White Paper

Workflow Patterns and BPMN

WP0121 | December 2013

Gregor Polančič

Gregor is an assistant professor at the University of Maribor and has a decade of experience in BPMN since its first version in 2004.

He participated in the development of one of the first BPMN modeling utilities - a package of plugins for Visio, which were introduced in 2004 and is the main author of the first BPMN poster.

In 2008, he was one of the first authors who published an article dedicated to the experiences and practical use of BPMN. The article was published in "BPM and Workflow Handbook" in association with the Workflow Management Coalition (WfMC).

He is currently researching BPMN from different technological and user aspects.

In general, a pattern describes a solution for a recurring problem. Patterns are commonly used in architecture as a formal way of documenting a solution to a design problem in a particular field of expertise. The idea of a pattern was introduced by the architect Christopher Alexander and has been adapted for various other disciplines, including software engineering.

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Patterns can be viewed as formalized best practices that a programmer must implement in the application. Most common are object-oriented software design patterns, which typically show relationships and interactions between classes or their instances, without specifying the final application classes or objects that are involved. An example on an object-oriented software design pattern is 'Singleton', which defines a way of implementing an object-oriented system that restricts the instantiation of a class to precisely one object.

Workflow

A workflow is an automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

What are workflow patterns?

A workflow pattern is a specialized form of a design pattern as defined in the area of software engineering or business process engineering. Workflow patterns refer specifically to recurrent problems and proven solutions related to the development of workflow applications. Workflow applications are tightly related to business processes. A workflow application or simplified a workflow is usually a centralized and dedicated IT solution, which automates specific business activities. In contrast to a workflow, a business process represents a holistic set of automated and manual activities dedicated to fulfill a business objective, where business process solutions tend to be more decentralized and generic as workflow ones. This means that workflow patterns can refer also to process-oriented applications.

Business Process

A business process is a set of one or more linked procedures or activities, which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

Workflow patterns follow the concept of efficient development. Their usage shall follow strategies of simplifying maintenance as well as reducing modeling activities. Most common are control-flow patterns, which capture aspects related to control-flow dependencies between various tasks (e.g. parallelism, choice and synchronization). However, the workflow patterns are not limited to control-flow. Other workflow pattern collections include data, resource, and exception handling. The data perspective deals with the passing of information and scoping of variables. The resource perspective deals with resource to task allocation and delegation of tasks. The last, exception-handling perspective, deals with various causes of exceptions and various actions that need to be taken because of exceptions occurring.

The remaining part of the article will focus on control-flow patterns. It will represent the main categories of control-flow patterns and their representatives in a BPMN view.

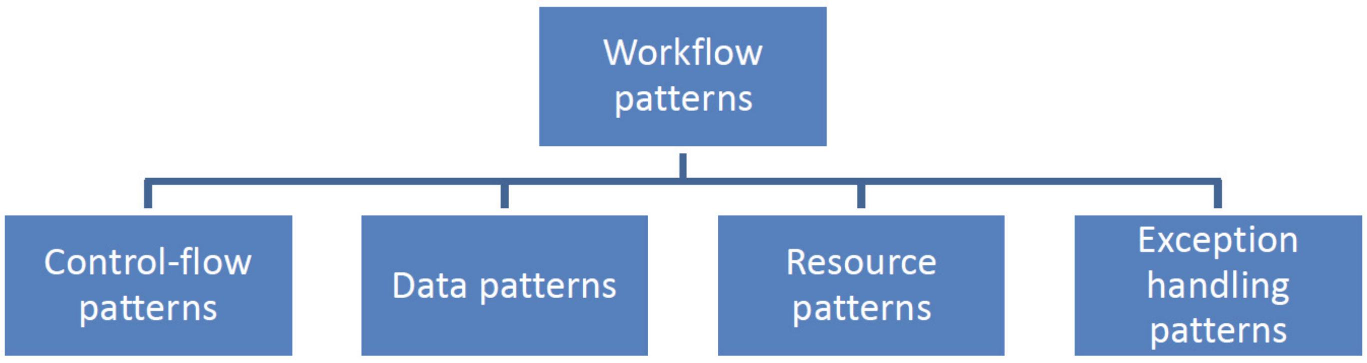

Figure 1: Man categories of workflow patterns

Control flow patterns

Control-flow patterns are the most popular workflow patterns. This collection of patterns focuses on one specific aspect of process-oriented application development, namely the description of control flow dependencies between activities in a workflow or process. Initially, Workflow Patterns Initiative (http://www.workflowpatterns.com) defined 20 control-flow patterns [1]. The basic set of 20 control-flow patterns is divided into six categories:

