GeneXus

Modelado de procesos con GeneXus BPM Suite

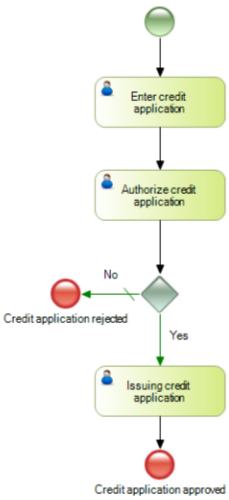
Versión X Evolution 3

GeneXus

Module 1

Modelado de procesos

GeneXus



¿Qué es el modelado de procesos?

Es la actividad de representar los procesos de negocio de una empresa con el objetivo de que estos puedan ser analizados, ejecutados y mejorados.

BPMN

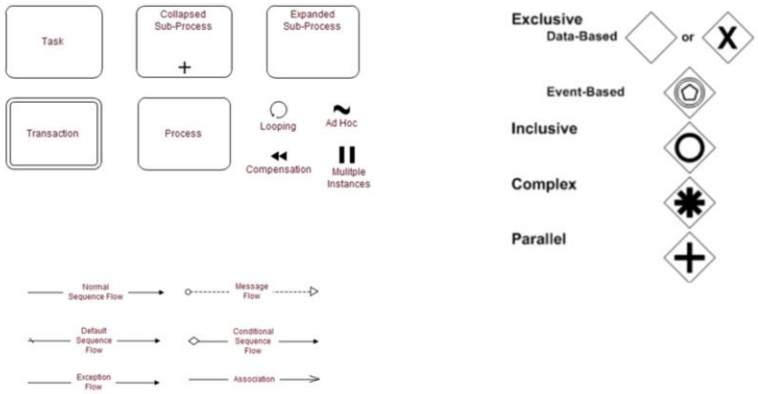
Business Process Modeling Notation

- Notación gráfica para representar flujos de procesos
- Estándar mantenido por el OMG
- Actualmente se encuentra en la versión 2.0

Para llevar a cabo esta actividad veremos el estándar **BPMN (Business Process Modeling Notation)**, este estándar define **símbolos y reglas** que nos permiten representar gráficamente un proceso de negocio.

GeneXus

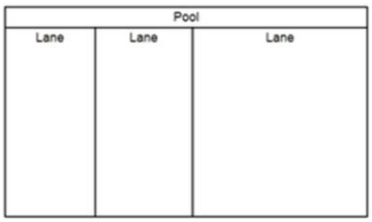
Symbols

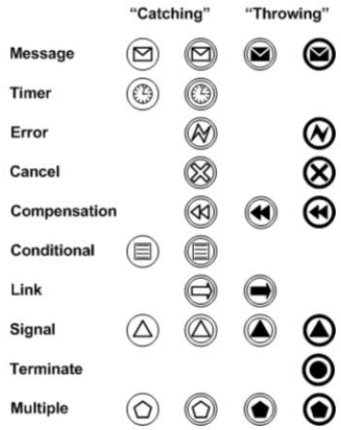


Los símbolos definidos por el estándar BPMN son **monocromáticos**, las herramientas como GeneXus que se basan en el estándar además agregan colores de forma que sea más claro para el usuario.

GeneXus

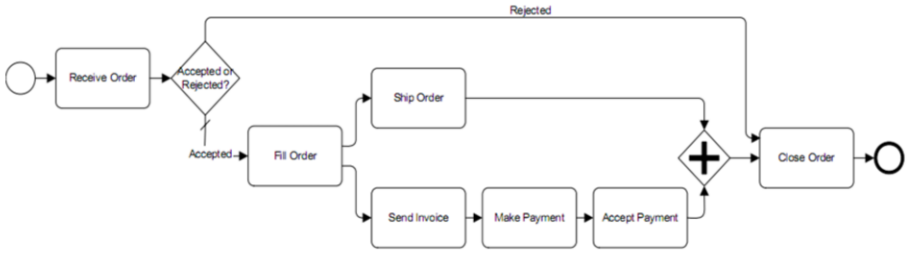
Symbols





GeneXus

Symbols



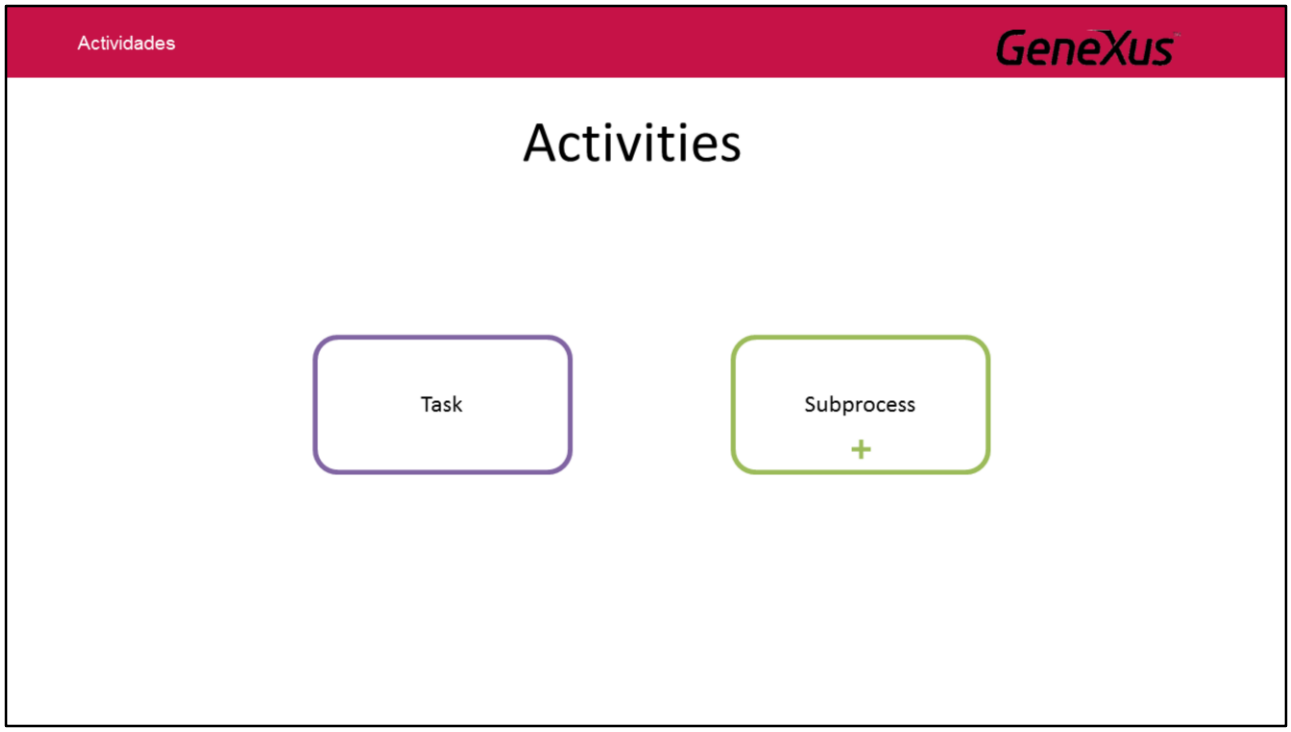
GeneXus

Ejemplo

Proceso de compras

1. Un funcionario ingresa un pedido de compras
2. El pedido es autorizado por el jefe del funcionario
3. Si el pedido es autorizado, se genera automáticamente la orden de compras o de lo contrario el proceso termina

Este ejemplo está compuesto por **tres actividades que deben ejecutarse en ese orden para obtener un resultado** que en este caso es la generación de la orden de compras.



Una **actividad** representa cierto trabajo a ser realizado dentro del proceso de negocio.
Una actividad toma cierto tiempo para ser ejecutada y utiliza uno o más recursos de la organización.
Requiere cierto tipo de entrada y usualmente produce algún tipo de salida.
Se representa gráficamente con un rectángulo redondeado.

Una actividad puede ser atómica o compuesta.
El tipo atómico es conocido como Tarea (Task)
Una actividad del tipo compuesta es llamada Subproceso (Subprocess).
Gráficamente, la única diferencia entre ellas es el signo de suma (+) utilizado en los suprocesos.

GeneXus

Actividades

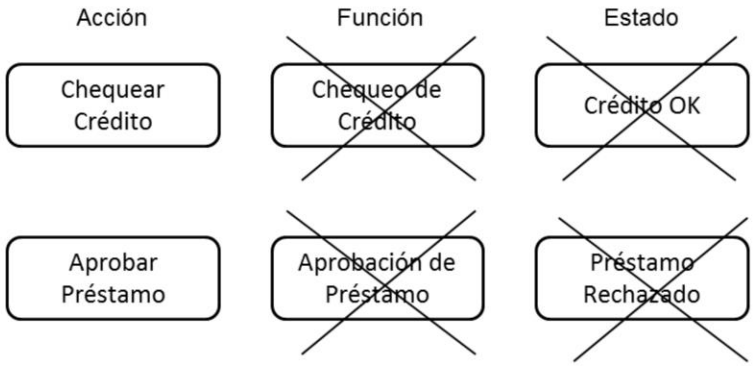
- Representan acciones que se llevan a cabo en el proceso
- NO representan estados o funciones
- Sus nombres deben ser de la forma Verbo – Sustantivo, Ej: “Ingresar factura”

Las actividades representan **acciones** que se llevan a cabo en el proceso.

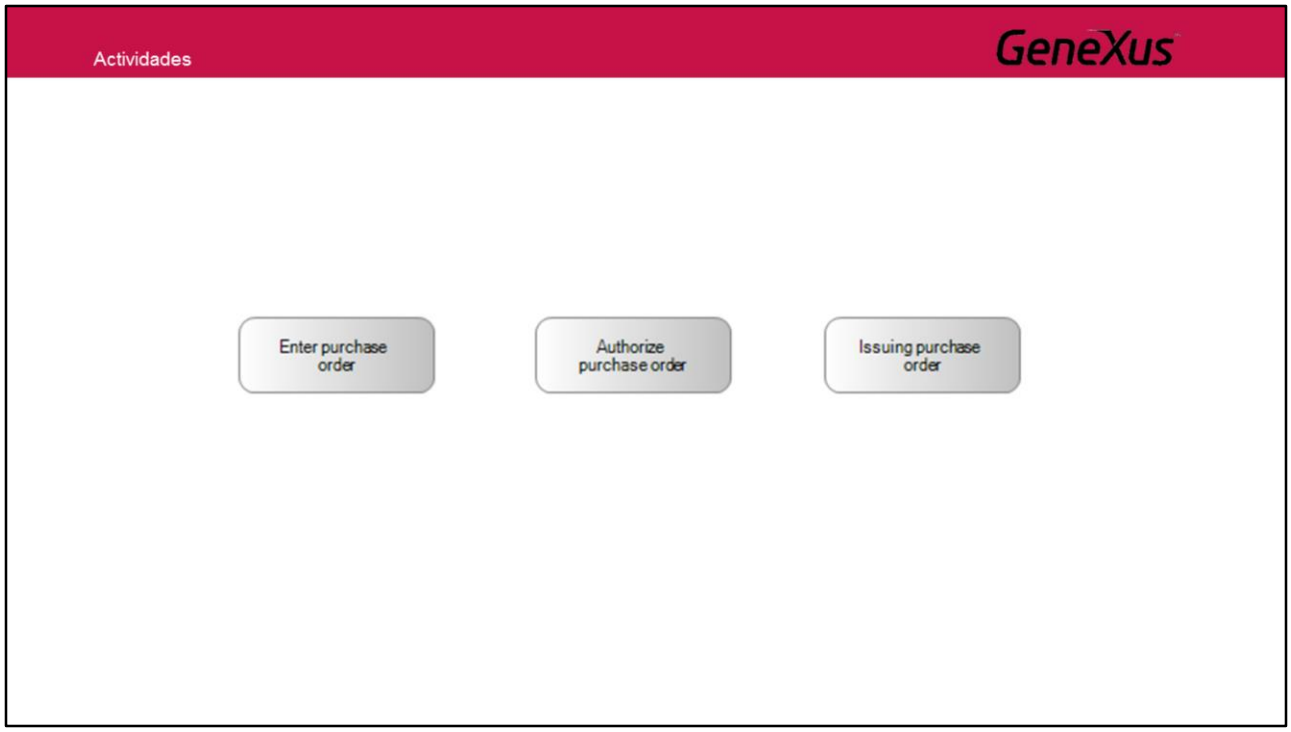
GeneXus

Actividades

Ejemplos:



A través de estos ejemplos podemos ver **cómo describir** las actividades de los procesos.

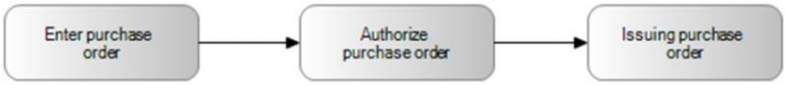


Entonces volviendo a nuestro ejemplo tenemos que representar tres actividades:

Ingresar Pedido de Compras
Autorizar Pedido de Compras
Generar Orden de Compras

GeneXus

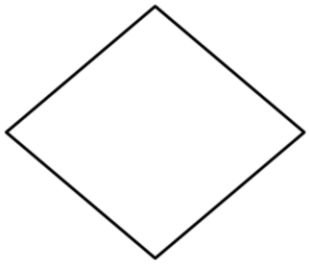
Conectores de Secuencia



Para indicar el orden de precedencia de estas actividades utilizamos el símbolo **Conector de Secuencia**.

GeneXus

Gateways (Compuertas)

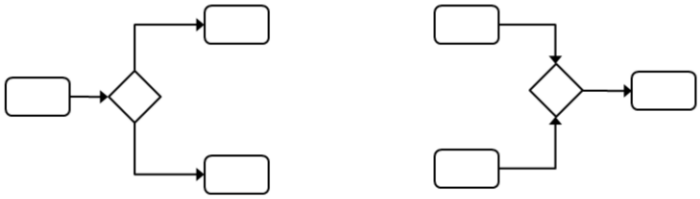


Hasta este punto el proceso que definimos es lineal, es decir que se procesan las tres actividades una a continuación de la otra sin excepciones. Pero en nuestro ejemplo luego de la tarea de autorización hay dos caminos posibles, uno es que se genere la orden de compras y el otro es que el proceso termine en base a la decisión tomada durante la autorización del pedido.

GeneXus

Gateways

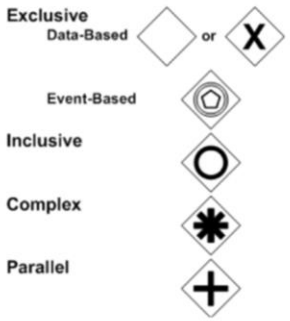
Se utilizan para controlar la ramificación (branching) y unificación (merging) de caminos en el flujo de un proceso



Para esto vamos a utilizar el símbolo **Gateway**, ya que nos permite ramificar o unificar el proceso.

Gateways

BPMN define varios tipos



BPMN define distintos tipos de gateways. Todos los Gateways se representan con una figura de **diamante**, utilizándose distintos **íconos en su interior** para distinguir el tipo.

Exclusive

Divergencia: el flujo seguirá por solamente uno de los caminos que tienen origen en el Gateway, la elección se hará evaluando las condiciones que se definen para el Gateway (se tendrá una por cada camino).

Convergencia: cuando múltiples caminos llegan al Gateway, este simplemente deja avanzar el flujo sin evaluar ninguna condición ni realizar ninguna sincronización.

Event

Divergencia: el flujo avanza por solamente uno de los caminos que tienen origen en el Gateway según la ocurrencia de un evento especificado.

Convergencia: tiene el mismo comportamiento que el Gateway Exclusive.

Inclusive

Divergencia: el flujo seguirá por todos los caminos que tienen origen en el Gateway que cumplan las condiciones del Gateway (se tendrá una por cada camino).

Convergencia: realiza una sincronización de todos los caminos que llegan al gateway.

Complex

Divergencia: el flujo seguirá por uno o más caminos según la evaluación de una única condición especificada para el Gateway.

Convergencia: realiza una sincronización de todos los caminos que llegan al Gateway y que cumplen una única condición especificada para el Gateway.

Parallel

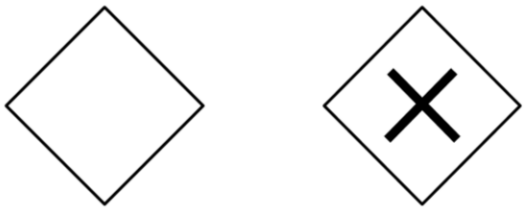
Divergencia: el flujo avanza por todos los caminos que tienen origen en el Gateway.

Convergencia: realiza una sincronización de todos los caminos que llegan al Gateway.

GeneXus

Ramificación exclusiva

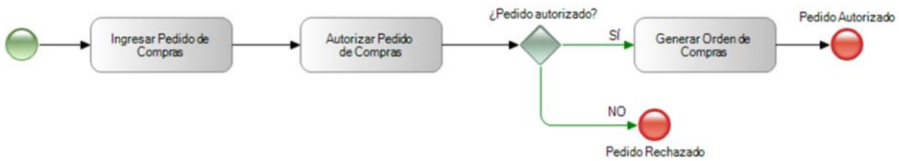
Exclusive Gateway



Se utiliza cuando el camino por donde debe seguir el flujo del proceso depende de la evaluación de una condición de tipo verdadero / falso

En nuestro proceso vamos a utilizar un gateway de tipo **Exclusive** para determinar si se genera la orden de compras o no.

Lo colocamos luego de la tarea de autorización y antes de la generación de la orden de compra.
Como vemos estamos utilizando el mismo símbolo de fin para ambos caminos, BPMN establece que lo correcto es utilizar uno para cada camino.



De esta forma podemos conocer cómo terminó el proceso, en este caso si el pedido fue autorizado o rechazado.
RR-¿cómo sabemos? Hay que hacer algo, con solo poner los end event distintos no alcanza…

GeneXus

Tareas

Son actividades atómicas

Tarea genérica que se puede utilizar si aún no se tiene más información acerca de la naturaleza de la tarea

Tarea ejecutada por una persona (interactiva)

Tarea ejecutada en forma automática (batch)

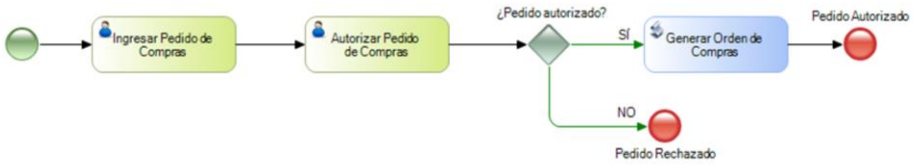
GeneXus

Ejemplo

1. **Un funcionario** ingresa un pedido de compras
2. El pedido es autorizado por el **jefe del funcionario**
3. Si el pedido es autorizado, **se genera automáticamente** la orden de compras o de lo contrario el proceso termina

GeneXus

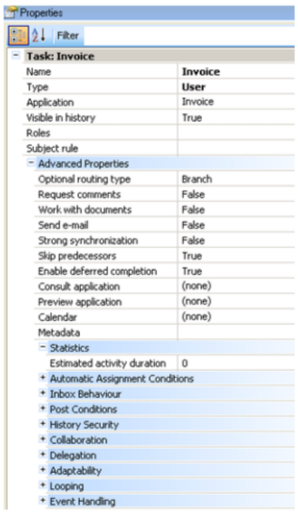
Ejemplo



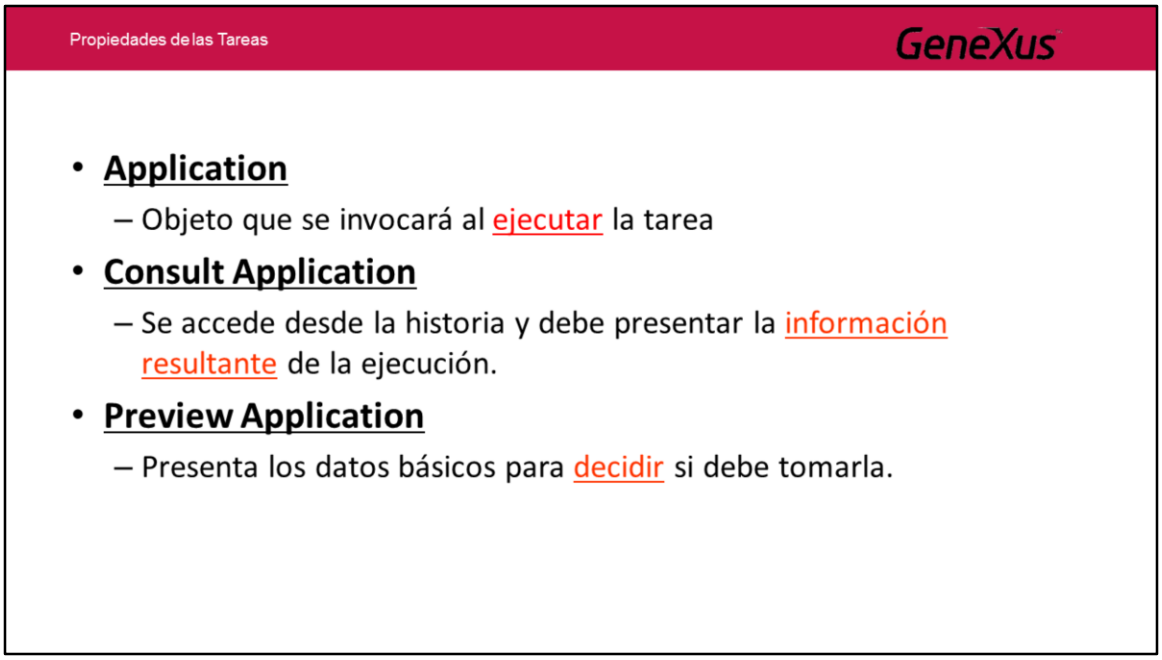
En nuestro ejemplo tanto la tarea de ingreso del pedido de compras como la tarea de autorización deben ser realizadas por usuarios por lo tanto les asignamos el tipo **User.** Para el caso de la generación de la orden como es automática no requiere la interacción de un usuario, por lo tanto le asignamos el tipo **Script**.

