BPM (Business Process Management) is a set of related activities, such as process modeling and design, process execution, process monitoring, and process optimization. This Refcard provides an overview of the BPM lifecycle together with the roles and results of business process modeling. It gives an overview of the BPMN (Business Process Modeling Notation) and presents the most important BPM patterns.

**BPM: Business Process Lifecycle**

A business process lifecycle covers the following phases (Figure 1):

- **Process modeling** - definition of the process models using the selected methodology and notation (such as BPMN).
- **Process implementation** - implementation of end-to-end IT support for the process. SOA provides technologies and tools to make the implementation phase quick and efficient.
- **Process execution and monitoring** - execution of the process and monitoring of the process to gather the Key Performance Indicators (KPI).
- **Process simulation** - simulated execution of the process with the objective gathering KPIs and identifying optimization points.
- **Process optimization** - improving the process efficiency, effectiveness, agility, flexibility, and transparence.

**KPIs are financial and non-financial metrics used to help an organization define and measure process efficiency. Examples of a KPI are “Average revenue per customer”, “Average time for response to a customer call”, “Average order amount”, etc.**

**Figure 1: BPM process lifecycle**

---

**About Business Process Management**

BPM (Business Process Management) is a set of related activities, such as process modeling and design, process execution, process monitoring, and process optimization. This Refcard provides an overview of the BPM lifecycle together with the roles and results of business process modeling. It gives an overview of the BPMN (Business Process Modeling Notation) and presents the most important BPM patterns.

**BPM: Modeling**

**Why do we Model Business Processes?**

<table>
<thead>
<tr>
<th>Design new business processes</th>
<th>Focus on business goals, KPIs, customer needs, and business partner expectations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model existing business processes</td>
<td>Ensure the right flow of activities. Identify inputs and outputs of activities. Identify key documents and sources.</td>
</tr>
<tr>
<td>Restructure existing business processes</td>
<td>Focus on the activities and their added value. Focus on lines of business and their relations.</td>
</tr>
<tr>
<td>Development of end-to-end IT support for business processes</td>
<td>Detailed modeling of process flow. Detailed modeling of data, documents, business objects, and interfaces. Detailed exception handling.</td>
</tr>
</tbody>
</table>

**Who should take part in process modeling?**

The team should include different profiles and encourage looking at the process from different angles. This is particularly important for optimizations. Four to six people is usually an optimal team size. The following table lists the various profiles that should comprise the team:

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line of Business Expert</td>
<td>Good, in-depth knowledge of the process.</td>
</tr>
<tr>
<td>Process Owner</td>
<td>Responsible for the overall execution of the process, approves process modifications.</td>
</tr>
<tr>
<td>Moderator</td>
<td>Responsible for the meeting, for asking questions for leading the discussion into the right direction.</td>
</tr>
<tr>
<td>Modeling Expert</td>
<td>Responsible for design the process model (during and after the meeting).</td>
</tr>
<tr>
<td>QA Owner</td>
<td>Responsible for the alignment of processes in aspect of total quality management.</td>
</tr>
</tbody>
</table>

**How do we model?**

<table>
<thead>
<tr>
<th>Approach</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-down</td>
<td>We start with the process architecture. First we identify the major process activities and their flow. Then we model each activity into more detail.</td>
</tr>
<tr>
<td></td>
<td>High-level process modeling requires good knowledge about the processes and some experience. Modeling lower levels can reveal inconsistencies on higher levels.</td>
</tr>
</tbody>
</table>

---

**Get More Refcardz**

(They’re free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & technologies
- Bonus content online
- New issue every 1-2 weeks

Subscribe Now for FREE!
Refcardz.com
We start with the identification of activities. We model sub processes and business transactions and merge them into processes.

• We get lost in the details.
• Getting overview of processes and their relations can become very difficult.
• We can focus on too many details.

We start with core processes. We expand them with adding support processes around core processes.

• It can be difficult to identify core processes and how to progress into the right direction.

The Inside-out approach is usually the most pragmatic approach to process modeling. Provide a brief explanation of why it is the most pragmatic approach.

How do we model?