- basic control flow patterns,
- advanced branching and synchronization patterns,
- structural patterns,
- patterns involving multiple instances,
- state-based patterns, and
- cancellation patterns.

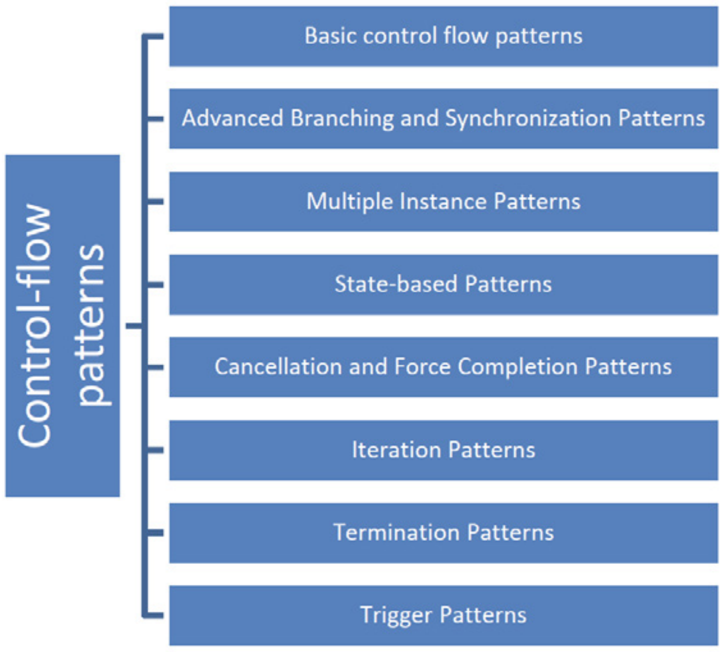
Control-flow patterns
Basic control flow patterns
Advanced Branching and Synchronization Patterns
Multiple Instance Patterns
State-based Patterns
Cancellation and Force Completion Patterns
Iteration Patterns
Termination Patterns
Trigger Patterns

Figure 2: Control-flow patterns categories

Since their release in 2003, these patterns have been widely used by practitioners, vendors and researchers. Review of the patterns associated with the control-flow perspective over the past few years has led to the recognition that there are a number of distinct modelling constructs that can be identified during process modelling that are not adequately captured by the original set of twenty patterns. So the basic set of control-flow patterns was extended with an additional twenty three control-flow patterns [2], classified into eight categories (Figure 2).

The next subsections will represent control-flow patterns categories and a representative of each category in a BPMN diagram [3].

Basic Control Flow Patterns

The Basic Control Flow Patterns category captures elementary aspects of process control and consists of five patterns: sequence, parallel split, synchronization exclusive choice and simple merge. The following figure represents BPMN view of the Sequence pattern, where a task in a process is enabled after the completion of a preceding task in the same process.

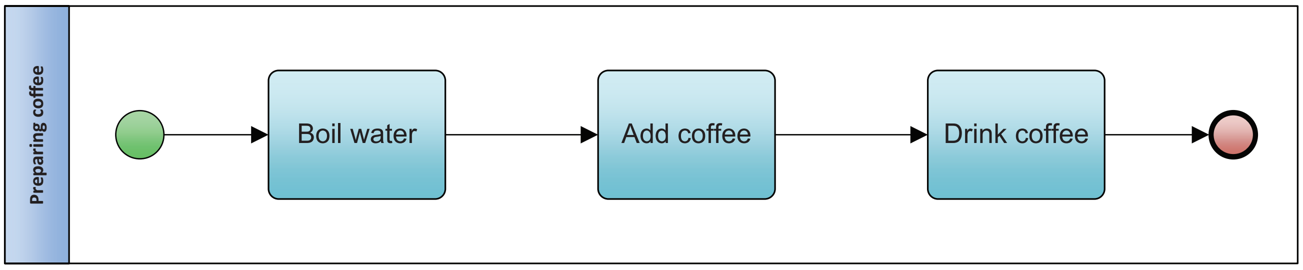

Figure 3: BPMN view of sequence pattern

Sequence pattern is used to model dependencies between tasks so that one task cannot start before another is finished (i.e. serial execution). To model this pattern in BPMN, it is necessary to connect the activities sequentially by using sequence flows.

