Anti-patterns

Overview

Anti-patters are modeling patterns wrongly used to model some business conditions. In general, Anti-patterns seem to work fine, but their use is not directly supported or is not BPMN compliant and could make Processes not work as expected. For these reasons, we do not recommend to use Anti-patters.

Below you find some of the most frequently used Anti-patterns:

Avoid following an Event-based gateway by elements other than Events and Tasks

Ex: Cancellation pattern with Parallel gateways

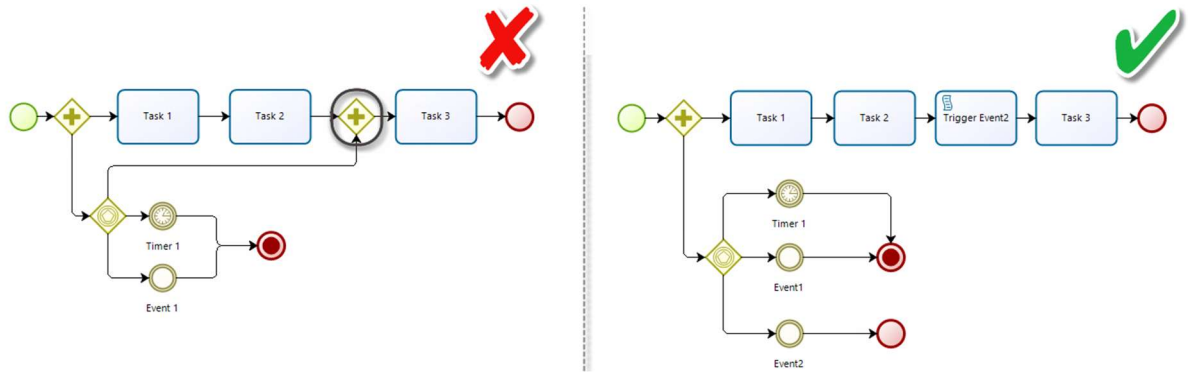
The Event-based gateway provides a useful modeling pattern, which should be used appropriately. Given that this type of gateway will enable and control different possibilities in the workflow. Make sure that this gateway is always followed by Tasks or Events. Other BPMN elements (such as Sub-Processes or other gateways) right after this gateway are not supported.

It is common to see processes automated using a cancellation pattern that involves an Event based Gateway followed by a Parallel gateway.

The whole process is canceled if an event is executed or a time is reached before a set of activities is completed, otherwise, the cancellation logic is disabled.

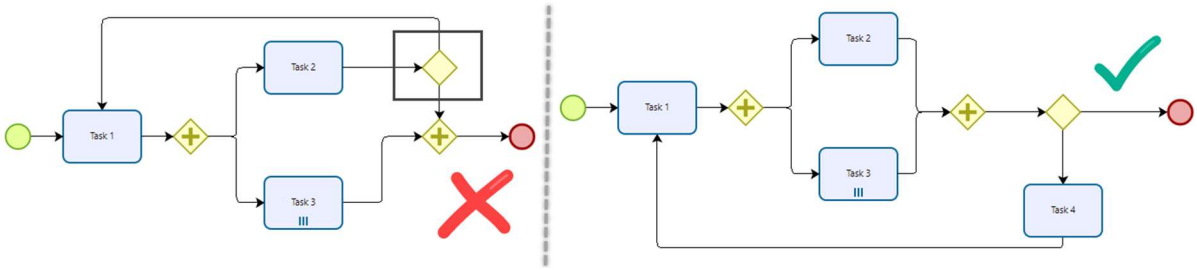
On the left the most common way to model this cancellation; this kind of pattern it is not BPMN compliant and, therefore, is not supported by Bizagi.

On the right an alternative way; An additional event is used to discard the cancellation logic once the necessary activities are accomplished.



When using a divergent gateway do not bifurcate the flow before converging

When your process flow uses a divergent gateway it is strongly recommended to use a convergent gateway of the same type, to collect all created tokens. You should not bifurcate the flow before converging, as Bizagi will not be able to control the paths created.



