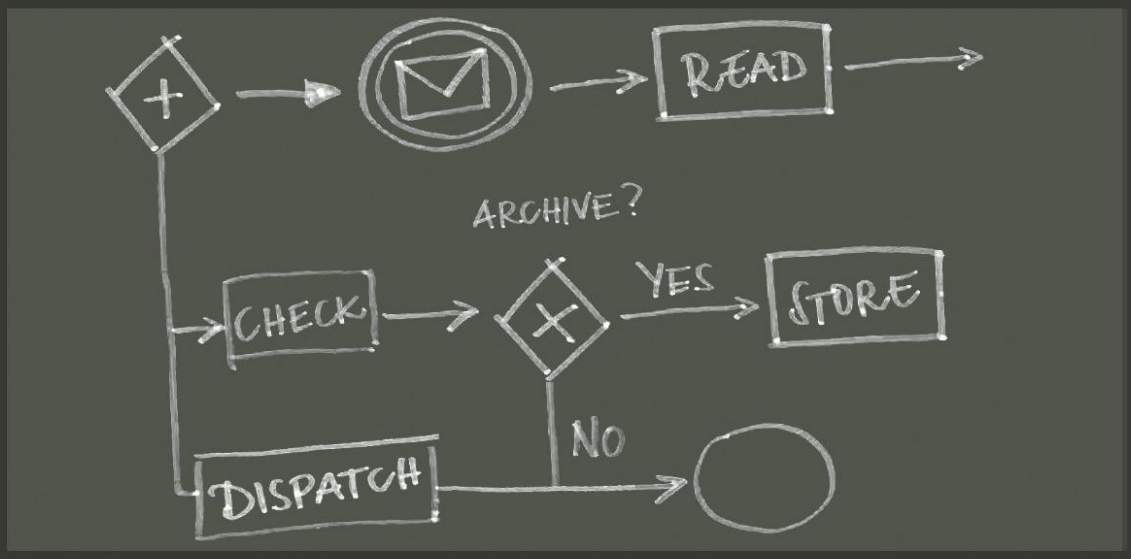
Real-Life BPMN

Jakob Freund
Bernd Rücker

Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company



camunda
the business process company

Real-Life BPMN

Jakob Freund
Bernd Rücker

Real-Life BPMN

Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company

Jakob Freund, Bernd Rücker
Founders of camunda services GmbH, Berlin, Germany
www.camunda.com

This first edition in English is based on the successful third German edition.
Also available in Spanish.

Editing for the English-language version of this book provided by James Venis of Lakewood, Colorado, USA, with assistance from Kristen Hannum.
www.jvenis.net

ISBN-10: 1480034983
ISBN-13: 978-1480034983

Contents

Preface

This is a book about Business Process Management (BPM) and Business Process Model and Notation (BPMN). Truth be told, there are several BPMN books on the market, and some of them are quite good. So why should you care about this one?

This book distills the BPMN project experience that we have accumulated while running camunda, our consulting company in Berlin, Germany. Our firm specializes in BPM. During the past five years, we have applied BPMN in practically every one of more than 250 customer engagements. These were big businesses, small companies, and public institutions.

In 2009, we published the first edition of our "BPMN Hands-On Guide." According to Amazon.de, it is still the highest-ranked book on BPMN in German. We are honored by the number of five-star-ratings that readers have awarded that book. And if you read their reviews, you see that what they like best about it are the real-life examples and the vivid language. (Most German books on this topic suffer from being abstract, and the writing is uninspired.)

We joined the Object Management Group (OMG) in 2009, and we helped to finalize BPMN 2.0. We also contributed chapters to the "BPMN 2.0 by Example" tutorial that the OMG provides to beginners. These interactions showed us that, even outside of Germany, few people have applied BPMN as often, as deeply, or as broadly as have we. We decided to translate our book into English, and we gave it a cooler-sounding title than "Hands-On Guide."

We hope you'll enjoy reading this book. Even more, we hope that you will find lots of help in it —great examples, meaningful tips and suggestions, and patterns that can lead you to solutions for your own real-life BPMN challenges.

You hear people bashing BPMN once in a while with arguments that are more or less pointless. You also hear valid critiques and useful suggestions. The good news is that we have a global community of people driving BPM generally and BPMN in particular, and we thank every one of them for contributing to a standard that, while not perfect, is definitely a big step in the right direction.

Our special thanks goes to Bruce Silver, whose own book, "BPMN Method & Style," is one of those "quite good BPMN books" on the market, and to James Venis, our editor for this English-language version. If you enjoy reading it, most of your praise should go to him.

Jakob Freund and Bernd Rücker
October 2012

1 Introduction

1.1 Business Process Management

This book is about Business Process Model and Notation (BPMN 2.0). To understand why BPMN was invented, we need first to understand Business Process Management (BPM).

1.1.1 Definition

Experts use different definitions for Business Process Management. We use the definition given by the European Association of BPM (EABPM) in its reference work, "BPM Common Body of Knowledge" [Eur09]:

> Business Process Management (BPM) is a systemic approach for capturing, designing, executing, documenting, measuring, monitoring, and controlling both automated and non-automated processes to meet the objectives and business strategies of a company. BPM embraces the conscious, comprehensive, and increasingly technology-enabled definition, improvement, innovation, and maintenance of end-to-end processes. Through this systemic and conscious management of processes, companies achieve better results faster and more flexibly.
>
> Through BPM, processes can be aligned with the business strategy, and so help to improve company performance as a whole thanks to the optimization of processes within business divisions or even beyond company borders.

What "end-to-end process" really means is "from start to finish." The goal is to understand and thus to assess and improve an entire process —not just its components. We find the EABPM's definition helpful because it treats automated and non-automated processes as both equally important and equally subject to the power of BPM. This understanding is essential to applying BPM successfully because it is rarely sufficient to improve only organizational procedures or the supporting technologies; most often we must improve both the procedures and the technology cooperatively.

1.1.2 BPM in practice

As consultants who specialize in BPM, our new projects almost always involve one of the following three scenarios:

1. The client wants to improve a process using Information Technology (IT).
2. The client wants current processes documented.
3. The client wants to introduce entirely new processes.

A vast majority of the time, we encounter the first scenario: the client seeks to improve a process with IT. The motivation often is a desire to improve efficiency —for example, to use software to eliminate manual keying or re-keying of data. A client may want to implement IT-based monitoring and analysis of routine processes based on key performance indicators (KPIs).

The second scenario, documenting processes, usually comes about because the client needs the documentation to guide the work of the people involved. Another rationale is that the documentation is mandated by regulation, or it is required to obtain certification such as ISO 9000.

The third scenario happens least often. We find that when companies want to introduce entirely new processes, it is usually because they are being forced to adapt to changed market conditions, develop new channels of distribution, or introduce new products.

In public announcements, companies may speak in generalities: they have an interest in exploring BPM or they want to increase their process orientation. In practice, especially in large organizations, the argument for BPM is usually well-defined and specific, but it can take two forms:

1. There is an acute reason for using BPM. The project concerns essential processes to be created, improved, or documented.
2. The reason for BPM is "strategic." There will be no direct or immediate benefit, and the project likely was initiated by some manager trying to advance his or her career.

As you can imagine, serious people don't greet the second argument with enthusiasm. It is our own experience, however, which makes us advocate strongly for this view: BPM, process management, or whatever you want to call it, is not an end in itself.

We always recommend introducing BPM in steps. Each step should yield a practical, measurable benefit that justifies the time and effort that it took to accomplish it. Once the justification of the first step is established, take the next step. You may think that this approach produces solutions isolated from each other, but what we mean to emphasize here is the controlled nature of the approach. Each step contributes to the big picture: the company's process orientation. A hiker may use a map and a compass to guide his or her steps. When you introduce BPM, you should use a good procedure model and common sense as your guides.

1.1.3 camunda BPM life cycle

Procedure models always seem to be either too simple or too complex. The too-simple ones contain only the most painfully obvious elements. They may be useful for marketing

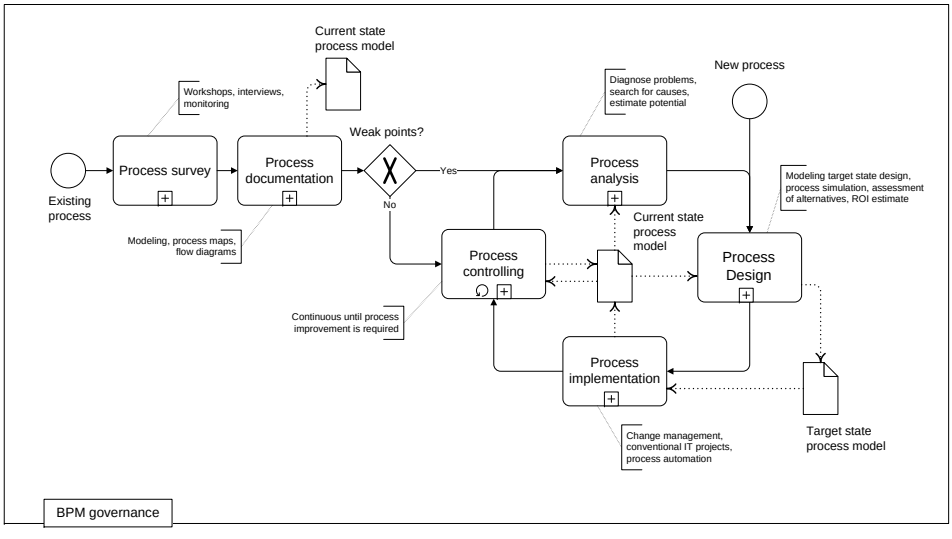


FIGURE 1.1 The camunda BPM life cycle

presentations, but not much else. On the other hand, overly complex models work so hard at anticipating every contingency that they trap the user like a fly in amber. They are unrealistically rigid. Still, without a model, we wouldn't have our "map" to orient us.