Advanced branching and synchronization patterns

The Advanced branching and synchronization patterns category consists of patterns which characterize more complex branching and merging concepts of business processes. The last revision of this category consists of 14 patterns:

- multi-choice,
- structured synchronizing merge,
- multi-merge,
- structured discriminator,
- blocking discriminator,
- cancelling discriminator,
- structured partial join,
- blocking partial join,
- cancelling partial join,
- generalized and-join,
- local synchronizing merge,
- general synchronizing merge,
- thread merge and thread split.

Figure 4 below represents BPMN view of the multi-choice pattern.

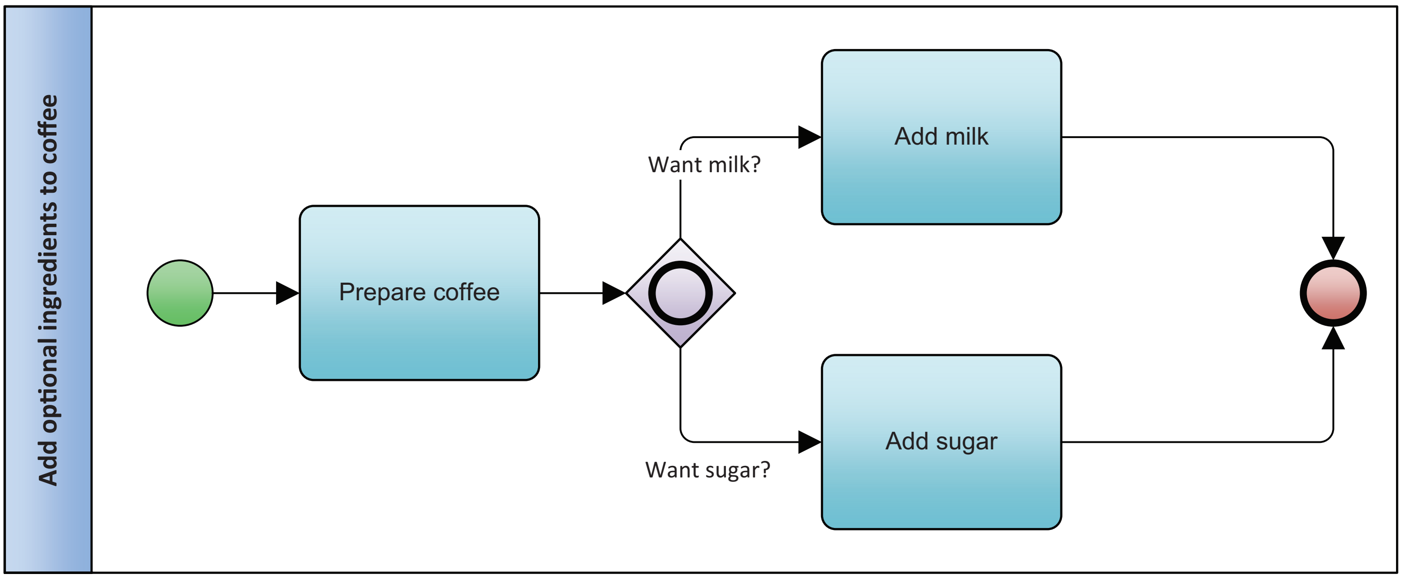

Figure 4: BPMN view of multi-choice pattern

Multi-choice pattern presents the divergence of a branch into two or more branches such that when the incoming branch is enabled, the thread of control is immediately passed to one or more of the outgoing branches based on a mechanism that selects one or more outgoing branches. To implement the multi-choice pattern in BPMN it is necessary to use an inclusive (OR) gateway, since it allows enabling one or more paths according to process data.