GeneXus

Propiedades de Tareas



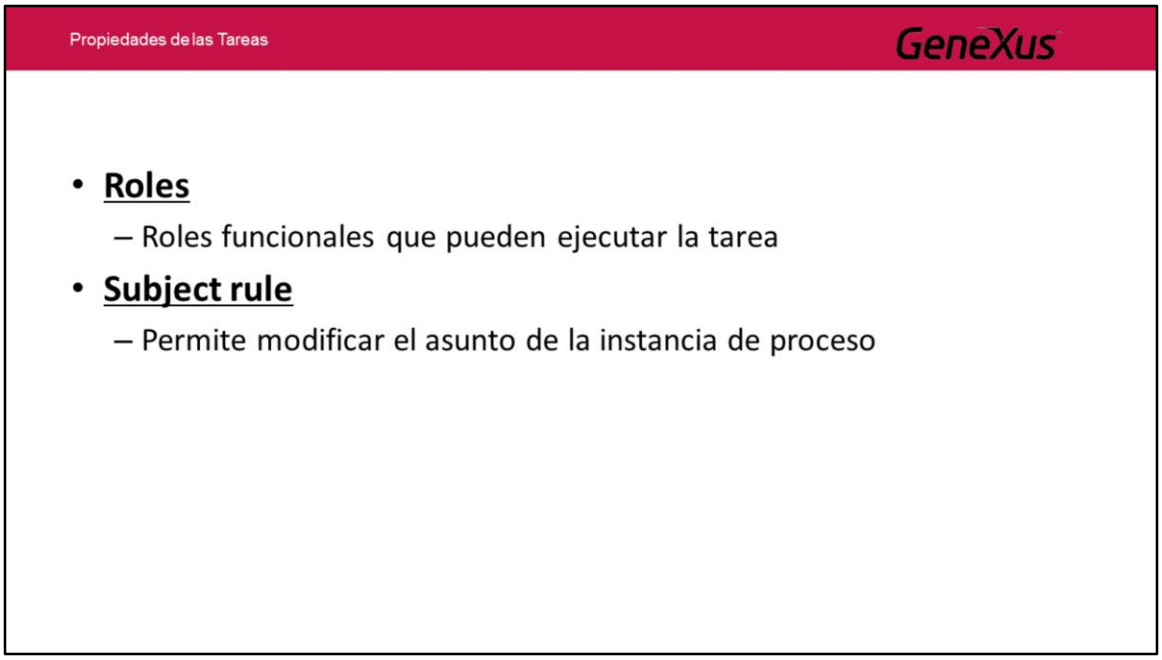
Las tareas tiene asociadas diferentes propiedades de configuración.
A continuación se listan algunas de ellas:
-Application
-Consult Application
-Preview Application
-Roles
-Subject Rule
-Request comments
-Send email



- **Application:** Permite indicar el objeto GeneXus que se invocará al ejecutar la tarea.

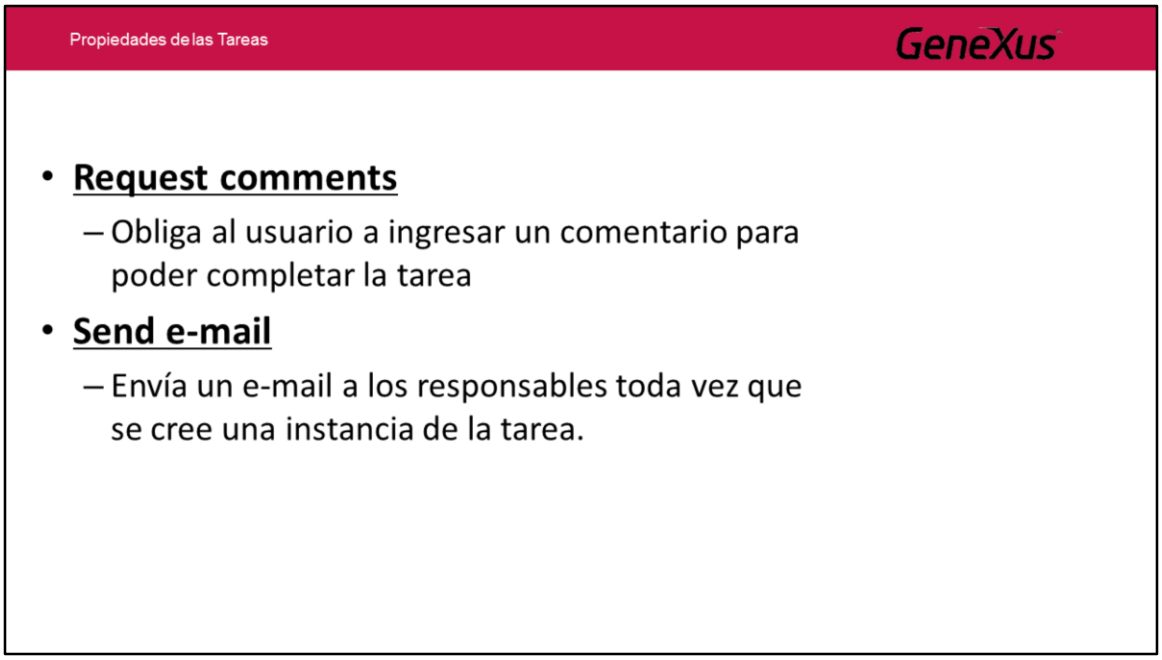
- **Consult Application:** Permite indicar el objeto GeneXus que se invocará desde la historia del proceso para mostrar la información resultante de la ejecución.

- **Preview Application:** Permite indicar el objeto GeneXus que se invocará desde la opción Preview de una tarea para decidir si se debe o no tomar la tarea.



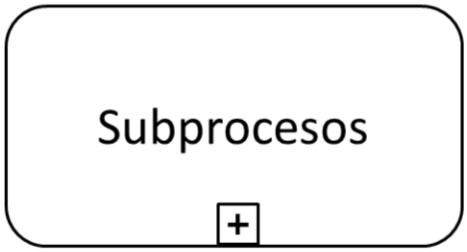
- **Roles:** Se indican los roles que tendrán permisos para ejecutar esa tarea. La definición de los roles se realiza desde Preferences/Workflow/Roles.

- **Subject Rule:** Permite modificar el asunto de la instancia de proceso. Se pueden utilizar caracteres, atributos, datos relevantes. Por ejemplo se puede colocar "Customer registration number“ + CustomerId + &relevantData



- **Request comments:** Indica si el usuario tendrá que ingresar comentarios antes de finalizar la tarea.

- **Send email:** Indica que se envíe un email a los responsables de una tarea cada vez que se cree una instancia de ella, teniendo la posibilidad de usar la plantilla por default de envío de email o crear una personalizada.



Una actividad puede ser atómica o compuesta.
El tipo atómico se conoce como **Tarea**.
El tipo compuesto de una actividad es llamado **Subproceso**.
Gráficamente se diferencian por el símbolo de más (+) que utiliza el Subproceso.

GeneXus

Subprocesos

- Actividades compuestas
- Existen dos tipos:
  - Reusable: permiten reusar un proceso definido en otro diagrama
  - Embebido: permiten definir el subproceso embebido dentro del diagrama que lo contiene

La finalidad de los sub-procesos es poder encapsular un proceso como una tarea de otro proceso. Esto implica dos ventajas: la mejora de la visibilidad de las definiciones de procesos y la reutilización de los mismos.

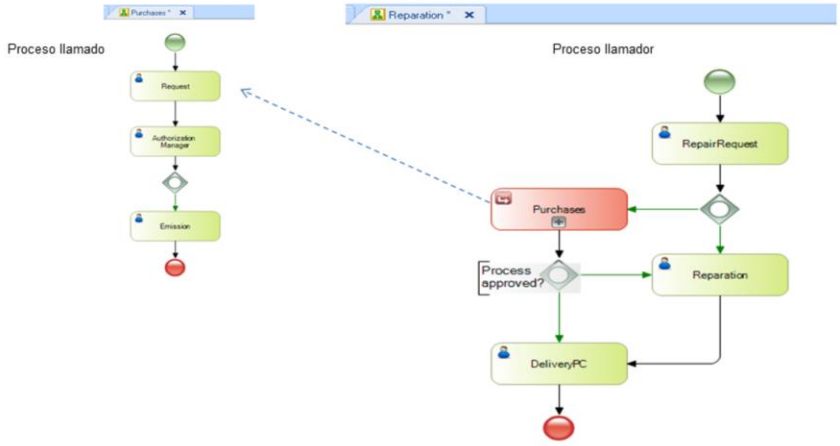
GeneXus

Subprocesos

En GeneXus estos son los símbolos que representan cada tipo.

GeneXus

Subproceso Reusable

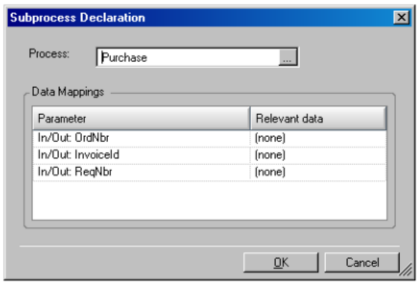


Un subproceso reusable (proceso llamado) es un proceso que es invocado desde otro proceso (proceso llamador).
En GeneXus, un subproceso reusable se asocia con otro objeto Business Process Diagram.
Los datos relevantes del proceso llamado son independientes de los datos del proceso llamador pero existe un mecanismo de pasaje de parámetros entre ellos.

GeneXus

Mapeo de datos relevantes entre subprocesos

- Permite comunicar los datos de un proceso con los del otro
- Dos formas
  - Desde la ruta que llega al subproceso
  - Desde el dialogo de selección del subproceso

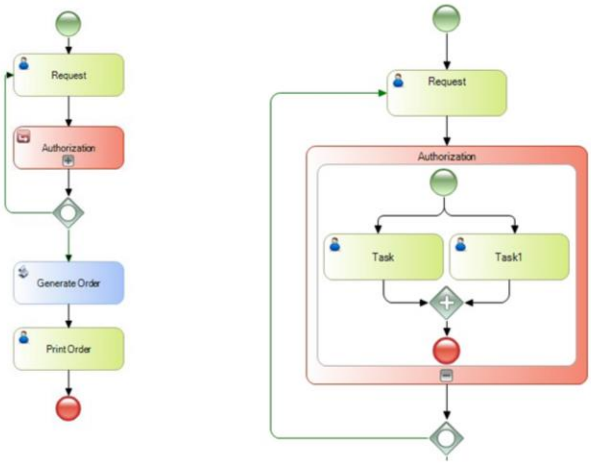


Cuando se llama a un subproceso muchas veces es necesario intercambiar información o recibir información de ese proceso. Para la comunicación de los datos se utilizan los datos relevantes.

Para acceder a la configuración de que datos relevantes que se van a intercambiar se puede hacer desde: la ruta que llega al subproceso o desde el diálogo de selección del subproceso.

GeneXus

Subprocesos embebidos



Los Subprocesos embebidos, son procesos encapsulados dentro del mismo objeto Business Process Diagram.

¿Cuándo usarlo? Por ejemplo, en caso de tener un diagrama complejo se puede detectar un grupo de actividades que puedan ser encapsuladas en un subproceso embebido, teniendo como resultado un diagrama más compacto con una visión global y entendible del proceso o si queremos ver un detalle en particular se puede hacer una expansión inline (+) y conocer las actividades embebidas, lo que nos dará una ganancia en la documentación del proceso.

Otra opción es cuando a un conjunto de tareas se quiere hacer algo, por ejemplo poner un timer, en este caso en vez de crear otro proceso para agrupar esas tareas se hace dentro del mismo.

Sus principales características son:

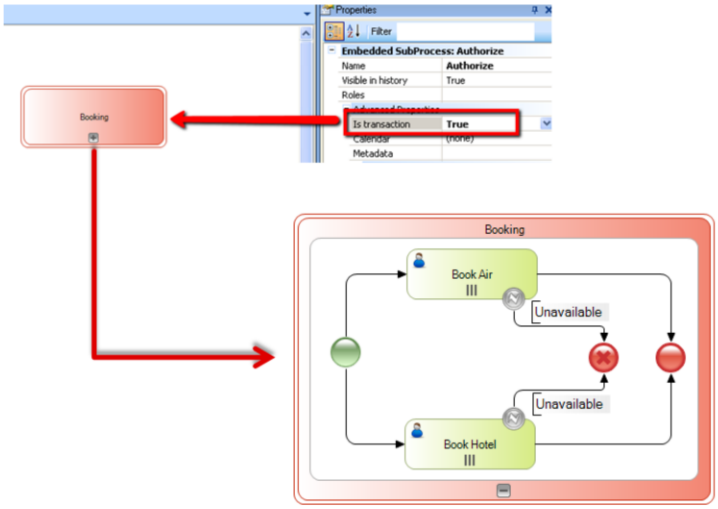
- Nos permiten definir un subproceso dentro de un proceso padre y no es reusable por otros procesos.

- Los datos relevantes del proceso padre estarán disponibles para el subproceso sin necesidad de realizar mapeos.
- No pueden iniciarse con eventos de tipo triggers (por ejemplo un timer) ya que sólo el proceso padre decide cuando deben iniciarse.

Tip: Para editar un subproceso embebido se puede hacer dentro del proceso padre o mediante el doble clic en el subproceso, esto abre el subproceso en una nueva pestaña y es útil para editar procesos complejos o que se anidan en varios niveles de subprocesos.

GeneXus

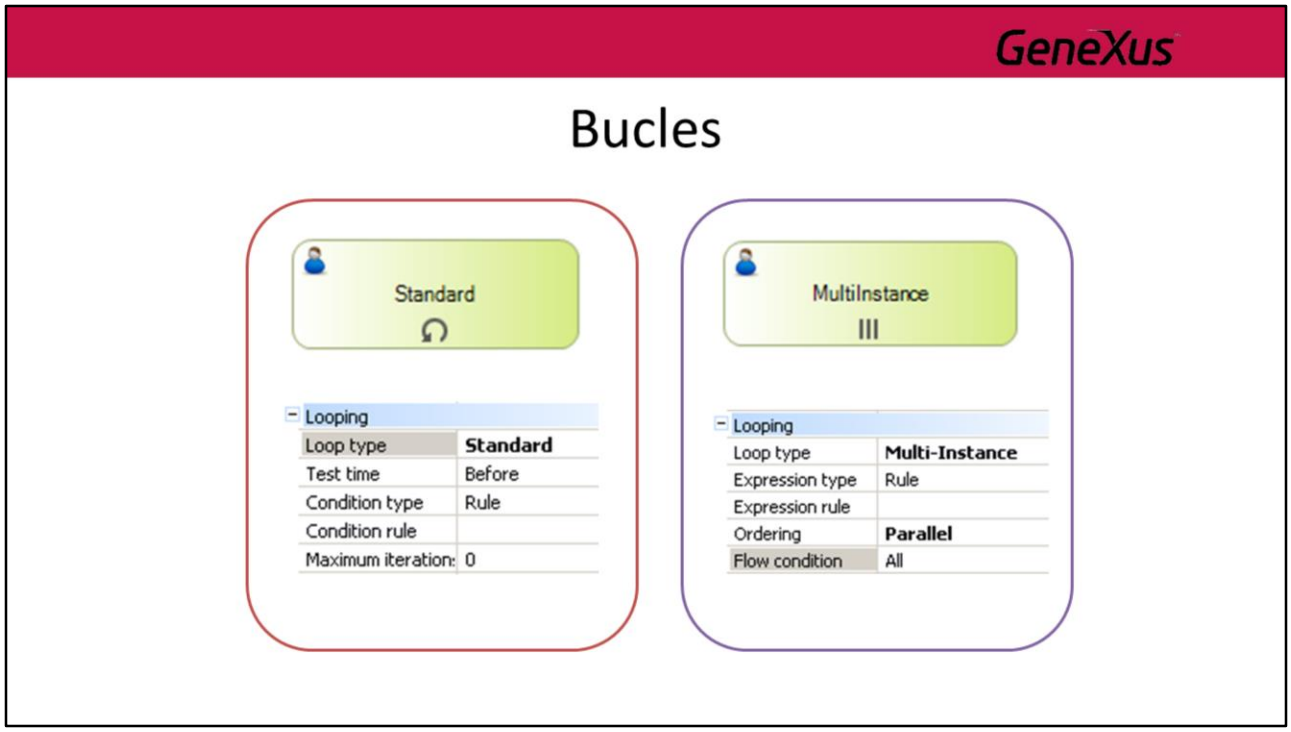
Subproceso transaccional



Un subproceso transaccional, es un subproceso embebido que puede ser considerado como una unidad de trabajo lógica (transacción), es decir, que se deben completar todas las tareas que están dentro del subproceso, o de lo contrario el proceso se deshace, deshaciendo todas las actividades que lo componen (rollback).

Para definir a un subproceso transaccional hay que habilitar la propiedad 'Is transaction' del subproceso y se identificará por tener un borde doble.

Como se muestra en el ejemplo tenemos un subproceso que hace reservas de vuelos y hoteles. Si ocurre un error de falta de disponibilidad en cualquiera de las reservas, el flujo avanza hacia un evento de finalización de tipo Cancel. Esto accionará el rollback del proceso y cualquier actividad de reserva que haya sido completada quedará como "undone".



Existen varias maneras de definir bucles en el modelado de los procesos.
Los bucles pueden ser definidos en una sola actividad (ya sea tarea o subproceso), pare ello hay que modificar la propiedad *Looping type* que puede tomar los siguientes valores:

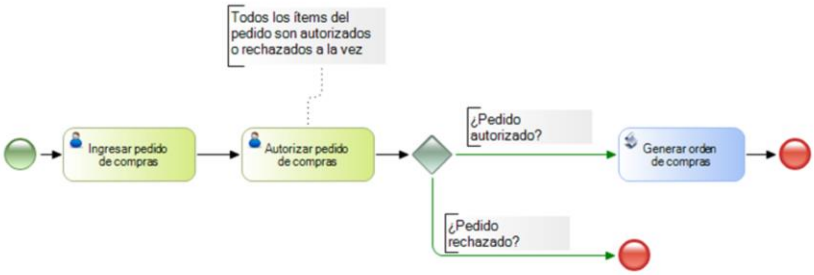
Standard
Una actividad de tipo bucle **estándar** tiene asociada una expresión booleana que se evalúa después de cada ciclo del bucle. Si la expresión es todavía cierta, entonces el ciclo continuará.
Existen dos variantes del bucle, que reflejan las construcciones de programación ”While” y “Until”.
En un bucle while la expresión se evalúa antes de ejecutar la actividad, lo que significa que la actividad puede ejecutarse cero veces.
En un bucle until la expresión se evalúa después de que la actividad se ha ejecutado, lo que significa que la actividad se llevará a cabo al menos una vez.
El tipo de bucle se configura con la propiedad “Test Time” (Before = While, After = Until).
La condición (booleana) puede ser definida mediante una regla o mediante un procedimiento (para casos complejos).

Multi-Instance
Un bucle Multi-Instance refleja la construcción de programación “for each”.
La expresión de un bucle Multi-Instance es una expresión numérica que se evalúa sólo una vez antes de realizar la actividad. El resultado de la evaluación de la expresión es un número entero que especificará el número de veces que la actividad deberá repetirse.
Al igual que con el bucle estándar la expresión puede definirse con una regla o con un procedimiento.
Hay también dos variantes del bucle Multi-Instance dependiendo si las diferentes instancias ejecutan en forma secuencial o en paralelo, lo cual se configura en la propiedad Ordering.

Bucles

- Ejemplo: Proceso de compras
  1. Un funcionario ingresa un pedido de compras
  2. Cada ítem del pedido es autorizado por separado por el jefe del funcionario
  3. Si al menos uno de los ítems del pedido fue autorizado entonces se genera la orden de compras correspondiente

Bucles


Todos los ítems del pedido son autorizados o rechazados a la vez
Ingresar pedido de compras
Autorizar pedido de compras
¿Pedido autorizado?
Generar orden de compras
¿Pedido rechazado?