After examining the simple BPM life cycle, which is the most well-established BPM procedure model, we refined it according to our experience. We wanted to create a relatively lightweight model without too many restrictions. We thought this would be more practical than the brightly colored marketing materials we see so often at conferences and in meetings. We call ours the "camunda BPM life cycle." See it in figure 1.1.

We intend the camunda BPM life cycle to describe one process at a time. Any process can repeat independently of any other process, and the process can be at a different stage each time it repeats. The cycle triggers when one of the following situations arises:

- An existing process is to be documented or improved.
- A new process is to be introduced.

We have to start by examining an existing process. The **process discovery** clearly differentiates the subject process from other processes both upstream and downstream. The discovery reveals the output generated by the subject process as well as the importance of that output for the client. We use techniques such as workshops and one-on-one interviews to identify not only what needs to be accomplished, but also who needs to be involved, and which IT systems.

We document the findings from the process discovery in a current state process model. This **process documentation** may include many different charts and descriptions; it usually has multiple flow charts. A systematic examination of the current state process clearly identifies weak points and their causes.

We conduct **process analysis** either because first-time documentation or continuous process control has revealed a weakness of a process that cannot be remedied easily.

The causes of weak points identified by a process analysis become the starting point for another **process design**. If necessary, different process designs can be evaluated by means of the process simulation. We also conduct a process design when introducing a new process. The result in either case is a target state process model.

In reality, we normally want to **implement** the target state process model as a change in business or organizational procedures as well as an IT project. Change management, especially process communication, plays a decisive role in successful organizational change. For the IT implementation, the process can be automated or software can be developed, adapted, or procured. The result of the process implementation is a current state process corresponding to the target state process model that, conveniently, has already been documented.

In most cases, we find all the stages from process discovery to process implementation to be necessary. Because **process monitoring** takes place continuously, however, it reveals more about the ongoing operation of the process.

The most important tasks of process control are the continuous monitoring of individual process instances and the analysis of key data so that weak points can be recognized as quickly as possible. Problems with individual entities require direct remedies, and so do structural problems if that's possible. If necessary, the current state process model has to be adjusted.

If the structural causes of the problems are unclear or complex, this calls for an improvement project that —once again —starts with a systematic process analysis of the weak points. The decision to initiate such a project lies with the process owner and anyone else who depends on the process. It is common to regard continuous process control as something that follows process implementation, though it may be better to have it follow the initial documentation. This is especially true when doubt exists about the necessity of the improvement.

Given the importance of the process model within the BPM life cycle, you can imagine the importance of a modeling standard such as BPMN. Yet you may also notice that process modeling is *not* a stage in the camunda BPM life cycle. That's because process modeling is a method that affects *all* stages, especially process documentation and process design. As consultants, we constantly encounter people who try to insert process modeling as a stage at the same level as current state documentation. We think that's a misconception.

The BPM life cycle describes a simple way to achieve continuous improvement. To apply it requires coordination of the "triad." That means the responsible parties, the applied methods, and the supporting software tools. Getting the triad moving toward a common goal is the task of BPM governance, which has authority over all processes and all BPM projects in an organization.

The EABPM's definition of BPM used the term "process automation," and we've also used that term in describing the camunda BPM life cycle. BPMN was developed to automate processes better. Even if you are not an IT expert, you need to understand what process automation means because it will help you to grasp how BPMN builds bridges between business and technology.

1.1.4 Process automation

Here's a simple process: A potential bank customer mails a paper credit application, which ends up on the desk of a bank accountant. The accountant examines the application, then checks the potential customer's credit worthiness through the web site of a credit rating agency. The results are positive, so the accountant records the application in a special software —let's call it "BankSoft" —and then forwards the documents to a manager for approval.

Here's the same process automated: A potential bank customer mails a paper credit application. At the bank, a clerk scans the application into electronic form. Software called a "process engine" takes over the document and routes it to the bank accountant's virtual task list. The accountant accesses the task list, perhaps through the bank's web site or an email program like Microsoft Outlook, examines the application on screen, then clicks a button. The process engine accesses the credit rating agency, transfers the pertinent details, and receives the report. Since the report is positive, the process engine passes the information to BankSoft, and it creates an approval task in the manager's task list.

Whether this example represents optimal processing is not the point. It's here only to illustrate the following principles of process automation:

- Process automation does *not* necessarily mean that the entire process is fully automated.
- The central component of process automation is the **process engine**, which executes an executable process model.
- The process engine **controls** the process by informing humans of tasks that they need to do, and it handles the result of what the people do. (This is human workflow management.) It also communicates with internal and external IT systems. (This is service orchestration.)
- The process engine **decides** which tasks or service calls take place or not, under what conditions, and according to the result of the task execution or service call. Thus the people involved still can influence the operational sequence of an automated process.

Figure 1.2 on the next page illustrates these principles.

If you think that process automation is just a kind of software development, you are right. The process engine is the compiler or interpreter, and the executable process model is the program code. A process engine is the mechanism of choice where process automation is concerned.

- The process engine specializes in representing process logic. The services it provides would have required extensive programming in the past; using a process engine now can make you considerably more productive than before. (Or perhaps productivity is not an issue for you, and so you develop your own spreadsheet, word-processing, and drawing programs!)
- A process engine combines workflow management with application integration. This makes it a powerful tool for implementing all kinds of processes from start to end, regardless of other applications or the geography of people in the process. In some BPM software solutions, we can add a separate Enterprise Service Bus (ESB) or other components to the process engine to make the whole more versatile.

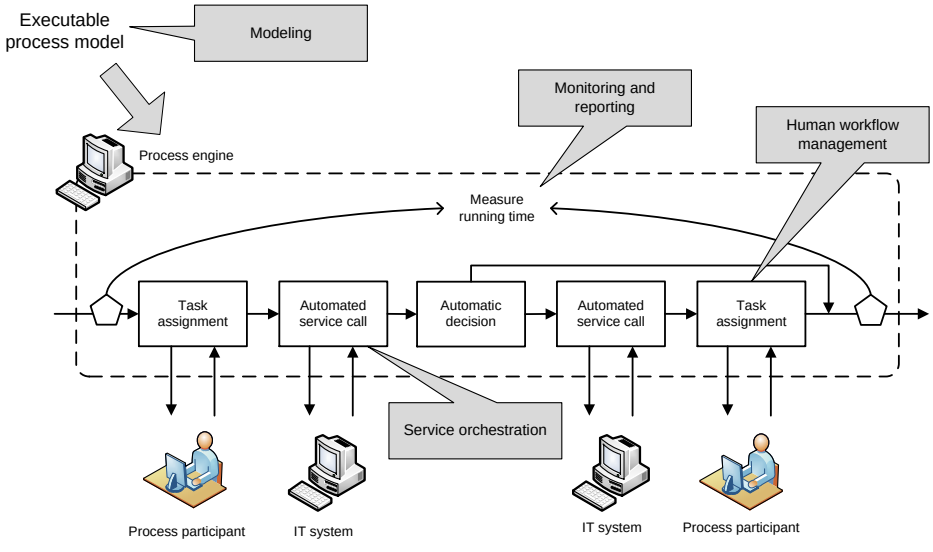


FIGURE 1.2 Process automation with a process engine

- As the process engine controls the process, it tracks everything. It always knows the current stage of the process and how long each task took to complete. Because the process engine monitors key performance indicators directly, it provides a means to analyze performance as well. This offers huge potential for successful process control.

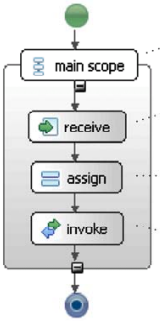
The three features above would themselves justify implementing a process engine, but there is a fourth justification: The process engine works on the basis of an executable process model. In the best cases, this model can be developed —or at least understood —by someone who is not a technician. This promotes genuinely good communication between business and IT, and it can even result in process documentation that corresponds to reality.

This leads us to BPMN.

1.2 Why BPMN?

Figure 1.3 on the facing page models a process being handled by a process engine. We created the model using an XML-based standard called Business Process Execution Language (BPEL). While some BPM products can depict BPEL graphically, BPEL defines no symbols. The resulting models fail to convey enough meaning about what's going on to help non-programmers make decisions. This has kept BPEL from being widely accepted.