Multiple instance patterns

Multiple instance patterns describe process behavior with multiple threads of execution active in a process model, which relate to the same activity. Multiple instances can arise in three cases:

1. An activity is able to initiate multiple instances of itself (e.g. delivering a coffee multiple times),
2. A given activity is initiated multiple times as a consequence of receiving several independent triggering (e.g. preparing an invoice is triggered after delivering a drink and a coffee), and
3. Two or more activities in a process share the same implementation definition. This may be the same activity definition in the case of a multiple instance activity or a common sub-process definition in the case of a block activity.

Multiple instance patterns are primary dedicated to the first case (multiple instances) since they require the triggering and synchronization of multiple concurrent activity instances. This group of control-flow patterns consist of seven patterns:

- multiple instances without synchronization,
- multiple instances with a priori design-time knowledge,
- multiple instances with a priori run-time knowledge,
- multiple instances without a priori run-time knowledge,
- static partial join for multiple instances,
- cancelling partial join for multiple instances and
- dynamic partial join for multiple instances.

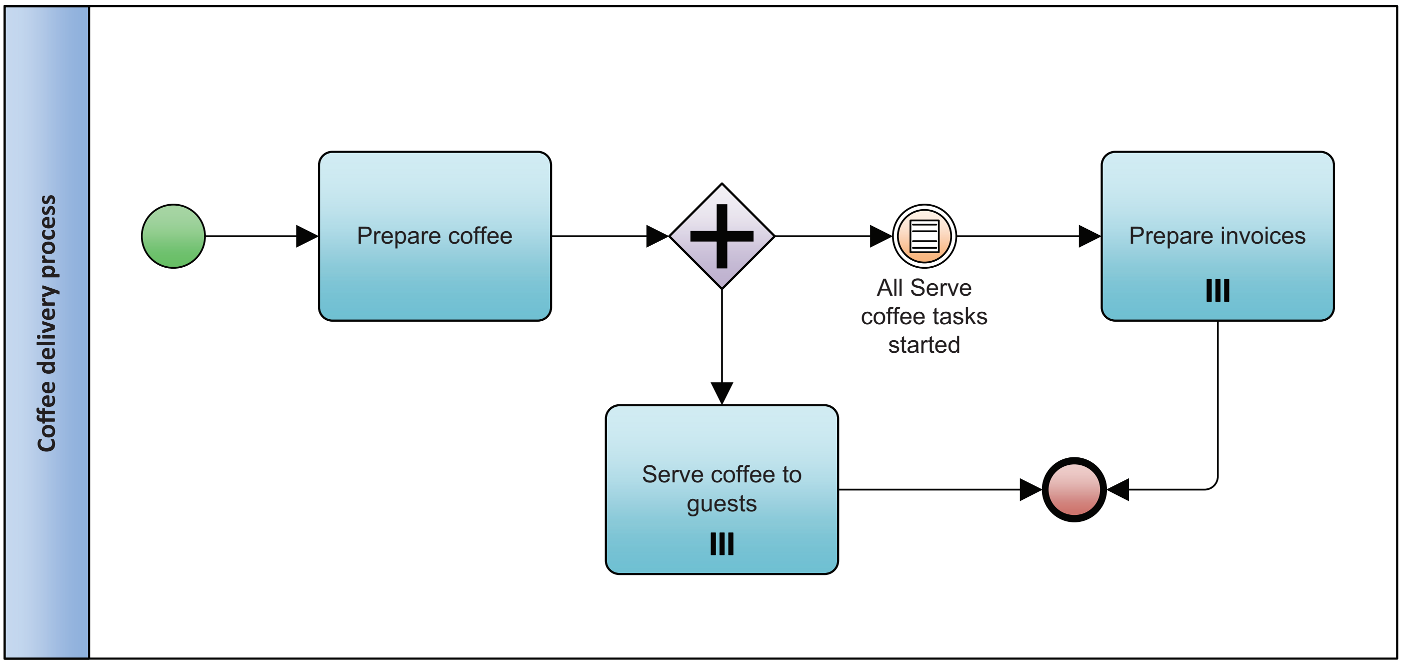

Figure 5: BPMN view of multi-instance without synchronization pattern

Figure 5 on the previous page represents a BPMN view of the 'multiple instances without synchronization' pattern, which is used to model activities that have to be instantiated many times within a process and do not need to be synchronized for the flow to continue.