Bucles

- Es necesario un mecanismo para declarar que una actividad será ejecutada potencialmente más de una vez
- El número de veces que será ejecutada no es conocido durante el diseño

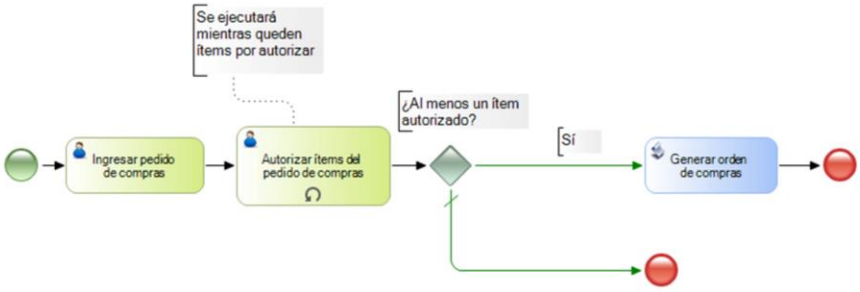
Standard Looping

- La actividad será ejecutada secuencialmente mientras se cumpla una condición de tipo verdadero/falso

Standard Looping

- Otras propiedades:
  - *Condition rule*: permite especificar la condición que debe cumplirse para que se siga ejecutando la actividad (implementación)
  - *Test time:* especifica el momento en que se evalúa la condición
    - *Before:* la evaluación se hace antes de ejecutar la actividad
    - *After:* primero se ejecuta la actividad y luego se evalúa la condición

Standard Looping



Multi-instance

- La actividad será ejecutada un número de veces que será determinado mediante la evaluación de una expresión numérica
- Las actividades pueden ejecutar en formas secuencial o paralela

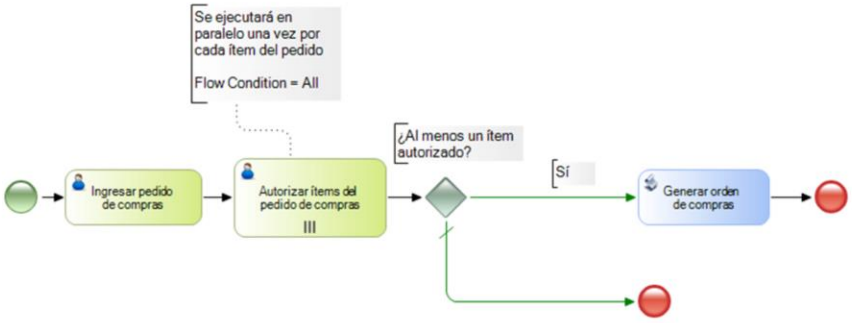
Multi-instance

- Otras propiedades:
  - Expresion rule: expresión numérica que indica el número de veces que se ejecutará la actividad (implementación)
  - Ordering
    - Sequential: se ejecutará en forma secuencial
    - Parallel: se ejecutará en forma paralela

Multi-instance

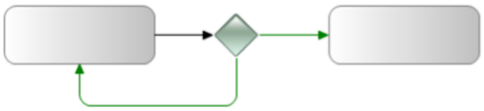
- Otras propiedades:
  - Flow condition (aplica solamente si *Ordering = Parallel)*
    - Determina cómo se unifican los caminos luego de que se ejecutan las diferentes instancias de una actividad
    - Valores:
      - » **None**: No hay unificación. Luego de que cada instancia termina, se continúa con el flujo que sigue a la misma, por lo que este flujo se repite por cada instancia.
      - » **One**: No hay unificación. El proceso continúa luego de que la primera de las instancias termina de ejecutarse y se ejecuta sólo el flujo que parte de ella.
      - » **All**: Similar a una parallel Gateway, el proceso continúa solamente si todas las instancias de la actividad finalizaron
      - » **Complex**: permite especificar una condición que indicará cuando puede continuar el proceso. La condición se evalúa cada vez que finaliza una instancia.

Multi-instance



GeneXus

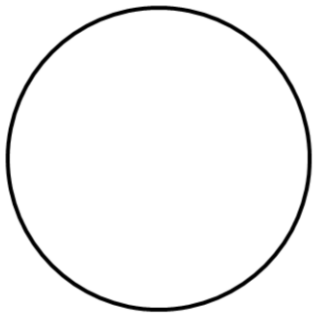
Bucles



Los bucles también pueden crearse utilizando flujos de secuencia que conectan con elementos “anteriores” del diagrama.

GeneXus

Eventos



Hasta aquí no hemos mencionado dónde comienza y termina el proceso para esto vamos a utilizar el símbolo **Evento.**

Los eventos representan **sucesos que ocurren durante el proceso**.
Hay tres tipos de eventos:

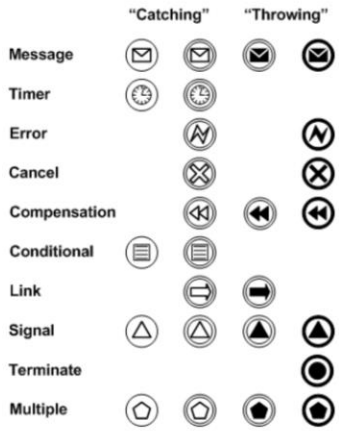
Inicio: Permite indicar dónde inicia el proceso. Un evento inicial **NO** puede tener conectores entrantes.
Intermedio: Permite indicar un suceso entre el inicio y el fin de un proceso.
Fin: Permite indicar dónde termina el proceso. Un evento final **NO** puede tener conectores salientes

Estos se diferencian por el trazo, los de **inicio tienen trazo fino**, los **intermedios tienen doble trazo fino** y los de **fin tienen trazo grueso**.

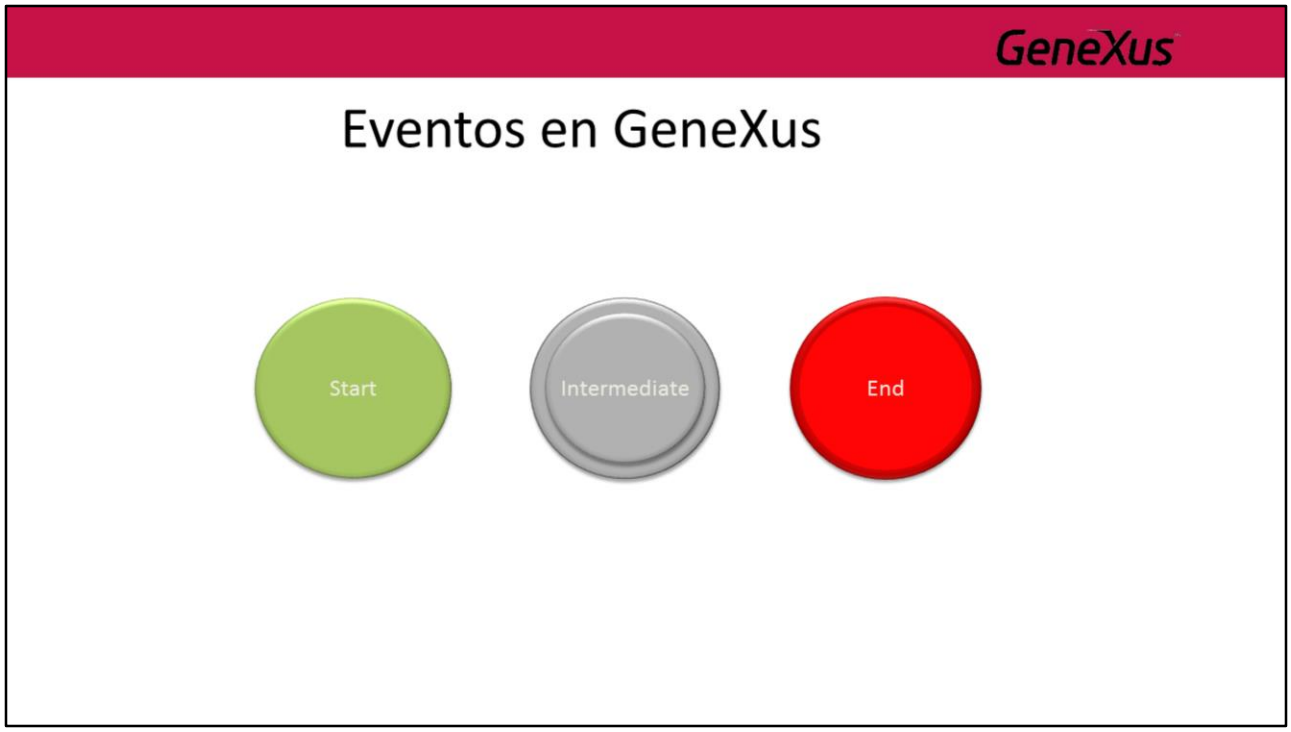
GeneXus

Eventos en BPMN

Para cada tipo de evento BPMN define subtipos que permiten especificar la naturaleza del evento



Para cada uno de estos tres tipos BPMN define **subtipos** que permiten especificar la **naturaleza del evento**.



Los eventos se representan con un círculo. Es algo que “pasa” durante el curso del proceso de negocio. Estos eventos afectan al flujo del proceso y suelen tener una causa (trigger) o un impacto (resultado). Los eventos representados con un círculo con centro abierto permiten a los marcadores internos diferenciar diferentes triggers y resultados.
Hay tres tipos de eventos, basados en cuando afectan al flujo:

Start
Como su nombre lo indica, un evento Inicial indica por donde va a iniciar un proceso. Su utilización es opcional. En caso de **NO** utilizarlo, cualquier actividad que **NO** le lleguen conectores se considerará como **Inicial**.
Se recomienda su utilización pues hace mas legible el diagrama. BPMN establece que los eventos de inicio se representan con un círculo de trazo fino (el color es agregado por GeneXus).
Un evento inicial **NO** puede tener conectores entrantes.

Intermediate
El evento Intermedio indica que algo pasa (un evento) en algún lugar entre el inicio y el final de un proceso.

Afecta el flujo del proceso, pero **NO** Inicia o termina (directamente) el proceso. BPMN establece que los eventos intermedios se representan con un círculo de trazo doble (el color es agregado por GeneXus).

End
El evento de Fin indica que un proceso va a terminar.
Su utilización es opcional. En caso de **NO** utilizarlo, cualquier actividad que **NO** tenga conectores salientes marca el fin de ese camino en el proceso (puede haber varios caminos paralelos).
Un evento Final **NO** puede tener conectores salientes. BPMN establece que los eventos de fin se representan con un círculo de trazo grueso (el color es agregado por GeneXus).

GeneXus

Eventos en GeneXus

	Inicial	Intermedio		Final
	Catching		Throwing	
None				
Timer				
Conditional				
Signal				
Error				

Los eventos pueden se pueden clasificar como “Catch” o “Throw” dependiendo si atrapan el evento o si lo disparan.

Tipos de Evento

None
Es un evento que no tiene un tipo definido.
En un evento intermedio puede ser utilizado para marcar un hito en el proceso.

Timer
Cuando es utilizado como un elemento independiente permite especificar un “delay” en el proceso. Esto es, una espera por un tiempo determinado.
Cuando es utilizado como adjunto a una actividad, permite modelar cuanto quiero esperar por esa actividad (deadline).

Conditional
Este evento es utilizado para esperar hasta que cierta condición sea verdadera.

Signal
Este evento es utilizado para enviar o recibir señales dentro o fuera del proceso (en este último caso el alcance es la jerarquía de procesos).

Error

El evento error interrumpe el proceso o requiere corrección.

Cuando es intermedio se utiliza adjunto a una actividad y permite definir un camino alternativo si existe la posible ocurrencia de un error al ejecutar dicha actividad. En caso de ser de tipo final interrumpirá el proceso.

GeneXus

Eventos en GeneXus

	Inicial	Intermedio		Final
	Catching		Throwing	
Message				
Compensate				
Link				
Cancel				
Terminate				

Message

Este evento envía o recibe un mensaje desde otra entidad de negocio o participante (otra Pool).

Compensate

Este evento es utilizado para deshacer actividades en el caso de que un subproceso transacción sea cancelado o necesite deshacerse (rollback).

En un evento catch solamente pueden estar adjuntos a actividades. Cuando es throw se puede colocar en el flujo normal del proceso

Link

El evento link siempre se utiliza de a par: (throw-catch), es necesario que ambos eventos tengan el mismo nombre para hacer el vínculo entre ellos. Su objetivo es crear un flujo de secuencia virtual.

Solo puede haber un evento link del tipo catch y existir muchos eventos link del tipo trhow con el mismo nombre.

Son utilizados únicamente dentro de un mismo proceso. Para comunicación entre procesos se recomienda el uso de señales.

Cancel

Sirve para cancelar un subproceso transaccional. Un evento de finalización cancel indica que se debe realizar un rollback a dicho subproceso.

Terminate
Un evento “terminate” da por terminado el proceso. Esto es, a diferencia del evento final “none”, no solo termina el camino actual sino todo los caminos paralelos que puedan existir.

GeneXus

Eventos de inicio y de fin

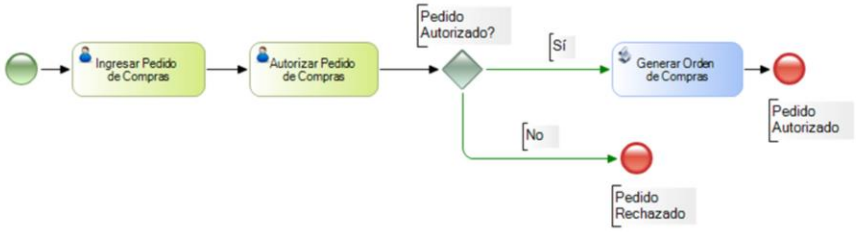
Entonces agregamos a nuestro proceso el inicio y fin para indicar dónde comienza y dónde termina.

Terminación de Procesos

- Si el flujo del proceso llega a determinado punto, se requiere que el proceso finalice, independientemente de si se tienen otros caminos paralelos que todavía están activos

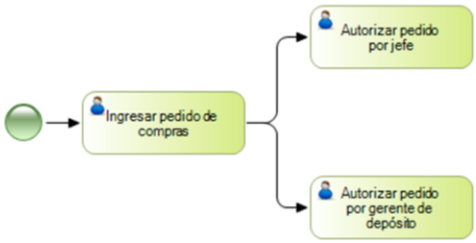
Terminación de Procesos

- El evento de finalización “None” alcanza para indicar el fin de un proceso si solamente se tiene un camino activo

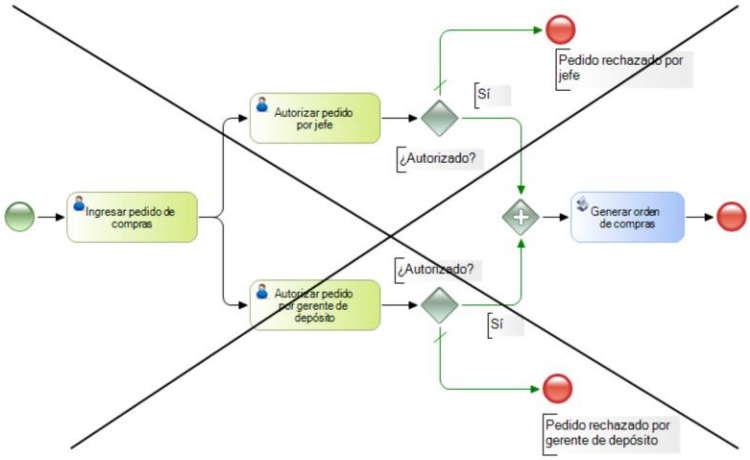


Terminación de Procesos

- ¿Qué pasas si tenemos dos autorizaciones en paralelo?



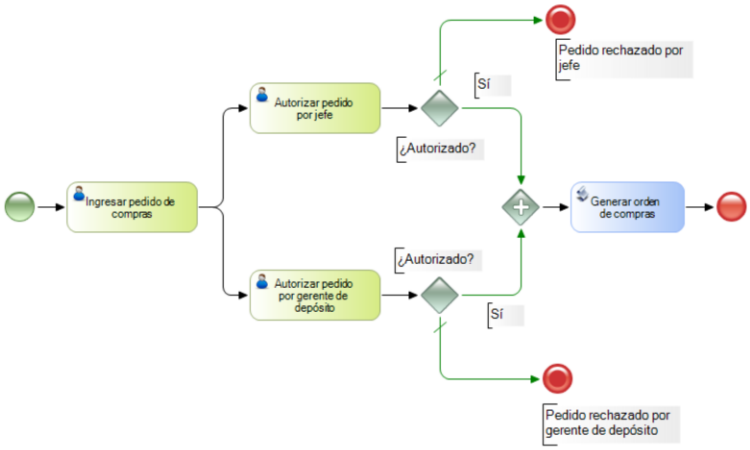
Terminación de Procesos



Terminación de Procesos

- Teminate End Event
  - Finaliza un proceso interrumpiendo todas las actividades que estén activas en ese momento
  - Si el proceso se utiliza como subproceso, NO se finaliza el proceso padre

Terminación de Procesos

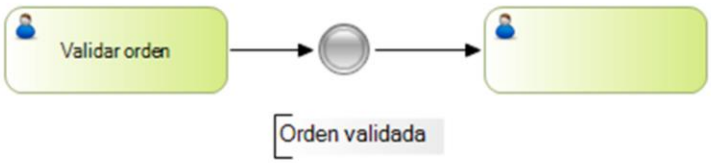


GeneXus

Eventos Intermediarios Adjuntos

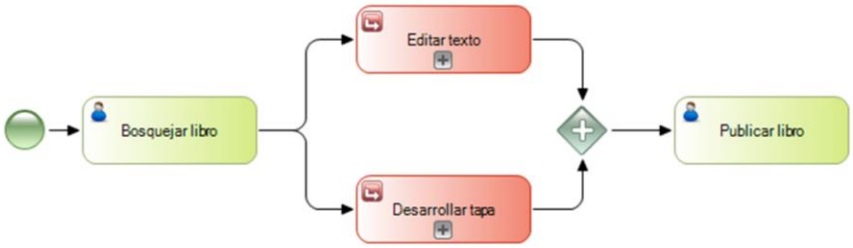
Es posible adjuntar eventos al borde de una actividad (tarea o subproceso) para modelar excepciones al flujo normal del proceso.
Esto permite que si durante el transcurso de tiempo en el que la actividad esta activa, ocurre dicho evento, se puede disparar un flujo alternativo con la posibilidad de interrumpir la actividad.

Indicación de Hitos



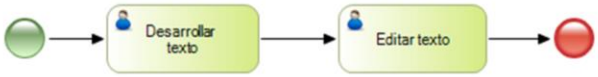
Señales

- Comunicación entre procesos

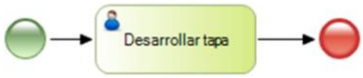


Señales

- Comunicación entre procesos
  - Subproceso *Editar Texto*



  - Subproceso *Desarrollar tapa*



Señales

- Signal Intermediate Event
  - Permite señalizar la ocurrencia de un evento que podrá ser detectado en cualquier parte del proceso, subprocesos e incluso procesos ancestros
  - Dos tipos

Throw
(para disparar la señal)

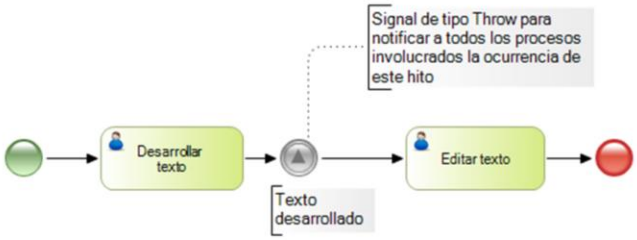
Catch
(para atrapar la señal)

Señales

- Signal Intermediate Event
  - En GeneXus
    - Propiedad *Trigger = Signal*
    - Las señales se identifican por su nombre (propiedad *Name)*
    - Si es un *Throw* hay que configurar la propiedad *Is Throw = true*

Señales

- Intercomunicación de procesos
  - Subproceso *Editar Texto*



Señales

- Intercomunicación de procesos
  - Subproceso *Desarrollar tapa*

Señales

- Comunicación entre procesos
  - Subproceso *Editar Texto*

  - Subproceso *Desarrollar tapa*

GeneXus

Propiedades de Eventos

Intermediate Event: Event	
Name	Event
Trigger	Signal
Is throw	False

Intermediate Event: Event	
Name	Event
Trigger	Conditional
Condition type	Rule
Condition rule	

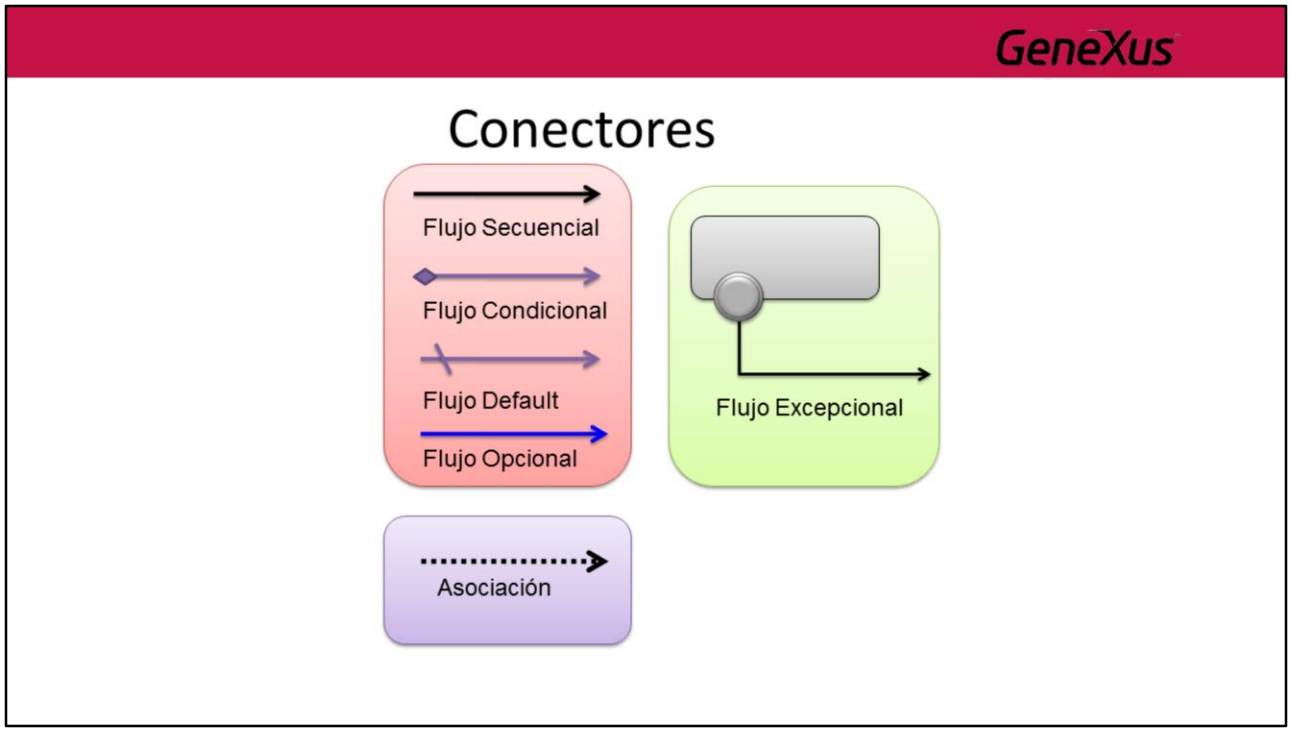
Intermediate Event: Event	
Name	Event
Trigger	Timer
Submit to calendar	False
Time unit	Minutes
Lapse expression type	Rule
Lapse expression rule	

Todos los eventos tienen la propiedad Nombre y Trigger que indica cual es el tipo de evento. Adicionalmente y dependiendo del tipo de evento pueden existir otras propiedades.