When the Business Process Management Initiative (BPMI) published BPMN in 2004, the abbreviation stood for "Business Process Modeling Notation." The goal was to create process notation with standard graphics that also could be used for process automation. In 2005, the Object Management Group (OMG) took over development of BPMN. In 2011, the OMG approved version 2.0 (to which camunda contributed). Now, BPMN means




```
<?xml version="1.0" encoding="UTF-8"?>
<bpws:process name="SimpleBPEL"
  targetNamespace="http://www.camunda.com/example" ... >
  ...
  <bpws:partnerLinks>
    <bpws:partnerLink name="client"
                      myRole="SimpleBPELProvider"
                      partnerLinkType="tns:SimpleBPEL"/>
    <bpws:partnerLink name="someWebservice"
                      partnerRole="ServiceProvider"
                      partnerLinkType="tns:SimpleService"/>
  </bpws:partnerLinks>
  <bpws:variables>
    <bpws:variable messageType="tns:RequestMessage" name="input"/>
    <bpws:variable messageType="tns:InvokeMessage"  name="parameter"/>
  </bpws:variables>
  <bpws:sequence name="main scope">
    <bpws:receive name="receive"
                  createInstance="yes"
                  operation="initiate"
                  variable="input"
                  partnerLink="client"
                  portType="tns:SimpleBPEL"/>
    <bpws:assign name="assign" validate="no">
      ...
    </bpws:assign>
    <bpws:invoke name="invoke"
                 operation="methodName"
                 inputVariable="parameter"
                 partnerLink="someWebservice"
                 portType="tns:ServiceInvoke"/>
  </bpws:sequence>
</bpws:process>
```

FIGURE 1.3 Example of a simple BPEL process, graphic and as XML

"Business Process Model and Notation." Not only has the notation been defined, so has the "meta model."

The OMG is highly regarded. It is famous for the Unified Modeling Language (UML) standard used in software design. The OMG's sponsorship of BPMN adds credibility to the rationale that company managers already have for choosing a standardized model.

BPMN is a specification. It exists as a simple document that you can download for free in PDF format from the OMG [Obj09]. BPMN version 1.2 had about 320 pages; BPMN version 2.0 has about 500 pages. (Both are available only in English.) These documents define all BPM symbols, their meanings, and the rules for using them.

Before version 2.0, it was not possible to execute BPMN process models directly in process engines. Version 1.2 had not yet defined all the technical attributes required for execution, and this resulted in several unfortunate attempts to convert (or "map") BPMN models to BPEL models (see section 5.7 on page 178). BPMN 2.0 enabled direct execution of BPMN process models in process engines, the first important criterion in favor of its use. The second important criterion was standardization, which yields the following benefits:

- You become less dependent on certain BPM tools because you don't need to learn a new notation every time you change tools. Even now, there are more than 70 BPMN tools available (and many of those tools are free, which is wonderful, but it lowers the resistance to changing from one tool to another).
- The likelihood increases that your customers, suppliers, consultants, and so on have a grasp of BPMN and will therefore understand your process models more readily.

- When you hire new staff members, the odds are higher that they already read and can create BPMN process models.
- You can benefit from the investment made by universities and private businesses that share their advanced BPMN solutions. The BPMN framework we are going to present in the following section is an example. We never would have developed it unless BPMN was a standard.

1.3 Can BPMN bridge the gap?

1.3.1 The dilemma

First, BPMN provides a set of symbols. Second, it implies a methodology that expresses itself as rules for combining the symbols graphically. Third, the symbol definitions and the rules for applying them is called syntax. Fourth, the meaning of the symbols and constructs that you can model with the symbols is called semantics.

Unfortunately, just knowing the BPMN symbols is not enough for you to create useful process models. Since 2007, we have used BPMN extensively and often, and you can believe that we have suffered terribly! Mainly, we suffered because we always tried for models with correct syntax and consistent semantics —in other words, unambiguous models. Others took the easy way out by saying, "Our process model is not really syntactically correct, and it's not really unambiguous. But that doesn't matter because the main thing is that the consumer understands it!" This attitude backfires because:

- When you apply BPMN in a syntactically incorrect way, you lose all benefits of standardization. (After all, what do you need a standard for if the models all look different in the end?) Many BPMN tools won't even enable syntactically incorrect modeling.
- Semantic inaccuracies or inconsistencies always create the risk that your model will be misinterpreted. This risk is particularly high if you create an inconsistent target state process model and then send it to IT to implement.

If you want to supply your process model directly to the process engine, you must make your model correct, precise, and consistent. At that point, you still have to reconcile two contradictory objectives:

1. Different consumers must understand and accept the process model. Making the model easy to comprehend helps to reach agreement.
2. Because the process model has to meet the requirements of formal modeling, there's usually an unavoidable level of complexity to it. This makes it harder to achieve the comprehension that leads to agreement.

Failure to reconcile the objectives, to bridge the gap in understanding between business and technology, is the main reason that process models have had limited success in the past. The really bad news is that BPMN alone also will not succeed!

Just as with spoken language, you can use BPMN and either succeed or fail. As with spoken language, successful use of BPMN depends on whom you want to communicate

with and about what. You speak to your colleagues about the work you all know so well differently than you speak to your three-year-old about why the cat doesn't *like* to be pulled by its tail. Similarly, you will need other BPMN process models for coordinating with your co-workers than for presenting the future process to upper management. (Decide for yourself if the latter scenario is akin to the toddler-and-cat situation.)

On one hand, different BPMN process models are required for specific audiences and topics so that they can be understood. On the other hand, each model must contain all the detail necessary for the topic. BPMN may be a "common language" for business and IT, but the phrasing will remain different nevertheless.

The following understanding is therefore imperative for your work with BPMN:

The precision and formal correctness of the process model must vary depending on the objective of modeling and the consumers to be expected.

1.3.2 The customers of a process model

Whenever we model processes, we have to work in a customer-focused way. We must always keep the consumer of our model in mind. We must put ourselves in his or her place. This sounds simple, but few process models actually support this orientation.

As we have been saying, the knowledge, skills, and interests of the people who view our process models vary a great deal. In the following list, we have compiled the types we encounter in our BPM projects. These descriptions are for the roles played in relation to the project; they are not the titles of people in any organization. What we find is that the more experience an enterprise develops with BPM, the more consistently we see these roles fulfilled. We recommend that you become familiar with:

- **Process Owner:** Process owners have strategic responsibilities for their processes. They are vitally interested in optimizing performance. They often have budget authority, but before they sign off, they need to be convinced that your improvement plan will work. In most companies, process owners inhabit the first or second tier of management. They may be members of management committees or heads of major divisions.
- **Process Manager:** Process managers have operational responsibility for their processes. They report directly or indirectly to the process owners. They apply for improvement projects, acting as the ordering party for external services. Process managers are often low- or middle-level managers.
- **Process Participant:** Process participants work with the processes and actually create value. Their relationship to the process manager varies greatly. In companies organized by functional divisions —sales, logistics, and so on —a process manager is a functional executive for the division in which the process is carried out. Process participants report directly to that functional executive. (If the process is carried out across departments, which is common especially in process matrix organizations, see figure 1.4 on the following page, conflicts can arise between department executives. Process modeling alone cannot resolve such issues, which is why we do not examine them further in this book.)
- **Process Analyst:** The core competencies of process analysts are BPM in general and BPMN in particular. They support process managers as internal or external service

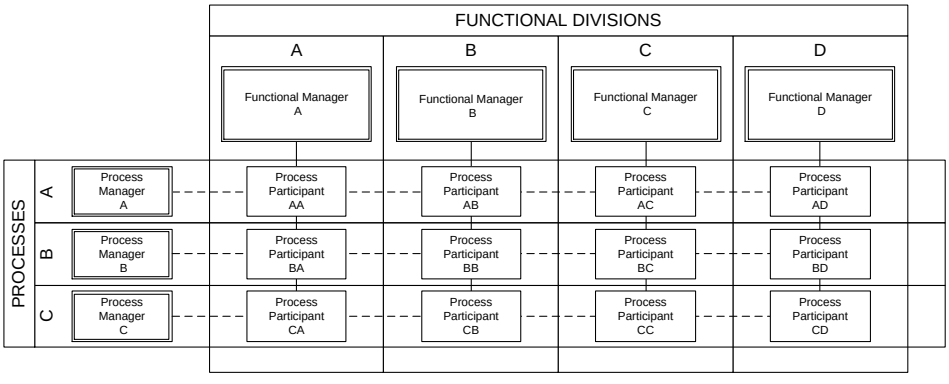


FIGURE 1.4 The process matrix organization

providers through all stages of the BPM life cycle. A process analyst may be the contact for external service providers or may act as the process manager's representative. Within the company, process analysts usually have either their own sphere of competence in BPM, such as the business organization, or they are part of their IT divisions. It is rare, however, for a process analyst to be responsible for technical implementation.

The analyst may like technical work, may know BPMN backwards and forwards, but his or her strengths are as an organizer and communicator. As the builder of bridges between business and IT, the process analyst is the center of every BPM project. About 70 percent of the people who claim or are assigned to this role, in our experience, are poorly qualified because they lack the proper analytic predisposition. The most important qualification of a process analyst is not a facility for sending out information, but a facility for receiving it. Good process analysts naturally want to understand everything thoroughly. At the same time, they have plenty of empathy in relating to the other people involved, and they can tailor their communication for every group. They remember every detail, but they also sensibly shield details from those for whom the details would just be a distraction.

Do project managers make good process analysts? No, nor should the project manager be the same person as the process analyst. Most project managers see themselves as "dynamic, action-oriented individuals" who constantly have to "get someone on board" or "pull chestnuts out of the fire." They may be extremely skilled at delegating responsibility (and honestly, some are clueless windbags). It may seem ideal to have a good process analyst also manage a BPM project, but it rarely works.

- **Process Engineer:** Process engineers use technology to implement the target state process modeled by process analysts. In the best cases, they do so in the process engine, which automates the process. You can call a programmer who programs the process logic in Java, C#, or another language a process engineer. The programmer's major work takes place during the implementation stage of the BPM life cycle, though the process analyst may get the process engineer involved at other stages as well.

Now that we've outlined the potential customers of a process model, we can talk about what the models should look like to keep these customers happy.

