermedios Condicionales

El Evento Intermedio Condicional representa una situación en la cual un Proceso está a la espera de que una *condición* predefinida se torne *verdadera*. Este tipo de Evento tiene como símbolo la figura de un papel con rayas, dentro de la forma del Evento (véase Figura 8-54).

Figura 8-37—Evento Intermedio Condicional

Existen tres maneras de utilizar Eventos Intermedios Condicionales:

- En el *flujo normal*, pero sólo como Eventos *capturadores*. Los Eventos Condicionales no son *lanzados*.
- Adjunto al límite de una Actividad para interrumpirla.
- Como parte de un Gateway Basado en Eventos (véase página 134)

Este tipo de Evento es disparado por un cambio en los datos relacionados al Proceso. Por ejemplo, un Evento Intermedio Condicional puede activarse si las ventas trimestrales de una empresa se ubican un 20 por ciento por debajo de lo esperado, o si el tipo de interés bancario base prevaleciente varía. Para ver un ejemplo más detallado y un descripción de las *condiciones*, consulte "Eventos de Inicio Condicionales" en la página 88.

Sería raro utilizar un Evento Intermedio Condicional en el *flujo normal*, pero es posible. Cuando un *token* llega al Evento, esperará allí hasta que

el Evento sea activado (la *condición* se vuelva *verdadera*). Cuando la *condición* se *vuelve* verdadera, entonces el Proceso continuará. Sin embargo, si la *condición* nunca se vuelve *verdadera*, entonces el Proceso se quedará estancado en ese punto y nunca se completará con normalidad.

En la mayoría de los casos, un Evento Intermedio Condicional es adjuntado al límite de una Actividad para que el cambio en la *condición* interrumpa la Actividad. Consulte “Interrupción de Actividades mediante Eventos” en la página 95 para una descripción general acerca de cómo los Eventos interrumpen las Actividades.

En BPMN 1.1, el Evento Intermedio de Regla pasó a denominarse Evento Intermedio Condicional, ya que el mismo representa una descripción más adecuada del comportamiento.

Eventos Intermedios de Vínculo

Los Eventos Intermedios de Vínculo son utilizados siempre en pares, con un Evento *origen* y un Evento *destino* (véase Figura 8-55). Informalmente, puede llamarse a los mismos Eventos de Vínculo. Para garantizar la vinculación, tanto el Evento de Vínculo *Origen* como el *Destino* deben tener la misma etiqueta. El Evento de Vínculo *Origen* es un Evento Intermedio *lanzador* (con el símbolo de una flecha con relleno color negro) y el Evento de Vínculo *Destino* es un Evento Intermedio *capturador* (con el símbolo de una flecha con relleno color blanco).

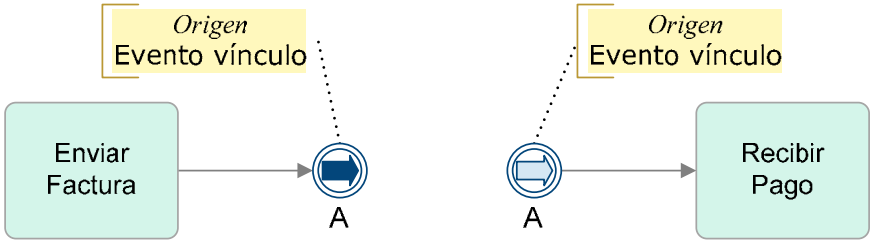


Figura 8-38—Un Flujo de Secuencia virtual es creado

Usando un par de Eventos de Vínculo se crea un Flujo de Secuencia virtual. Esto significa que el diagrama en la Figura 8-55 citado anteriormente, con el par de Eventos de Vínculo ubicados entre las dos Actividades, es equivalente al de la Figura 8-56, más abajo, en el cual un solo Flujo de Secuencia conecta las Actividades.

Figura 8-39—Comportamiento equivalente a un par de Eventos de Vínculo

El alcance de los Eventos Intermedios de Vínculo ha cambiado en BPMN 1.1. Como resultado, los Eventos de Vínculo ahora son sólo utilizados

como Eventos Intermedios y deben existir dentro de un mismo nivel del Proceso. Los Eventos de Vínculo ya no se utilizan para establecer comunicaciones entre Procesos o niveles de Procesos—Los Eventos de Señal, los cuales son nuevos en BPMN 1.1, son utilizados en su lugar.

Los Eventos de Vínculo se utilizan de dos formas:

- Como Conectores 'Off-Page'
- Como Objetos 'Go-To'

Comportamiento de Evento Intermedio de Vínculo

Cuando un token llega a un Evento de Vínculo Origen (desde el Flujo de Secuencia entrante), el Evento es disparado inmediatamente (véase Figura 8-57). Nótese que la distancia entre los Eventos es usualmente mucho mayor que la mostrada en la figura.

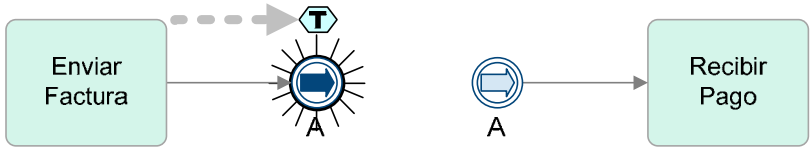


Figura 8-40—Un *token* llega al Evento de Vínculo *lanzador*

Una vez que el Evento de Vínculo Origen es disparado (el *lanzador*), el token inmediatamente salta hacia el (*Destino*) Evento de Vínculo *capturador* (véase Figura 8-58). La llegada del token al Evento de Vínculo Destino inmediatamente activa el Evento.

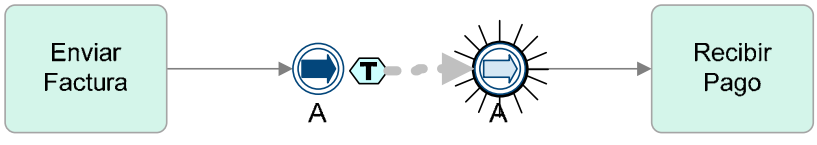


Figura 8-41—El *token* inmediatamente salta hacia el Evento de Vínculo *capturador*

Después de que el Evento de Vínculo Destino es activado, el *token* se mueve inmediatamente hacia el Flujo de Secuencia *saliente* del Evento (véase Figura 8-59).

Figura 8-42—El *token* se mueve hacia el Flujo de Secuencia *saliente*

En relación con la idea de que los Eventos de Vínculo vinculados actúen como un Flujo de Secuencia virtual, el token recorre el Flujo de Secuencia, saltando entre Eventos y moviéndose hacia el segundo Flujo de Se-

cuencia; todo en el mismo tiempo que le llevaría al token recorrer un único Flujo de Secuencia (es decir, instantáneamente).

Off-Page Connectors

Los Eventos de Vínculo pueden mostrar cómo un Flujo de Secuencia continúa de una página a otra. La Figura 8-60 muestra un segmento de un Proceso que puede caber en una página. El extremo derecho de la página tiene el primero de un par de Eventos de Vínculo que conectan ese segmento del Proceso a otro segmento del Proceso en otra página. La Figura 8-61 muestra el Evento de Vínculo que realiza el matcheo.

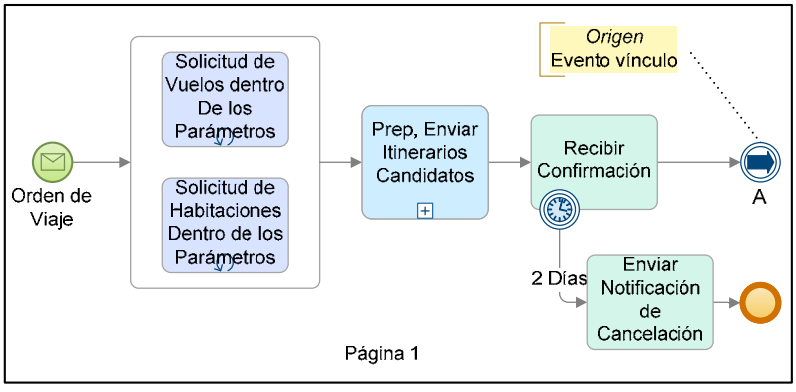


Figura 8-43—Un Evento de Vínculo *Origen* al *final* de una página impresa

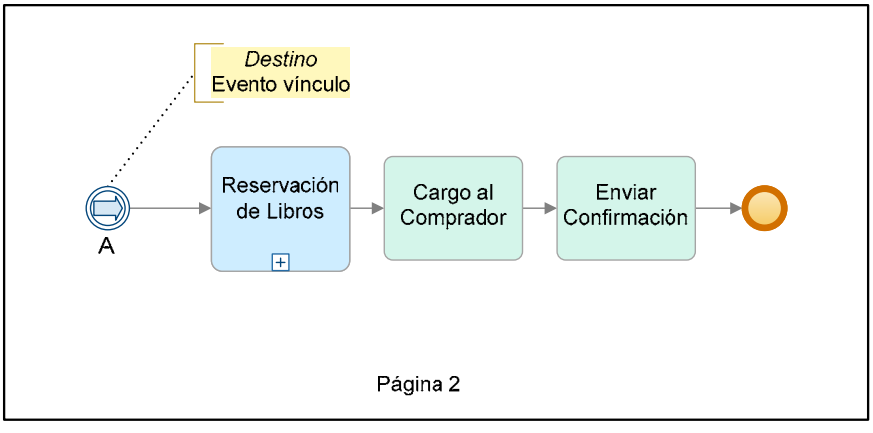


Figura 8-44—Un Evento de Vínculo *destino* al *comienzo* de una página impresa

Entre ellos, el par de Eventos de Vínculo crean un Flujo de Secuencia *virtual* que conecta la última Actividad en la Página 1 (“Recibir Confirmación”) con la primera Actividad en la Página 2 (“Reservaciones de Libros”).

Esto es útil para imprimir diagramas o para metodologías en las cuales se tiene un número limitado de objetos en una página (por ejemplo, IDEF limita el número de Actividades a 5 o 6 por página).

Objetos Go-To

Otro modo de usar Eventos de Vínculo es como "Objetos Go-To" (como se ve en la Figura 8-62).

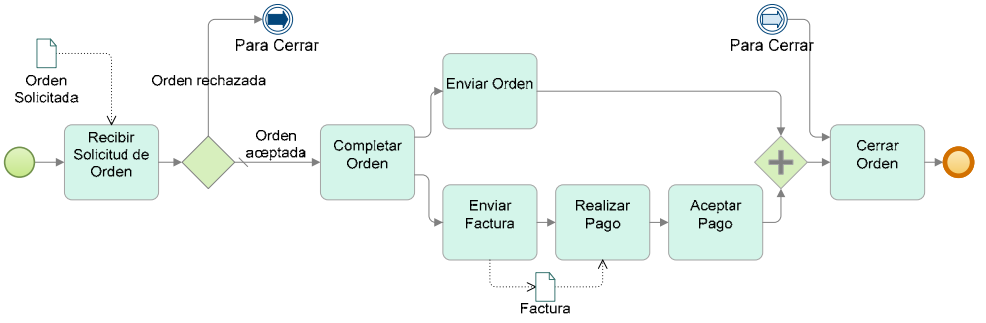


Figura 8-45—Eventos de Vínculo usados como objetos "Go-To"

El Proceso anterior contiene un ejemplo de un flujo que salta de un Evento de Vínculo a otro. Generalmente, los Eventos de Vínculo usados de esta manera evitan Flujos de Secuencia muy largos en diagramas (en general para diagramas que son mucho más grandes que este pequeño ejemplo).

Los Eventos de Vínculo pueden dirigir el flujo hacia abajo, como en el ejemplo, o pueden dirigir el flujo hacia *arriba* de nuevo, creando un loop. La única restricción es que el Flujo de Secuencia *virtual* creado debe ser una conexión válida.