It demonstrates that within a given process instance, multiple instances of Task B can be created. These instances are independent of each other and run concurrently and there is no requirement to synchronize them before completion. Each of the instances of Task B must execute within the context of the process instance from which they were started (i.e. they must share the same case identifier and have access to the same data elements) and each of them must execute independently from and without reference to the task that started them.

State-based patterns

State-based patterns reflect situations for which solutions are seamlessly accomplished in process environments that support the concept of a state. In this context, we consider the state of a process instance to include the broad collection of data associated with current execution including the status of various activities as well as process-relevant working data such as activity and case data elements. State-based patterns category consists of following five patterns:

- milestone,
- deferred choice,
- interleaved parallel routing,
- critical section and
- interleaved routing.

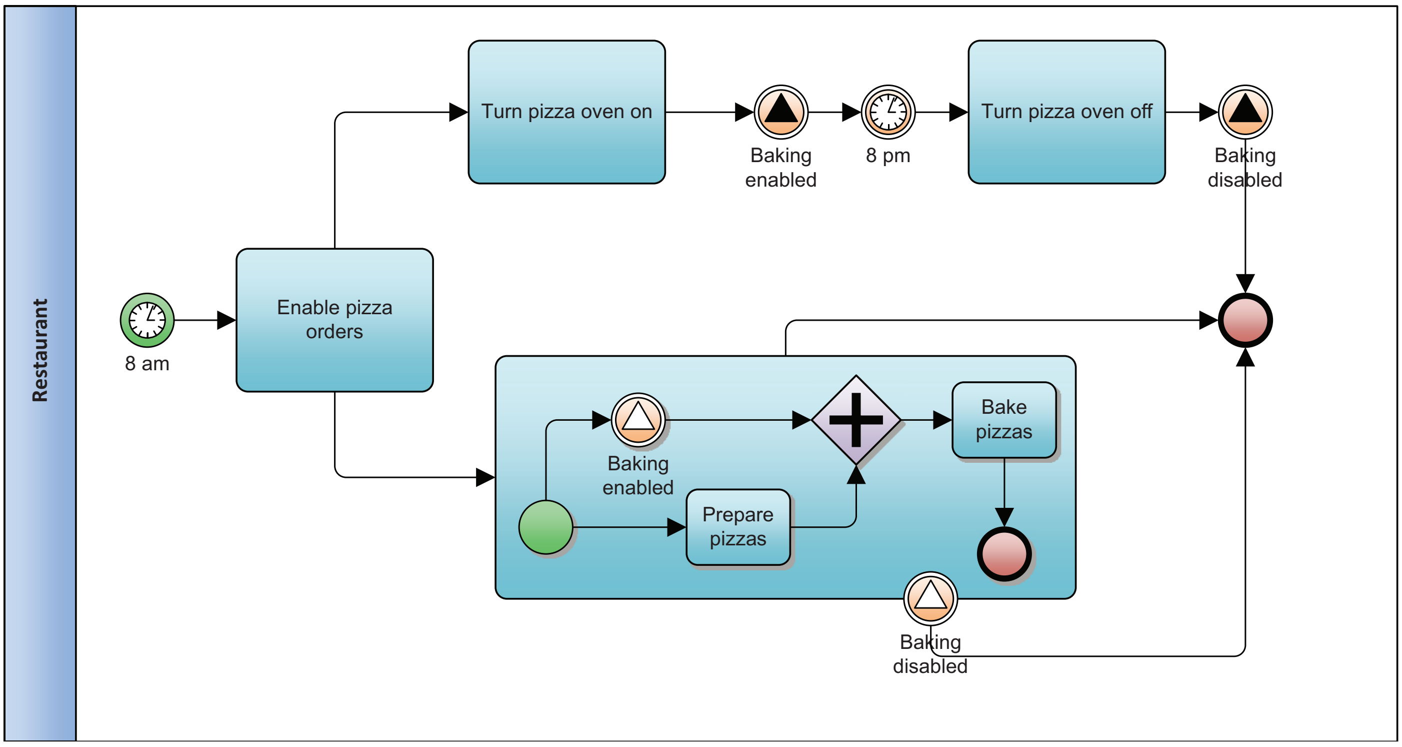

Figure 6: BPMN view of milestone pattern

Figure 6 represents a BPMN view of the ‘milestone’ pattern, which defines that an activity is enabled only when the process instance is in specific state (milestone). If the process instance has progressed beyond this state, then the task can no longer be enabled.

As evident from *Figure 6*, task 5 is only enabled when the process instance is in a specific state (typically implemented with a parallel branch). The state is assumed to be a specific execution point (also known as a milestone) in the process model. When this execution point is reached, task 5 can be enabled. If the process instance has progressed beyond this state, then task 5 cannot be enabled now or at any future time (e.g. the deadline has expired).

Cancellation and Force Completion Patterns

Cancellation and Force Completion Patterns utilize the concept of activity cancellation where enabled or active activity instance are withdrawn. This category of control-flow patterns consists of following five patterns: cancel task, cancel case, cancel region, cancel multiple instance activity and complete multiple instance activity. The next example presents a BPMN view of a cancel activity pattern.

Figure 7 shows that Task 4 is withdrawn prior to it commencing execution. If the task has started, it is disabled and, where possible, the currently running instance is halted and removed.