1.3.3 A method framework for BPMN

In our consulting projects and in our workshops, we have introduced a great many people from all kinds of enterprises to BPMN. From that collected experience, we developed a practical framework for applying BPMN. The framework helps us decide which BPMN symbols and constructs to use in which situations —and also when to hold back in the interest of simplicity. The framework focuses on projects with processes that need improved technological support and where it is the target state that needs to be modeled. In principle, what we show as modeling patterns can also be applied to other scenarios, such as the discovery, documentation, and analysis of current-state processes. We and our customers have had positive experiences with this framework, but even if you do not want to copy it exactly, try using it as a reference for your own modeling conventions.

The camunda BPMN framework (caBPMN) shown in figure 1.5 consists of four levels. Levels 1, 2, and 3a are the most interesting ones, which we will discuss in detail in the rest of this book.

- **Out of scope: process landscape:** We developed our framework specifically for projects involving an individual process or a manageable group of related processes. (For now, we won't deal with modeling entire process landscapes using, for instance, so-called process maps. BPMN's portfolio does not contain process maps.) Even when we've already modeled one or more process landscapes using BPMN at a customer's request, primarily with the collapsed pools and message flows described in section 2.9 on page 69, we cannot recommend doing this. If you want a process landscape, you should

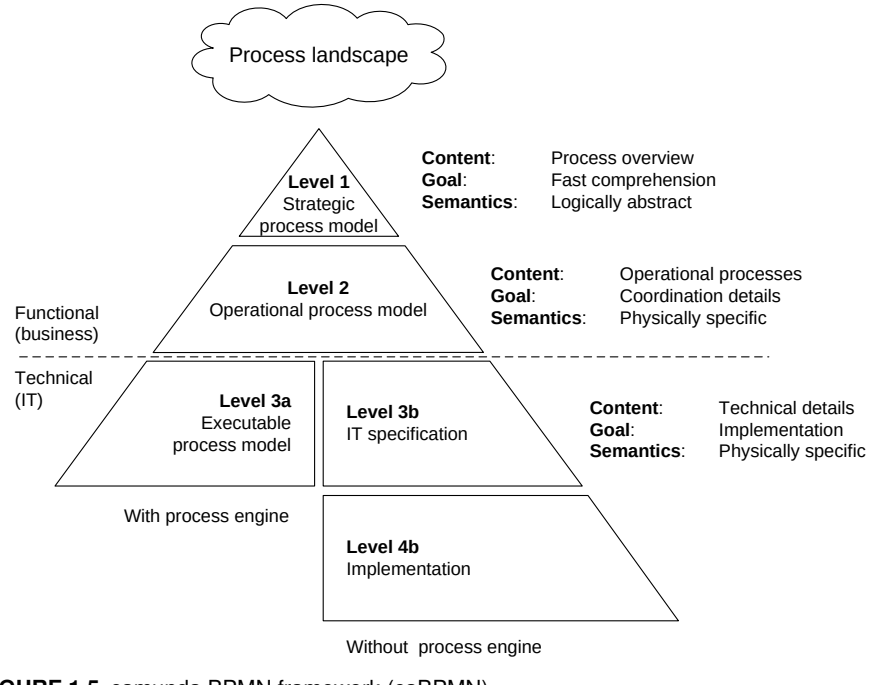


FIGURE 1.5 camunda BPMN framework (caBPMN)

use a more appropriate tool —perhaps a proprietary one that uses block arrows and rectangles and lots of colors. You can, of course, refine a process landscape with BPMN diagrams by linking the individual elements with flow charts.

- **Level 1: Strategic process model:** The primary target group for Level 1 process models are process owners and process managers. A secondary group early in the project may include process participants and process analysts. We provide this as a general, result-oriented representation of the process. We want to create the quickest possible understanding of the procedure for an audience that has no special BPMN knowledge. We sketch the process in a few steps as an overview. We don't show errors or variations. See Chapter 3 for more detailed information on creating Level 1 models.
- **Level 2: Operational process model:** At this level, we investigate operational details of the actual process. These models are necessary to guide the participants in their daily work, but they also are crucial for the process analyst to analyze for weak points and possible improvements. The great accomplishment of a process analyst is to develop a Level 2 process model that not only coordinates with the organizational process model, but which also can be handed to the process engineer for further refinement and actual implementation in Level 3. In Chapter 4, we describe our procedure for this, tailored to BPMN.
- **Level 3a: Executable process model:** We recommend implementing the process in a process engine. Since this isn't always possible, we subdivide the caBPMN framework: Level 3a deals with refining the process model created at Level 2 for a process engine. This only became possible with BPMN version 2.0; we explain it in Chapter 5.
- **Level 3b: IT specification:** Without a process engine, you must implement the process logic with a conventional programming language —or an adjustable ERP system or something similar. This usually requires additional technical specification before implementation begins, and the procedure depends on the platform. BPMN plays a subordinate role in such cases. We address this possibility in section 4.4.5 on page 125 in terms of change documentation, but it is not otherwise in scope for this book.
- **Level 4: Implementation:** This is where the process is actually technically implemented on a "conventional" platform. If you use a process engine, you do not need separate IT specifications, which is why our pyramid is asymmetrical.

The caBPMN framework is purely methodical. In other words, it works independently of particular software tools, although certain tool functions make it easier to apply. We deal with this in section 6.4.2 on page 205.

About half of this book is detailed description of the caBPMN framework. Because those chapters offer so much practical information, we encourage you to read them even if you are unconvinced of our framework's utility. If that's the case, just think of our framework as a classification system for our advice on the practical application of BPMN.

We look forward to your comments, not just on this book, but also on the caBPMN framework itself. It isn't perfect and it isn't final —and it can always be better. With your help, perhaps we can make it better for everyone!

2 The notation in detail

2.1 Understanding BPMN

What does a monkey know about the taste of ginger?

This Indian proverb expresses the truism that you can't fully appreciate what you don't understand. We see a corollary in the English expression, "Don't cast pearls before swine."

BPMN is a pearl not everyone can appreciate because not everyone understands it. If you are new to the standard, you won't regret spending some time to familiarize yourself with its underlying principles. For those who already know the BPMN specification, this chapter provides explanation and tips beyond the specification itself. It also describes the visual conventions we use when applying symbols. This is our BPMN etiquette.

A full understanding makes BPMN an extremely powerful tool for any modern BPM project. In our experience, however, even those with high confidence in their BPMN knowledge still may fail to understand certain fundamental principles, and they often express surprise that sequence flows must not be drawn across pool boundaries.

2.1.1 Things BPMN does *not* do

BPMN was developed to model processes: logical, chronological sequences of events. That's all. Nonetheless, you often hear BPMN criticized for *not* representing:

- Process landscapes
- Organizational structures
- Data
- Strategies
- Business rules
- IT landscapes

We appreciate how important it is to incorporate these topics into process documentation. We also know that many process professionals come from the systematic world of Architecture of Integrated Information Systems (ARIS) (see section 2.12.1 on page 80). They have worked with event-driven process chains (EPCs), and they may regard BPMN as insufficient. But feasible (and even partly standardized) notations exist

for the topics in the list above, and we are glad for it! It relieves BPMN of over-complication and keeps BPMN from being a monstrosity that nobody would want to compile, develop, or even understand. We remind those professionals that:

- BPMN process models are easy to combine with other types of diagrams. It is just a question of the tools used.
- BPMN provides extension options, including custom symbols. We explain this in section 2.11.2 on page 79.

Obviously it would be wonderful if BPMN could provide a complete, out-of-the-box alternative for the ARIS methodology. We admit that's not the case for the pure standard, but precisely because BPMN *is* a standard, software tools are now being created to use BPMN for the other necessary views.

2.1.2 A map: The basic elements of BPMN

When you draw a process diagram in BPMN, you use symbols that can be assigned to the categories shown in figure 2.1. We refer to these categories as the basic elements of BPMN.

In general, certain tasks have to be carried out during a process (*activities*), perhaps under certain conditions (*gateways*), and things may happen (*events*). What connects these three flow objects are *sequence flows*, but only within a *pool*. If connections cross pool boundaries, the process resorts to *message flows*.

Furthermore, *artifacts* provide additional information on the process, but these cannot influence the order of flow objects directly. Every artifact can connect to every flow object

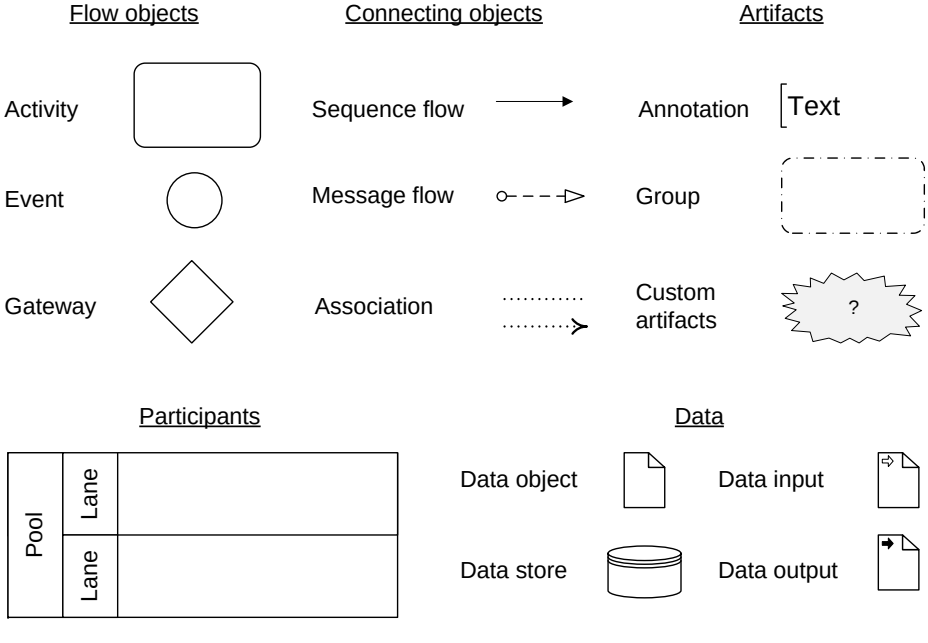


FIGURE 2.1 The basic elements of BPMN

through *associations*. (You can also incorporate your own symbols as additional artifacts into a BPMN palette all your own. We detail this option in section 2.11.2 on page 79.)