Sólo puede haber un Evento de Vínculo *Destino*, pero pueden haber múltiples Eventos de Vínculo *Origen* vinculados con el mismo Evento de Vínculo *capturador*. Habrá un Flujo de Secuencia virtual separado para cada uno de los Eventos de Vínculo *Origen*.

Eventos Intermedios Múltiples

El Evento Intermedio Múltiple usa el símbolo de un pentágono dentro de la forma del Evento (véase Figura 8-63). Este representa una colección de *disparadores* de Evento Intermedio válidos. Sin embargo, la colección de *disparadores* debe ser o bien todos Eventos *lanzadores* o bien todos Eventos *capturadores*.

Figura 8-46—Eventos Intermedios Múltiples

Al *lanzar* la colección de *disparadores*, entonces el símbolo del pentágono tendrá relleno color negro. Los Eventos Intermedios de Vínculo, al ser un caso especial de Eventos vinculados, no pueden ser utilizados en un

Evento Intermedio Múltiple. En consecuencia, los *disparadores* que son válidos para este tipo de Eventos son: Mensaje, Compensación y Señal. Cuando un *token* al Evento este se activa (lanza) el conjunto entero de *disparadores* en la colección.

Al *capturar* la colección de *disparadores*, el símbolo del pentágono tiene un relleno blanco. Cualquier disparador dentro de la colección de *disparadores* activará el Evento.

El ícono para los Evento Intermedio Múltiple ha cambiado de una estrella de seis puntas a uno en forma de pentágono en BPMN 1.1 (como se muestra en la Figura 8-63, citada anteriormente).

Eventos de Fin

Un Evento de Fin marca cuando un Proceso, o más específicamente un "camino" dentro de un Proceso, finaliza.[24] Un Evento de Fin es un pequeño círculo abierto con una única línea gruesa marcando su límite (véase Figura 8-64).

Figura 8-47—Un Evento de Fin

Al igual que los Eventos de Inicio y Eventos Intermedios, existen diferentes tipos de Eventos de Fin que indican diferentes categorías de *resultados* para el Proceso. Un *resultado* es algo que ocurre al final de un camino particular del Proceso (por ejemplo, un *mensaje* es enviado, o una *señal* es transmitida).

Todos los Eventos de Fin son *lanzadores de resultados* (es decir, no tiene sentido *capturar* al final de un Proceso). En consecuencia, símbolos de los Eventos tienen relleno color negro. Hay ocho tipos diferentes de Eventos de Fin, cada uno con su propia representación gráfica.

Nuevamente, se han agrupado los mismos de acuerdo a su tipo en *básicos* y *avanzados*:

Eventos de Fin Básicos (véase Figura 8-65):

- **Básico**—No se define ningún *resultado*.
- **Mensaje**—Comunicación con otra Entidad de Negocio (*participante* o Proceso).
- **Señal**—Define un Evento "broadcast" el cual cualquier otro Proceso puede ver y al cual puede reaccionar.
- **Terminador**—Detiene todas las Actividades del Proceso, incluso si están en curso en otros hilos de ejecución (Caminos Paralelos).

[24] Los Caminos se denominan a veces *threads*.

Figura 8-48—Tipos de Eventos de Fin *básicos*

Eventos de Inicio Avanzados (véase Figura 8-66):

- **Error**—Un estado final que interrumpirá el Proceso o requerirá corrección.
- **Cancelación**—Usado junto con el Sub-Proceso de Transacción, este Evento causa la cancelación de este tipo de Sub-Procesos. Es el *lanzador* para el *capturador* que está en el límite del Sub-Proceso de Transacción (véase “Transacciones y Compensación” en la página 174).
- **Compensación**—Usado además como parte del comportamiento del Sub-Proceso de Transacción, este Evento *lanza* el *disparador* para deshacer (en caso que la *instancia* necesite ser deshecha). Puede estar vinculado a una Actividad específica, o puede dejarse como un evento general de Compensación, caso en el cual se aplica globalmente a esta *instancia*.
- **Múltiple**—Define dos o más *resultados* Mensaje, Error, Compensación, o Señal (activa todos los *disparadores*).

Figura 8-49—Tipos de Eventos de Fin *avanzados*

En BPMN 1.1, el Evento de Fin de Señal reemplaza al Evento de Fin de Vínculo. El Evento de Señal es un mecanismo más general de comunicación dentro o entre Procesos. Este es un nuevo tipo de Evento en BPMN 1.1. Además, el símbolo del Evento para el Evento de Fin Múltiple cambió

en BPMN 1.1. Otra adición en 1.1 fue la inclusión de un efecto global para el Evento de Fin de Compensación.

Conectando Eventos de Fin

Los Eventos de Fin tienen una restricción similar pero opuesta para las conexiones, al igual que los Eventos de Inicio. Solo los Flujos de Secuencia *entrantes* son permitidos—es decir, los Flujos de Secuencia no pueden salir desde un Evento de Fin—solo pueden llegar a un Evento de Fin (véase Figura 8-67).

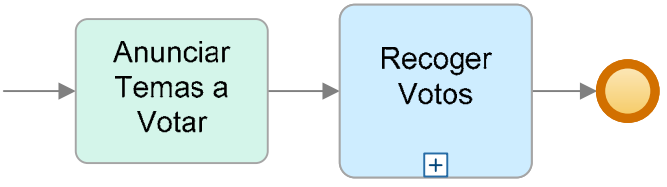


Figura 8-50—Un Evento de Fin usado en un Proceso

Comportamiento de Eventos de Fin

Los Eventos de Fin representan cuando el flujo de un Proceso termina, y por lo tanto, es cuando los tokens se consumen. Cuando un token llega a un Evento de Fin, el resultado del Evento, si lo hubiera, ocurre y el token es consumido (véase Figura 8-68).

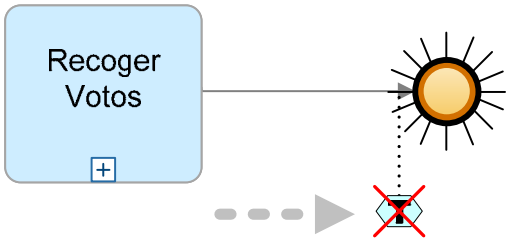


Figura 8-51—El *token* es consumido cuando llega a un Evento de Fin.

El camino del Proceso se completa en el momento que el *token* es consumido por el Evento de Fin. Por supuesto, pueden aparecer múltiples Eventos de Fin dentro de un Proceso (véase Figura 8-69). Debido a esto, es posible tener uno o más hilos de ejecución que continúan incluso después de que el *token* en un camino llagó a un Evento de Fin y fue consumido. Si el Proceso sigue conteniendo un *token* no consumido, entonces el Proceso se mantendrá "activo". Luego de que todos los hilos de ejecución activos hayan llegado a un Evento de Fin, el Proceso se completa. Nótese que algunos caminos de un Proceso pueden no ser atravesados por un *token* durante una ejecución específica de un Proceso.

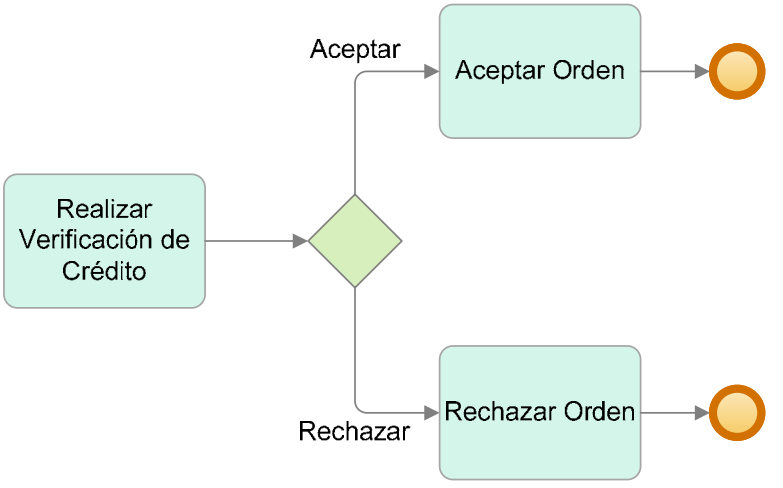


Figura 8-52—Múltiples Eventos de Fin en un Proceso.

Eventos de Fin Básicos

Un Evento de Fin sin *resultado* es conocido como Evento de Fin Básico. Dado que no hay un *resultado* definido, no hay ningún símbolo en el centro de la forma (véase Figura 8-70). Además, un Evento de Fin Básico es siempre usado para marcar el fin de un Sub-Proceso (pasando de un nivel al siguiente).

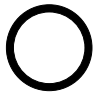


Figura 8-53—Un Evento de Fin Básico

Eventos de Fin Mensaje

El Evento de Fin de Mensaje usa el símbolo de un sobre dentro de la forma del Evento (véase Figura 8-66, citada anteriormente). Este indica que el fin del camino de un Proceso resulta en el envío de un *mensaje* a otro *participante* o Proceso (es decir, no puede comunicarse con un Evento Intermedio Mensaje *capturador* en el mismo Pool). Cuando un *token* llega a un Evento de Fin de *mensaje*, el *mensaje* es enviado y el *token* es consumido (véase Figura 8-71).

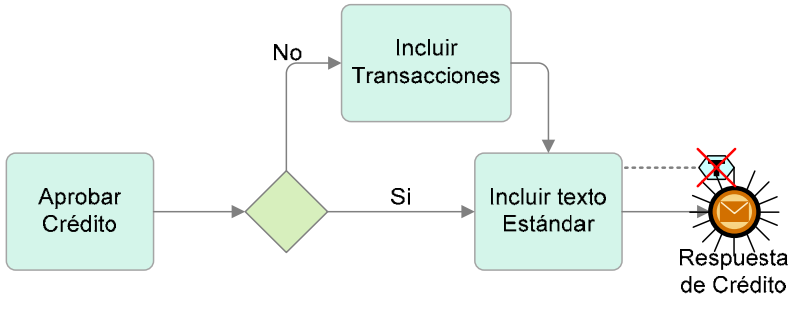


Figura 8-54—Ejemplo de un Evento de Fin de Mensaje

Evento de Fin Señal

El Evento de Fin de Señal usa el símbolo de un triángulo relleno de color negro dentro de la forma del Evento (véase Figura 8-72). Este indica que el fin del camino de un Proceso resulta en la transmisión de una *señal*. Cuando un *token* alcanza el Evento de Fin de Señal, este provoca la transmisión antes de consumir el *token*.

Figura 8-55—Un Evento de Fin Señal

Para más detalles acerca de cómo son utilizados los Eventos de Señal, consulte “Eventos Intermedios Señal” en la página 105.

Evento de Fin Terminador

El Evento de Fin Terminador usa el símbolo de un círculo con relleno color negro dentro de la forma del Evento (véase Figura 8-66, citada anteriormente). Este Evento de Fin tiene una característica especial que lo diferencia de todos los otros tipos de Eventos de Fin. El Evento de Fin Terminador causará la inmediata suspensión de la instancia del Proceso en su nivel actual y para cualquiera de sus Sub-Procesos (aún cuando todavía hay actividad en curso), pero no va a terminar un Proceso padre de nivel superior. Efectivamente, termina el hilos de ejecución actual y causa que todos los otros hilos de ejecución activos finalicen inmediatamente, independientemente de sus respectivos estados.

La Figura 8-73 provee un ejemplo de cómo un Evento de Fin Terminador es usado a menudo. En este Proceso, hay dos hilos de ejecución. El camino superior es efectivamente un loop infinito que envía un mensaje cada siete días. Cuando el camino inferior llega al Evento de Fin Terminador el trabajo del camino superior será detenido, con lo que se detendrá el loop infinito.