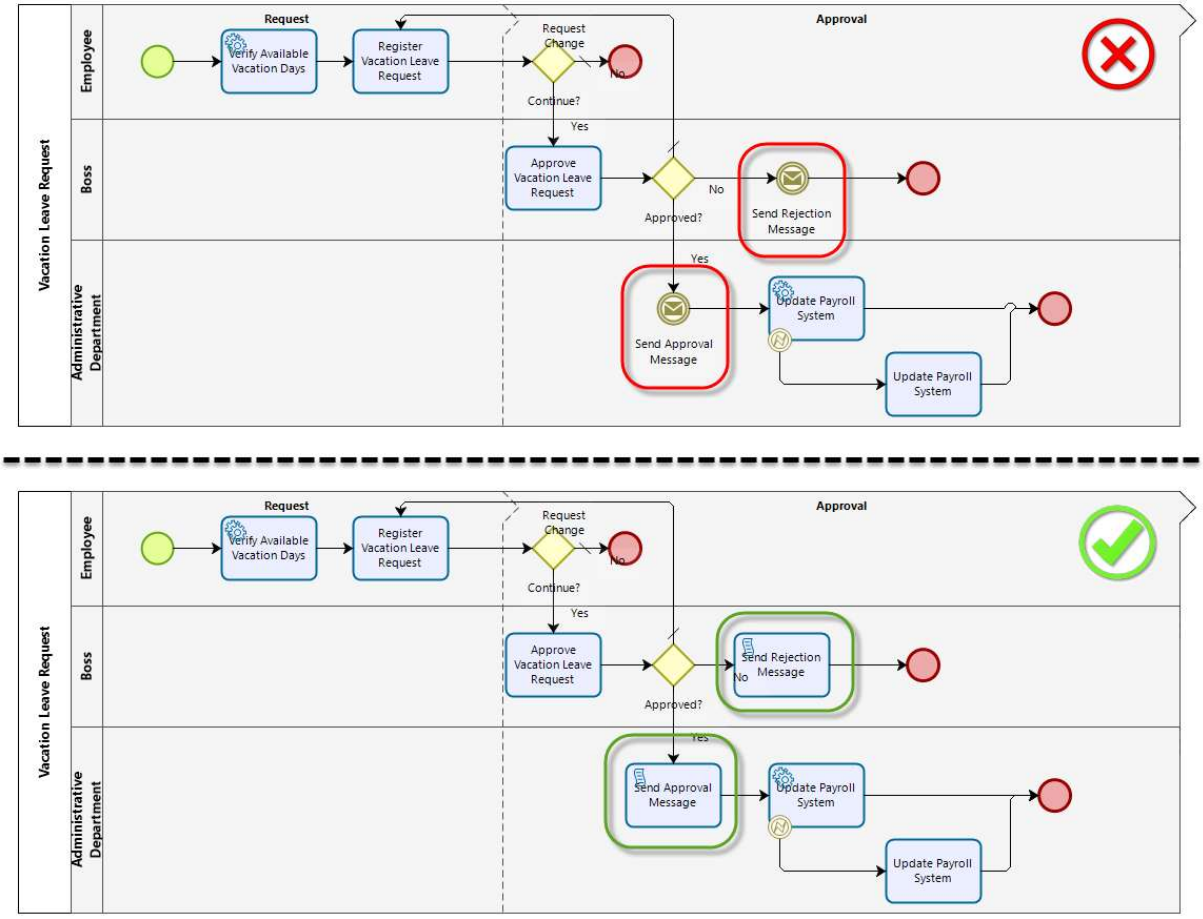
Never use signals or messages in embedded Sub-Processes

Embedded sub processes (a.k.a. Modules) should not contain signals or messages. These may cause the Sub-Process to be opened whilst waiting for something to activate them, and not allow the parent process to finish correctly.

Never use Message events to send electronic mails

Messages events should not be used to send electronic mails during the process flow. These elements must be used to let a process to communicate with another process.

Electronic mails must be send using Activity actions.



Don't use signals nor messages to activate paths inside the process

Don't use signals nor messages events to activate paths inside the process

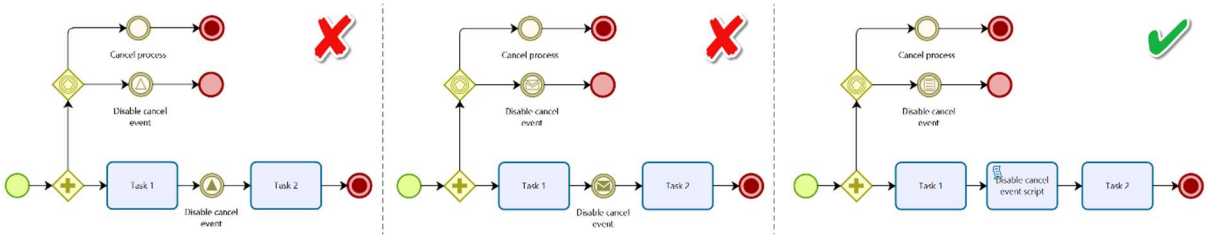
Signals are listened by all active processes and it is caught by all the process at the same time. If a signal is used to activate a path in the process this path activates all the active processes, not just inside the process that launched the signal.