BPMN 2.0 contains an additional **data** category. This refers to the creation, processing, and filing of information that may become relevant within the scope of process handling, thus the category's symbols usually connect to activities through associations.

There are three more aspects necessary to a full understanding of BPMN:

- The advanced ideas and rules behind this simple scheme
- The full range of symbols and
- The practical know-how to apply this stuff

The ideas and rules and the full range of symbols are explained later in this chapter. The practical know-how is acquired through experience, but we offer our knowledge in the subsequent chapters to help speed your progress. We've also devised a few "recipes" for applying BPMN. They may help you to avoid some traps that often snare beginners.

2.1.3 Perspectives in process analysis

Someone accustomed to modeling processes with other notation systems may have trouble adjusting to an extremely important aspect of BPMN: everything depends on perspective.

BPMN is based on the assumption that one or more *participants* can exist within one diagram. Do not, however, jump to the conclusion that a participant functions like a role, a department, or an employee! In BPMN, a participant is a *logical* element to which the following rules apply:

- There can be only one participant for each process. (This means logical participants; there may be many human participants.)
- The participant has complete control over the process flow.
- The participant is fully responsible for its process.
- Other participants cannot influence a participant's process; they may not even know how it works.
- If a participant wants to interact with other participants within the context of the process, the participant must communicate with the others, and they affect their own processes accordingly.

The same process may look completely different for each participant, and how it looks depends on its perspective. This results in different process models.

In BPMN, the symbol for a participant and for its process is the pool; each process gets its own pool. Logically, however, a participant can control more than one process.

If you learn to handle pools properly, you will have mastered the most significant principle of process modeling —assuming you're aiming for modern BPM aligned with necessary business IT. In section 2.9 on page 69, we detail this subject and also solve the riddle of why there can be only one logical participant for each process.

2.1.4 Models, instances, tokens, and correlations

In the specification for BPMN 2.0, Chapter 7 contains a new section titled *Understanding the behavior of diagrams*. It introduces the idea that the behavior of the diagrams must be understood as well as the processes they describe. (Note: Because a diagram may contain several pools, a single diagram implies *n* processes). This is easier in theory than in practice because some process models are so complex that it becomes hard to know how to handle some circumstances. Remember the following:

- **Process model**: The basic description of a process. A diagram may describe one or more process models.
- **Process instance**: A process carried out in reality —what a layperson calls an "operation." A customer complaint is an instance of a complaint process, for example. Some processes may be instantiated only a few times in a year, such as end-of-quarter reporting in the accounting department. Other instances occur more often. Think of the millions of credit-report requests in a year's time.
- **Token**: You can apply the token model, if you have a process model in mind and want to find out which process paths must or may be used during a process instance. A token is a concept we compare to a car: A car follows a road. At an intersection, its driver must decide to continue in a straight path or to turn left or right. Or perhaps the car turns *and* a clone of the car continues straight on. This is where the car metaphor breaks down, but we hope you get the gist: that the road system corresponds to a process model and that any particular route the car takes represents an instance. The token model can help you understand even the most complex BPMN process models, so tokens are also explained in the above-mentioned section of the BPMN specification. We apply this method frequently in examples throughout this book.
- **Correlation**: Do you ever get letters with a transaction key or a file number? When you reply, you are expected to reference the key or number to make it easier for your correspondent to allocate your communication properly. This allocation based on an unambiguous key is called correlation. Another example is when you pay a bill, and you are asked to write the invoice number on your check. If you don't comply, your payment may not be properly allocated, and the lack of correlation can lead to reminder notices, late-payment fees, and other unpleasantness. Correlation is often crucial to the success of processes, from both organizational and technical points of view. Some of the costliest mistakes come from carelessness with the issue of appropriate correlation.

2.1.5 Symbols and attributes

The BPMN specification describes the symbols provided for process modeling. It also describes the many attributes that you can assign to the symbols. Many of these attributes don't appear in diagrams, though they are stored in the modeling tool and used when a process engine executes the modeled process.

2.2 Simple tasks and none events

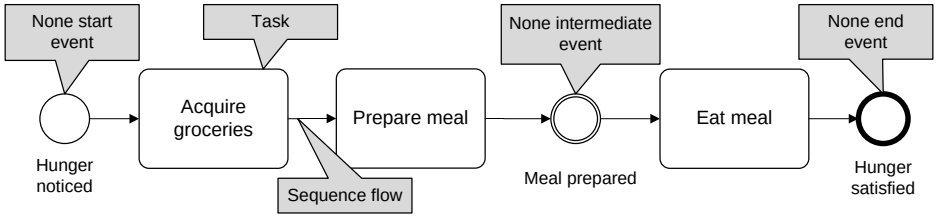


FIGURE 2.2 Our first process

Figure 2.2 shows a simple process triggered by someone being hungry. The result is that someone must shop for groceries and prepare a meal. After that, someone will eat the meal and have his or her hunger satisfied. You will easily recognize the following symbols and their meanings in the diagram:

Tasks

Tasks are the heart of the process. Ultimately, something has to happen to bring about a desired outcome. In BPMN, a task technically is part of the activities category, which also includes the subprocess explained in section 2.8 on page 57.

Our BPMN Etiquette

When naming tasks, we try to adhere to the object-orientated design principle of using the [verb] + [object] pattern. We would say "acquire groceries," for example, not "first take care of shopping for groceries."

Events

Events describe significant things that happen before, during, or at the end of a process. The example in Figure 2.2 uses only "none events." None events can be used in a process flow to indicate a status or a milestone. We explain about more event types later.

- **Start events** show which event causes the process to start.
- **Intermediate events** stand for a status that is reached in the process and that is modeled explicitly. They are used infrequently, but intermediate events can be useful, for example, if you regard reaching a certain status as a milestone and you want to measure the time until the milestone is reached.
- **End events** mark the status reached at the end of the process path.

Even for these simple events, we have to make further distinctions:

- Start events are catching events. That means something happened independent of the process, but the process has to wait for this event, or react to it.
- Intermediate events may occur, or they may be caused or triggered by the process itself (throwing events). The none intermediate event marks a status reached by the process and is therefore a throwing event. (Again, we will explain about more event types later, including more types of intermediate events to be classified as catching events.)

- End events take place when the process can no longer react to them. As a consequence, only the process can trigger them.

Our BPMN Etiquette

Events refer to something that has already happened regardless of the process (if they are catching events) and as a result of the process (if they are throwing events). For this reason, we use the [object] and make the [verb] passive in voice, so we write "hunger noticed." BPMN does not require you to model start and end events for a process —you can leave them out —but *if* you model a start event, you must model an end event for each path. The same is true for end events that require start events. We always create our models with start and end events for two reasons: first, that way it's possible to determine the process trigger, and second, you can describe the final status of each path end. We only sometimes abandon this practice with subprocesses. More on this later.

Sequence flows

The sequence flow describes the time-logic sequence of the flow elements: tasks, events, and the gateways we describe later.

The process path taken by our token is also a sequence flow. It is "born" with the process instance because of the start event. Through the sequence flow and by means of the tasks and the intermediate events, it reaches the end event, where it is "consumed" and disappears. This also leads to the "death" of our process instance.

Our BPMN Etiquette

We always draw our process diagrams horizontally, from left to right, but there is nothing to keep you from orienting your flow differently. You may orient your diagrams vertically instead of horizontally, for example, although that is unusual.

2.3 Design process paths with gateways

2.3.1 Data-based exclusive gateway

Certain things can only be done under certain circumstances, so few processes always take the same course.

In our simple example (figure 2.3 on the next page), we want to go into the details of cookery. Driven by hunger, we think about what we are going to cook today. We only know three recipes, so we choose one. We can either cook pasta *or* cook a steak *or* prepare a salad. Let's say that these options are exclusive —we will never prepare more than one at a time. The point of decision on what to do next is called a "gateway." We decide based on available data (the chosen recipe) and we follow only one of the paths, which is a data-based exclusive gateway. We abbreviate "exclusive gateway" as **XOR**.

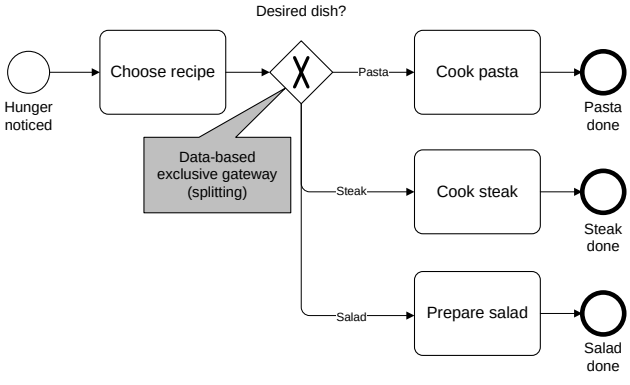


FIGURE 2.3 The XOR gateway.