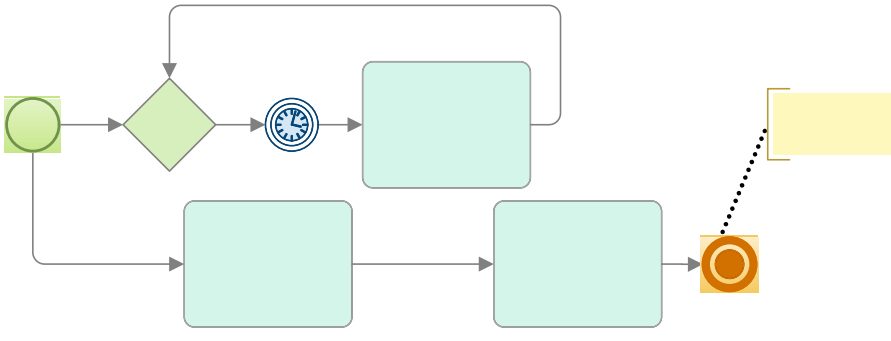


Figura 8-56—Un Evento de Fin Terminador

Los Eventos de Fin Terminadores son utilizados ampliamente ya que pueden facilitar una gran flexibilidad en combinación con otras carac-

terísticas de BPMN. Por ejemplo, un hilos de ejecución separado podría disparar escaladas y alertas no interrumpibles, salvo que el proceso se complete a tiempo (con el Evento de Fin Terminador); luego el otro hilo de ejecución nunca se iniciará. Vea la Figura 8-36 para un ejemplo.

Evento de Fin de Error

El Evento de Fin de Error representa una situación donde el fin del camino de un Proceso resulta en un *error*. Este tipo de Eventos tiene el símbolo de un rayo dentro de la forma del Evento.

La Figura 8-74 muestra un ejemplo donde es usado un Evento de Fin de Error. El *error* lanzado por el Evento será capturado por un Evento Intermedio en un nivel superior (véase Figura 8-75, más abajo).

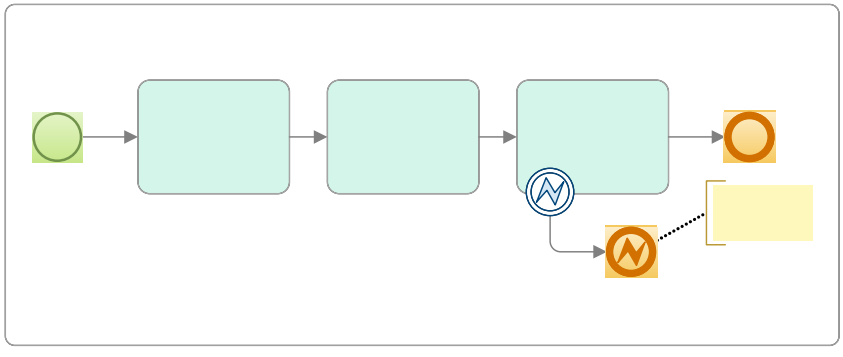


Figura 8-57—Ejemplo de un Evento de Fin de Error.

Además del nombre del Evento, la definición del *error* incluye un *código de error*. Este código de error es usado por Eventos que están esperando para *capturar el error*.

A diferencia de las *señales*, los *errores* no son emitidos a lo largo o a través de Procesos. Los *errores* tienen un alcance específico de visibilidad. Un *error* puede ser visto solamente por un Proceso *padre*. Otros Procesos en el mismo nivel o dentro de Pools diferentes no pueden ver el error. Los errores sólo ascienden en la jerarquía de Procesos. Si llegara a ocurrir que más de un nivel de Proceso fuera superior que el Evento de Fin de Error, entonces el primer nivel que tenga un Evento Intermedio de Error *capturador* adjunto a su límite será interrumpido, aun cuando haya niveles superiores que posiblemente podrían *capturar* el mismo *error*.

La Figura 8-75 muestra cómo los Eventos Intermedios de Error adjuntos al límite de un Sub-Proceso son usados para capturar *errores* lanzados dentro de las actividades internas. El *error* lanzado por el Evento de Fin dentro del Sub-Proceso “Manejar Envío” (como se muestra en la Figura 8-74, citada anteriormente) es atrapado por el Evento Intermedio de Error “Falló Envío”.

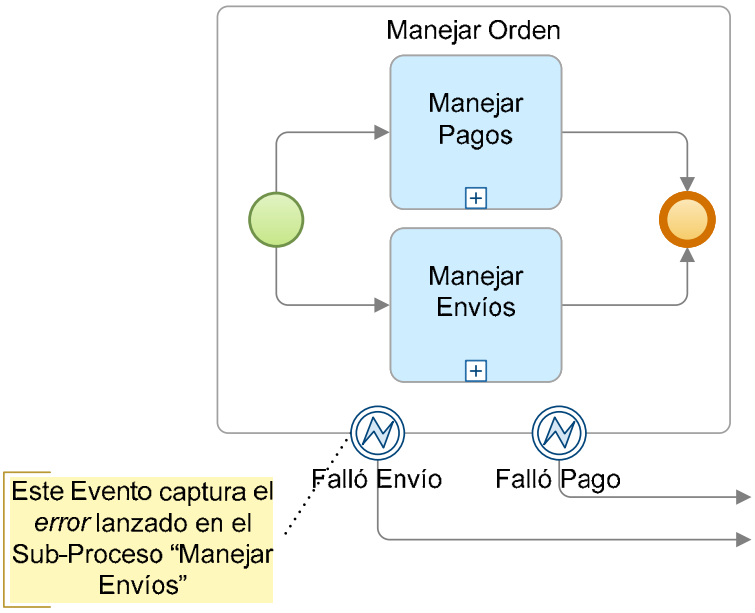


Figura 8-58—Capturando el *error* de un Evento de Fin de Error

Nótese que el Sub-Proceso anterior tiene dos Eventos Intermedios de Error diferentes adjuntos a su límite. Cada uno esta diseñado para manejar un *error* diferente.

Evento de Fin de Cancelación

El Evento de Fin de Cancelación usa el símbolo "X" dentro de la forma del Evento (véase Figura 8-76). Este indica que el fin del camino de un Proceso resulta en la cancelación de un Sub-Proceso de Transacción.

Figura 8-59—Un Evento de Fin de Cancelación

Para *cancelar* el Sub-Proceso de Transacción, el Evento de Fin de Cancelación debe estar contenido dentro del Sub-Proceso o dentro de un Sub-Proceso *hijo* de nivel inferior. Consulte "Transacciones y Compensación" en la página 174 para más detalles acerca de cómo los Sub-Proceso de Transacción son cancelados.

Evento de Fin de Compensación

El Evento de Fin de Compensación indica que el final del camino de un Proceso resulta en la activación de una *compensación*. Es distinguido de otros tipos de Eventos de Fin por el símbolo de "rebobinar" que es colocado dentro de la forma del Evento (véase Figura 8-77).

Figura 8-60—Un Evento de Fin de Compensación

En la definición del Evento de Fin de Compensación, el nombre de una Actividad puede ser identificado como la Actividad que debe ser *compensada*. La Actividad debe estar dentro del Proceso, ya sea en el Proceso de nivel más alto o dentro de un Sub-Proceso. Si la Actividad nombrada fue completada y tiene un Evento Intermedio de Compensación adjunto, entonces esa Actividad será *compensada* (vea más acerca de “Evento Intermedio de Compensación” en la página 114).

Si una Actividad no es identificada en la definición del Evento Intermedio de Compensación, entonces el comportamiento resultante es una *compensación vacía*. Todas las Actividades *completadas* dentro de la instancia del Proceso que tengan un Evento Intermedio de Compensación adjunto son *compensadas*.

Un ejemplo sobre cómo es manejada la *compensación* es detallado en “Transacciones y Compensación” en la página 174.

Evento de Fin Múltiple

El Evento de Fin Múltiple usa el símbolo de un pentágono dentro de la forma del Evento (véase Figura 8-78). Representa una colección de dos o más *resultados* de Eventos de Fin. Los *resultados* pueden ser cualquier combinación de *mensajes*, *errores*, *compensaciones*, y/o *señales*. Cuando un *token* llega al Evento, este dispara (lanza) el conjunto entero de *resultados* en la colección.

Figura 8-61—Un Evento de Fin Múltiple

El ícono para el Evento de Fin Múltiple ha cambiado de una estrella de seis puntas a la forma de un pentágono, en BPMN 1.1 (como se muestra en la Figura 8-78).

Capítulo 9. Gateways

Los Gateways son elementos de modelado que controlan cómo el Proceso diverge o converge—es decir, representan puntos de control para los caminos dentro de los Procesos. Dividen y unifican el flujo de un Proceso (a través del Flujo de Secuencia). Todos los Gateways tienen en común la forma de un diamante (véase Figura 9-1).

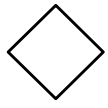


Figura 9-1—Un Gateway

La idea subyacente es que los Gateways son innecesarios si el Flujo de Secuencia no requiere ser controlado. Ejemplos de control de flujo incluye los caminos alternativos de una decisión en un Proceso—por ejemplo, elegir un camino si "Si" y el otro si "No"; o esperar por dos caminos separados a fin de alcanzar cierto punto antes de que el Proceso pueda continuar (un punto de sincronización). Ambos ejemplos usarían un Gateway para controlar el flujo.

Un Gateway *divide* el flujo cuando este tiene múltiples Flujos de Secuencia *salientes* y unificará el flujo cuando este tiene múltiples Flujos de Secuencia *entrantes* (véase Figura 9-2). Un solo Gateway puede tener múltiples Flujos de Secuencia tanto *entrantes* como *salientes* (es decir, ambos *unifican* y *dividen* al mismo tiempo).

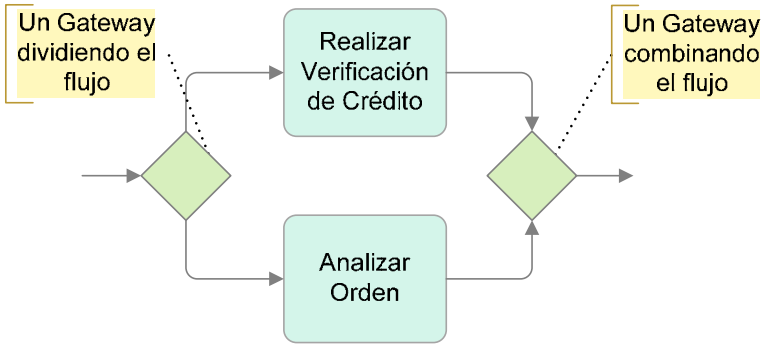


Figura 9-2—Gateways *dividiendo* y *unificando* el flujo de Proceso

Dado que hay diferentes formas de controlar el flujo de un Proceso, hay diferentes tipos de Gateways. Mientras que todos comparten la misma forma básica (un diamante), los símbolos internos diferencian el comportamiento que cada uno representa. Los dos Gateways comúnmente más usados son el Exclusivo y Paralelo. El Gateway Evento es comúnmente menos usado (en este punto), pero se cree que se volverán más importante cuando los modeladores se familiaricen (eduquen) con sus capacida-

des.[25] Estos tres Gateways integran el set de Gateways *básicos* (véase Figura 9-3).

Gateways *Básicos*:

- **Exclusivo**—*Dividiendo*: el Gateway enviará un *token* a través de solo un camino *saliente* (exclusivamente) dependiendo de la evaluación de las *condiciones* del Flujo de Secuencia. *Unificando*: el Gateway "hará pasar" cualquier *token* de cualquiera de los caminos *entrantes*.
- **Evento**—*Dividiendo*: el Gateway envía un *token* a través de solo un camino *saliente* (exclusivamente) dependiendo de la ocurrencia de un Evento específico (por ejemplo, el arribo de un *mensaje*). *Unificando*: el Gateway "hará pasar" cualquier *token* de cualquiera de los caminos *entrantes*.
- **Paralelo**—*Dividiendo*: el Gateway enviará un *token* a través de todos los caminos *salientes* (en paralelo). *Unificando*: el Gateway esperará por un *token* de todos los caminos *entrantes*.