Event Signal
La propiedad “Is throw” indica si esta disparando o esperando por el evento.
El nombre del evento se utiliza para discriminar entre diferentes señales. Esto es, si tengo una evento de señal de tipo “catch” y nombre “Signal” va a esperar por un evento de señal de tipo “throw” y de mismo nombre.

Event Conditional
Un evento condicional tiene asociada una condición. Dicha condición se puede expresar con una regla o mediante el llamado a un procedimiento.
Las reglas de condición de un evento condicional se evalúan por el motor de workflow cuando se modifican los datos relevantes de la instancia de proceso a la que pertenece el evento o de un dato relevante compartido en una jerarquía de procesos.
Como consecuencia, la expresión a definir tiene que estar basada en datos relevantes ya que si utilizara atributos la regla nunca se llegaría a evaluar.
Los eventos condicionales siempre son de tipo “catch”.

Event Timer
Un evento de tiempo tiene asociada una expresión numérica (o un procedimiento) que indicará el tiempo que debe pasar para que ocurra el evento (contado desde el momento de su creación). Se puede especificar la unidad de tiempo y si esta sujeto a un calendario particular.



Los elementos del diagrama de procesos se conectan entre ellos en un diagrama para crear el esqueleto básico de la estructura de un proceso de negocio.
Hay tres tipos de conectores que hacen esta función. Estos conectores son:

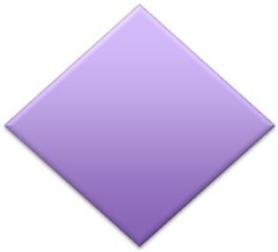
Flujo Secuencial
El flujo secuencial se representa por una línea sólida con una cabeza de flecha sólida y se usa para mostrar el orden (la secuencia) en el que las diferentes actividades se ejecutarán en el Proceso.

Flujo Condicional
El flujo condicional se representa por una línea sólida con una cabeza de flecha sólida. Si el flujo parte desde una actividad entonces en el comienzo de la línea se utiliza un mini rombo. Si el flujo parte desde una compuerta entonces el rombo no se utiliza porque el tipo de flujo queda implícito.
Un flujo condicional tiene asociado una regla de condición que es evaluada en tiempo de ejecución para determinar si dicho flujo debe continuar o no.

Flujo Default
Este flujo se utiliza en exclusive o inclusive gateways y es utilizado para modelar que es el flujo que debe continuar cuando los restantes flujos condicionales salientes no evalúan como verdadero en tiempo de ejecución.
Para diferenciarlo se utiliza barra diagonal que lo corta transversalmente en el comienzo de la línea.

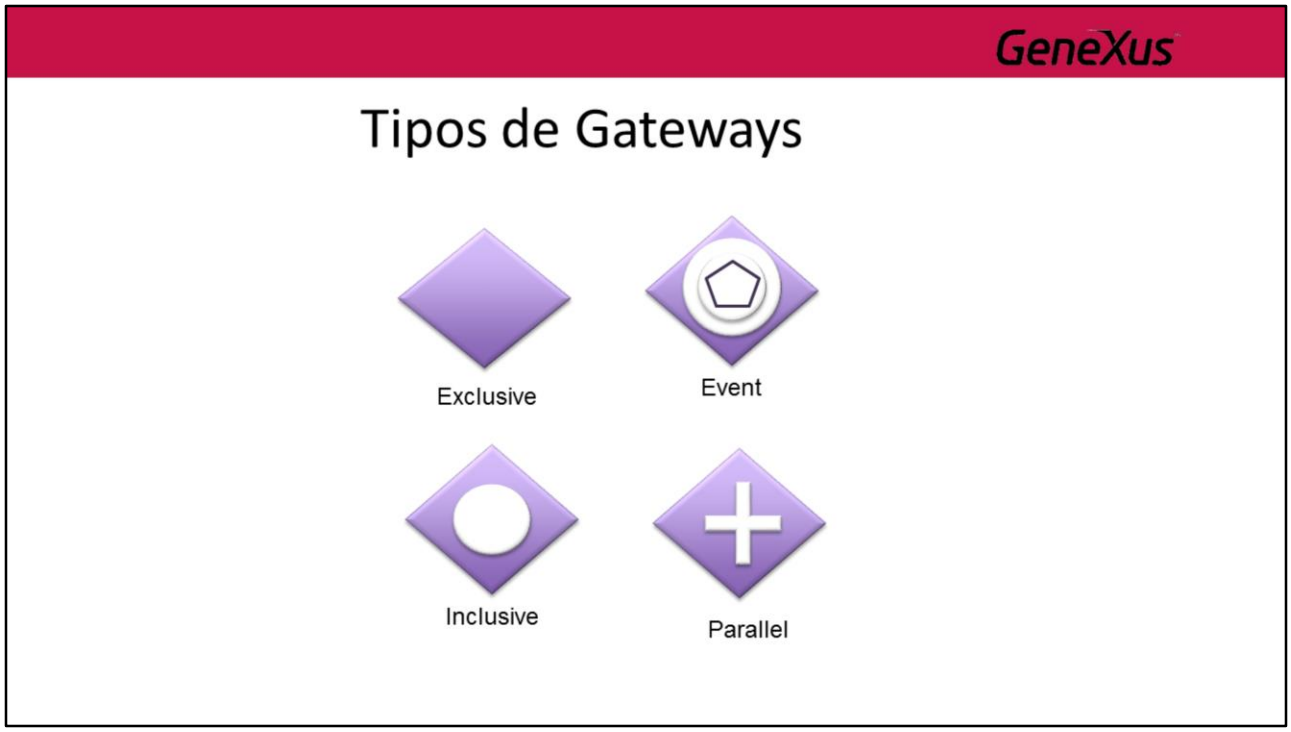
GeneXus

Gateways



Los Gateways o Compuertas son elementos de modelado que controlan como un proceso Diverge o Converge. Cuando el flujo no necesita ser controlado entonces los gateways no son necesarios.

Todos los Gateways se representan con una figura de diamante, utilizándose distintos íconos en su interior para distinguir el tipo de Gateway.



Existen diferentes tipos de gateways las cuales se pueden utilizar tanto para convergencia como para divergencia. A continuación se detallan las mismas:

- **Exclusive:** *Divergencia:* el flujo seguirá por solamente uno de los caminos que tienen origen en el Gateway, la elección se hará evaluando las condiciones que se definen para el Gateway (se tendrá una por cada camino).
Convergencia: cuando múltiples caminos llegan al Gateway, este simplemente deja avanzar el flujo sin evaluar ninguna condición ni realizar ninguna sincronización.

- **Event:** *Divergencia:* el flujo avanza por solamente uno de los caminos que tienen origen en el Gateway según la ocurrencia de un evento especificado.
Convergencia: tiene el mismo comportamiento que el Gateway Exclusive.

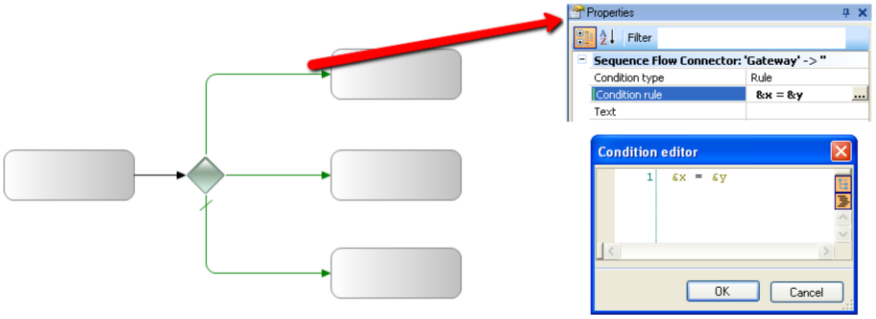
- **Inclusive:** *Divergencia:* el flujo seguirá por todos los caminos que tienen origen en el Gateway que cumplan las condiciones del Gateway (se tendrá una por cada camino).
Convergencia: realiza una sincronización de todos los caminos por donde el flujo del proceso efectivamente llegará al gateway.

- **Parallel:** *Divergencia:* el flujo avanza por todos los caminos que tienen origen en el Gateway.
Convergencia: realiza una sincronización de todos los caminos que llegan al Gateway

Luego veremos en detalle la utilización de cada uno de ellos en diferentes ejemplos de modelado de procesos

GeneXus

Exclusive Gateway



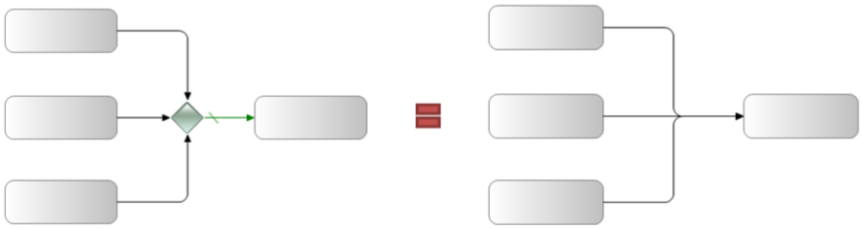
Flujo -> Uno de los caminos

Los **Exclusive** gateways (decisiones) son lugares dentro de un proceso de negocio donde la secuencia de flujo puede tomar dos o más caminos alternativos. Para una determinada instancia del proceso, sólo se puede tomar uno de los caminos.
Una decisión no es una actividad desde la perspectiva del proceso de negocio, es un tipo de puerta de enlace que controla la secuencia de flujo entre las actividades.
Puede ser pensado como una pregunta que se hace en ese momento en el proceso y que tiene un conjunto definido de respuestas alternativas (solo una de las cuales puede ser verdadera en una determinada instancia).

A cada conector saliente se le asocia una regla de condición.
Durante la ejecución del gateway se evalúan las reglas condicionales de cada una de los conectores salientes. La primera que evalúe como verdadera será la que determine el camino a seguir.
Si ninguna de las reglas evalúa en verdadero entonces se sigue por el camino default (el conector que esta marcada con una raya transversal).
El conector default no es obligatoria, en ese caso se debe asegurar que en cada instancia al menos una de las reglas evaluará como verdadero.

GeneXus

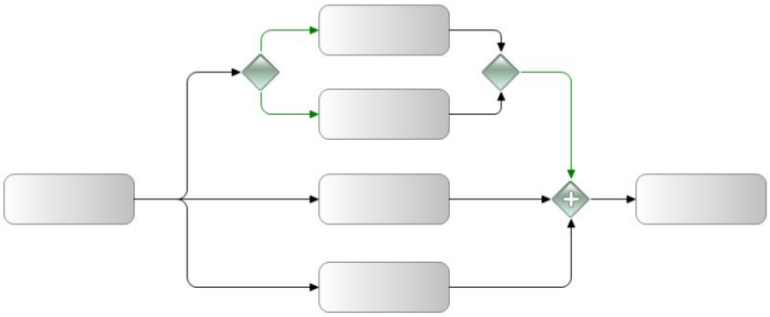
Exclusive Gateway



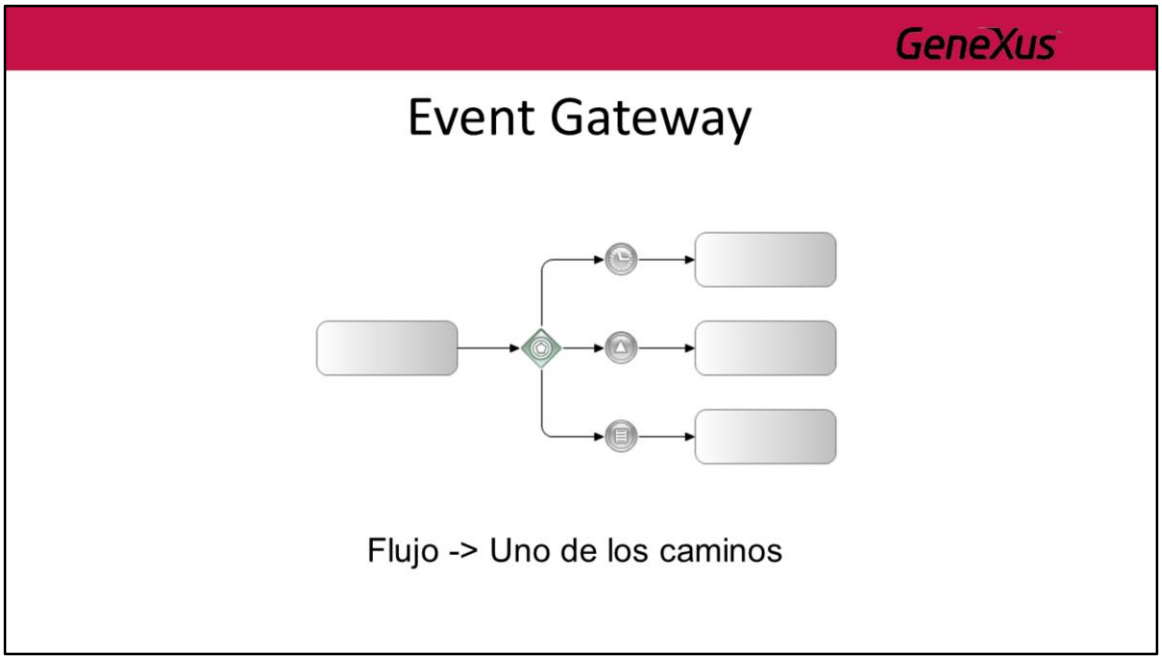
Los **Exclusive** gateway también pueden ser utilizadas para sincronización, aunque su utilización rara vez es necesaria para el modelado ya que generalmente puede ser modelado sin el mismo (como se ve en la otra figura) obteniendo el mismo comportamiento.

GeneXus

Exclusive Gateway



Hay ciertas situaciones en las que un **Exclusive** gateway si es requerida para sincronizar.
En el ejemplo de la figura, si no se utilizara el **Exclusive** para sincronizar el resultado de un gateway anterior, **Parallel** gateway tendría cuatro conectores de secuencia de entrada. Sin embargo, sólo tres de las cuatro secuencias de flujo podrían pasar en cada vez, por lo tanto, el proceso quedaría atascado en ese **Parallel** gateway.

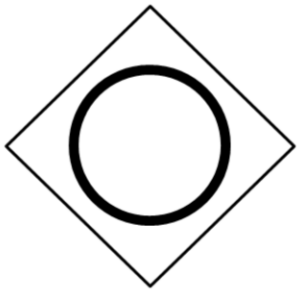


Un gateway de tipo **Event** es un caso particular de un **Exclusive** gateway (en el sentido que solo uno de los caminos posibles puede ser elegido) en que la decisión de que camino seguir depende de la ocurrencia de un evento y no de la evaluación de una regla de condición.

Cabe aclarar que si nunca ocurre ninguno de los eventos especificados, el proceso quedará estancado, por lo que se recomienda utilizar siempre un evento **Timer** como una de las alternativas.

GeneXus

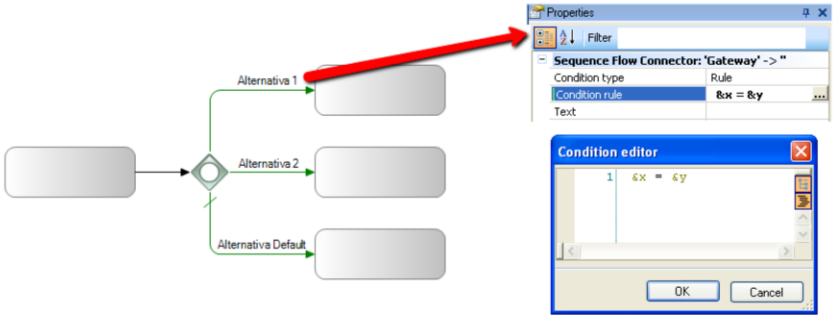
Inclusive Gateway



Un **Inclusive** gateway es similar al exclusive en el sentido que cada conector tiene asociada una regla condicional. La diferencia es que se evalúan todas las reglas y se sigue por todas aquellas que evalúan como verdadero.

Se recomienda el uso de caminos default a fin de evitar que el proceso quede atascado en cualquier situación.

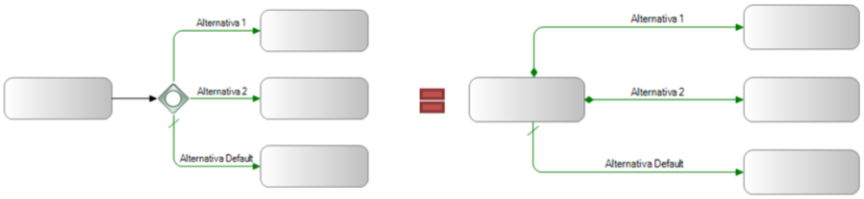
Inclusive Gateway (Ramificación)



Flujo -> Todos los caminos

GeneXus

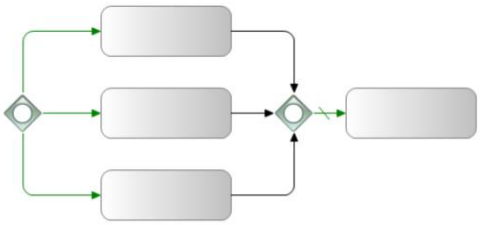
Inclusive Gateway (Ramificación)



Se puede modelar este mismo comportamiento sin utilizar una **Inclusive** gateway. En su lugar podemos utilizar secuencias de flujo de tipo condicional, las cuales también tienen asociada una regla de condición.

GeneXus

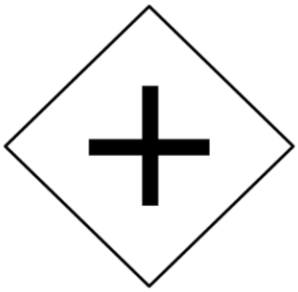
Inclusive Gateway (Sincronización)



Un **Inclusive** gateway también puede ser utilizado para sincronizar.
En este caso el gateway espera por todas secuencias de flujo que fueron seleccionadas en una paso anterior.
Actualmente el motor de gxflow solo soporta esta modalidad cuando el **Inclusive** gateway que sincroniza esta emparejada con otro **Inclusive** gateway como lo muestra la figura.

GeneXus

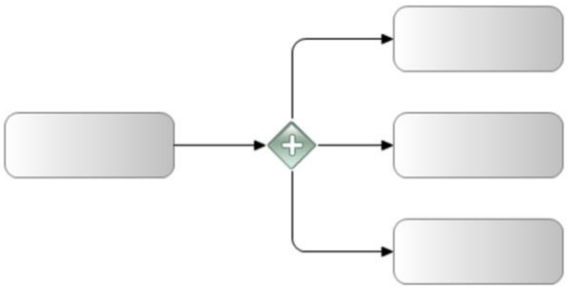
Paralell Gateway



Un **Paralell** gateway permite crear flujos paralelos (o sincronizarlos cuando se utiliza en esa otra modalidad).

GeneXus

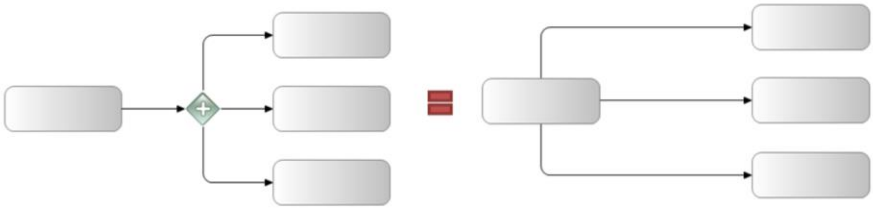
Paralell Gateway



Simplemente divide el flujo en dos o más caminos paralelos. No evalúa ninguna condición.