In this kind of situation a Conditional event should be used.

A common example, for this kind of situations, is when you want to disable the cancel process event.

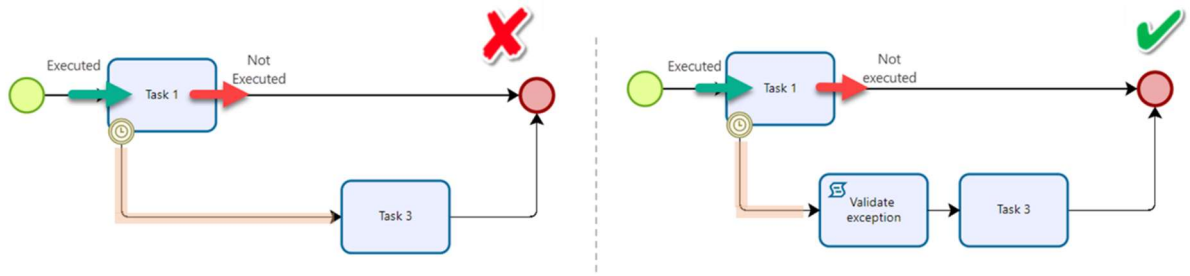
On the left and middle the most common ways to model this cancellation; these kind of patterns are not BPMN compliant and, therefore, are not supported by Bizagi.

On the right an alternative way; A conditional event has replaced the signal/message event with a script task which activates the event.



Using attached timer events

When you use attached timer events, and the exception path is triggered (when the timer reaches its duration), the case follows the next element in the exception path. However, the activity actions On Exit (red arrow), or forms validations included in the task (like mandatory fields) are not executed. Therefore, as a best practice, it is advisable to include a validation script task, that either reset or delete information that you do not want to persist when the path goes through the exception path (the timer path). For example a value that you need to either validate or keep empty if the case goes to the task 3.



Considerations about synchronous BPMN elements executed automatically after a Timer Event

Timer events are executed by the Scheduler. When a timer is executed in a process, a new scheduler job is created. You can see all the jobs in the Management Console.

If after a timer event you model BPMN elements that are executed automatically in a synchronous way (no user intervention), when the timer job reaches its execution time (duration of the timer) and executes the timer, Bizagi also executes the timer and all its subsequent synchronous tasks in the same transaction of the timer.

If any of the automatic BPMN elements fails, Bizagi rolls back to the element that started the transaction, the timer. This means that the timer is not executed again and the case gets stuck at the timer event.

Examples of elements that are executed automatically in a synchronous way (no user intervention) are:

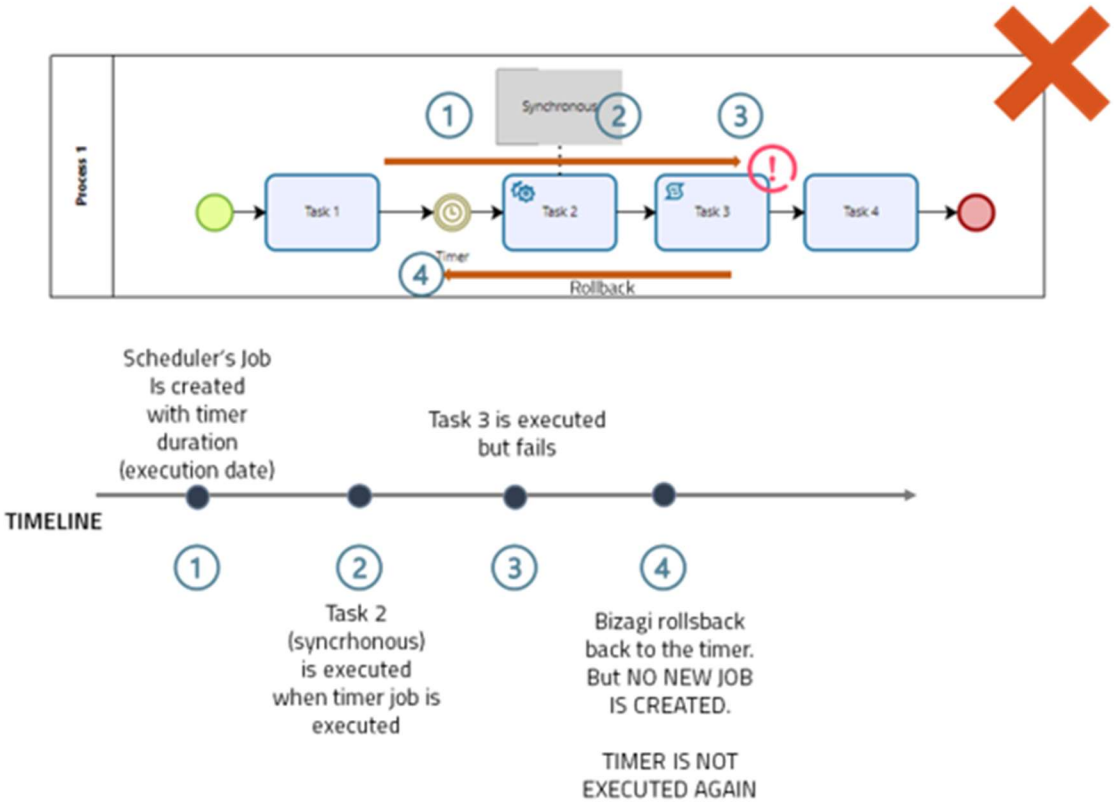
•All the automatic tasks executed in a synchronous way. For example:

-Service tasks as synchronous activities.

-Script tasks.

•Expressions on Exit of the timer event.

Let’s explain this behavior with the following example.



After Task 1 is executed, the following steps are executed:

1. A scheduler's job is created with the duration set in the timer. See how to set timer duration.

2. When the job reaches the timer duration, a transaction is executed with all the following elements:

•Expressions on Exit of the timer event.

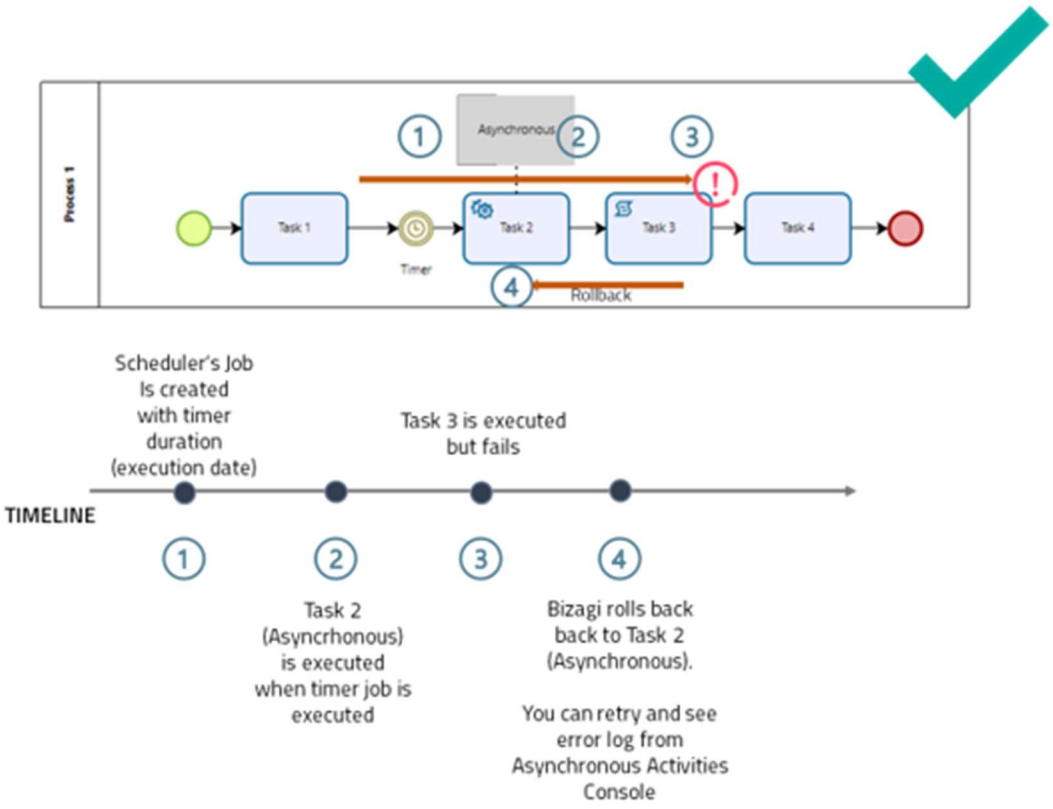
•All the automatic tasks executed in a synchronous way.

Therefore, in the example, Task 2 is executed (this task is set as synchronous).