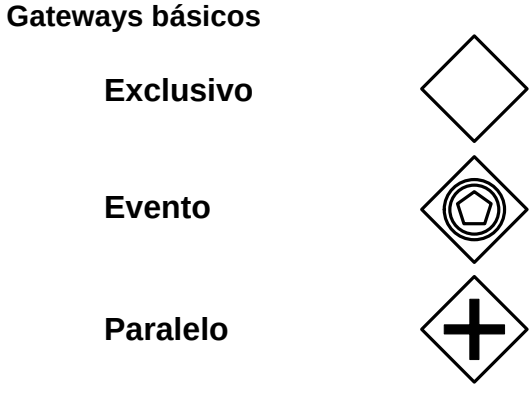


Figura 9-3—Gateways básicos

Los dos tipos restantes de Gateways (Inclusivos y Complejos) componen la lista de Gateways *avanzados* (véase Figura 9-4).

Gateways *Avanzados*:

- **Inclusivos**—*Dividiendo*: el Gateway enviará un *token* de uno a todos los caminos *salientes* (inclusivamente) dependiendo de la evaluación de todas las condiciones del Flujo de Secuencia. *Unificando*: el Gateway esperará por un *token* de uno a todos los caminos *entrantes* dependiendo de cuales caminos esperan un *token*.
- **Complejos**—*Dividiendo*: el Gateway enviará un *token* de uno a todos los caminos *salientes* (inclusivamente) dependiendo de la evaluación de una única condición del Gateway. *Unificando*: el Gate-

[25] Aunque el Gateway Basado en Eventos no es ampliamente usado en la actualidad, se cree que esto es debido al hecho de que pocas personas lo comprenden. Este es incluido en el conjunto *básico* dado que es una construcción muy útil.

way esperará por un *token* de uno a todos los caminos *entrantes* dependiendo de la evaluación de una única *condición* del Gateway.

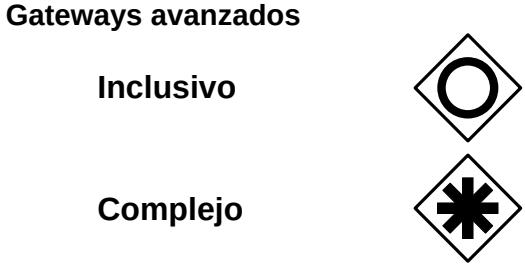


Figura 9-4—Gateways *Avanzados*

El tipo (dividiendo y unificando) para un solo Gateway debe ser correspondido—es decir, un Gateway no puede ser Paralelo en el lado de entrada, y Exclusivo en el lado saliente. Nótese que los Flujos de Secuencia entrantes y salientes pueden conectarse a cualquier punto en el límite del Gateway. No están obligados a conectarse a cualquier punto predeterminado de la forma de diamante perteneciente al Gateway, como las esquinas del mismo (aunque algunos fabricantes podrían imponer esta restricción en sus herramientas de modelado).

Cuatro de los Gateways (Exclusivo, Paralelo, Inclusivo, y Evento) tienen un comportamiento predefinido (es decir, maneras de controlar el flujo). El quinto tipo, el Gateway Complejo, provee un mecanismo para que un Modelador especifique (programe) cualquier comportamiento deseado.

Gateways Exclusivos

Los Gateways Exclusivos son puntos dentro de un Proceso donde hay dos o más caminos alternativos. Piense en ellos como una "bifurcación en el camino" del Proceso—usualmente, ellos representan una decisión. Al igual que todos los Gateways, el Gateways Exclusivo, utiliza una forma de diamante. Los criterios para la decisión, la cual representa el Gateways Exclusivo, existen como *condiciones* en cada uno de los Flujos de Secuencia *salientes*. Dependiendo del nivel de detalle del modelo, las condiciones están definidas como texto regular (por ejemplo, "Si" o "No") o como *expresiones* (por ejemplo, "monto_de_orden > $100,000.00").

Como todos los Gateways, los Gateways Exclusivos tienen un símbolo interno (una "X"). Sin embargo, la exhibición de este símbolo es opcional y, de hecho, la presentación por defecto del Gateway Exclusivo es sin el símbolo (véase Figura 9.5). [26]

[26] Todos los ejemplos de Procesos en este libro usarán el Gateway Exclusivo sin el símbolo interno.

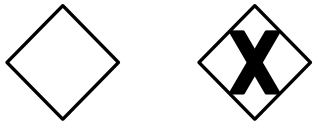


Figura 9-5—Un Gateway Exclusivo con y sin símbolo interno

Comportamiento Divisor de un Gateway Exclusivo

Los Gateways Exclusivos dividirán el flujo cuando tengan dos o más caminos *salientes*. El Proceso (un *token*) continuará a través de uno solo de ellos. De esta manera, al considerar los *tokens*, incluso cuando existan múltiples Flujos de Secuencia *saliente*, solo un *token* es pasado a través de uno de esos Flujos de Secuencia.

Cuando un *token* llega a un Gateway Exclusivo (véase Figura 9-6), ocurre una inmediata evaluación de las condiciones que hay en el Flujo de Secuencia *saliente* del Gateway. Una de esas *condiciones* siempre debe ser evaluada como verdadera.

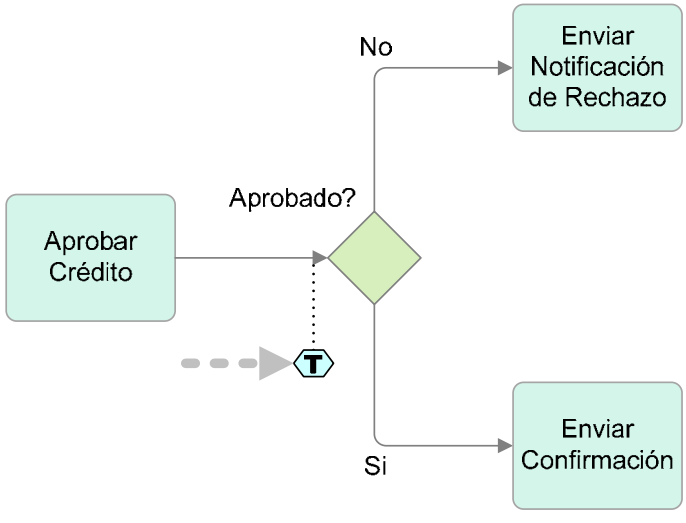


Figura 9-6—Un *token* llega a un Gateway Exclusivo y es dirigido hacia uno de los Flujos de Secuencia *salientes*

La evaluación de las *condiciones* del Flujo de Secuencia es hecha en realidad de a una por vez, en el orden en que son listadas en los atributos del Gateway (no necesariamente son presentadas en el diagrama en orden arbitrario). El *token* se mueve hacia el primer Flujo de Secuencia con la *condición* que evalúa en *verdadero*. Así que si sucede que más de una *condición* es *verdadera*, entonces después de la primera identificada, el Gateway ignorará todas las *condiciones verdaderas* restantes—nunca las evalúa. Utilice un Gateway Inclusivo si el Proceso necesita activar más de un Flujo de Secuencia *saliente* (consulte página 142).

La *condición* que se evalúa a *verdadero* suele ser diferente cada vez que el Proceso es ejecutado, o cada vez que las *condiciones* de un Gateway Ex-

clusivo son evaluadas (por ejemplo, si el Gateway es parte de un loop). Por ejemplo, en la Figura 9-7, si la *condición* para el Flujo de Secuencia *saliente* inferior ("Si") del Gateway es *verdadera*, entonces el *token* será enviado por ese camino.

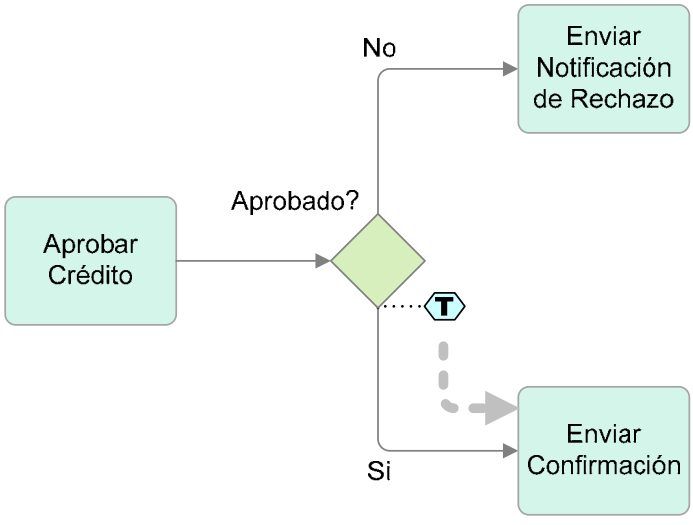


Figura 9-7—Un Gateway Exclusivo usando *condiciones* para bifurcar el Flujo de Secuencia

Quizás la próxima vez que el *token* alcance el Gateway Exclusivo, la *condición* para el Flujo de Secuencia superior ("No") sea verdadera y el *token* será enviado por ese camino.

Como se mencionó anteriormente, una de las *condiciones* en el Flujo de Secuencia *saliente* debe evaluarse a *verdadero*. Esto significa que el modelador debe definir las *condiciones* para cumplir este requerimiento. Si las *condiciones* son complicadas, podría no ser obvio que al menos una *condición* será *verdadera* para todas las ejecuciones del Proceso. Si resulta que ninguna *condición* es *verdadera*, entonces el Proceso quedará atascado en el Gateway y no se completará con normalidad.

Mejor Práctica:

***Usar una condición por defecto**—una forma que tiene el modelador de asegurarse que el Proceso no se quedará atascado en un Gateway Exclusivo es utilizando una* condición *por defecto para uno de los Flujos de Secuencia* salientes *(véase Figura 9-8). Esto crea un Flujo de Secuencia Predeterminado (consulte en la página 164). El Flujo de Secuencia Predeterminado es elegido si todas las otras* condiciones *de Flujos de Secuencia se evalúan en* falso.

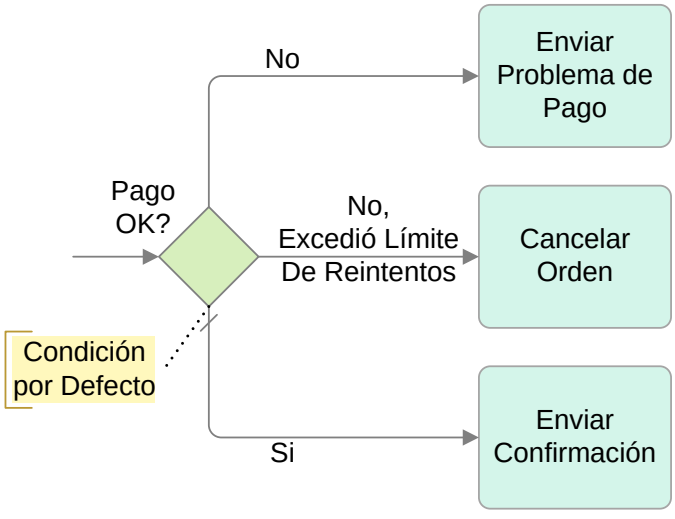


Figura 9-8—Un Gateway Exclusivo con un Flujo de Secuencia por Defecto

Comportamiento Unificador de un Gateway Exclusivo

Los Gateways Exclusivos pueden también unificar Flujos de Secuencia. Esto es, pueden tener múltiples Flujos de Secuencia *entrantes*. Sin embargo, cuando un *token* llega al Gateway Exclusivo, no hay evaluación de *condiciones* (en el Flujo de Secuencia *entrante*), ni hay algún tipo de sincronización de *tokens* que podrían venir de otro Flujo de Secuencia *entrante*. El *token*, cuando llega, inmediatamente se mueve hacia el Flujo de Secuencia *saliente*. Efectivamente, existe un inmediato "pasaje directo" del *token* (véase Figura 9-9).

