Subprocesos transaccionales

1.- Introducción

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 y en GeneXus los vemos de la siguiente manera:

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:

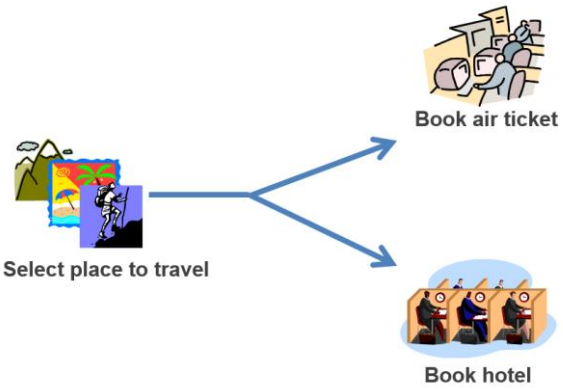
- Que todas las operaciones **finalicen satisfactoriamente**, con lo cual el proceso continúa por el flujo normal
- Que **ocurra una falla** y es necesario revertir las actividades que ya fueron completadas dentro del subproceso. En este caso deben ejecutarse ciertas tareas de compensación que se encargan de dejar al proceso en el estado inicial existente antes de iniciarse el subproceso transaccional.
- 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.

2.- Caso de Uso: Reserva de un viaje

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.



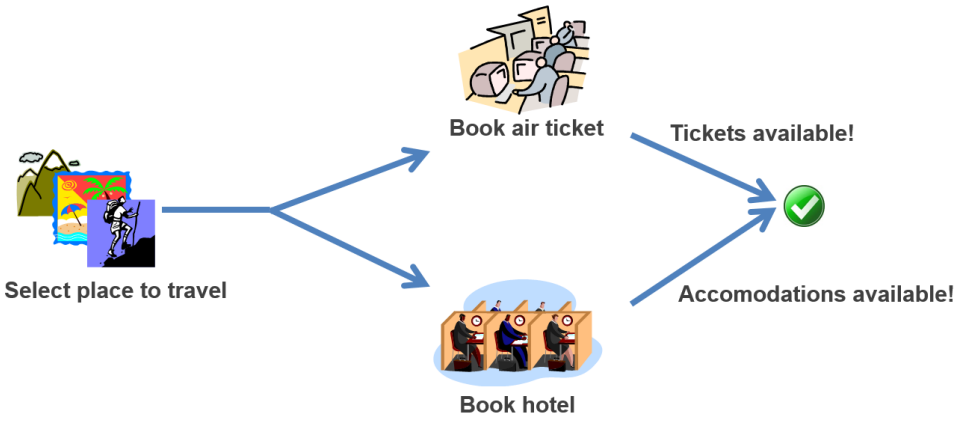
Consideremos un proceso de reserva que consiste de dos actividades: la reserva de un vuelo y la reserva de una habitación de hotel.

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.

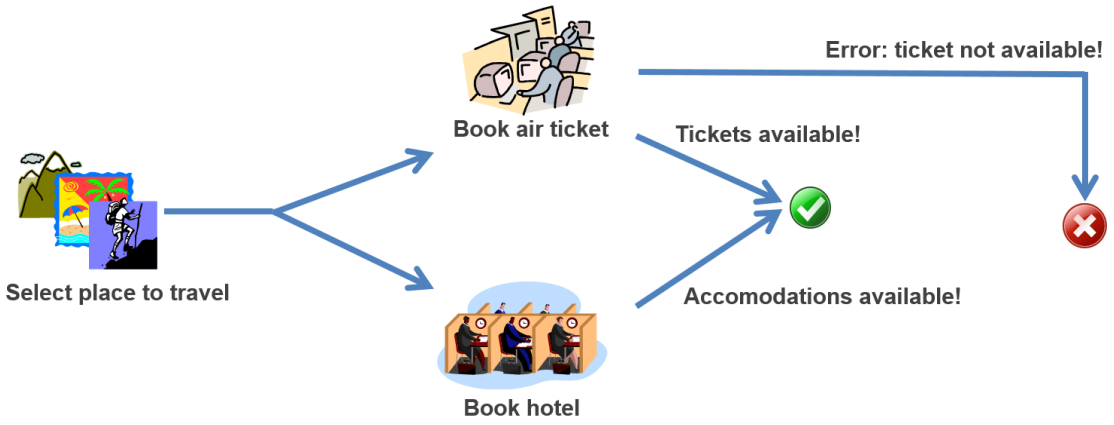
Esto implica que debemos considerar los tres casos antes vistos:

- 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.