As-is model
We model the process as it is currently executed. Knowing the current as-is state is necessary for any future optimizations. We need to clarify whether we will model the process as it should be performed, or as it is performed in reality. Often there are significant differences between the two. When we model the as-is process we should not make on-the-fly modifications - not even those which seem obvious. We should however make notes of all possible modifications for the to-be process model.

To-Be model
We model the optimized model, where we should consider:
• Extent of changes – do we want evolutionary or revolutionary changes
• How radical the changes to the process can be
• Organizational and other limitations
• How the to-be model will be accepted by the involved people and what organizational changes will it require

How to approach designing a process model:
We should model the process to understand the detailed structure of it. We should identify at least at the following:
• Process activities, on various levels of details (depending on the selected approach)
• Roles responsible for carrying-out the process activities
• Events, which trigger the process execution and events that interrupt the process flow
• Input and output documents exchanged within the process
• Business rules that are part of the process

Below is the most conventional approach for designing a process model, in order of occurrence:
1) Identify the roles
2) Identify the activities
3) Connect the activities with roles
4) Define the order of activities
5) Add events
6) Add documents

FLOW OBJECTS

Flow Objects
Flow objects are the main BPMN constructs that define the behavior of a business process. There are three categories of flow objects:
• Activities: they represent the work performed within a business process (see Figure 4).
• Gateways: they represent how a sequence flow diverges or converges in a business process (see Figure 5).
• Events: they depict that something happens in a business process (see Figure 6).
**Connecting Objects**

Connecting objects are used to connect flow objects to each other and to other information. There are three categories of connecting objects: Sequence flow (see Figure 7), Message flow (see Figure 9), Association (see Figure 11).

**Events**

<table>
<thead>
<tr>
<th>Events</th>
<th>Start Event</th>
<th>Intermediate Event</th>
<th>End Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catching</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throwing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- None
- Message
- Timer
- Conditional
- Signal
- Error
- Cancel
- Compensation
- Terminate
- Link
- Multiple

**Figure 5: Types of gateways**

**Figure 6: Events, event triggers and results**

**Sequence Flow**

- Defining the order of execution of flow objects.
- Sequence flow with a condition (conditional flow).
- Default flow, which is chosen if none of the conditions are satisfied.

**Figure 7: Sequence Flow**

**Figure 8: Construct that can be connected via sequence flow (blue shaded field represent a legal connection)**

**Figure 9: Message Flow**

**Figure 10: Construct that can be connected via message flow (blue shaded field represent a legal connection)**

**Figure 11: Association**

**Swimlanes**

- We use pools to model participants in the process (company, buyer, seller, ...)
- A pool can contain several lanes in order to organize and categorize activities in a pool.
- A pool can hide its process, for example, when it represents an external participant.

**Figure 13: Swimlanes and pools**
In order to model an exception flow, we use intermediate events attached to the boundary of an activity. If such event is triggered during the activity execution, the flow is redirected through the intermediate event.

Example: The activity Check With Supplier of the example process has an intermediate timer event attached to its boundary. If the supplier does not provide a response within a certain timeframe, we remove the item from the order.

**WORKFLOW PATTERNS WITH BPMN**

**Sequence**

Workflow Pattern Description: An activity starts after completion of another activity.

BPMN: Activities are connected by a sequence flow directed towards the subsequent activity.

Example: After checking if the supplier can provide the necessary items in the Check With Supplier task, we notify the customer about their order in the Notify Customer task.

**Parallel Split**

Workflow Pattern Description: A path diverges into two or more parallel subsequent paths. The subsequent paths execute concurrently.

BPMN: The pattern can be implemented in several ways:
- We use several outgoing sequence flows for a flow object;
- We use a parallel gateway to divide a sequence flow into several sequence flows.
- We use an expanded sub-process in which we place the activities to be performed in parallel.
- We use an inclusive gateway with equivalent conditions.

Example 1: After receiving payment for the order we prepare the ordered items for shipment and issue the receipt concurrently.

Example 2: If the order items are in stock we send the confirmation of the order to the customer and reserve the ordered items in the inventory. These tasks are performed in parallel. Otherwise we check if the supplier can deliver the items.