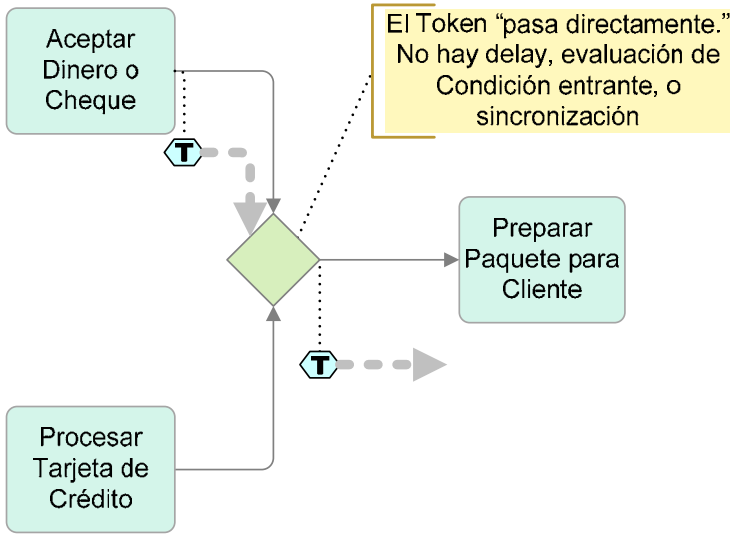


Figura 9-9—Un *token* "pasa directamente" uno de los Flujos de Secuencia *entrantes* de un Gateway Exclusivo

Nótese que cuando el Gateway Exclusivo tiene sólo un Flujo de Secuencia *saliente*, entonces ese Flujo de Secuencia no tendrá una *condición*. Si el Gateway tiene también múltiples Flujos de Secuencia *saliente*, entonces el Flujo de Secuencia tendrá *condiciones* a ser evaluadas como se describió en la sección anterior.

Si otro *token* llega desde otro Flujo de Secuencia *entrante*, entonces también pasará directamente de un lado a otro, activando el Flujo de Secuencia *saliente* nuevamente—es decir, habrán dos *instancias* del "Preparar Paquete para Cliente" en la Figura 9-9, citada anteriormente.

La mayor parte del tiempo, el modelador probablemente no necesitará preocuparse por este comportamiento ya que un Gateway Exclusivo *bifurcador* (*divisor*) normalmente precede la *unificación*. En tales condiciones, solo un *token* llegará a la *unificación* de cualquier manera. Algunas veces vendrá desde un camino y otras veces desde otro.

Por otra parte, puede haber situaciones en que hay (potencialmente) múltiples *tokens* llegando a un Gateway. En tales circunstancias, si el comportamiento esperado es que sólo el primer *token* debe pasar (ignorando el resto), entonces es necesario un Gateway Complejo (véase Figura 9-32).

El comportamiento de los otros tipos de Gateways usualmente incluye esperar por (sincronizar) otros *tokens* de otros Flujos de Secuencia *entrantes* (como se debatirá más tarde), pero este no es el caso de los Gateways Exclusivos.

Gateways Exclusivos Basados en Eventos

Los Gateways Exclusivos Basados en Eventos representan un punto de bifurcación alternativo donde la decisión está basada en dos o más Eventos que pueden ocurrir, en vez de *condiciones* orientadas a datos (como en un Gateway Exclusivo). Usualmente se hará referencia a "Gateway Evento", al referirse a este Gateway. Ya que más de un Evento controla el Gateway Basado en Eventos, el símbolo dentro del diamante es el mismo que el utilizado en el Evento Intermedio Múltiple (un pentágono rodeado por dos círculos concéntricos—véase Figura 9-10).

Figura 9-10—Un Gateway Evento

Los Procesos que involucran comunicaciones con un participante de negocio o alguna entidad externa, a menudo necesitan este tipo de comportamiento. Por ejemplo, si el participante de negocio envía un *mensaje* que dice "Si, hagámoslo" el Proceso se dirigirá hacia un camino. Si, por otro lado, el participante de negocio envía un *mensaje* que dice "No gracias", el Proceso necesita dirigirse hacia otro camino. Y si el participante de negocio no responde, entonces un Temporizador está obligado a prevenir

que el Proceso caiga en deadlock. Los Gateway Basado en Eventos permiten este tipo de flexibilidad. Varios ejemplos de Gateway Basado en Eventos son usados en los capítulos introductorios, como los mostrados en las Figuras 5-7 y la Figura 5-12.

Nótese que el símbolo para el Gateway Evento Exclusivo cambió en BPMN 1.1. Esto se debe a que el símbolo del Gateway es un Evento Intermedio Múltiple, el cual se ha convertido en un pentágono de una estrella de seis puntas).

Comportamiento Divisor de un Gateway Evento

El Gateway Evento es único en BPMN en el hecho de que su comportamiento normal esta realmente determinado por una combinación de *objetos de flujo*. El Gateway por si solo no es suficiente para lograr la *división* exclusiva del flujo. El Gateway Evento usa además una combinación de Eventos Intermedios para crear el comportamiento. Estos Eventos, los cuales deben ser de la variedad de *captura*, son los primeros objetos conectados por el Flujo de Secuencia *saliente* del Gateway (véase Figura 9-11—el Grupo que rodea la configuración del Gateway provee énfasis).

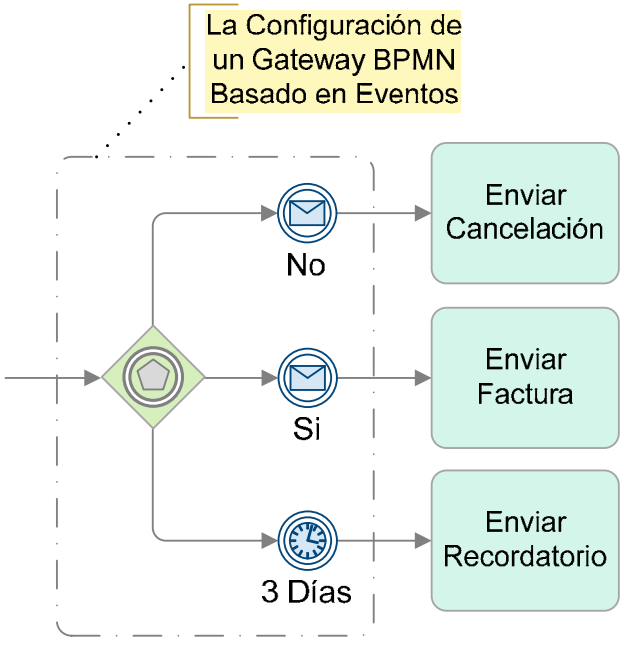


Figura 9-11—Una configuración de un Gateway Evento (con un Grupo añadido para hacer énfasis)

Los siguientes Eventos Intermedios *capturadores* son válidos en un Gateway Evento Exclusivo:

- Mensaje
- Temporizador
- Condicional

- Señal

En lugar de los Eventos Intermedios de Mensaje pueden ser usados también Tareas de Recepción. Sin embargo, no es posible combinar ambos (Evento Intermedio de Mensaje y Tareas de Recepción).

Cuando un *token* llega al Gateway (véase Figura 9-12) inmediatamente atravesará el Gateway y luego se dividirá enviando un *token* a cada uno de los Eventos que siguen al Gateway (véase Figura 9.13).

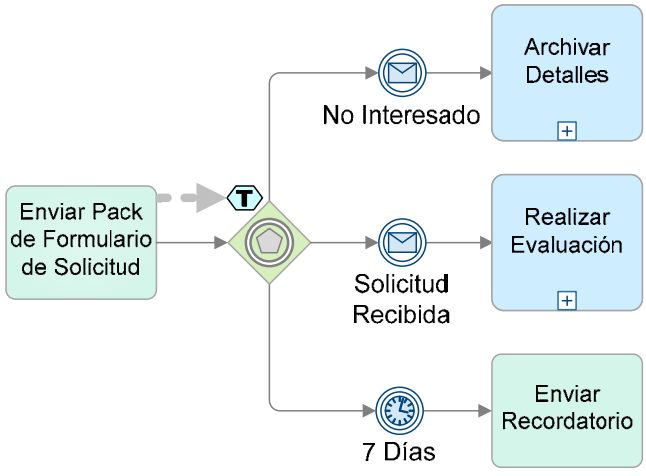


Figura 9-12—Un *token* llega a un Gateway Evento

Dado que todos los Eventos Intermedios son Eventos capturadores, los *tokens* esperarán allí hasta que uno de los Eventos sea disparado.

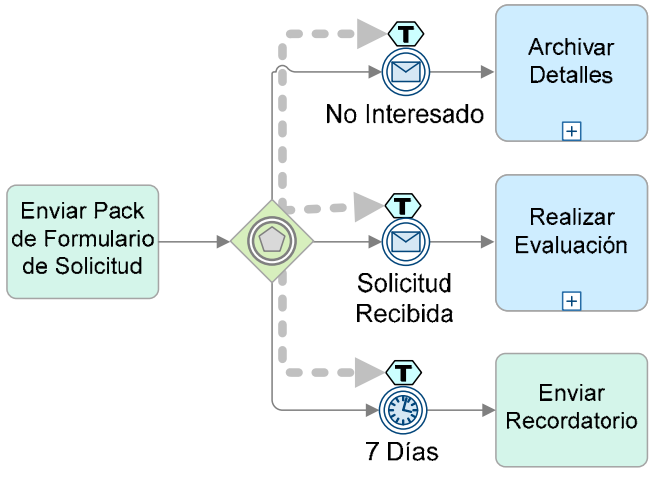


Figura 9-13—El *token* se divide y envía a todos los Flujos de Secuencia *salientes* del Gateway

La situación aquí es similar a una Actividad con Eventos Intermedios adjuntos (consulte “Interrupción de Actividades mediante Eventos” en la página 96). Los Eventos Intermedios que son parte de la configuración del Gateway se ven involucrados en una *condición de carrera*. Cualquiera

de ellos que termine primero (se dispare) ganará la *carrera* y tomará el control del Proceso junto con su *token*. Luego el *token* continuará inmediatamente hacia su Flujo de Secuencia *saliente* (desde ese Evento Intermedio—véase Figura 9-14). Los *tokens* esperando en los otros Eventos se consumen inmediatamente, inhabilitando esos caminos.

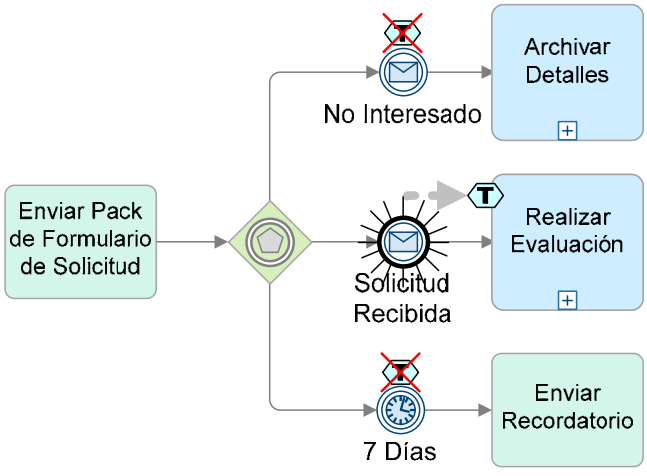


Figura 9-14—El Evento que es disparado "elige" el camino

Siempre existe el riesgo de que el Evento elegido de la configuración del Gateway Evento pueda no ocurrir para una *instancia* de Proceso dada. Si ninguno de los Eventos son disparados, no importa por qué razón, entonces el Proceso quedará atascado en el Gateway y no se completará con normalidad. Si ninguno de los otros Eventos se activa, entonces el *temporizador* eventualmente se disparará, permitiendo al Proceso continuar (y encargarse de por qué los otros Eventos nunca ocurrieron).