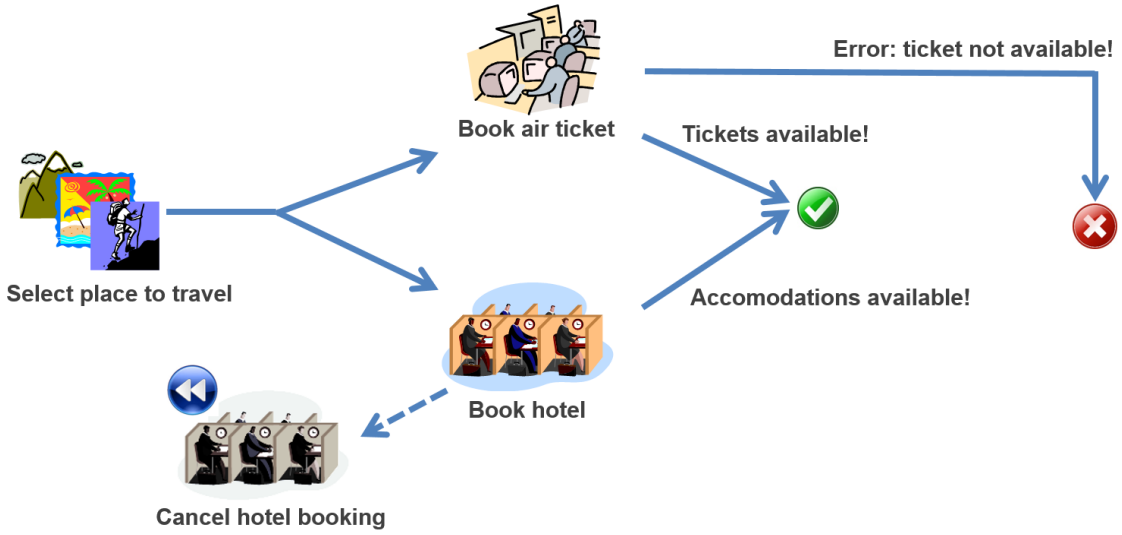
En el caso de que ambas reservas se obtengan satisfactoriamente, el proceso finaliza:



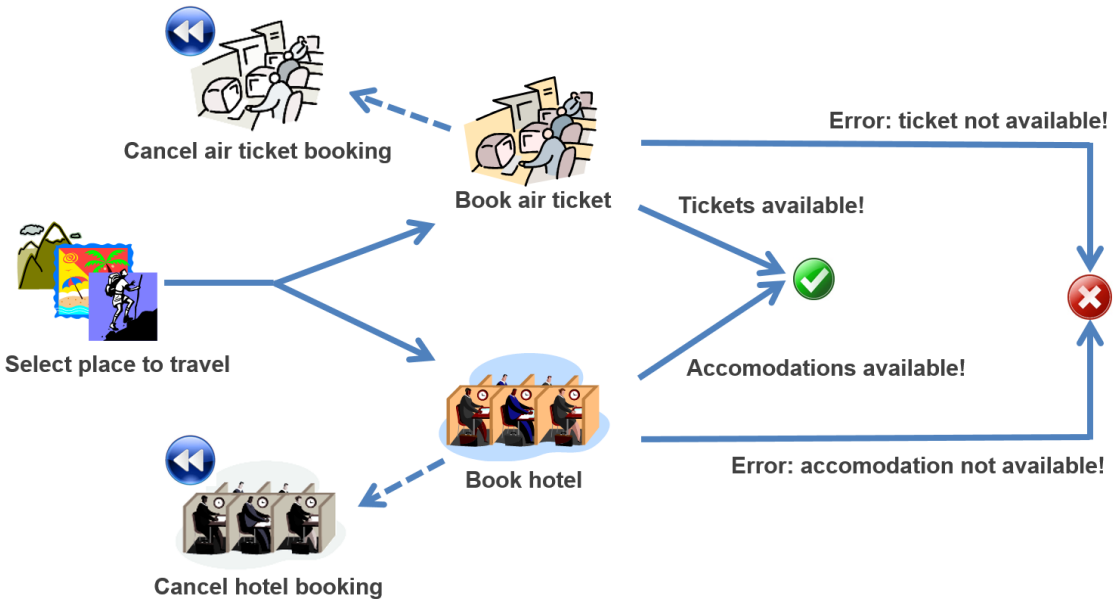
Pero si falla la obtención del pasaje aéreo y se logró finalizar la reserva del hotel, el proceso no puede darse por completado porque una de las actividades de la transacción no se cumplió, por lo que es necesario deshacer las actividades que llegaron a cumplirse.