GeneXus

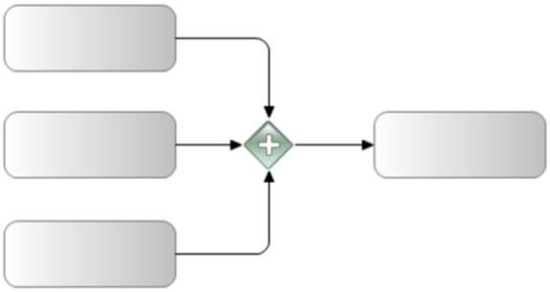
Paralell Gateway (Ramificación)



Se puede modelar este mismo comportamiento sin utilizar un **Paralell** gateway como muestra la figura. Sin embargo, el uso del gateway puede llegar a clarificar el diagrama en determinadas circunstancias.

GeneXus

Paralell Gateway (Sincronización)

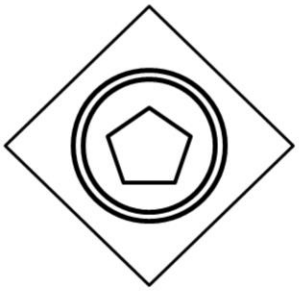


Cuando el **Paralell** gateway es utilizado para sincronización, esta espera por todas las secuencias de flujo entrantes para entonces si poder seguir adelante.

GeneXus

Event Gateway

- Dada una lista de posibles eventos se necesita esperar por la ocurrencia del primero de ellos



Un gateway de tipo **Event** es un caso particular de un **Exclusive** gateway (en el sentido que solo uno de los caminos posibles puede ser elegido) en que la decisión de que camino seguir depende de la ocurrencia de un evento y no de la evaluación de una regla de condición.

Cabe aclarar que si nunca ocurre ninguno de los eventos especificados, el proceso quedará estancado, por lo que se recomienda utilizar siempre un evento **Timer** como una de las alternativas.

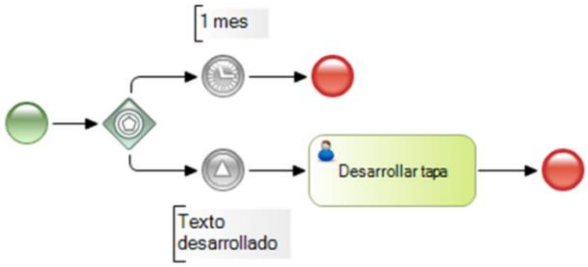
Event Gateway

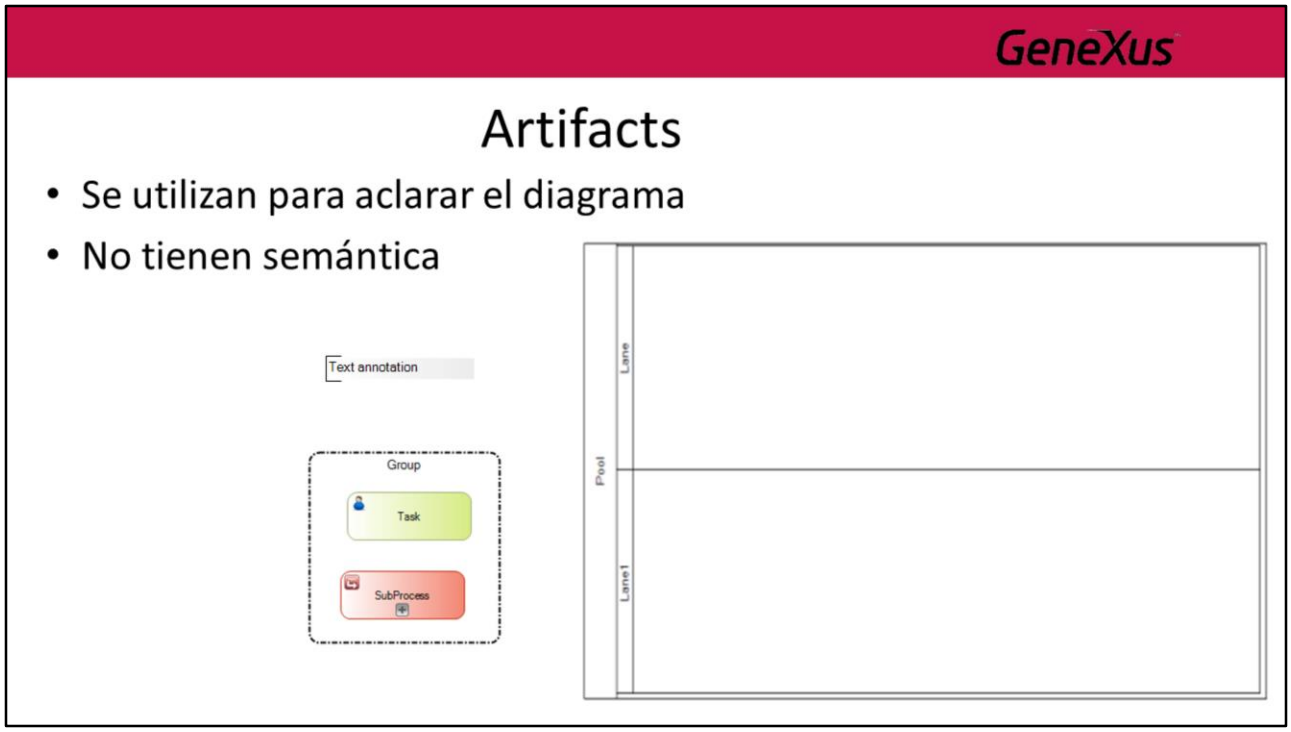
- El siguiente proceso puede quedarse indefinidamente bloqueado esperando la llegada de la señal

Ramificación basada en eventos

- Queremos evitar el bloqueo imponiendo un límite máximo de espera de 1 mes

Event Gateway





Los Artifacts son símbolos que se usan dentro de los diagramas para dar mas claridad al mismo. Los artifacts disponibles son:

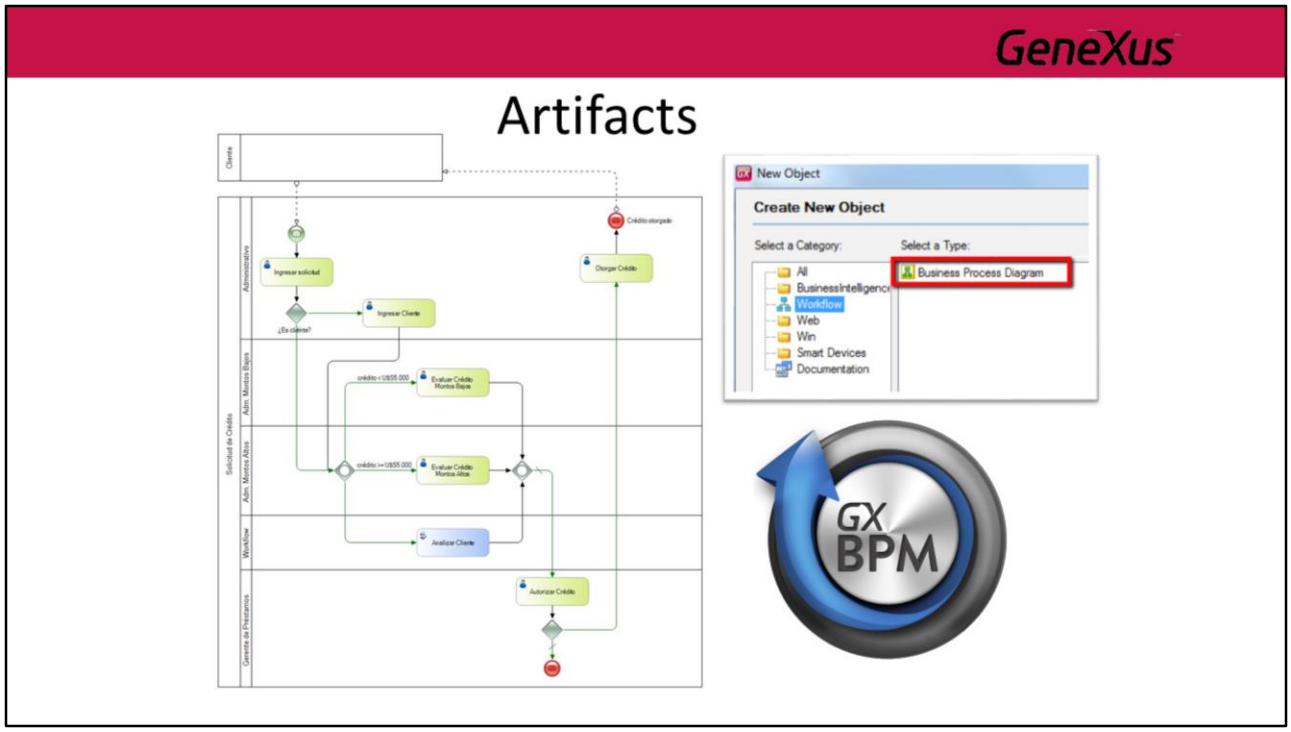
Text annotation: Las anotaciones de texto son un mecanismo para que el modelador pueda proporcionar información adicional para el lector del diagrama.
Gráficamente se representa con un rectángulo abierto con una línea de color negro sólido.
Un anotación de texto se puede conectar a un determinado elemento del diagrama utilizando una asociación (flecha punteada).

Group: Un grupo es un artefacto que proporciona un mecanismo visual para agrupar elementos de un diagrama de manera informal. Gráficamente se representa con un rectángulo de esquinas redondeadas formado por una línea punteada.

Pool: Representa el proceso de un participante y es utilizado cuando el diagrama involucra dos entidades de negocio o participantes separados. Las actividades dentro de los contenedores son considerados procesos

autónomos, por lo que el flujo de mensajes se definen como el mecanismo para tener comunicación entre dos participantes.

Lane: Permite asociar las actividades con una función o rol específicos de la empresa.



Para **modelar procesos de negocio** se puede utilizar el objeto Business Process Diagram que se encuentra en GeneXus pero además se puede utilizar la herramienta **GeneXus Business Process Modeler** (genexus.com/gxbpm) que se distribuye de forma independiente de GeneXus. Esta herramienta está pensada para resolver dos escenarios, los cuales se describen a continuación.

Escenario 1: fase de análisis

El primero en la fase inicial del proyecto donde tenemos a los analistas, que son funcionales que entienden muy bien del negocio de la empresa y conocen los procesos de negocio pero no necesariamente tienen un perfil técnico. Y lo que precisan es una herramienta que les permita capitalizar ese conocimiento, documentarlo y formalizarlo de forma que luego pueda ser automatizado, en este caso con GXflow.

Escenario 2: modificación de procesos que ya están en producción
El segundo escenario se ubica en el otro extremo del proyecto, cuando el sistema está en producción o se implantando. Por ejemplo un ERP que se está implantando

en una empresa y se necesita parametrizar determinadas reglas de negocio.

GeneXus Business Process Modeler provee todas las funcionalidades de edición del modelador de procesos incluido en la versión estándar de GeneXus.

Las funcionalidades más importantes son:

- Edición del flujo que define el proceso. Esto implica poder agregar o quitar cualquier tipo de elemento de la notación BPMN al diagrama además de poder conectar elementos.
- Edición de las propiedades de los elementos de un diagrama.
- Poder asociar el objeto GeneXus que se encargará de resolver la operativa de la tarea en ejecución (transacción, web panel o procedimiento).
- Edición de reglas del proceso (por ej.: reglas subject, de definición del lapso de un evento timer, condicionales, etc.)
- Creación de nuevos procesos. Solamente se permitiría crear nuevos diagramas de procesos de negocio pero no cualquier objeto GeneXus.
- En este escenario solamente se pueden editar diagramas de procesos, para los demás objetos GeneXus que se podrán asociar a los diagramas solo se podrán visualizar algunas de sus partes (por ej.: estructura de transacciones, regla parm y documentación) pero no editarlas.

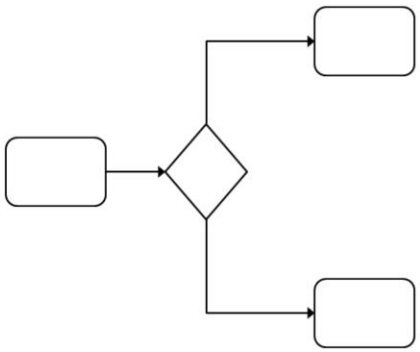
GeneXus

Módulo 2

Ramificación y unificación de caminos

GeneXus

Ramificación



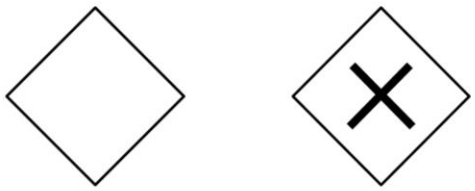
Ramificación exclusiva

- En determinado punto del proceso se tienen distintos caminos posibles por donde puede seguir el proceso
- Solamente se puede seguir por uno de esos caminos
- La elección del camino depende de la evaluación de condiciones de tipo verdadero/falso

GeneXus

Ramificación exclusiva

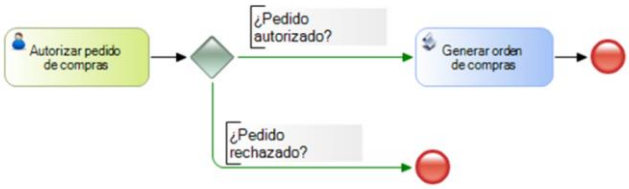
- Exclusive Gateway (compuerta exclusiva)



- En GeneXus
  - Propiedad *Type = Exclusive*

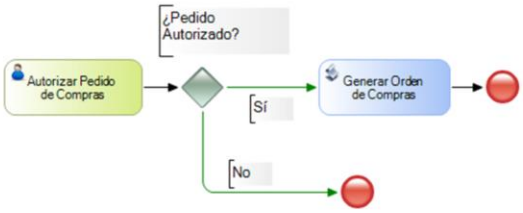
Ramificación exclusiva

- Exclusive Gateway
  - Para cada conector de salida se define una condición de tipo verdadero/falso



Ramificación exclusiva

- Exclusive Gateway
  - En general se recomienda etiquetar el gateway y los conectores



Ramificación exclusiva

- Exclusive Gateway
  - Se puede tener un conector *default* que se seguirá en caso de que las demás condiciones evalúen como falso

GeneXus

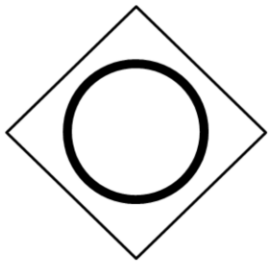
Ramificación Inclusiva

- En determinado punto del proceso se tienen distintos caminos posibles por donde puede seguir el proceso
- Se puede seguir por uno o más caminos
- La elección de los caminos a seguir dependen de la evaluación de condiciones

GeneXus

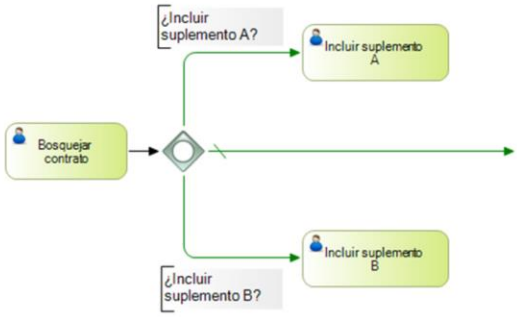
Ramificación Inclusiva

- Inclusive Gateway (compuerta inclusiva)



Ramificación Inclusiva

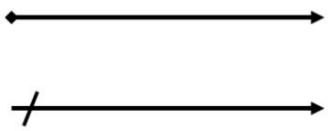
- Inclusive Gateway



GeneXus

Ramificación Inclusiva

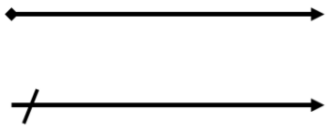
- Condition Sequence Flow Connector
  - Evitan tener que usar un inclusive gateway
  - En general es preferible usar el gateway porque es más claro



GeneXus

Ramificación Inclusiva

- Condition Sequence Flow Connector
  - En GeneXus
    - Propiedad *Type = Conditional*



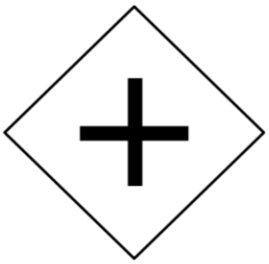
Ramificación Paralela

- El flujo debe avanzar por dos o más caminos paralelos sin excepciones

GeneXus

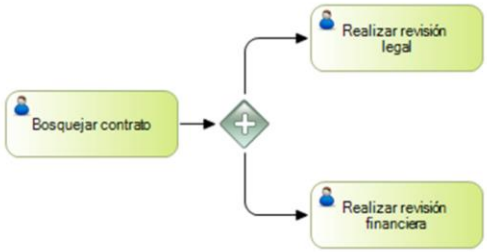
Ramificación Paralela

- Parallel Gateway (compuerta paralela)