Mejor Práctica: ***Use un Evento Intermedio Temporizado con un Gateway Evento****—Una forma que tiene el modelador para asegurarse de que el Proceso no se quedará atascado en el Gateway Exclusivo Basado en Eventos es utilizando un Evento Intermedio Temporizado como una de las opciones del Gateway (véase Figura 9-11, citada anteriormente).*

Comportamiento Unificador de un Gateway Evento

A diferencia de todos los otros Gateways, el Gateway Evento es siempre utilizado para *dividir* el flujo de Proceso. Sin embargo, al mismo tiempo, los mismos *unifican* el flujo de Proceso. El comportamiento unificador del Gateway Evento es exactamente el mismo que el comportamiento unificador del Gateway Exclusivo (consulte "Comportamiento Unificador de un Gateway Exclusivo" en la página 133).

Gateways Paralelos

Los Gateways Paralelos insertan una división en el Proceso para crear dos o más hilos de ejecución paralelos. Estos pueden además unificar

caminos paralelos. El símbolo “+” es usado para identificar este tipo de Gateway (véase Figura 9-15).

Figura 9-15—Un *Gateway Paralelo*

Gateway Paralelo Dividiendo

Los Gateways Paralelos dividirán el flujo cuando tengan dos o más caminos *salientes*. Cuando un *token* llega al Gateway Paralelo (véase Figura 9-16), no hay evaluación de ninguna *condición* en el Flujo de Secuencia *saliente* (a diferencia del Gateway Exclusivo).

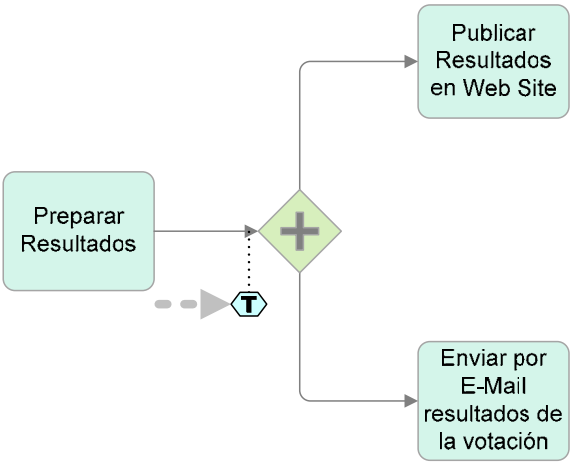


Figura 9-16—Un solo *token* llegando al Gateway Paralelo

Por definición, el Gateway Paralelo creará caminos paralelos. Esto significa que el Gateway creará un número de *tokens* equivalente al número de Flujos de Secuencia *salientes*. Un *token* se mueve hacia cada uno de estos Flujos de Secuencia *salientes* (véase Figura 9-17). No hay delay entre la llegada del *token* al Gateway y los *tokens* abandonando el Gateway.

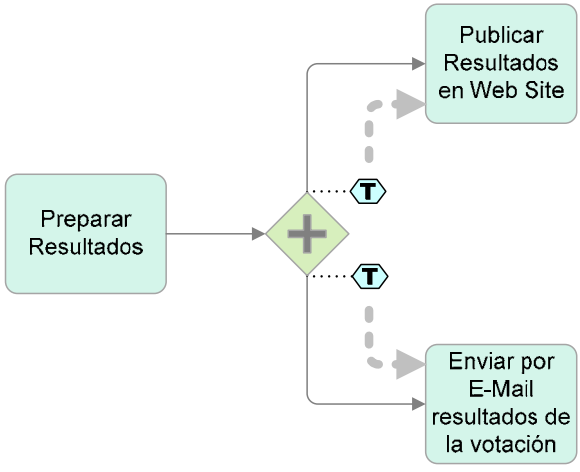


Figura 9-17—Dos *tokens* abandonando un Gateway Paralelo

Gateway Paralelo Unificando

Utilice un Gateway Paralelo unificador cuando caminos paralelos requieran ser sincronizados antes que el Proceso pueda continuar. Para sincronizar el flujo, el Gateway Paralelo esperará que un *token* llegue desde cada uno de los Flujos de Secuencia *entrantes*. Cuando el primer *token* llega, no hay evaluación de *condición* para el Flujo de Secuencia *entrante*, pero el *token* es "retenido" en el Gateway y no continúa (véase Figura 9-18).

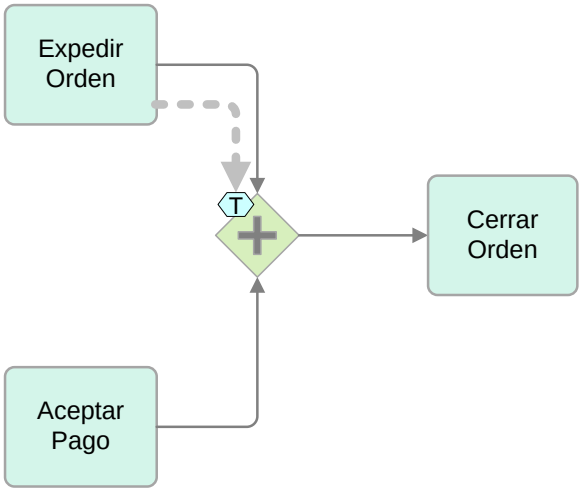


Figura 9-18—Un Gateway Paralelo *unificando* el Flujo de Secuencia

Estos *tokens* "retenidos" continuarán así hasta que llegue un *token* desde todos los Flujos de Secuencia *entrantes* (véase Figura 9-19).

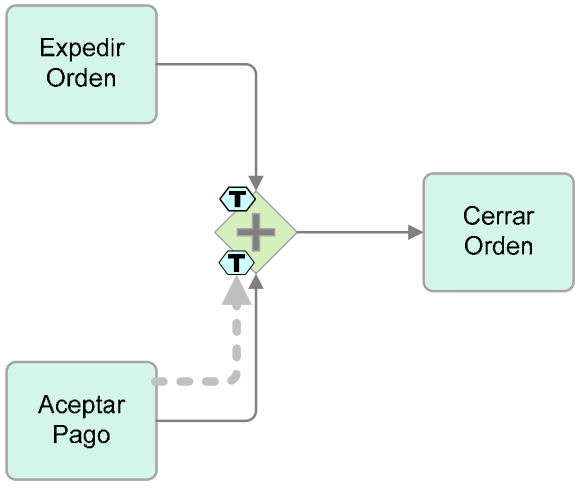


Figura 9-19—Los *tokens* son *unificados* en el Gateway Paralelo

Cuando todos los *tokens* hayan llegado, entonces los mismos son unificados y un *token* se mueve hacia el Flujo de Secuencia *saliente* (véase Figura 9-20).

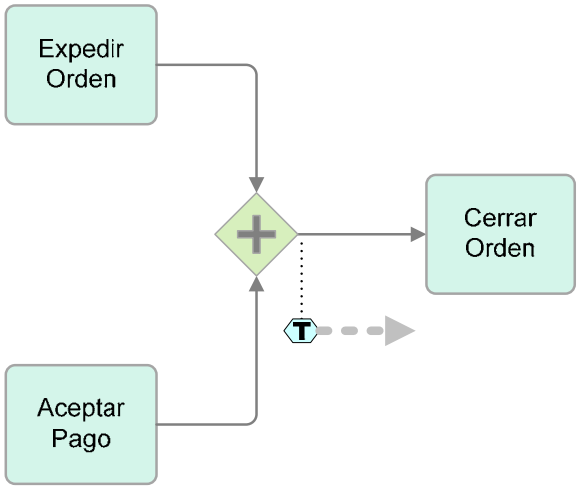


Figura 9-20—El *token* puede seguir su camino una vez que todos los *tokens entrantes* son recibidos

Si hay más de un Flujo de Secuencia *saliente* a la salida de un Gateway Paralelo *unificador*, entonces cada uno recibirá un *token* cuando todos los Flujos de Secuencia *entrantes* hayan arribado. En efecto, esto creará otro set de hilos de ejecución paralelos.

Mejor Práctica:

Asegúrese que el número de* Flujos de Secuencia entrantes *es correcto para el Gateway Paralelo *—El punto clave es practicar con cautela, asegurando que los Gateways Paralelos unificadores tienen el número correcto de Flujos de Secuencia entrantes—especialmente cuando son usados en conjunto con otros Gateways. Como guía, los modeladores deberían hacer concor-*

dar los Gateways Paralelos unificadores y divisores (si el comportamiento deseado es unificarlos nuevamente). [27].

Si el número de Flujos de Secuencia *entrantes* no concuerda con el número de *tokens* que realmente llegarán, entonces el proceso se atascará en el Gateway Paralelo, esperando por un *token* que nunca se materializará—es decir, el Proceso puede no completarse nunca correctamente. La Figura 9-21 provee un ejemplo de tal configuración de Proceso.

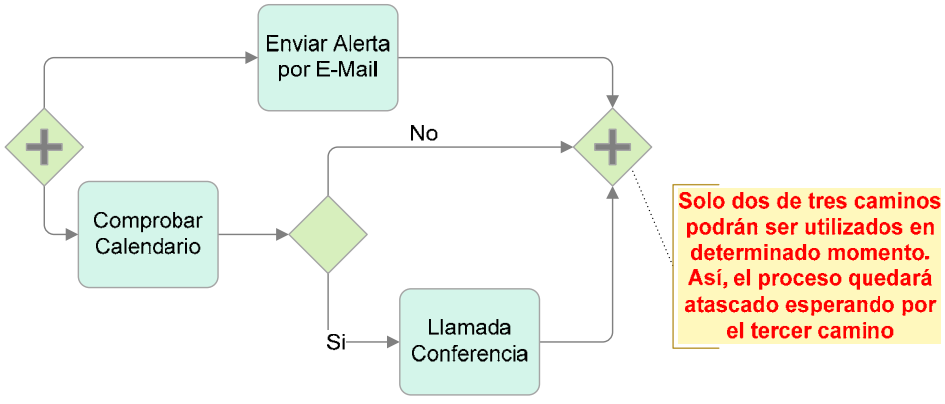


Figura 9-21—Un ejemplo de uso incorrecto de un Gateway Paralelo *Unificador*

La forma correcta de modelar esta situación es mostrada en la Figura 9-22. Esto implica colocar un Gateway Exclusivo antes que el Gateway Paralelo unificador, para reducir el número de Flujos de Secuencia *entrantes* (en este caso, hasta dos).

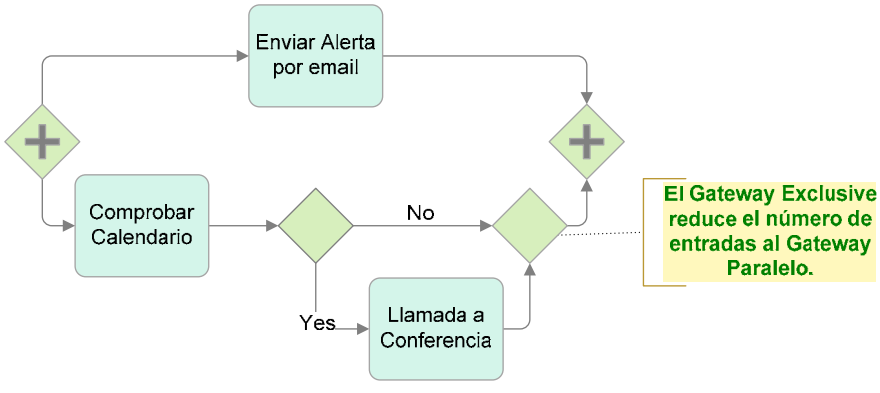


Figura 9-22—Haciendo concordar el número de entradas en un Gateway Paralelo *unificador*

[27] Por supuesto, es posible dividir el flujo en un Gateway Paralelo y nunca volver a recombinar los **hilos**. Al final, cada **hilo** finalizará en un Evento de Fin.