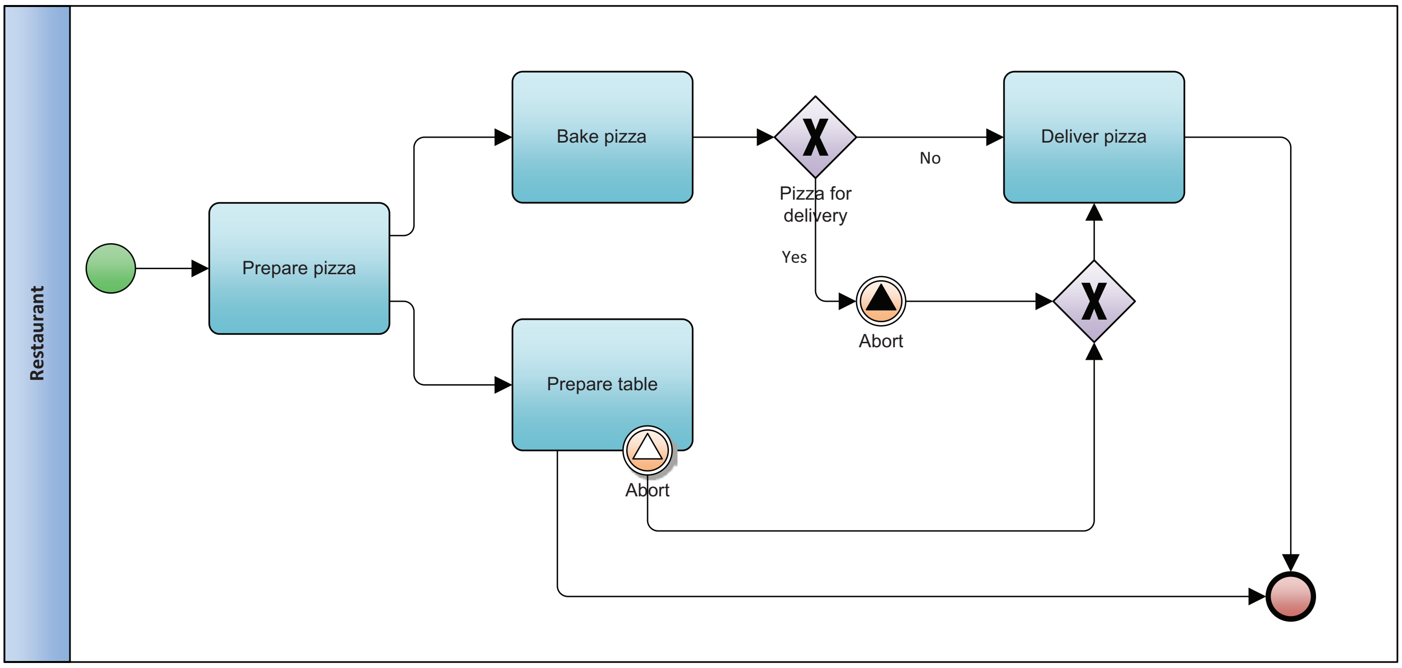

Figure 7: BPMN view of ‘cancel activity’ pattern

Iteration Patterns

The group of iteration patterns deals with capturing repetitive behavior in a workflow and consists of following three patterns: structured loop, arbitrary cycles and recursion. The following figure represents BPMN view of the 'structured loop' pattern, which describes the ability to execute an activity or sub-process repeatedly. The loop has either a pre-test or post-test condition associated with it.

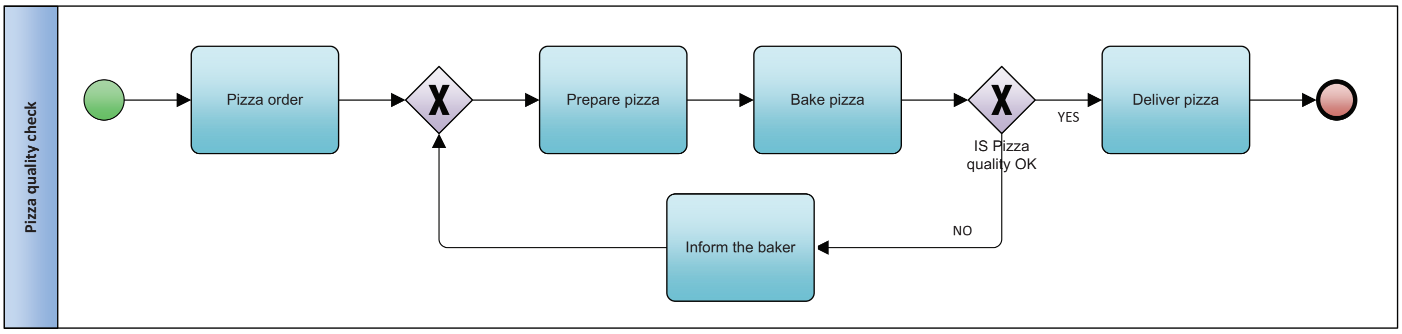

Figure 8: BPMN view of 'structured loop' pattern

The condition is either evaluated at the beginning or end of the loop to determine whether it should continue. The looping structure has a single entry and exit point.

Termination Patterns