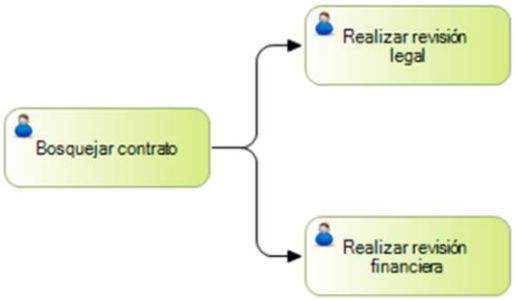
Ramificación Paralela

- Parallel Gateway



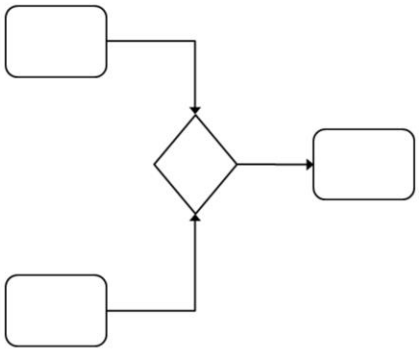
Ramificación Paralela

- Simplificación



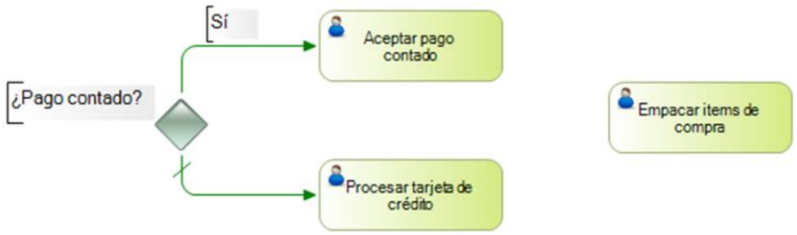
GeneXus

Unificación



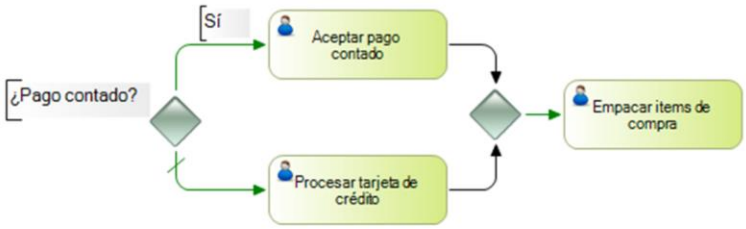
GeneXus

Unificación de caminos alternativos



GeneXus

Unificación de caminos alternativos



Unificación de caminos alternativos

- Simplificación

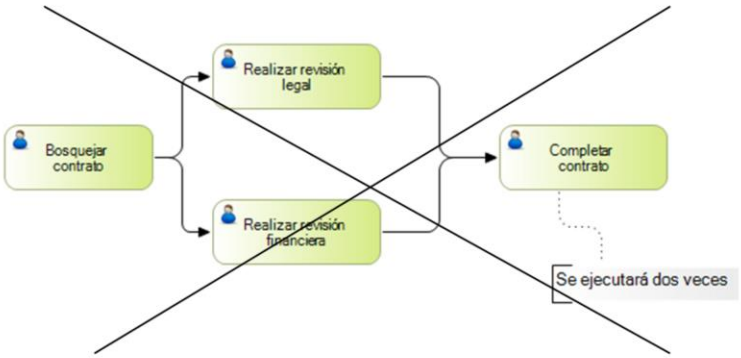
GeneXus

Unificación de caminos paralelos

Unificación de caminos paralelos

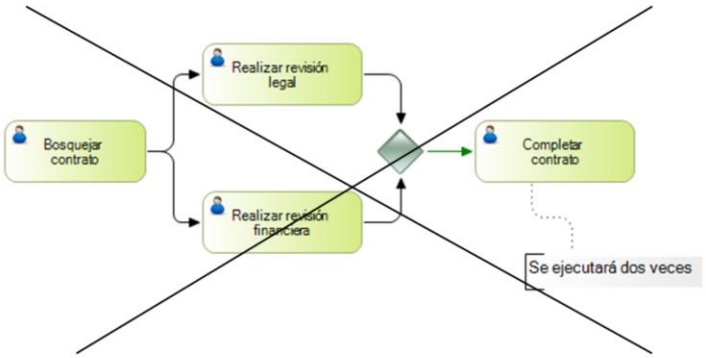
GeneXus

Unificación de caminos paralelos



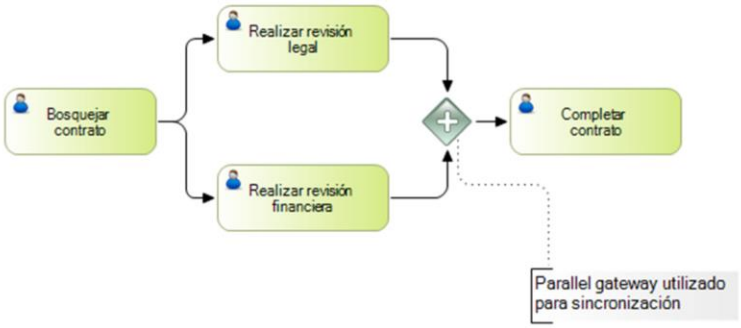
GeneXus

Unificación de caminos paralelos



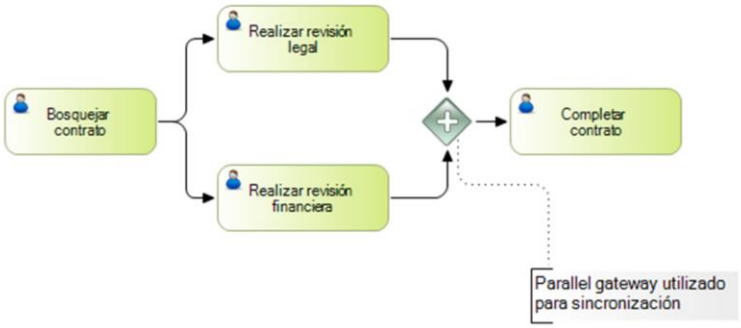
GeneXus

Unificación de caminos paralelos



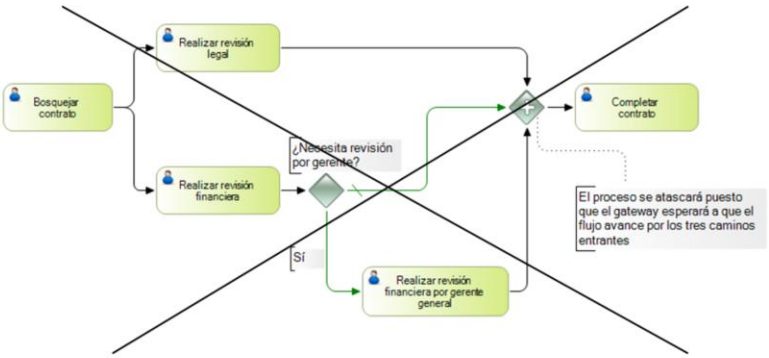
GeneXus

Unificación de caminos paralelos

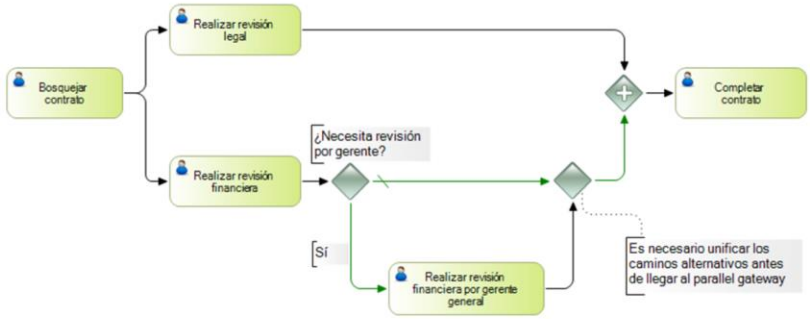


GeneXus

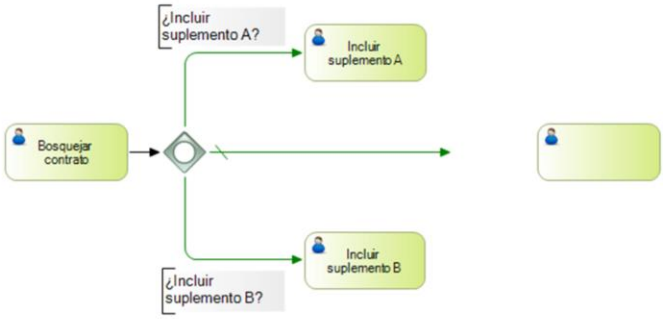
Unificación de caminos paralelos



Unificación de caminos paralelos

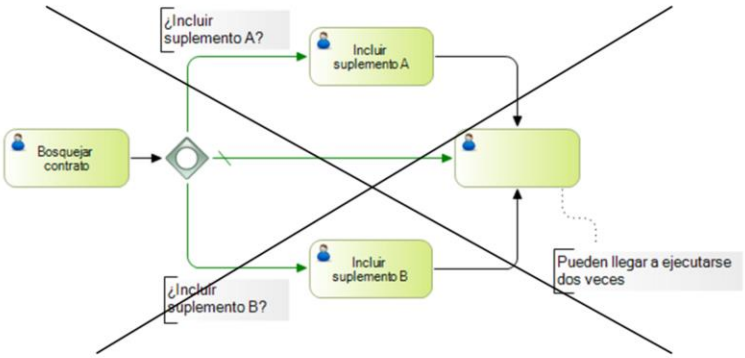


Unificación de caminos paralelos condicionales



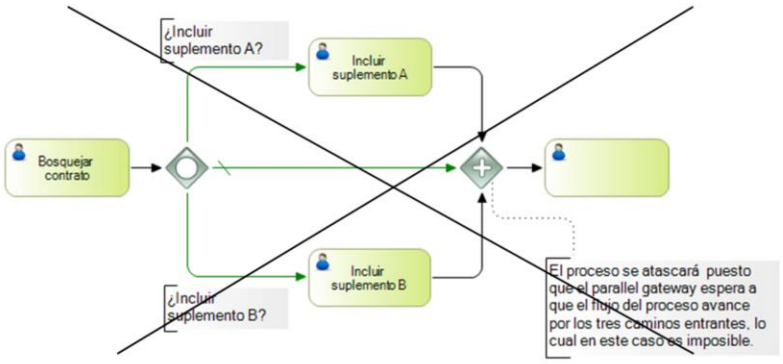
GeneXus

Unificación de caminos paralelos condicionales



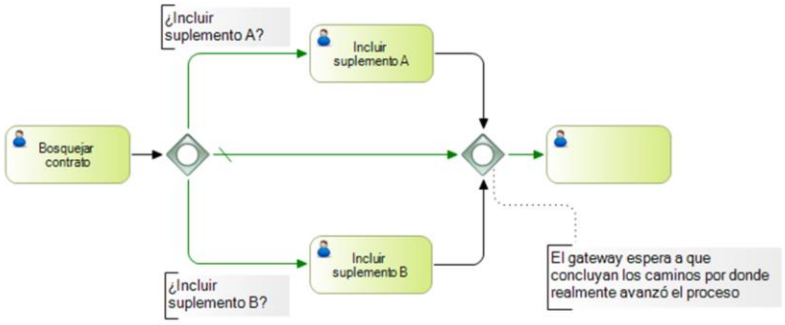
GeneXus

Unificación de caminos paralelos condicionales



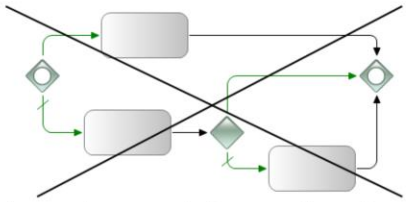
GeneXus

Unificación de caminos paralelos condicionales



Unificación de caminos paralelos condicionales

- Limitación en GeneXus
  - No pueden haber otros gateways entre el inclusive gateway de ramificación y el de unificación



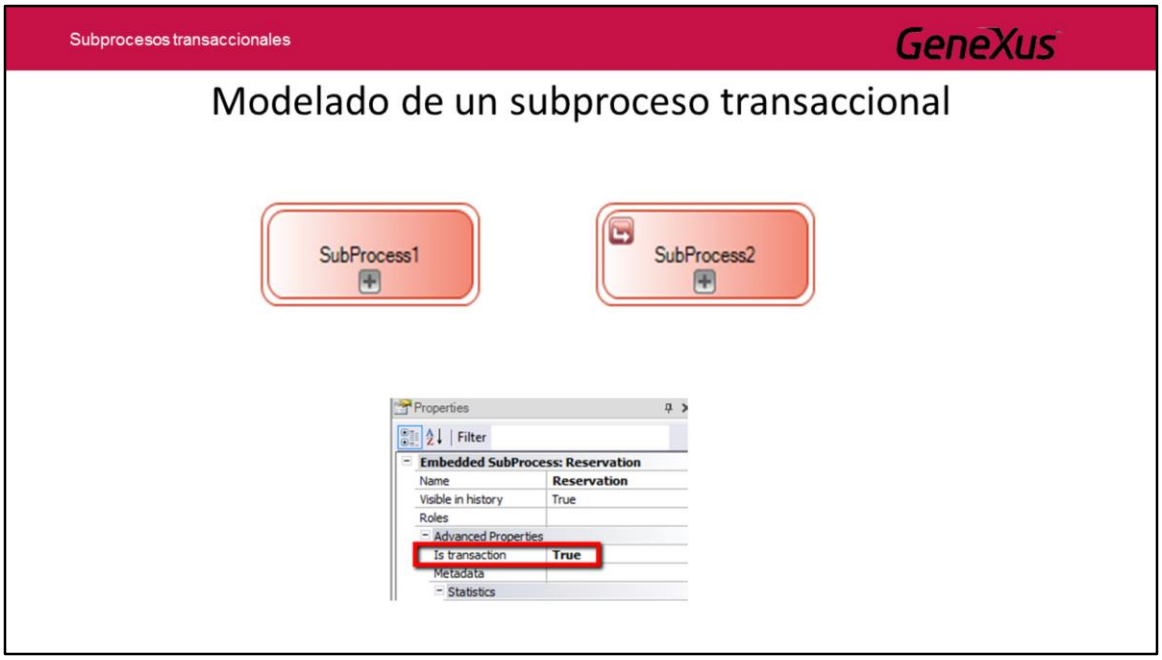
  - Solución: incluir los gateways intermedios dentro de un subproceso

GeneXus

Módulo 3

Subprocesos transaccionales

Hay procesos en los cuales es necesario coordinar varias actividades que necesitan cumplirse exitosamente todas ellas para que el flujo del proceso pueda seguir y en caso de que alguna no se cumpla satisfactoriamente, es necesario regresar a todas ellas a su estado inicial.
Estas actividades integran lo que denominamos una unidad de trabajo lógica (UTL), es decir una unidad atómica que debe ejecutarse en forma indivisible. Una UTL en un proceso de negocios podría durar días o incluso semanas.



Para modelar estos escenarios de negocio con transacciones, donde cada transacción está compuesta de varias operaciones que deben realizarse y terminarse correctamente todas ellas, o bien ninguna de ellas, utilizamos a los subprocesos transaccionales.

En BPMN se representan con un borde doble.

En GeneXus para definir un subproceso como transaccional, asignamos la propiedad **Is transaction** con el valor True. Al hacerlo, veremos que cambia el borde del símbolo de subproceso y se representa con un borde doble.

Resultados posibles de una transacción

1) Todas las operaciones finalizan satisfactoriamente

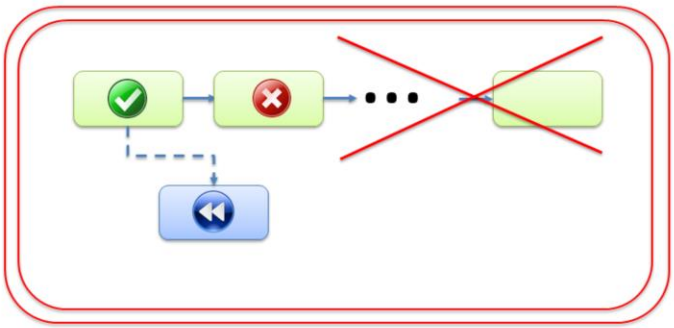
Un subproceso transaccional finaliza satisfactoriamente cuando los cambios de los datos en la base de datos terminan en un estado consistente. En el caso de que se produzca un fallo en la ejecución de una transacción, es necesario iniciar acciones que deshagan los cambios, llevando los datos al estado que tenían antes de estos cambios.

Las transacciones en un modelo de procesos de negocios pueden tener tres resultados posibles:

- Que todas las operaciones **finalicen satisfactoriamente**, con lo cual el proceso continúa por el flujo normal

Resultados posibles de una transacción

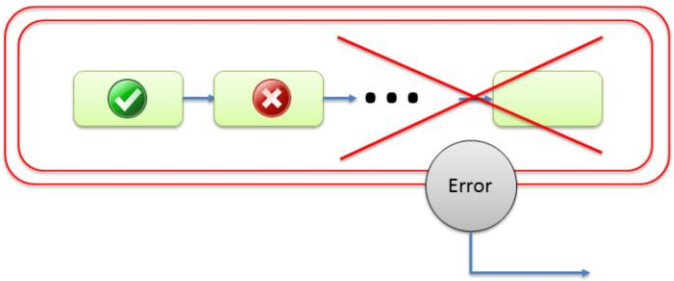
2) Ocurre una falla: es necesario revertir las tareas ya completadas



- Que **ocurra una falla** y es necesario revertir las actividades que ya fueron completadas dentro del subproceso. En este caso deben ejecutarse ciertas tareas de compensación que se encargan de dejar al proceso en el estado inicial existente antes de iniciarse el subproceso transaccional.

Resultados posibles de una transacción

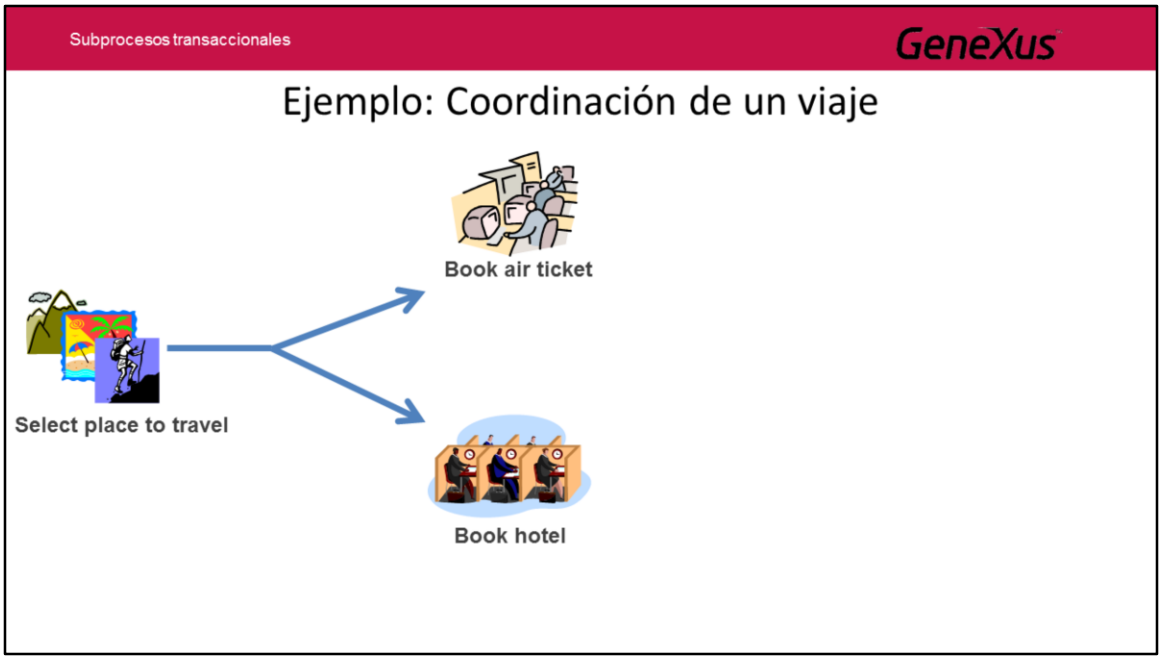
3) Ocurre un error inesperado: no se realiza ninguna compensación



- Que **suceda un error inesperado**, en cuyo caso las actividades del subproceso son interrumpidas y no se ejecuta ninguna actividad compensatoria. En este caso el proceso continúa con la ejecución de un evento intermedio de error.

Por lo tanto, para modelar un subproceso transaccional, es necesario capturar eventos que respondan a estas situaciones, es decir eventos de error y de cancelación de procesos debido a una falla.

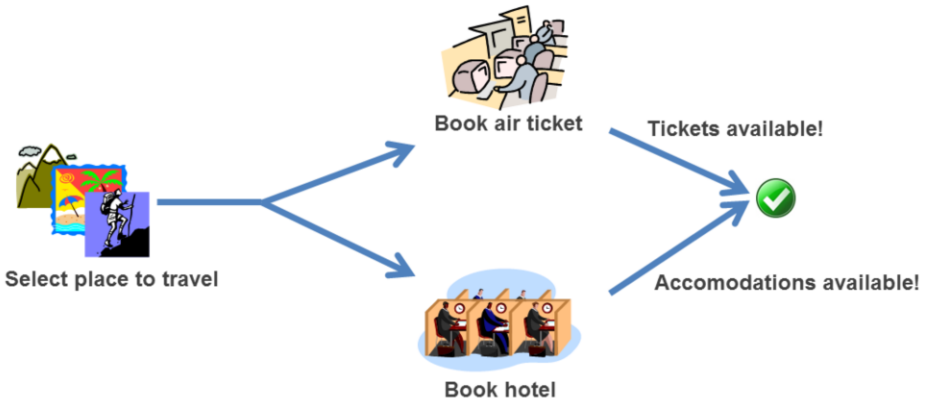
Veremos a continuación los conceptos mencionados hasta aquí, en base a un ejemplo.



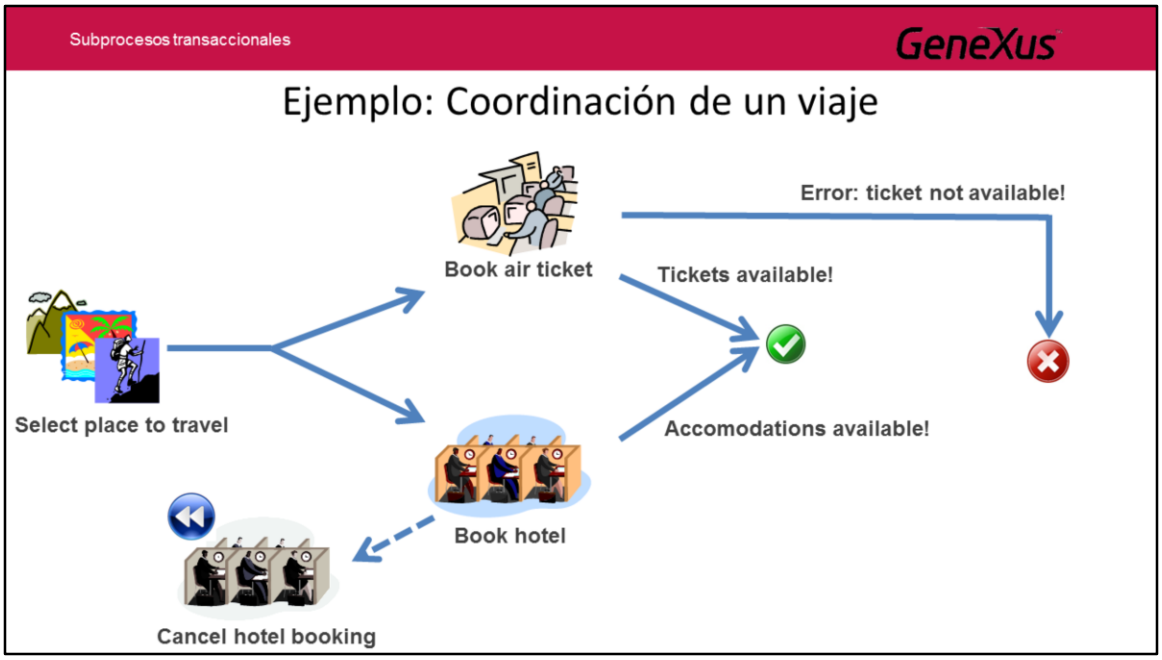
En la Agencia de Viajes en la cual hemos venido trabajando, la coordinación de un viaje podría ser un ejemplo de un subproceso transaccional, ya que para poder completarse correctamente hay que poder completar exitosamente varias actividades, como podrían ser la reserva del pasaje aéreo, la reserva del hotel, el alquiler de coche y la entrada a una atracción turística, entre otras.

Consideremos un proceso de reserva que consiste de dos actividades: la reserva de un vuelo y la reserva de una habitación de hotel.
Consideremos que ambas actividades se ejecutan en una única transacción, es decir que si no se consigue el pasaje aéreo y sí se consiguió la reserva del hotel, es necesario deshacer la reserva del hotel, y viceversa, si se consiguió el pasaje pero no el hotel, debe cancelarse la reserva del pasaje.
Esto implica que debemos considerar los tres casos antes vistos:
Que consigamos ambas reservas y el proceso finalice normalmente.
Que falle alguna de las reservas y el proceso se cancele, por lo que deberá deshacerse la reserva que se haya obtenido, ya que deben obtenerse ambas o ninguna.
Que suceda un error inesperado y el proceso termine.