Gateways Inclusivos

Los Gateways Inclusivos soportan las decisiones donde es posible más de un resultado en el punto de decisión. El símbolo "O" identifica este tipo de Gateway (véase Figura 9-23).

Figura 9-23—Un Gateway Inclusivo

Comportamiento Divisor de un Gateway Inclusivo

Al igual que los Gateways Exclusivos, un Gateway Inclusivo con múltiples Flujos de Secuencia salientes crea caminos alternativos basados en las *condiciones* de estos Flujos de Secuencia. La diferencia es que los Gateways Inclusivos activan uno o más caminos;[28] mientras que el Gateway Exclusivo activará solamente un Flujo de Secuencia *saliente* y el Gateway Paralelo activará todos los Flujos de Secuencia *salientes*.

Cuando un *token* llega a un Gateway Inclusivo (véase Figura 9-24), se da una inmediata evaluación de todas las *condiciones* existentes en el Flujo de Secuencia *saliente* del Gateway.

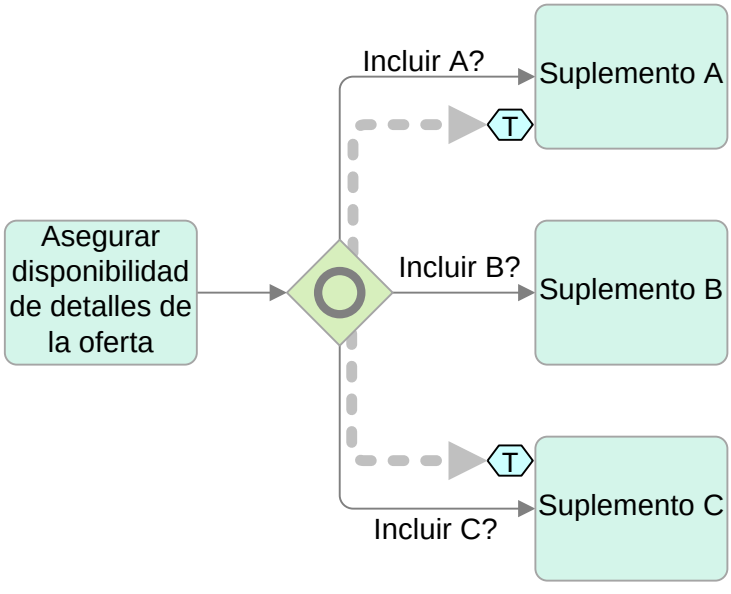


Figura 9-24—Un *token* llega a un Gateway Inclusivo

[28] Existe otro mecanismo para crear una división inclusiva. Este es logrado utilizando múltiples Flujos de Secuencia Condicionales salientes de una Actividad (consulte Flujo de Secuencia Condicional en la página 187).

Cada *condición* que se evalúa a *verdadero* resultará en un *token* moviéndose hacia ese Flujo de Secuencia. Al igual que el Gateway Exclusivo, al menos una de esas *condiciones* debe evaluarse a *verdadero*. Esto significa que cualquier combinación de Flujo de Secuencia, de uno de ellos a todos, puede recibir un *token* cada vez que el Gateway es utilizado. En la Figura 9-25, si la *condición* para el Flujo de Secuencia *saliente* superior ("Incluir A?") y la condición para el Flujo de Secuencia *saliente* intermedio ("Incluir B?") son verdaderas, entonces los *tokens* se moverán hacia ambos caminos.

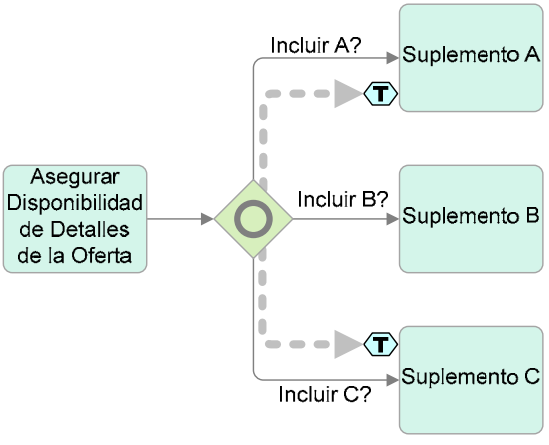


Figura 9-25—Un *token* es dirigido hacia uno o más de los Flujos de Secuencia *salientes*

Otra vez que un *token* alcance el Gateway Exclusivo, podría decidir que la condición del Flujo de Secuencia *saliente* intermedio ("Incluir B?") es la única que se evalúa a *verdadero*, conduciendo a un *token* a continuar por ese camino.

Como se describió para en "Comportamiento Divisor de un Gateway Exclusivo", anteriormente (véase página 131), una de las *condiciones* en el Flujo de Secuencia *saliente* debe evaluarse a *verdadero* o el Proceso quedará atascado en el Gateway y no se completará con normalidad.

Mejor Práctica: ***Usar una condición por defecto*** *—Una forma que tiene el modelador para asegurarse que el Proceso no se quedará atascado en el Gateway Inclusivo, es utilizando una condición por defecto para uno de los Flujos de Secuencia salientes. Este Flujo de Secuencia por Defecto se evaluará siempre a verdadero si todas las otras condiciones de Flujos de Secuencia resultan ser falsas (véase "Flujo de Secuencia Predeterminado" en la página 164).*

Comportamiento Unificador de un Gateway Inclusivo

El comportamiento *unificador* del Gateway Inclusivo es una de las partes más complejas de entender de BPMN. El Gateway sincronizará el flujo del

Proceso al igual que el Gateway Paralelo (consulte página 137), pero lo hace de un modo que refleja la salida del Gateway Inclusivo *divisor*. Esto es, el Gateway sincronizará de uno a todos los Flujos de Secuencia *entrantes* del Gateway. El Gateway determinará cuántos caminos se espera que tengan un *token*.

Mediante un ejemplo se ilustrará cómo el Gateway Inclusivo realiza esta sincronización. Cuando el primer *token* llega al Gateway (véase Figura 9-26), el mismo "mirará" hacia cada uno de los otros Flujos de Secuencia *entrantes* para ver si hay un *token* que podría llegar en un momento posterior. En la figura, hay otro *token* en el camino inferior, parado en la Tarea "Suplemento C", pero no hay un *token* en el camino intermedio (en la Tarea "Suplemento B" desde la que pueda llegar).

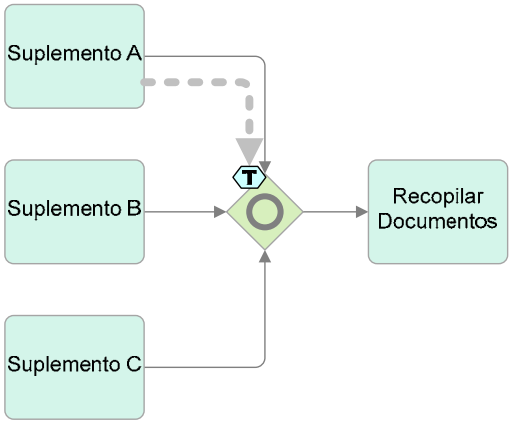


Figura 9-26—Un *token* llega a un Gateway Inclusivo

De esta manera, el Gateway mantendrá el primer *token* que arribó en el camino superior, hasta que otro *token* del camino inferior llegue (véase Figura 9-27).

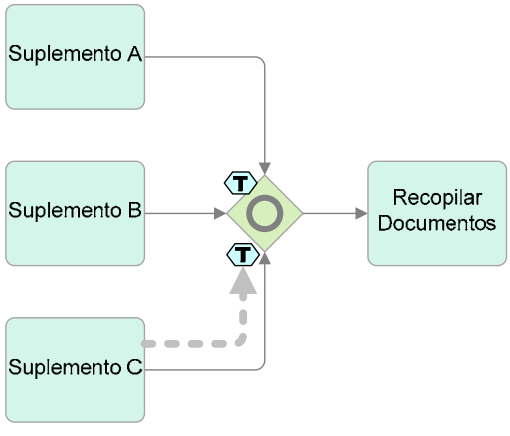


Figura 9-27—Un segundo *token* llega a un Gateway Inclusivo, sincronizando el flujo

Cuando todos los *tokens* esperados hayan arribado al Gateway, el flujo de Proceso es sincronizado (los *tokens entrantes* son *unificados*) y entonces un *token* se mueve hacia el Flujo de Secuencia *saliente* del Gateway (véase Figura 9-28).

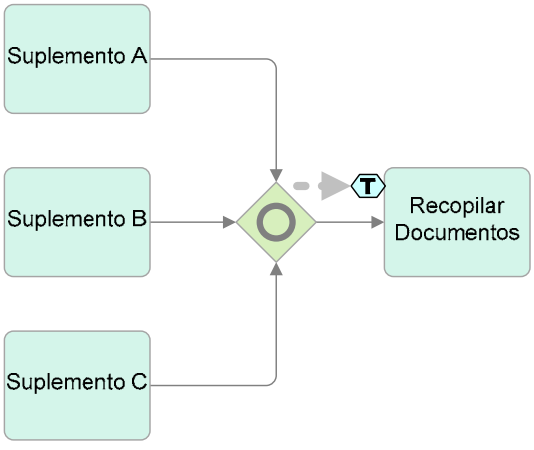


Figura 9-28—El *token* es enviado hacia el Flujo de Secuencia *saliente*

La Figura 9-29 muestra una *instancia* diferente del Gateway, donde un *token* llega desde el camino intermedio mientras no hay ningún otro *token* esperando otro que pueda llegar, ya sea en el camino superior o en el inferior. En ese caso, el *token* se moverá inmediatamente hacia el Flujo de Secuencia *saliente* (como en la Figura 9-28, citada anteriormente).

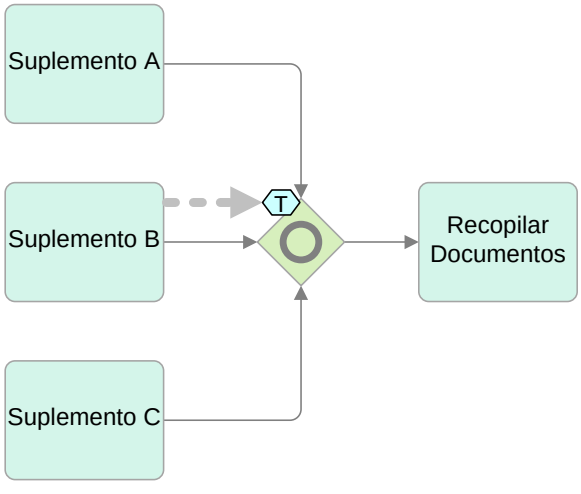


Figura 9-29—Un *token* llega a un Gateway Inclusivo que no necesita sincronización

Nótese que cuando el Gateway Inclusivo tiene solo un Flujo de Secuencia *saliente*, entonces ese Flujo de Secuencia no tendrá una *condición*. Si el Gateway tiene además múltiples Flujos de Secuencia *salientes*, entonces

el Flujo de Secuencia tendrá *condiciones* a ser evaluada como fue descrito en la sección anterior.