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.

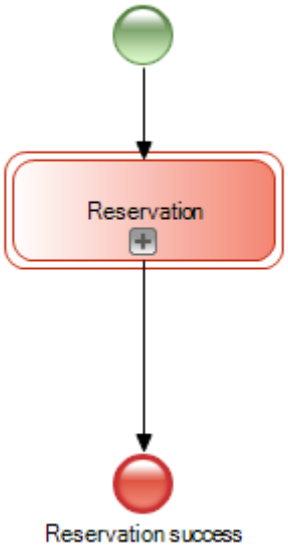
En nuestro ejemplo, cuando el proceso de reserva no finaliza exitosamente, se detecta esa situación y se le comunica al cliente que no es posible realizar la reserva.

2.- Implementación en GeneXus: Reserva de un viaje

Veamos como modelamos este proceso con GeneXus BPM Suite.

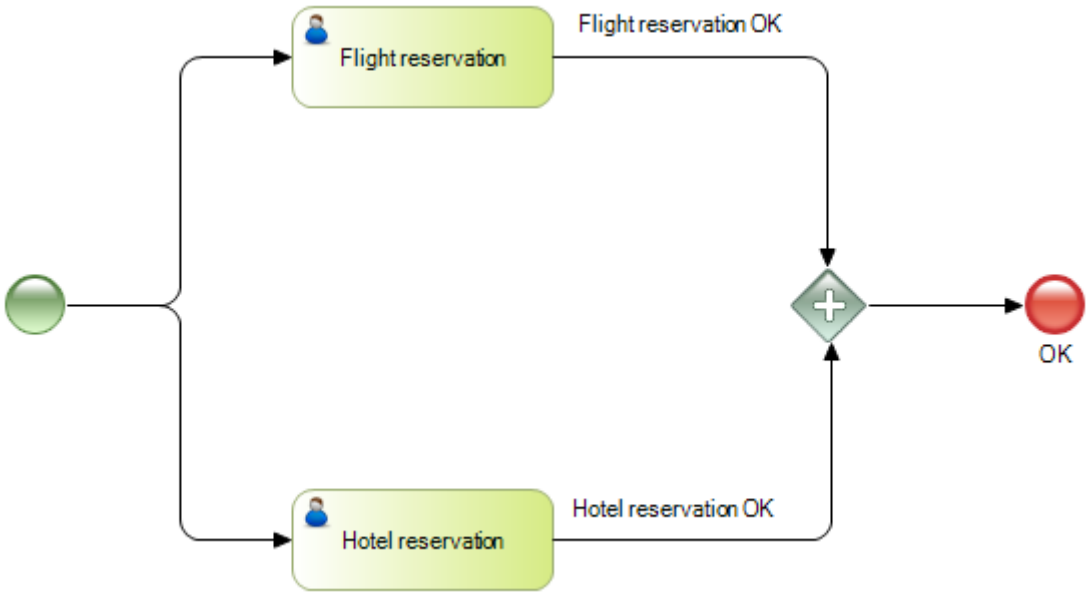
En primer lugar ejecutamos el IDE de GeneXus o el GeneXus Business Process Modeler y en nuestra Knowledge Base, creamos un objeto Business Process Diagram.

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.