**Simple Merge**

Workflow Pattern Description: Two or more alternative paths converge into a single subsequent path.

BPMN: The pattern can be implemented in two ways:
- We use an exclusive merge gateway to merge alternative paths.
- We use a flow object with two or more incoming sequence flows. The incoming sequence flows represent the ends of alternative paths. Any one of the incoming sequence flows trigger the flow object.

Note: The behavior is the same in both cases provided that the incoming sequence flows are alternative.

Example: The two alternative paths used to calculate the final price of the ordered items are merged using the exclusive merge or by sequence flows leading to the "Check Inventory" task.

**EXCEPTION FLOW**

In order to model an exception flow, we use intermediate events attached to the boundary of an activity. If such event is triggered during the activity execution, the flow is redirected through the intermediate event.

Example: The activity Check With Supplier of the example process has an intermediate timer event attached to its boundary. If the supplier does not provide a response within a certain timeframe, we remove the item from the order.

**Synchronization**

Workflow Pattern Description: Two or more paths converge into one subsequent path. The subsequent path is enabled when all the preceding paths complete (and-join).

BPMN: The Pattern can be implemented in two ways:
- We use a parallel gateway to merge several sequence flows into a single flow.
- We use an expanded sub-process in which we place the activities to be performed in parallel. Expanded sub-process completes after all the activities it contains complete.

Example 1: After preparing the ordered items for shipment and issuing the receipt, we ship the package to the customer.

Solution 1: Synchronization using a parallel gateway.

Solution 2: Synchronization using an expanded sub-process.

**Exclusive Choice**

Workflow Pattern Description: A path diverges into two or more subsequent paths. When the incoming path is enabled exactly one of the subsequent paths is selected and enabled.

BPMN: We use an exclusive gateway.

After analyzing the order we check whether the customer has provided a promotional code. If a promotional code is provided we collect discount information and use it to calculate final price. Otherwise, we calculate final price for the order without discounts.

Example 1: Exclusive choice with data-based exclusive gateway

Example 2: Exclusive choice with event-based exclusive gateway

**Simple Merge**

Workflow Pattern Description: Two or more alternative paths converge into a single subsequent path.

BPMN: The pattern can be implemented in two ways:
- We use an exclusive merge gateway to merge alternative paths.
- We use a flow object with two or more incoming sequence flows. The incoming sequence flows represent the ends of alternative paths. Any one of the incoming sequence flows trigger the flow object.

Note: The behavior is the same in both cases provided that the incoming sequence flows are alternative.

Example: The two alternative paths used to calculate the final price of the ordered items are merged using the exclusive merge or by sequence flows leading to the "Check Inventory" task.

Solution 1: Simple merge with exclusive merge gateway

Solution 2: Simple merge with sequence flows to a flow object
ADVANCED BRANCHING AND SYNCHRONIZATION PATTERNS

Multi-Choice
Workflow Pattern Description: A path is diverged into two or more subsequent paths. One or more subsequent paths may be executed.

BPMN: The pattern can be implemented in several ways:
• We use an inclusive gateway.
• We use a collection of conditional sequence flows.
• We use a complex gateway.

Example 1: Based on requirements the customer specified in the order, we confirm the order via e-mail, by regular mail or both. Example solutions 1 nd 2 represent equivalent behavior.

Solution 1: Multi-Choice with an inclusive gateway

Solution 2: Multi-Choice with conditional sequence flows

Structured Synchronizing Merge (Synchronizing join)
Workflow Pattern Description: Two or more paths converge into a single subsequent path. Several incoming paths may be enabled, in which case they are synchronized before the subsequent path is activated. In different process instances different number of incoming paths may be taken.

BPMN: We use an inclusive gateway.

Example: Based on requirements the customer specified in the order, we confirm the order via e-mail, by regular mail or both. If both activities are required to be executed, paths have to be synchronized before the process can continue.

Multi-Merge (Multiple Merge)
Workflow Pattern Description: Two or more paths converge into a single subsequent path. Each incoming path activates the subsequent path.

BPMN: We use sequential flow for every ending of a converging path directed towards the flow object of the beginning of the subsequent path.