Bear in mind that a gateway is not a task! You have to determine facts and needs before reaching a gateway. We will encounter this again in Business Rules Management (see section 4.5.4 on page 134).

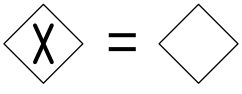


FIGURE 2.4 Both symbols mean the same.

BPMN uses two symbols for XOR gateways (see figure 2.4). They are identical in meaning. We always use the version with the X because it seems less ambiguous, but use what works for you.

Our BPMN Etiquette

As in figure 2.3, we place the crucial question before the gateway. This is our convention, which has proved its value in our projects. Possible answers go on parallel paths after the gateway, which is how the BPMN specification shows them. We always work with XOR gateways as follows:

1. Model the task that requires a decision for the XOR gateway.
2. Model the XOR gateway after that. Create a question with mutually exclusive answers.
3. Model one outgoing path (or sequence flow) for each possible answer, and label the path with the answer.

An XOR gateway can have as many outgoing paths as you like. We start some paths in the upper left corner and the others in the bottom left corner, but these are just our style conventions.

By the way, it is not unusual to have three end events nor for the process to result in three end states. Recognizing this possibility can help you with more complex diagrams. Later,

we will give more reasons for working with different end events. BPMN is not a block-oriented process notation, so you need not merge a split process path later —you can, but you don't have to.

Certainly, it may make semantic sense to merge the three paths. The meal is eaten after it's prepared, regardless of the recipe chosen. We can use the XOR gateway for merging also, and doing so leads the tokens from the three incoming paths into a single outgoing path. (See figure 2.5.)

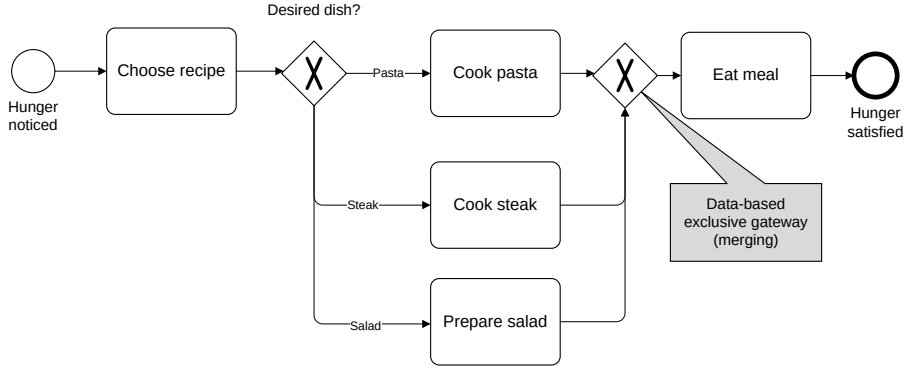


FIGURE 2.5 XOR gateways can also merge.

The double application of the XOR gateway —splitting and merging or "XOR split" and "XOR merge" —may confuse beginners. You can even model an XOR gateway that merges *and* splits at once! (See figure 2.6.) You have to decide if you prefer to compact your diagrams this way. For our part, we usually choose not to do that, and instead draw the two XOR gateways in succession. This method prevents misinterpretation.

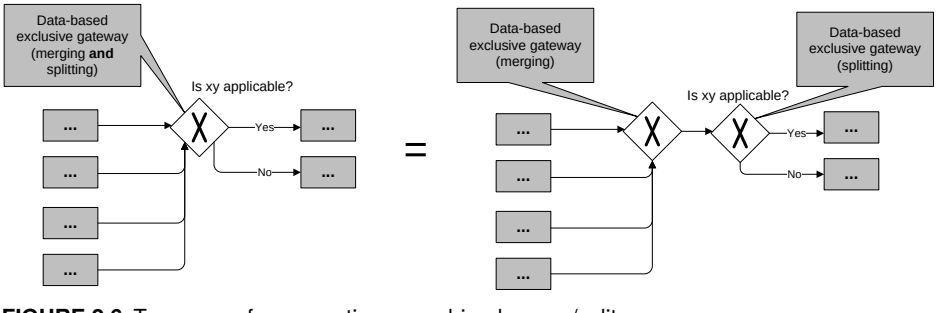


FIGURE 2.6 Two ways of representing a combined merge/split.

2.3.2 Parallel gateway

Suppose that now we want a salad on the side. If you want salad no matter what, you could model it as we have done in figure 2.7 on the next page.

Here, we've introduced another symbol, the (text) annotation. This is an artifact that you can associate with any flow object (in this case, tasks). You can enter any text; in our

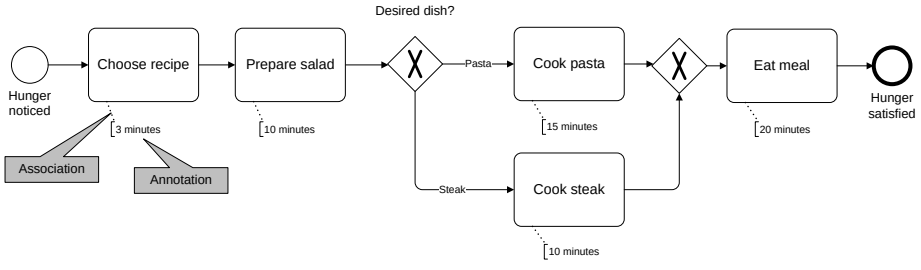


FIGURE 2.7 Preparing salad and main course.

example, we entered the average time to carry out the associated task. The total of the task times equals the running time of the process, which was a total of 48 minutes for pasta and 43 minutes for steak. Congratulations: you've just analyzed your first process based on key data!

Still, this means waiting 23 or even 28 minutes until you can start to eat. Insufferable! You're really hungry, but what can you do? Maybe you don't prepare the salad first and then cook the pasta or the steak, but you work on both at the same time —in parallel. The appropriate symbol is the parallel gateway, or the "AND gateway" for short, as shown in figure 2.8.

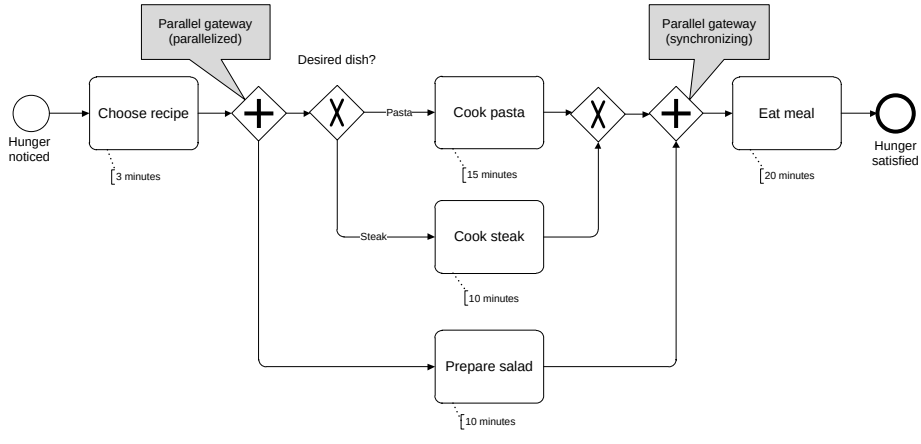


FIGURE 2.8 Preparing salad and main course at the same time.

Diagramming tasks as parallel does not make simultaneous processing compulsory. In contrast to the example shown in figure 2.7, it is also not imperative that you prepare the salad before starting other tasks. Parallel preparation does, however, reduce our total time by 10 minutes. It is classic process optimization to make tasks parallel as much as possible.

As the example shows, the process is not only parallel (it uses an "AND split"), but the paths also synchronize later (an "AND merge"). The reason is easy to understand: you can only start to eat after both main course and side dish are prepared.

How would the concept of tokens apply to an instance of this process? The token is "born" at the start event, it runs through the "choose recipe" task, and then it plunges into the AND split. One token emerges from the gateway for each path. That means two tokens in this example: The first token enters the XOR split, and its outgoing path depends on the recipe selected.

Let's assume we want to cook pasta. The token enters the task and stays there 15 minutes. At the same time, the second token enters the second, "prepare salad" task, where it stays only 10 minutes. After 10 minutes, it moves on to the AND merge. The number of incoming paths determines the number of related tokens the gateway is waiting for, so here, it waits for two tokens of the same process instance.

In our scenario, the second token arrives at the AND merge after 10 minutes, while the first token stays in "cook pasta" for a total of 15 minutes. This means the AND merge waits until the first token arrives —an additional 5 minutes. At that point, the tokens happily merge into a single token, which continues on the outgoing path.

Does that sound too abstract or technical? It is not. The AND merge behavior is identical to your own: The salad is ready, but the pasta is not, so you wait. When the pasta finally is ready, you eat.

Why the seemingly complicated token concept then? Think of 90 million process instances created by credit agencies, for instance, every year. Surely, these don't executed in strict sequence. They overlap. Correctly to define and carry out such complex processes and their various parallel operations, branchings, mergings, and synchronizings every day, the token approach is not only extremely helpful in conceptual design and implementation, but also necessary. We hope it is clear by now that process instances are not identical to tokens: Many tokens can run within the bounds of a single process instance.