Ejemplo: Coordinación de un viaje

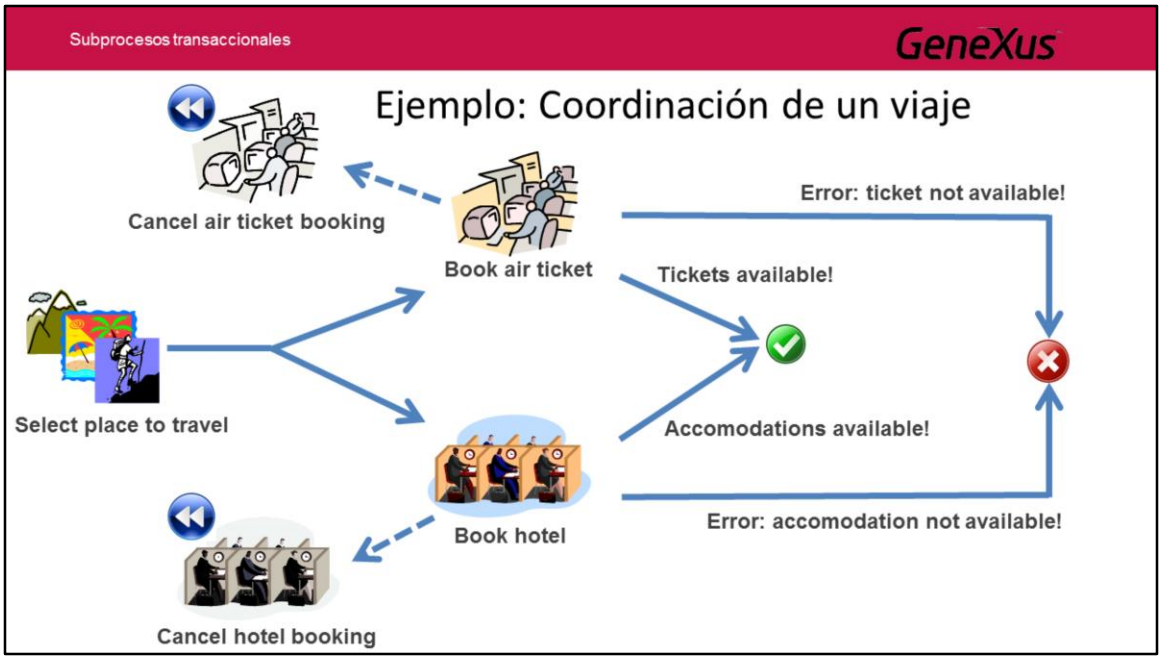


En el caso de que ambas reservas se obtengan satisfactoriamente, el proceso finaliza.



Pero si falla la obtención del pasaje aéreo y se logró finalizar la reserva del hotel, el proceso no puede darse por completado porque una de las actividades de la transacción no se cumplió, por lo que es necesario deshacer las actividades que llegaron a cumplirse.

En este caso sería necesario ejecutar un proceso compensatorio que deshaga la reserva del hotel, de modo que el proceso de coordinación de viaje queda completamente sin efectuar.



Si por el contrario, la reserva del ticket aéreo hubiera sido exitosa y no fue posible obtener lugar en el hotel, es necesario deshacer la reserva del pasaje.

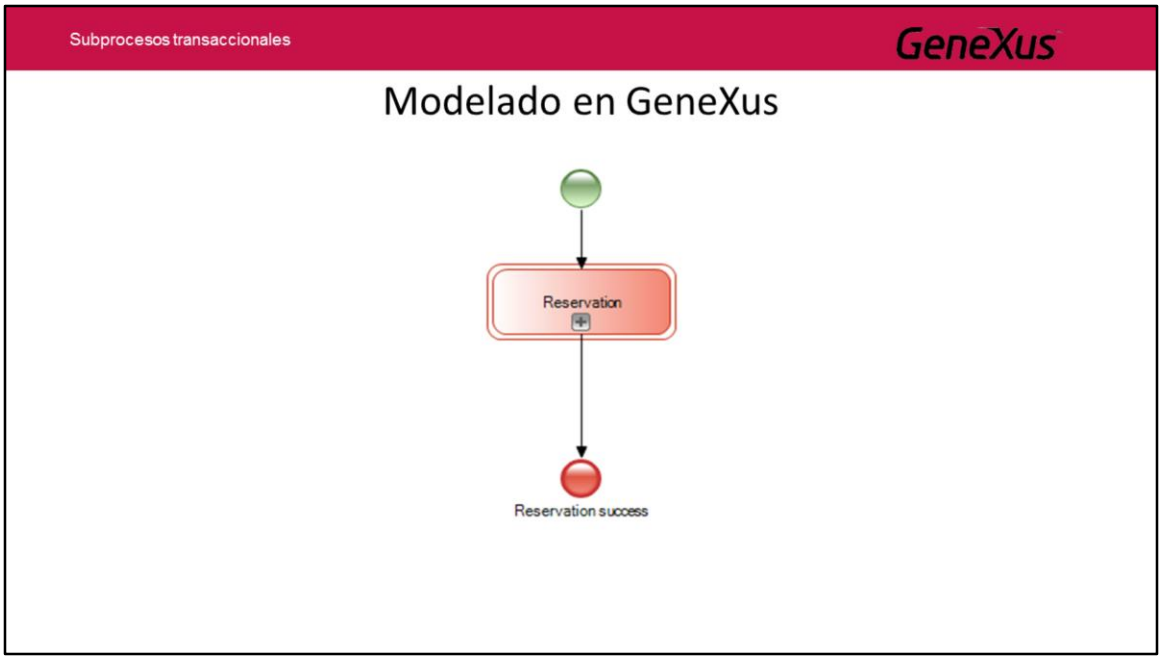
Las tareas de cancelar las reservas de hotel o de los pasajes se conocen como actividades de compensación y muchas veces son realizadas por un sistema externo.

Hasta ahora hemos visto el caso de que el proceso transaccional finalice exitosamente porque todas las actividades que lo integran finalizaron correctamente, o el caso de que se produzca una falla en alguna de las actividades del proceso transaccional y esta falla produce una cancelación del proceso, siendo necesario deshacer las actividades que llegaron a completarse de modo de que el proceso se revierta hasta el estado anterior al comienzo del mismo.

El tercer caso tiene que ver cuando se produce un error que no puede ser manejado por el subproceso y evita que el mismo continúe, como por ejemplo que un servidor no responda o que se produzca una caída del sistema. En este caso las actividades del subproceso son interrumpidas sin compensación y la base de datos hace rollback hasta el último commit anterior al comienzo del subproceso.

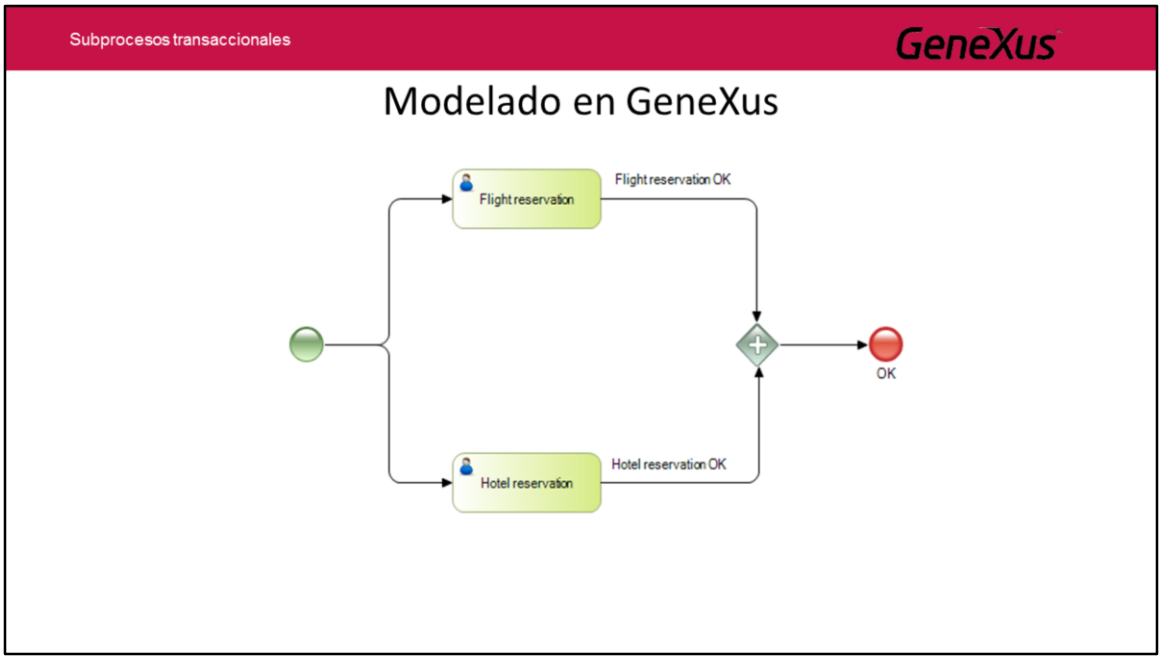
En nuestro ejemplo, cuando el proceso de reserva no finaliza exitosamente, se detecta esa situación y se le comunica al cliente que no es posible realizar la reserva.

Veamos como modelamos este proceso con GeneXus BPM Suite.



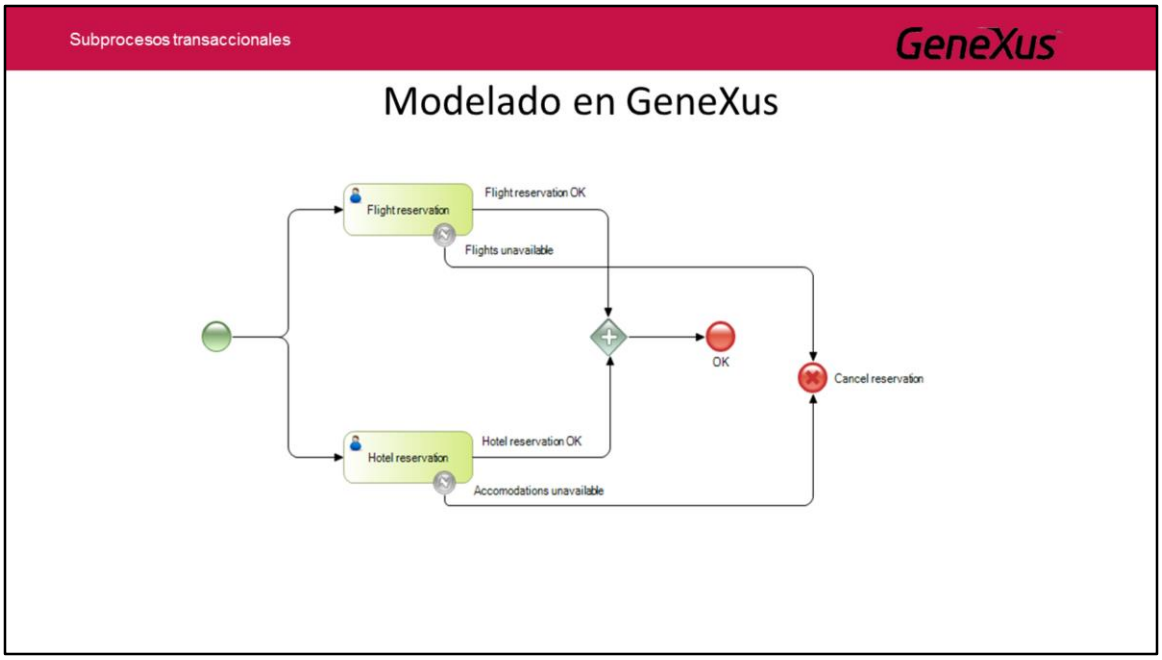
En primer lugar ejecutamos el IDE de GeneXus o el GeneXus Business Process Modeler y en nuestra Knowledge Base, creamos un objeto Business Process Diagram.

Arrastramos desde la toolbar un None Start Event, un símbolo de subproceso al que conectaremos desde el Start Event y un None End Event al que conectamos desde el subproceso. En las propiedades del subproceso, asignaremos a la propiedad **Is transaction** el valor **True.** Esto hará que el símbolo del subproceso cambie y se vea con doble borde, que es la forma en que en BPMN se representa a un subprocesso transaccional.



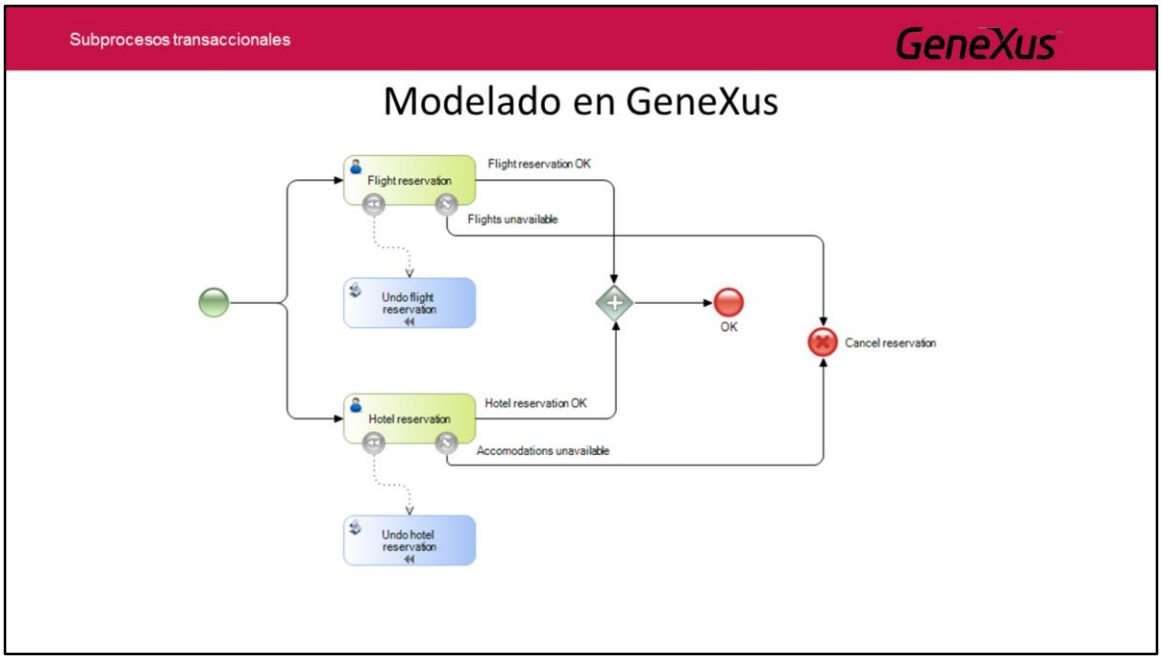
Ahora abrimos el subproceso, comenzamos agregando un None Start Event, luego insertamos 2 tareas interactivas, una para la reserva de vuelos y otra para la reserva del hotel y las conectamos desde el evento de Start. La salida de ambas las unimos en un Parallel Gateway, y la salida del mismo la conectamos a un None End Event.

Con esto estamos modelando el caso de que ambas tareas finalicen correctamente, en cuyo caso el Parallel Gateway podrá sincronizar ambos flujos que le llegan y el flujo saliente seguirá hasta el End event. Sin embargo es necesario tomar en cuenta los otros casos, ya que si uno de las dos tareas falla, el flujo jamás podrá avanzar debido al Parallel Gateway.



Para tomar en cuenta las posibles fallas de las reservas, agregamos sendos Intermediate Error Event a cada tarea y las salida de los mismos las conectamos a un Cancel End Event.

Este evento Cancel se disparará cuando alguna de las dos tareas interactivas falle. El evento Cancel es un evento especial utilizado en los procesos transaccionales, que se dispara **durante** la ejecución del subproceso, en lugar de dispararse al finalizar el mismo.



Cuando la transacción se cancela, antes de que pase el control al proceso padre, se disparan las **tareas de compensación**. Esto es que todas las actividades que terminaron exitosamente hasta ese momento, deben ser deshechas ejecutando las tareas de compensación definidas para cada una de ellas.

En nuestro caso, debemos definir tareas que deshagan la reserva del pasaje o la reserva del hotel, según el caso.

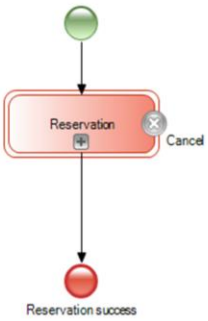
Para esto agregamos un Intermediate Cancel Event adjunto a la tarea de reserva de vuelo, insertamos una tarea script con el nombre Undo flight reservation y la unimos desde el Intermediate Cancel Event.

Repetimos lo mismo para la tarea de reserva de hotel, insertando otro Intermediate Cancel Event y una tarea no interactiva con el nombre Undo hotel reservation, que conectamos desde el Intermediate Cancel.

Observamos que desde el momento que conectamos las tareas script con los Intermediate Cancel Event, cambia el aspecto de la conexión ya que ahora el conector es del tipo Association y aparece sobre las tareas un símbolo “<<”. Este tipo de conector se representa con una línea punteada y permite establecer una relación diferente a la de los conectores de secuencia ya que las actividades asociadas no pueden ser parte de ninguna secuencia de flujo, es decir, no pueden tener conectores de secuencia de entrada o salida.

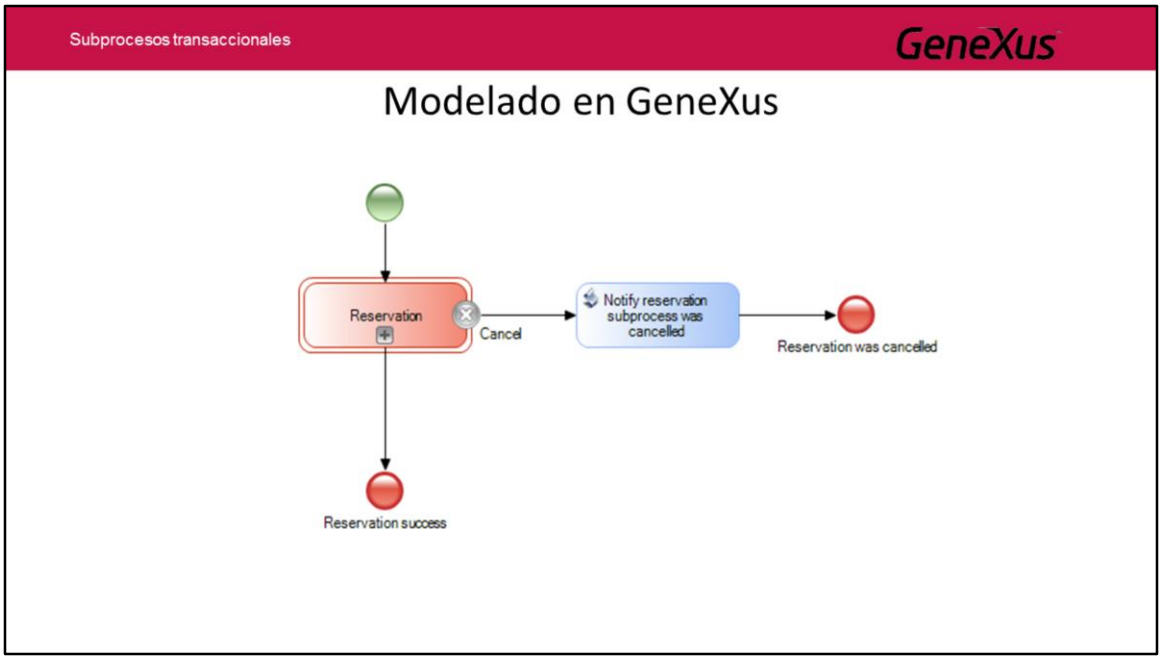
GeneXus

Modelado en GeneXus



Una vez que la compensación de la transacción se completa, se interrumpe la ejecución en el subproceso y se dispara un evento de cancelación desde el subproceso al proceso padre.

Para capturar este evento desde el proceso padre, debemos adjuntar un Cancel Intermediate Event al símbolo del subproceso.



Al recibirse la cancelación en el proceso padre, comunicamos al usuario que el proceso de reserva fue cancelado y terminamos el proceso.

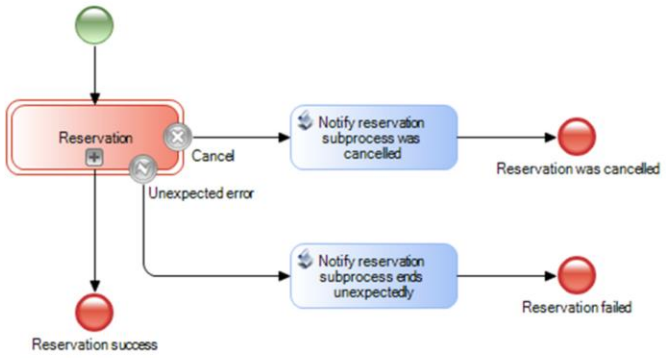
Al igual que vimos anteriormente con el evento Error, el Cancel utiliza un mecanismo de throw-catch desde el Cancel End Event del subproceso, hacia el Cancel Intermediate Event adjunto al subproceso, en el diagrama padre.

Este evento cancel dispara automáticamente las tareas de compensación que hayan sido definidas en el subproceso, asegurando la integridad transaccional del mismo.

Cualquier otro tipo de interrupción que sufra el subproceso como un evento de error, se aborta la ejecución del mismo sin ninguna compensación.

GeneXus

Modelado en GeneXus



http://wiki.genexus.com/commwiki/servlet/hwikibypageid?24922

A fin de proveer un manejo de este error, insertamos un Intermediate Error Event adjunto al símbolo del subproceso y mediante una tarea script enviamos un aviso de que el proceso de reserva finalizó en forma inesperada.