Example: We confirm the order via e-mail, by regular mail or both. If either of the activities takes place, the order information file needs to be updated.

TERMINATION PATTERNS

Implicit Termination
Workflow Pattern Description: A process or a subprocess instance terminates when there is nothing else to be done and it is not deadlocked. The instance has completed successfully.

BPMN: The pattern can be implemented in one of the following ways:
• We end every path of the process or subprocess with an end event. If we use a start event we must use at least one end event.
• An end of a path in the process is indicated by a flow object without an outgoing sequence flow. The process completes when all tokens that were generated for the instance are consumed.

Note: We must either conclude all paths with an end event (with an exception of compensation activities) or not use end event for the given process/subprocess.

MULTIPLE INSTANCE PATTERNS

Multiple Instances without Synchronization
Workflow Pattern Description: Multiple instances of a task or a subprocess are created. They run concurrently and are not synchronized on completion.

BPMN: We set the values of activity attributes as follows:
• LoopType attribute to “multInstance”,
• MI FlowCondition to “None”.
• we set the value of the MI_Ordering attribute to “Parallel”.

Example: For every order in the order list an instance of the Process Order subprocess is invoked. The subprocess instances are executed concurrently. Every instance generates a token that continues after the instance is completed.

Multiple Instances with a Priori Design-Time Knowledge
Workflow Pattern Description: Multiple instances of a task or a subprocess are created. The number of instances is known at design time. They run concurrently and are synchronized at completion before the process continues.

BPMN: We set the attributes of the activity as follows:
• we set the value of the LoopType attribute to “MultInstance”,
• the expression of the MI.Condition attribute returns an integer representing the number of instances known at design time,
• we set the value of the MI.Ordering attribute to “All”.
• we set the value of the MI.Ordering attribute to “Parallel”.

Example: If a request for a loan exceeds 1000 USD the loan needs to be checked for approval by 3 eligible employees.

Multiple Instances with a Priori Run-Time Knowledge
Workflow Pattern Description: Multiple instances of a task or a subprocess are created. The number of instances depends on various run-time factors. Instances run concurrently and are synchronized at completion before the process continues.

BPMN: We set the attributes of the activity as follows:
• we set the value of the LoopType attribute to “MultInstance”,
• the expression of the MI.Condition attribute depends on the number of orders in the list, which can be different for every process instance. For every order in the order list an instance of the Process Order subprocess is created. The subprocess instances are executed concurrently. After all subprocess instances are completed, the process continues.

Example: The process receives a list of all orders. The expression of the MI.Condition attribute is based on run-time factors and returns the actual number of required instances at run-time,
• we set the value of the MI.Ordering attribute to “All”.
• we set the value of the MI.Ordering attribute to “Parallel”.

Termination Patterns
Implicit Termination
Workflow Pattern Description: A process or a subprocess instance terminates when there is nothing else to be done and it is not deadlocked. The instance has completed successfully.

BPMN: The pattern can be implemented in one of the following ways:
• We end every path of the process or subprocess with an end event. If we use a start event we must use at least one end event.
• An end of a path in the process is indicated by a flow object without an outgoing sequence flow. The process completes when all tokens that were generated for the instance are consumed.

Note: We must either conclude all paths with an end event (with an exception of compensation activities) or not use end event for the given process/subprocess.

Multiple Instance Patterns

Multiple instances can be implemented in several ways. Examples are:

1. Multi-Choice
2. Multi-Merge
3. Structured Synchronizing Merge

Each method has its own use cases and benefits. For instance, Multi-Choice allows for flexible decision-making, while Multi-Merge ensures that all incoming paths activate the subsequent path simultaneously. Structured Synchronizing Merge enables more complex synchronization patterns.

Example: In a loan application process, multiple instances of a verification task can be executed concurrently. Depending on the outcome, different subsequent paths are activated.

Example: A purchase order process can be designed with multiple instances, allowing for parallel processing of orders.
BPM is essential for continuous improvement of business process efficiency and effectiveness with the overall goal to produce business results faster, cheaper, better. This Refcard has provided the overview of the BPM lifecycle, presented the BPMN notation and demonstrated the most important patterns.