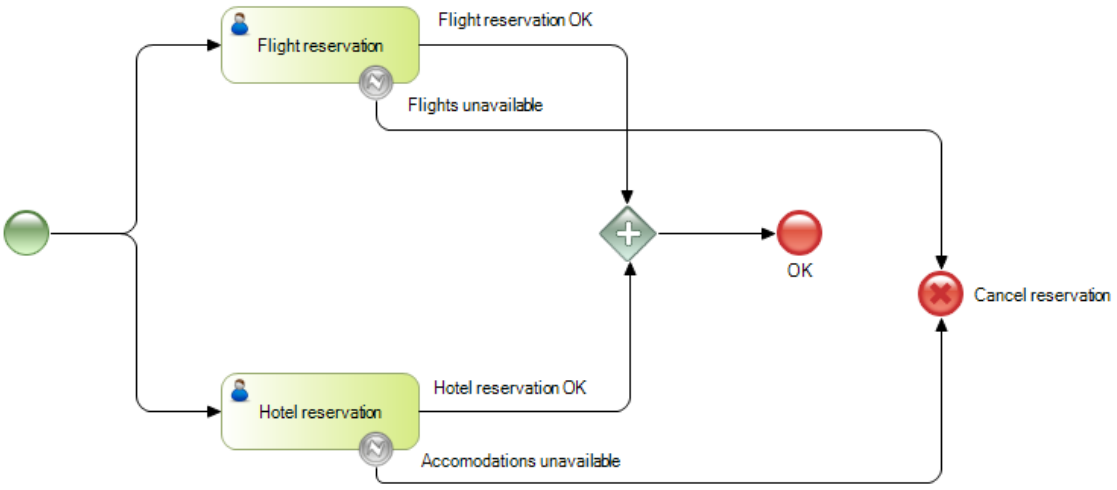
Ahora abrimos el subproceso, comenzamos agregando un None Start Event, luego insertamos 2 tareas interactivas, una para la reserva de vuelos y otra para la reserva del hotel y las

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.



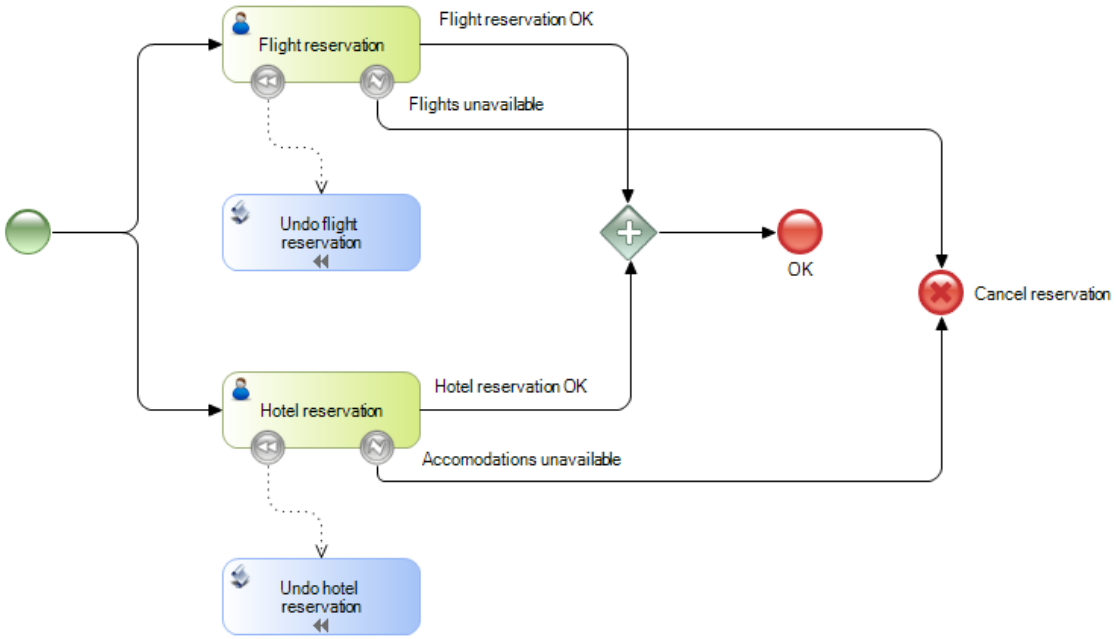
Este evento Cancel se disparará cuando alguna de las dos tareas interactivas falle. El evento Cancel es un evento especial utilizado en los procesos transaccionales, que se dispara **durante** la ejecución del subproceso, en lugar de dispararse al finalizar el mismo.

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.

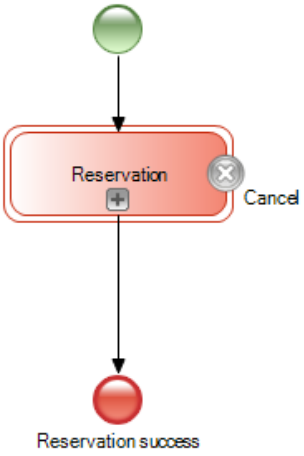
Repetimos lo mismo para la tarea de reserva de hotel, insertando otro Intermediate Cancel Event y una tarea no interactiva con el nombre Undo hotel reservation, que conectamos desde el Intermediate Cancel.