3. In the same transaction Bizagi executes the Script task, Task 3. Now, let's assume that Task 3 fails at this point.

4. Bizagi rolls back to the timer element. However, Bizagi DOES NOT create a new Scheduler job. **This means that the timer is not executed again and the case gets stuck at the timer event.**

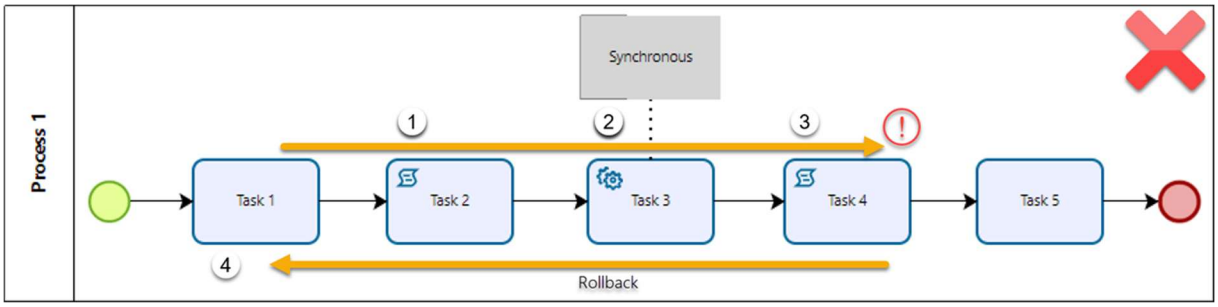
To avoid this behavior, you must set the service task (Task 2) as an asynchronous activity, see Using Asynchronous Activities, and move any expression on exit of the timer to an asynchronous task too. By doing this, if Task 2 or Task 3 fails, after executing the timer, Bizagi rolls back the transaction to Task 2, so you can configure retries, or retry manually from the Asynchronous Activities Console.



Using service tasks for email notifications

Usually email notifications are configured using a Script task. This type of task is executed automatically in a synchronous way, and all synchronous tasks in sequence are executed in the same transaction. If one of the activities fails, Bizagi rolls back to the element that started the transaction. If a Script task is included in this sequence, the email will be sent again, and could end on repeated communications.

Consider the following process to understand this behavior.



In this process, Task 4 is an Script task and is used for email notification, and Task 3 is a Service task configured as a synchronous activity. Hence, the transaction after Task 1 follows these steps:

1. Task 2 (Script task) is executed.

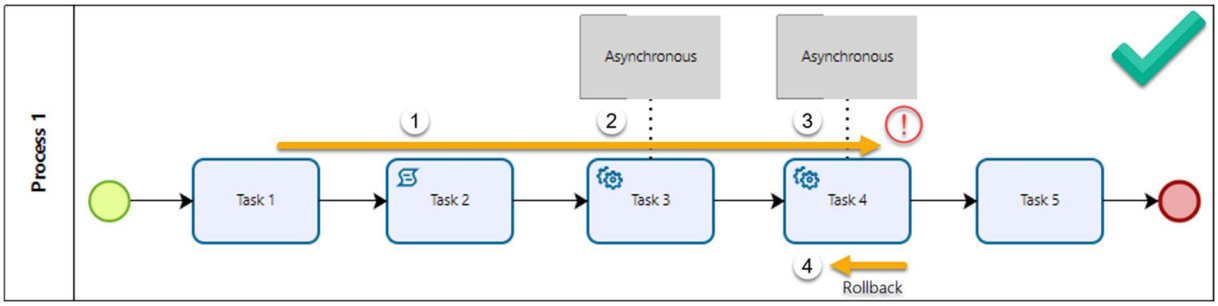
2. Task 3 (Service task) is executed. As it is configured as a synchronous activity, it is executed automatically in the same transaction.

3. Bizagi continues with the execution of Task 4 (Script task). Remember that this task is used for email notification and Script tasks are executed automatically in a synchronous way. At this point, assume that this task fails.

4. As none of the previous tasks were configured as asynchronous activities, Bizagi rolls back to Task 1. Considering that Task 4 was the one that failed, this means that Task 2 and Task 3 are executed all over again.

To avoid this behavior, bear in mind that for the execution of automatic tasks in sequence, the best practice is to convert as many automatic tasks as possible to asynchronous activities. Then, in this example it is highly advisable to set Task 3 (Service task) as an asynchronous activity. In this way, Bizagi rolls back to this task if it (or any subsequent task) fails, and you will be able to configure retries and observe the error log in the Work Portal's Asynchronous Activities Console.

Additionally, remember that it is possible to configure email notifications from a Service task. Therefore, as a best practice for the execution of automatic tasks in sequence, it is recommended to change Script tasks used for email notification (i.e. Task 4) to Service tasks set as asynchronous activities and configured to serve the same purpose. In such manner, you can halt Bizagi's roll back to already executed tasks if an email notification task fails, and resume the execution of the transaction from it.



Last Updated 7/5/2023 10:56:47 AM