Two patterns deal with the circumstances under which a workflow is considered to be completed. These are implicit termination and explicit termination. The next diagram presents 'implicit termination' pattern, which is used to determine when a process instance is considered as complete. A given process instance should terminate when there are no remaining work items that are able to be done either now or at any time in the future and the process instance is not in deadlock.

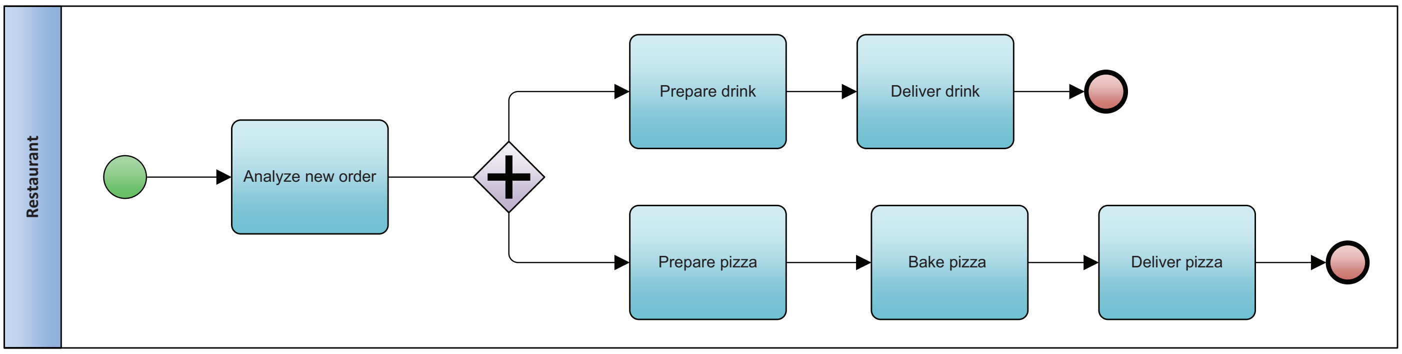

Figure 9: BPMN view of 'structured loop' pattern

The above implementation of implicit termination pattern is represented by adding end events at the end of the paths that must be completed. The process will be considered as finished when each end event is reached.