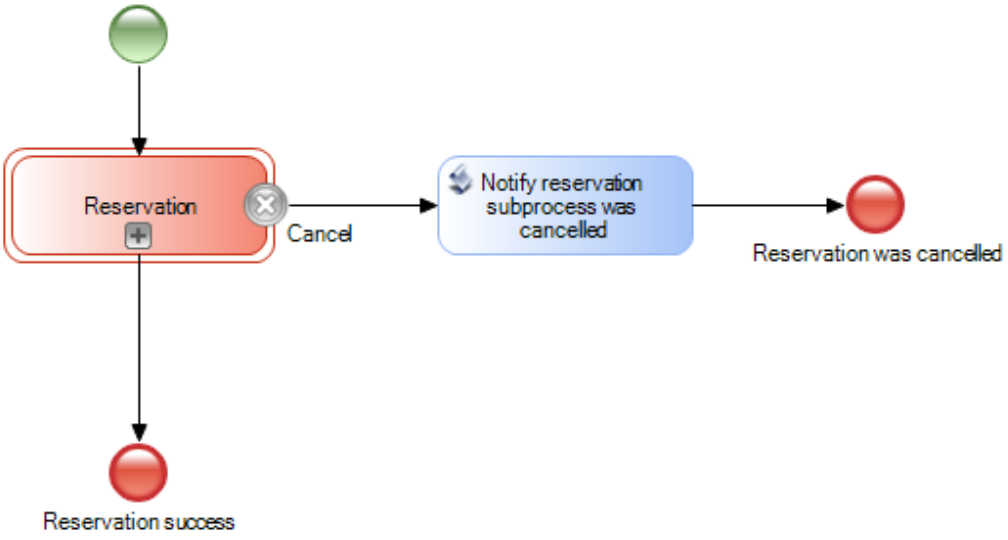
Observamos que desde el momento que conectamos las tareas script con los Intermediate Cancel Event, cambia el aspecto de la conexión ya que ahora el conector es del tipo Association y aparece sobre las tareas un símbolo "<<". Este tipo de conector se representa con una línea punteada y permite establecer 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.

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

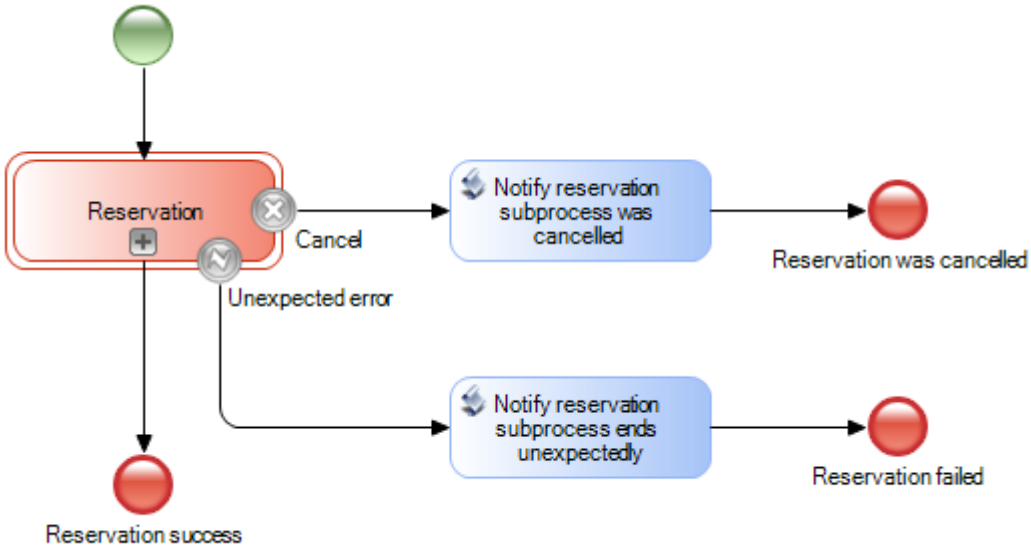


Al igual que vimos anteriormente con el evento Error en el video “Definición de tareas simultáneas, detección e identificación de errores”, el Cancel utiliza un mecanismo de throw-catch desde el Cancel End Event del subproceso, hacia el Cancel Intermediate Event adjunto al subproceso, en el diagrama padre.

Como vimos, 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.

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.



Puede encontrar más información, en el siguiente link:

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