Un Gateway Inclusivo que *unifica* el flujo suele ir de la mano con su correspondiente Gateway Inclusivo *divisor*. Cuando se los utiliza en pares, de esta manera, el Gateway *divisor* continuará el flujo hacia uno o hasta todos los caminos, y el Gateway *unificador* sincronizará (esperará por) todos los caminos que fueron generados para esa *instancia*.

Desde luego, es posible crear Procesos que incluyan diferentes combinaciones de Gateways Inclusivos *divisores* y *unificadores*. En algunos casos, será difícil determinar la vinculación de los Gateways. Es difícil lograr llegar a un deadlock en el Proceso (que se atasque) cuando se utiliza un Gateway Inclusivo, pero el comportamiento real del Proceso puede no ser el que el modelador espera. Por esto, proceda con cuidado cuando utilice Gateways Inclusivos unificadores. [29]

Mejor Práctica: ***Utilice siempre Gateways Inclusivos en pares*** *—Una forma de evitar comportamientos inesperados, es creando modelos donde un Gateway Inclusivo unificador siga a un Gateway Inclusivo divisor, y que ese número de Flujos de Secuencia se correspondan entre sí (véase Figura 9-30).*

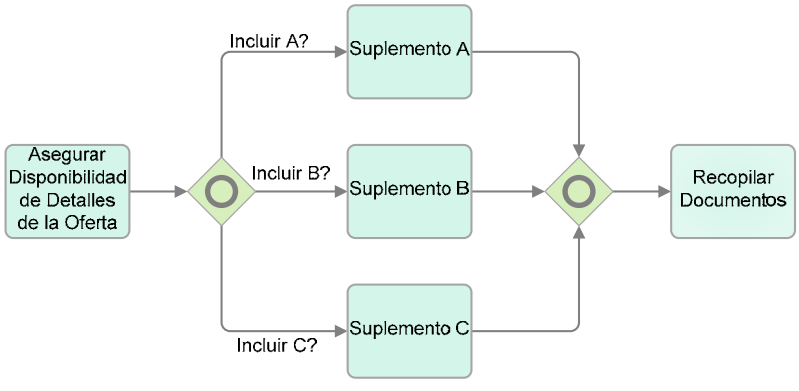


Figura 9-30—Gateways Inclusivos vinculados para *dividir* y *unificar* Flujos de Secuencia.

Gateways Complejos

Los Gateways Complejos usan un asterisco en negrita como símbolo dentro de la forma de diamante (véase Figura 9-31). Estos forman parte de BPMN para manejar situaciones donde los otros tipos de Gateways no proveen soporte para el comportamiento deseado.

[29] La directiva general utilizada en el desarrollo de BPMN fue desarrollar una capacidad modeladora que no restringa demasiado a los usuarios, haciendo la notación lo más flexible posible. Como resultado, el modelado imprudente puede conducir a problemas.

Figura 9-31—Un Gateway Complejo

Los modeladores proveen sus propias expresiones que determinan el comportamiento divisor y/o unificador del Gateway. Ellos a menudo reemplazan un conjunto de Gateways estándar, combinando los mismos en un simple y más compacto Gateway.

Comportamiento Divisor de un Gateway Complejo

El comportamiento divisor del Gateway Complejo es el lado menos interesante de los Gateways. Este es similar al comportamiento divisor del Gateway Inclusivo. La diferencia es que el Gateway Complejo utiliza una sola *asignación saliente* dentro del Gateway, en vez de un set de *condiciones* separadas en el Flujo de Secuencia *saliente*. El resultado es el mismo en el hecho de que el Gateway activa uno o más caminos *salientes* (consulte "Comportamiento Divisor de un Gateway Inclusivo" en la página 142 para un ejemplo de este tipo de comportamiento).

Una *asignación* tiene dos partes: una *condición* y una *acción*. Cuando una *asignación* es realizada, la misma evalúa la *condición* y si esta es *verdadera*, entonces realiza la *acción* como la actualización del valor de un Proceso o propiedad de un Objeto de Datos. En el caso de un Gateway Complejo, la *asignación saliente* puede enviar un *token* hacia uno o más de los Flujos de Secuencia *salientes* del Gateway. La *asignación saliente* puede hacer referencia a datos del Proceso o sus Objetos de Datos y el estatus del Flujo de Secuencia *entrante* (es decir, allí hay un *token*). Por ejemplo, una *asignación saliente* puede evaluar los datos del Proceso y luego seleccionar diferentes conjuntos de Flujos de Secuencia *salientes*, basándose en los resultados de la evaluación. Sin embargo, la *asignación saliente* debería garantizar que al menos uno de los Flujos de Secuencia *salientes* siempre será elegido.

Comportamiento Unificador de un Gateway Complejo

El lado más interesante de los Gateways Complejos es su comportamiento *unificador*. Hay muchos patrones que pueden ser realizados junto con el Gateway Complejo, como el comportamiento típico de un Gateway Inclusivo, procesamiento de múltiples *tokens*, aceptar *tokens* de algunos caminos pero ignorar los *tokens* de otros, etc. El Gateway se ve de igual manera para cada uno de estos patrones, por lo que el modelador debe usar Anotación de Texto para informarle al lector del diagrama cómo se utiliza.

Mejor Práctica: ***Utilice una Anotación de Texto con el Gateway Complejo—*** *Dado que el comportamiento real de un Gateway Complejo puede variar para cada uso del Gateway, utilice una Anotación*

de Texto para informarle al lector del diagrama qué comportamiento tiene establecido para realizar el Gateway .

El Gateway Complejo usa una *asignación entrante* cuando un *token* llega. La *condición* de la *asignación entrante* puede referirse a información de un Proceso u Objeto de Dato, y el estado del Flujo de Secuencia *entrante*. Si la *condición* es *falsa* no ocurre nada aparte de que el *token* es mantenido allí. Si la *condición* es *verdadera*, entonces la *acción* podría ser configurada para pasar el *token* al lado saliente del Gateway, y por ende activando la asignación *saliente*, o la *acción* podría ser configurada para consumir el *token*.

De las diversas formas de utilizar un Gateway Complejo, se utilizara el patrón discriminador como demostración. En este patrón, hay dos o más actividades paralelas. Cuando una de las Actividades se completa, entonces las Actividades que la siguen pueden comenzar, excepto que no tiene importancia cual Actividad se completa. Este es otro ejemplo de *condición de carrera*. Todas las Actividades restantes se completarán con normalidad, pero no tendrán impacto en el Flujo de Procesos.

En la Figura 9-32, la Tarea “Comenzar Análisis” debe empezar después de “Test A” o “Test B”, no importa cuál de las dos finaliza primero. Pero, después de que finalice la segunda, la Tarea “Comenzar Análisis” no debe empezar de nuevo. En cambio, si el modelador elige un Gateway Exclusivo para unificar el flujo, la Tarea “Comenzar Análisis” comenzará de nuevo (es decir, dos *instancias*).

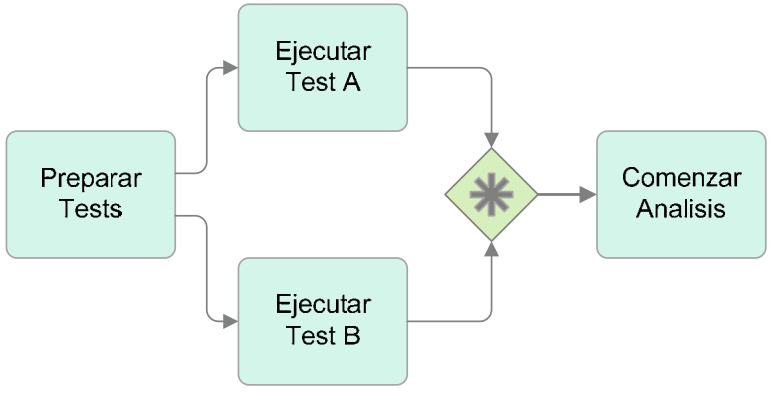


Figura 9-32—Un Gateway Complejo Unificando el Flujo de Secuencia

Si la Tarea “Ejecutar Test A” finaliza primero, su *token* es enviado al Gateway mientras que el otro *token* se mantiene en la Tarea “Ejecutar Test B” (véase Figura 9-33).

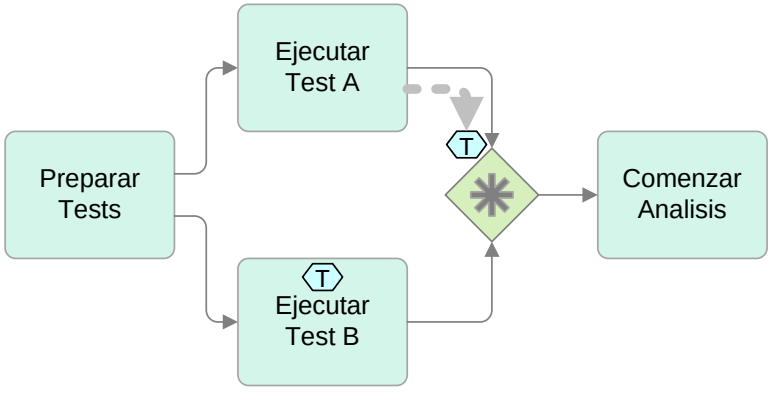


Figura 9-33—Un *token* llega a un Gateway Complejo

Para este patrón, el primer *token* que llega es enviado inmediatamente mediante el Flujo de Secuencia *saliente* hacia la siguiente Actividad (véase Figura 9-34).

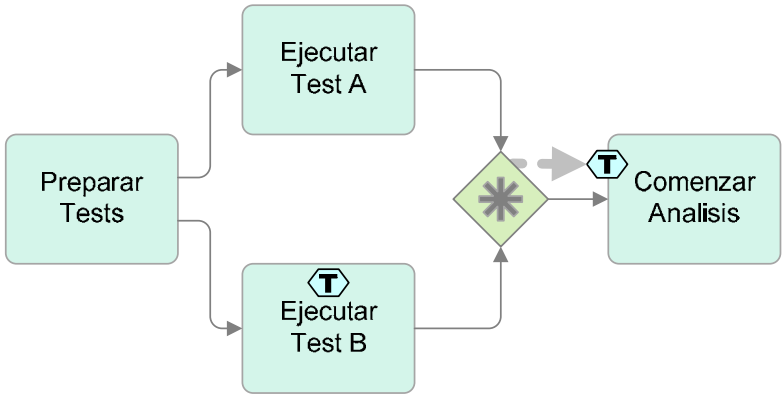


Figura 9-34—El primer *token* pasa a través del Gateway Complejo

Cuando el otro *token* finalmente llega, el mismo es consumido por el Gateway (véase Figura 9-35).

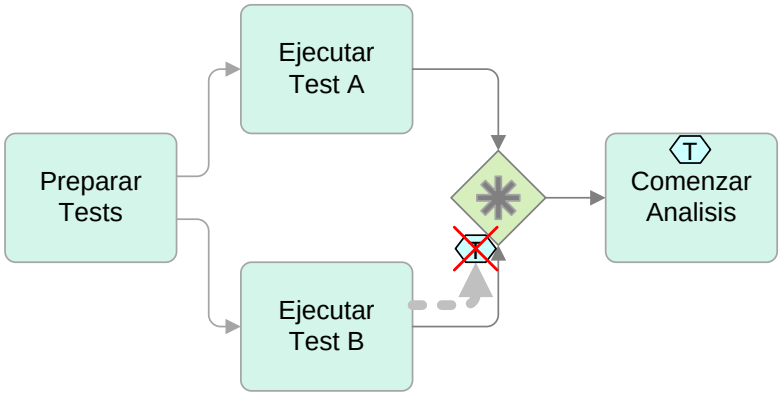


Figura 9-35—El segundo *token* se detiene en el Gateway Complejo