Trigger Patterns

Trigger patterns deal with the external signals that may be required to start certain tasks. Currently, there are two trigger patterns: transient trigger and persistent trigger. The following BPMN diagram presents 'persistent trigger' pattern, which allows an activity to be triggered by a signal from another part of the process or from the external environment. These triggers are persistent in form and are retained by the workflow until they can be acted upon by the receiving activity.

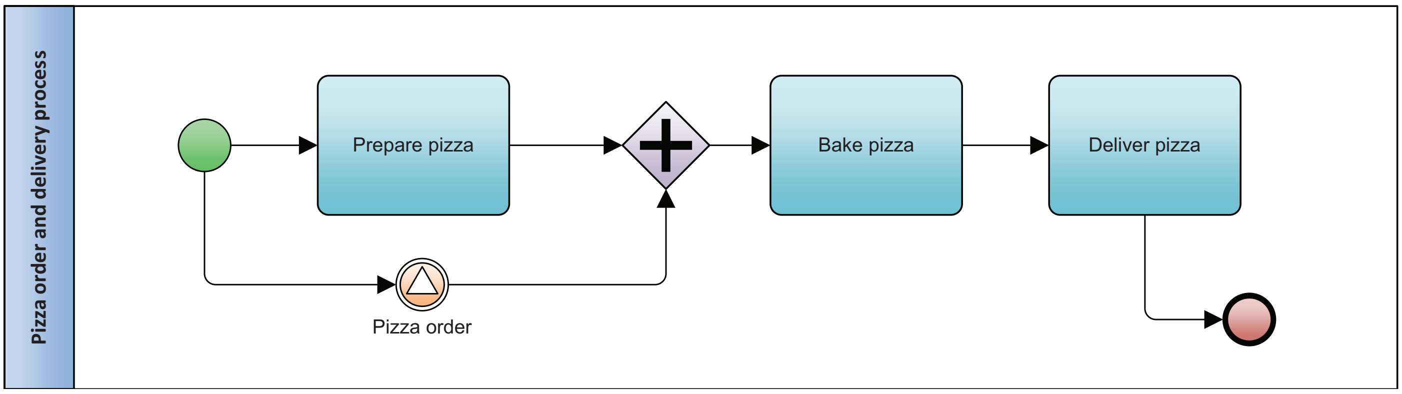

Figure 10: BPMN view of 'Persistent trigger' pattern

As evident from *Figure 10* above, the external signal will enable the process to proceed to Task 3.

Conclusion

This article presented the concept of workflow patterns, which are a specialized form of design patterns, dedicated to solve the recurrent problems in the development of workflow applications and process engines. Workflow patterns are commonly classified into four categories:

- control-flow,
- data,
- resource, and
- exception handling.

Most common are control-flow patterns, which capture aspects related to control-flow dependencies between various tasks. In this article eight different control flow patterns were presented, each belonging to different control-flow pattern sub-category. The patterns were represented as BPMN diagrams, since BPMN is a rich notation, which is capable of representing the majority of workflow patterns.

Resources

[1] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow Patterns,” Distrib. Parallel Databases, vol. 14, no. 1, pp. 5–51, Jul. 2003.

[2] N. Russell, A. H. M. T. Hofstede, and N. Mulyar, “Workflow ControlFlow Patterns: A Revised View,” 2006.

[3] “BPMN 2 workflow patterns.” [Online]. Available: http://www.ariscommunity.com/users/sstein/2010-07-20-bpmn-2-workflow-patterns. [Accessed: 03-Dec-2013].

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com