De esta forma hemos visto el funcionamiento de los subprocesos transaccionales y cómo los modelamos en GeneXus.

Puede encontrar más información, en el siguiente link:
http://wiki.genexus.com/commwiki/servlet/hwikibypageid?24922

Módulo 4

Actividades múltiples

Ejemplo

- Proceso de compras
  1. Un funcionario ingresa un pedido de compras
  2. Cada ítem del pedido es autorizado por separado por el jefe del funcionario
  3. Si al menos uno de los ítems del pedido fue autorizado entonces se genera la orden de compras correspondiente


Todos los ítems del pedido son autorizados o rechazados a la vez
Ingresar pedido de compras
Autorizar pedido de compras
¿Pedido autorizado?
Generar orden de compras
¿Pedido rechazado?

GeneXus

- Es necesario un mecanismo para declarar que una actividad será ejecutada potencialmente más de una vez
- El número de veces que será ejecutada no es conocido durante el diseño

GeneXus

- Para estos casos BPMN define el concepto de *Looping*
- Dos tipos
  - Standard
  - Multi-instance

GeneXus

Standard Looping

- La actividad será ejecutada secuencialmente mientras se cumpla una condición de tipo verdadero/falso

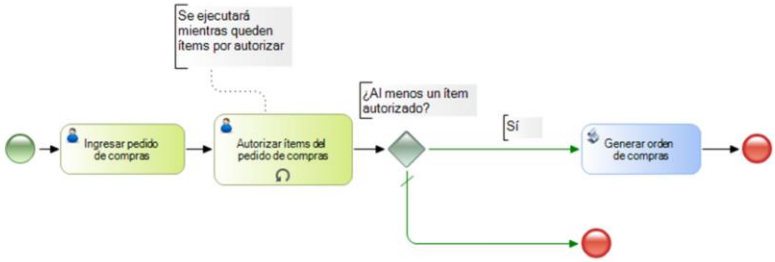
Standard Looping

- En GeneXus se debe especificar la propiedad *Loop type* = Standard

Standard Looping

- Otras propiedades:
  - *Condition rule*: permite especificar la condición que debe cumplirse para que se siga ejecutando la actividad (implementación)
  - *Test time:* especifica el momento en que se evalúa la condición
    - *Before:* la evaluación se hace antes de ejecutar la actividad
    - *After:* primero se ejecuta la actividad y luego se evalúa la condición

Standard Looping



GeneXus

Multi-instance

- La actividad será ejecutada un número de veces que será determinado mediante la evaluación de una expresión numérica
- Las actividades pueden ejecutar en formas secuencial o paralela
- En GeneXus se debe configurar la propiedad *Looping type = Multi-instance*

GeneXus

Multi-instance

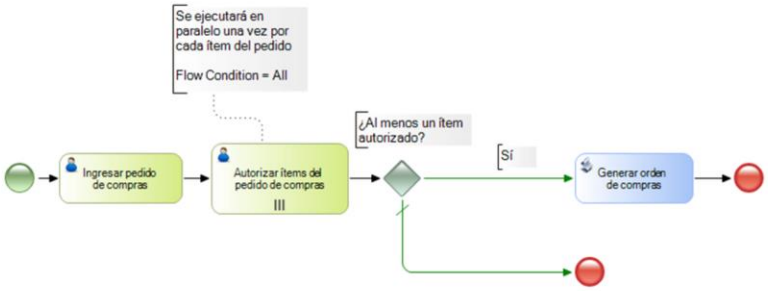
- Otras propiedades:
  - Expresion rule: expresión numérica que indica el número de veces que se ejecutará la actividad (implementación)
  - Ordering
    - Sequential: se ejecutará en forma secuencial
    - Parallel: se ejecutará en forma paralela

Multi-instance

- Otras propiedades:
  - Flow condition (aplica solamente si *Ordering = Parallel)*
    - Determina cómo se unifican los caminos luego de que se ejecute la actividad
    - Valores:
      - » None: el proceso continúa inmediatamente sin unificar nada
      - » One: el proceso luego que se termina de ejecutar la actividad una vez
      - » **All**: similar a una parallel gateway
      - » Complex: permite especificar una condición que indicará cuando puede continuar el proceso

GeneXus

Multi-instance

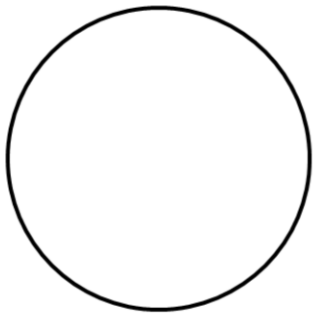


Módulo 5

Eventos

GeneXus

Eventos



Hasta aquí no hemos mencionado dónde comienza y termina el proceso para esto vamos a utilizar el símbolo **Evento.**

Los eventos representan **sucesos que ocurren durante el proceso**.
Hay tres tipos de eventos:

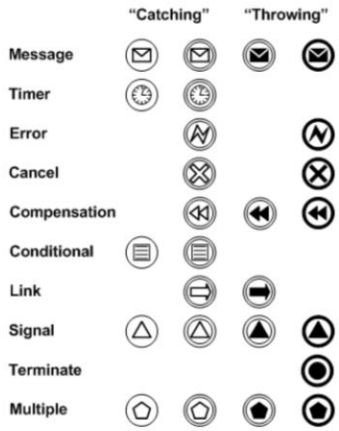
Inicio: Permite indicar dónde inicia el proceso. Un evento inicial **NO** puede tener conectores entrantes.
Intermedio: Permite indicar un suceso entre el inicio y el fin de un proceso.
Fin: Permite indicar dónde termina el proceso. Un evento final **NO** puede tener conectores salientes

Estos se diferencian por el trazo, los de **inicio tienen trazo fino**, los **intermedios tienen doble trazo fino** y los de **fin tienen trazo grueso**.

GeneXus

Eventos

Para cada tipo de evento BPMN define subtipos que permiten especificar la naturaleza del evento



Para cada uno de estos tres tipos BPMN define **subtipos** que permiten especificar la **naturaleza del evento**.

GeneXus

Eventos

Entonces agregamos a nuestro proceso el inicio y fin para indicar dónde comienza y dónde termina.

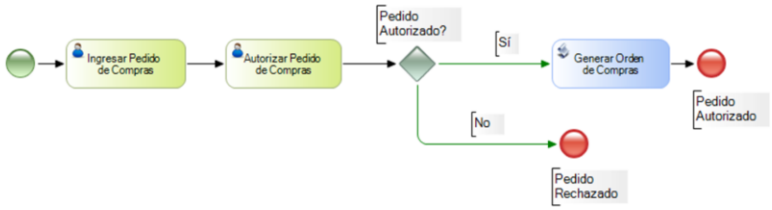
GeneXus

Terminación de Procesos

- Si el flujo del proceso llega a determinado punto, se requiere que el proceso finalice, independientemente de si se tienen otros caminos paralelos que todavía están activos

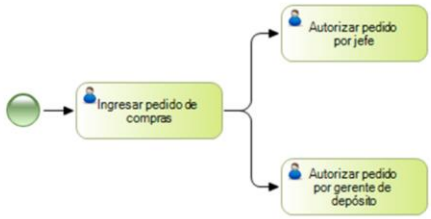
Terminación de Procesos

- El evento de finalización “None” alcanza para indicar el fin de un proceso si solamente se tiene un camino activo



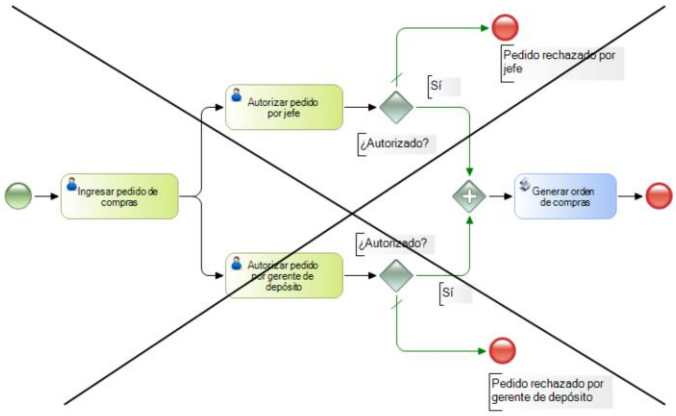
Terminación de Procesos

- ¿Qué pasas si tenemos dos autorizaciones en paralelo?



GeneXus

Terminación de Procesos

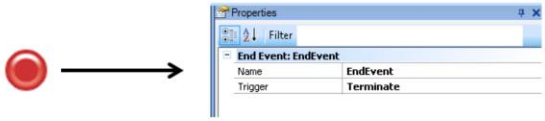


Terminación de Procesos

- Teminate End Event
  - Finaliza un proceso interrumpiendo todas las actividades que estén activas en ese momento
  - Si el proceso se utiliza como subproceso, NO se finaliza el proceso padre

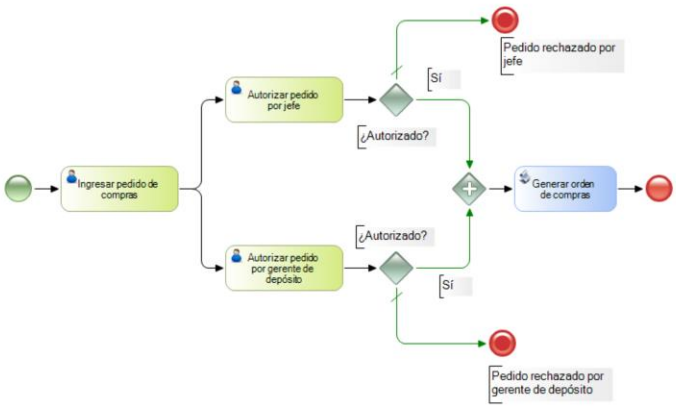
Terminación de Procesos

- Terminate End Event
  - En GeneXus para que un evento de fin sea de tipo *terminate,* se debe establecer la propiedad *Trigger* con el valor *Terminate*



GeneXus

Terminación de Procesos



GeneXus

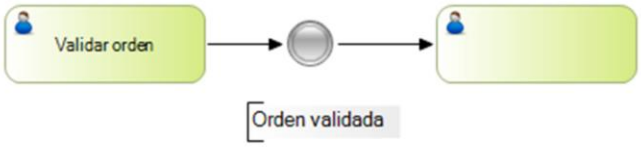
Indicación de Hitos

- Se necesita enfatizar en alguna parte del diagrama que se alcanzó un determinado hito

None Intermediate Event

GeneXus

Indicación de Hitos



Orden validada

Monitoreo de Tiempos

- Pausa
  - En determinado punto el flujo del proceso debe detenerse por un lapso de tiempo

Monitoreo de Tiempos

- Timer Intermediate Event (evento intermedio de temporización)

- En GeneXus:
  - Propiedad *Trigger = Timer*
  - También se tienen propiedades para declarar el lapso de tiempo pero esto corresponde a la implementación

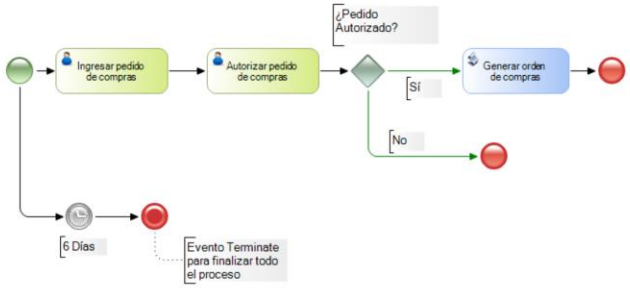
GeneXus

Monitoreo de Tiempos

- Pausa

Monitoreo de Tiempos

- Deadlines de procesos
  - Se necesita acotar la duración total de un proceso

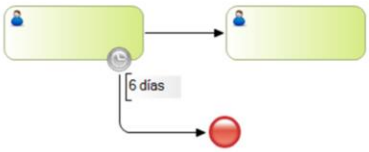


Monitoreo de Tiempos

- Deadlines de actividades
  - Se necesita acotar la duración total de una actividad
  - Para esto se puede adjuntar un evento timer al borde de la actividad

Monitoreo de Tiempos

- Deadlines de actividades
  - Por defecto la actividad es interrumpida si no se completa antes de que ocurra el evento de tiempo especificado y el flujo puede avanzar por un camino alternativo

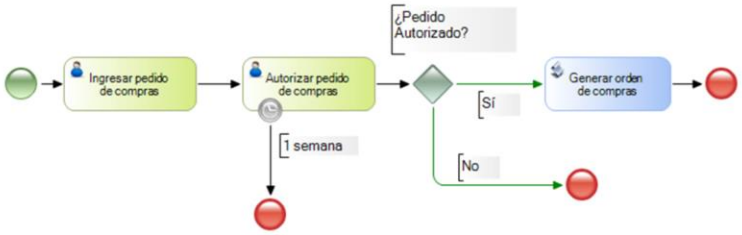


Monitoreo de Tiempos

- Deadlines de actividades
  - Ejemplo
    - "En el proceso de compras si la autorización del pedido demora más de una semana esta se debe interrumpir y el proceso debe terminar"

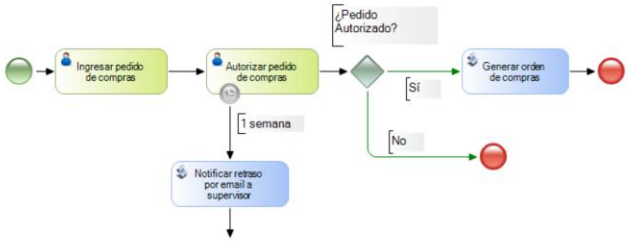
Monitoreo de Tiempos

- Deadlines de actividades
  - Ejemplo



Monitoreo de Tiempos

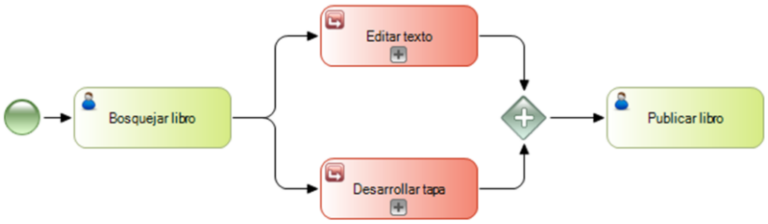
- El evento timer adjunto también puede no interrumpir la tarea
  - En GeneXus
    - Propiedad *Interrupts activity = false*



GeneXus

Señales

- Comunicación entre procesos



Señales

- Comunicación entre procesos
  - Subproceso *Editar Texto*

  - Subproceso *Desarrollar tapa*

Señales

- Signal Intermediate Event
  - Permite señalizar la ocurrencia de un evento que podrá ser detectado en cualquier parte del proceso, subprocesos e incluso procesos ancestros
  - Dos tipos

Throw
(para disparar la señal)

Catch
(para atrapar la señal)

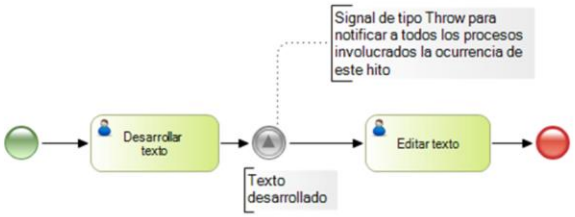
GeneXus

Señales

- Signal Intermediate Event
  - En GeneXus
    - Propiedad *Trigger = Signal*
    - Las señales se identifican por su nombre (propiedad *Name)*
    - Si es un *Throw* hay que configurar la propiedad *Is Throw = true*

Señales

- Intercomunicación de procesos
  - Subproceso *Editar Texto*



Señales

- Intercomunicación de procesos
  - Subproceso *Desarrollar tapa*

Señales

- Comunicación entre procesos
  - Subproceso *Editar Texto*

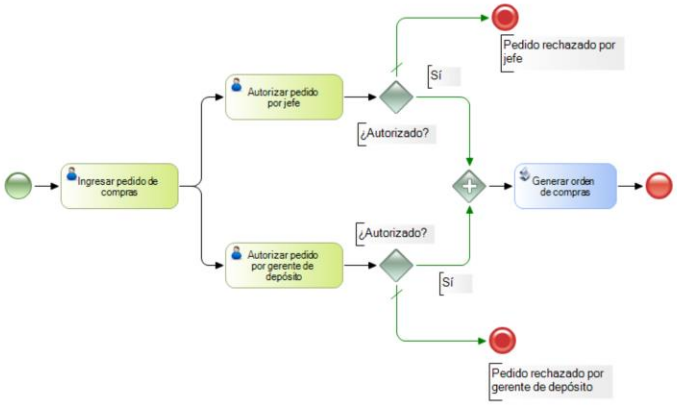
  - Subproceso *Desarrollar tapa*

Señales

- Otro ejemplo
  - Proceso de compras
    1. Un funcionario ingresa un pedido de compras
    2. El pedido es autorizado por el jefe del funcionario y por el gerente de depósito
    3. Si el pedido es autorizado por ambas personas se genera la orden de compras correspondiente
    4. Si es rechazado debe volver al funcionario que lo ingresó para que sea modificado

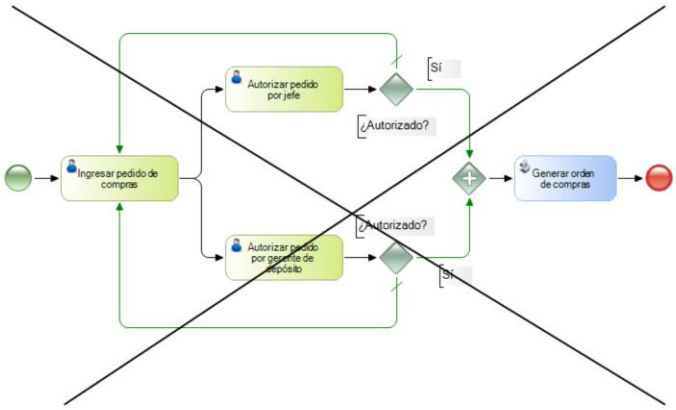
GeneXus

Señales

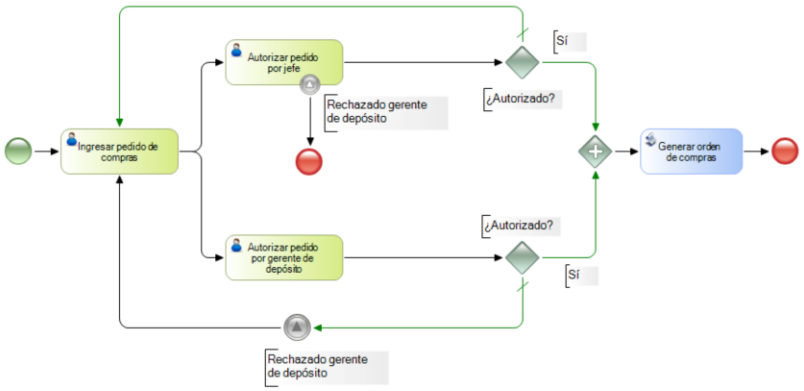


GeneXus

Señales

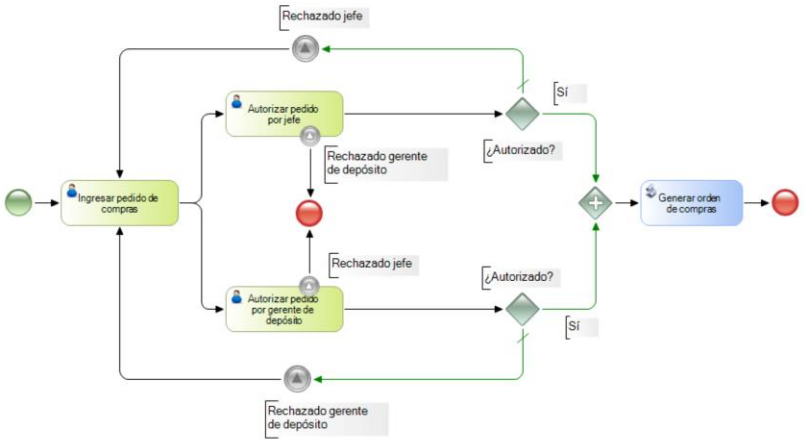


Señales


Autorizar pedido por jefe
Rechazado gerente de depósito
Sí
¿Autorizado?
Ingresar pedido de compras
Generar orden de compras
¿Autorizado?
Autorizar pedido por gerente de depósito
Sí
Rechazado gerente de depósito

GeneXus

Señales



GeneXus

Señales

- Signal End Event
  - Similar a un *None End Event* pero además dispara una señal

GeneXus

Señales

- Signal Start Event
  - Permite iniciar un proceso si ocurre una señal determinada

Ramificación basada en eventos

- Dada una lista de posibles eventos se necesita esperar por la ocurrencia del primero de ellos

Ramificación basada en eventos

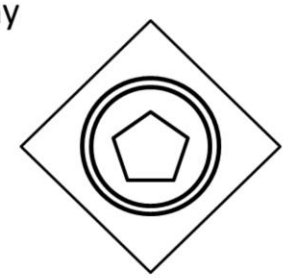
- El siguiente proceso puede quedarse indefinidamente bloqueado esperando la llegada de la señal

Ramificación basada en eventos

- Queremos evitar el bloqueo imponiendo un límite máximo de espera de 1 mes

Ramificación basada en eventos

- Event Gateway



GeneXus

Ramificación basada en eventos