Check your understanding with the following questions:

Question: Figure 2.9 shows the same process, but the AND merge was left out for lack of space, and the path from the "prepare salad" task leads directly to the XOR merge. What happens if we instantiate the process, and we decide in favor of pasta?

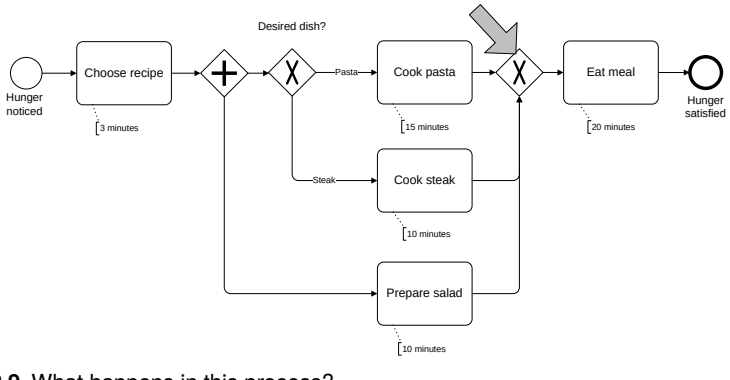


FIGURE 2.9 What happens in this process?

Answer: The token is generated and then cloned as always at the AND split. As soon as we finish preparing the salad, the token passes through the XOR merge and "eat meal"

executes. Five minutes later,"cook pasta" also completes. Its token passes through the XOR merge and "eat meal" executes again! That's not the behavior we wanted.

Question: Figure 2.10 shows a process that consists of two tasks only. Once instantiated, how long does the process instance live?

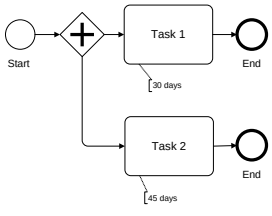


FIGURE 2.10 How long does the process instance live?

Answer: It lives 45 days, which corresponds to the run time of the process. Even though the token generated in the AND split passes through task 1 after 30 days and then is consumed by the upper end event, the second token stays in task 2 for an additional 15 days. The process instance continues to live until the second token is consumed by the lower end event.

Note: As long as just one token lives within the process, the process instance lives too! The instance cannot finish until all tokens generated are consumed.

2.3.3 Data-based inclusive gateway

We want to make our process even more flexible: When we are hungry, we want to eat

- Only a salad,
- A salad and "something real," like pasta or steak, or
- Only something real.

Using the symbols you have learned so far, you could model the situation as shown in figure 2.11.

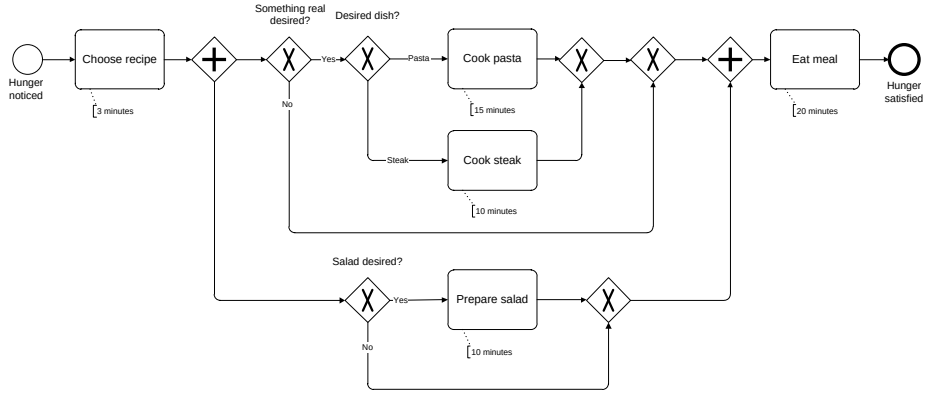


FIGURE 2.11 Various options in the combination of our meal.

If you want a more compact representation, you can use the data-based inclusive gateway —the OR gateway for short. (See figure 2.12.) Use OR gateways to describe and/or types of situations, in which processing can flow along one, many, or all outgoing paths. OR gateways can keep diagrams from becoming overly complex.

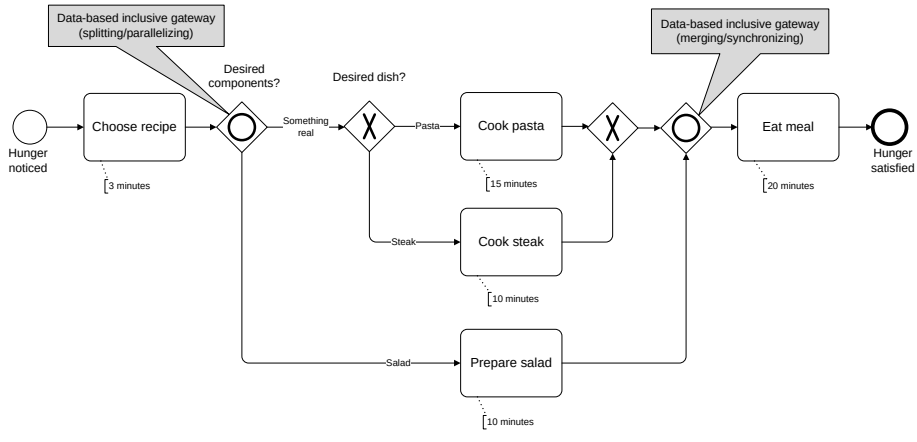


FIGURE 2.12 The OR gateway enables the compact representation of complex path variants.

We can use OR gateways to combine paths too: Depending on whether we want to eat just a salad *or* something real, or a salad *and* something real, we have to wait either for one token to arrive (merge) or for both tokens (synchronize) before we can eat. Note the difference between this and figure 2.11 on the preceding page, however. In the version without the OR gateway, we could have resolved not to prepare anything (neither salad nor something real), but we ate after this decision. The OR gateway excludes this absurdity. We have to decide at least in favor of a salad and/or something real, otherwise the token gets stuck in the gateway. Strictly speaking, the BPMN specification determines that a runtime error occurs in such a case, and that's important when it comes to technical process implementation.

In practice, handling OR gateways is not as simple as these examples imply. It's easy to understand that progress depends on waiting for another token to reach an OR merge. It can be harder to trace the synchronization rules with complex diagrams that sprawl across several pages. Just memorizing the conditions that apply at the OR split isn't a solution.

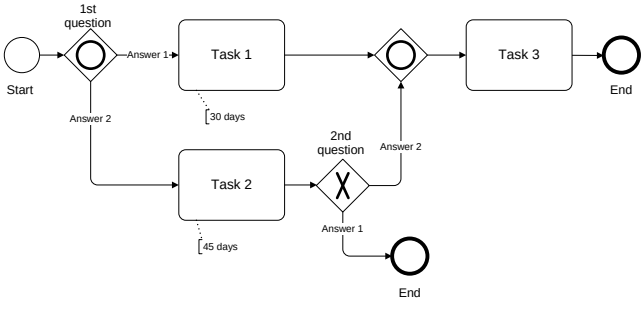


FIGURE 2.13 How long does the second gateway have to wait?

Consider figure 2.13 on the preceding page: whether the OR merge needs to synchronize or not depends on whether the OR split runs through one or more paths. Here's the scenario: The first token reaches the OR merge after 30 days. Because answer 2 applied to the previous OR split too, another token is on its way, and it will stay in task 2 for another 15 days. This task is completed, so it becomes possible that a decision made at the XOR split results in the second token being routed through the answer 1 path, and being consumed by the end event. What happens to the first token at the synchronizing OR merge? *The OR gateway must register that the second token has vanished, and it must forward the first token.*

This could cause problems in three circumstances:

- You come across an OR merge in your process manual on page 10, and you have to rummage through the previous 9 pages to understand what conditions require which waiting times.
- You implement such a process in an organization that makes a person responsible for task 3 but permits that person no control over the process.
- A process engine runs the process and controls the synchronizing behavior. It is expensive to implement such a check, and it is bound to fail. In some cases it may be impossible.

There are a couple of reasons for using the OR gateway —with caution.

Question: Can we model the process as shown in figure 2.14?

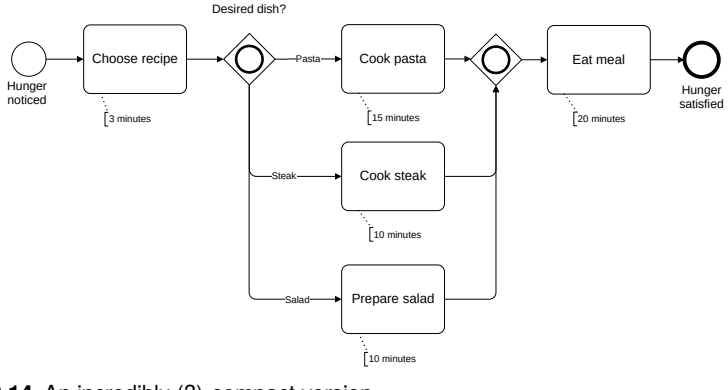


FIGURE 2.14 An incredibly (?) compact version.

Answer: Sure, this makes the model more compact, but it changes the meaning. This process model produces the following outcomes:

- We eat only pasta.
- We eat only steak.
- We eat only salad.
- We eat pasta and salad.
- We eat steak and salad.
- We eat pasta and steak.
- We eat pasta, steak, and salad.

And the last two outcomes aren't what we intend!

2.3.4 Default flow and getting stuck

There's another aspect to working with XOR and OR gateways. (To simplify matters, let's set the salad aside for now and focus on real meals.) What happens if we want neither pasta nor steak? In the previous models, this situation meant that our token could never get beyond the XOR split for "desired dish." According to the BPMN specification, that "throws an exception." In other words, a runtime error occurs.

Don't get angry because we are talking about throwing exceptions! (We'll come back to this issue and show why it doesn't concern only IT.)

The so-called default flow protects us from runtime errors. We indicate the default flow with the small slash shown in figure 2.15. The principle behind default flows is simply that all outgoing paths are examined; when none of the other paths apply, the process uses the default. Don't mistake the default flow for the usual flow, however. The symbol does not mean that the default applies most of the time. That's a different question.

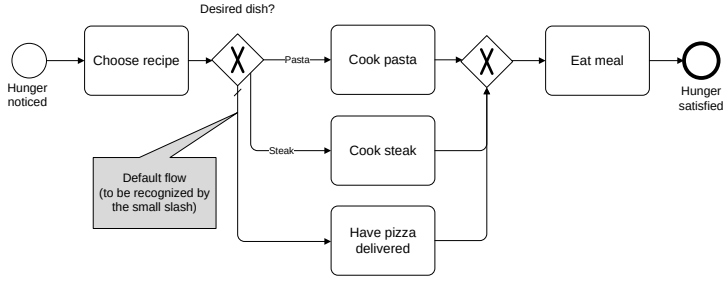


FIGURE 2.15 The default flow.

Our BPMN Etiquette

You don't have to use a default flow, of course. You can draw a normal sequence flow instead and label it "other" or whatever you like. We use the default flow any time there's a risk of getting stuck, and we want to avoid disruption to the organization. If a diagrammed decision has Yes or No outflows only, risk is zero; more complex decisions present greater risk.

In our models, default flows help us to determine if we have limited the risk of getting stuck. In terms of aligning business and IT goals, this is certainly good business practice.

2.3.5 Complex gateway

The complex gateway is a category apart. While it isn't used often, there are situations that justify its use. An example: we want to order pizza. We peruse the menu of our favorite supplier, but just for a change, we also look on the Internet. Once we find something we want to try after researching *both* sources, we order the pizza.

How can we model that? The attempt shown in figure 2.16 results in ordering the pizza only after the research in *both* sources completes. In figure 2.17, neither is an option: Based on the token concept, we would execute the "order pizza" task twice. (Remember the test question in section 2.3.2 on page 20?) Nor does the OR merge in figure 2.18 solve the problem: When a token arrives at the OR merge, the process waits for corresponding tokens that may never get there. The OR merge behavior is thus the same as an AND gateway.

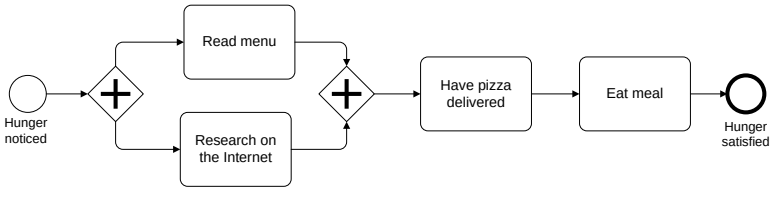


FIGURE 2.16 Pizza research with AND merge.

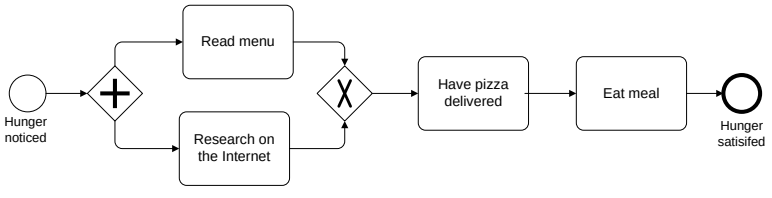


FIGURE 2.17 Pizza research with XOR merge.

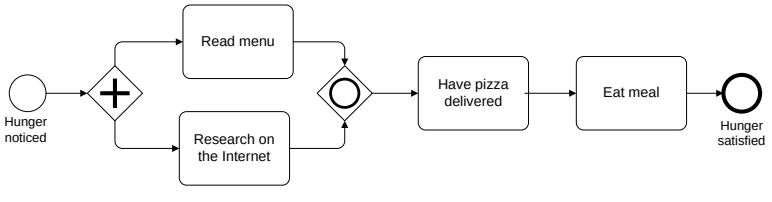


FIGURE 2.18 Pizza research with OR merge.

The solution is the complex gateway combined with an annotation, as shown in figure 2.19. As soon as one of the two tasks completes, the complex merge sends the token to the "order pizza" task. When the next token reaches the complex merge, it is consumed. It vanishes.

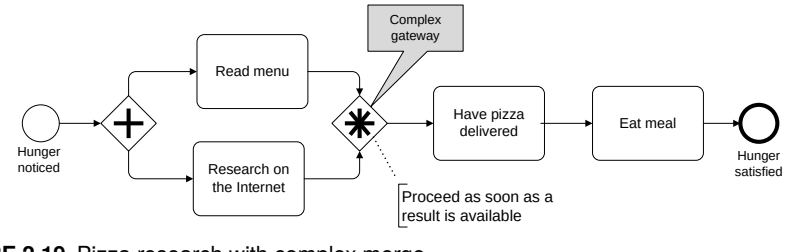


FIGURE 2.19 Pizza research with complex merge.

Here's a similar situation: Assume we execute four tasks at once. There's a fifth task to execute once three of the first four tasks complete. For example, we ask four friends what pizza place they want to order from. Once three friends have offered opinions, we decide. We can model our synchronizing behavior with a complex gateway. (See figure 2.20.)

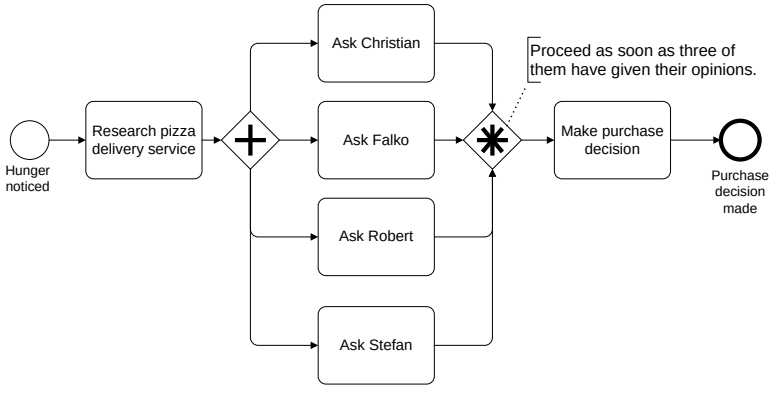


FIGURE 2.20 Using complex gateways to realize *m* out of *n* merges.

In principle, a complex gateway also can be applied as a split —to summarize several different gateways in one symbol to save some space, for instance. The OR split from the process in figure 2.14 on page 25 could be replaced with a complex gateway by writing the split semantics in an annotation. That doesn't really make sense, though, and we have never used the complex gateway as a split nor seen it used in any practical model.

2.4 Design process paths without gateways

Some people don't like gateways. They think gateways make process diagrams too comprehensive or even inflated, and they would rather do without all those diamonds. While gateways *are* optional —you can instead model the logic of the XOR, AND, and OR gateways directly with the tasks —you have to be careful. It's rare that you can eliminate gateways entirely.

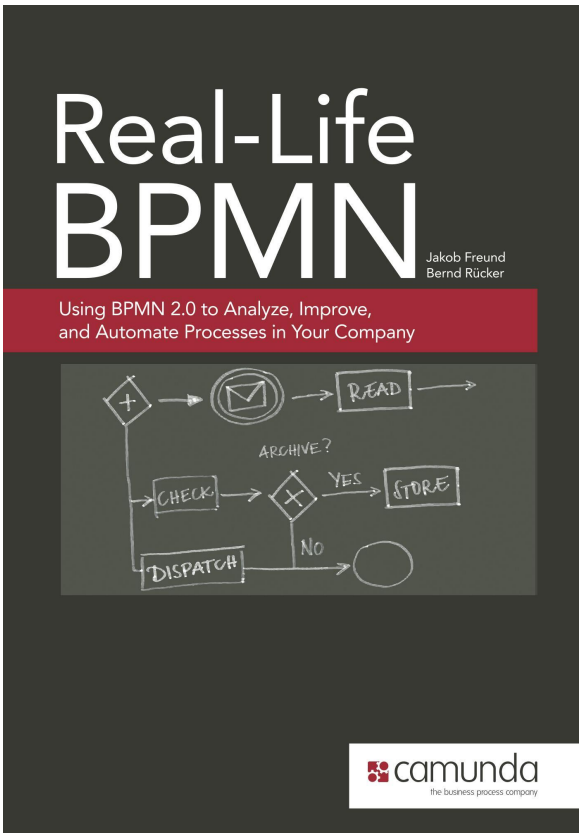
Figure 2.21 on the next page illustrates an alternative to the OR split as well as to the XOR merge. The upper and the lower process models are equivalent, but the upper model shows two flows routing directly to task 4. It also represents the OR split with conditional flow symbols: the small diamonds connected to task 1. Conditional flow symbols may connect only to tasks or subprocesses, and only as outlets. They may not be used with gateways or events.

If you read section 2.3 on page 18 carefully, you likely see the problem with this: If only one of the two conditions applies, everything is okay, but if both apply, they generate two tokens in the OR split and so trigger task 4 twice thanks to the XOR merge. This isn't

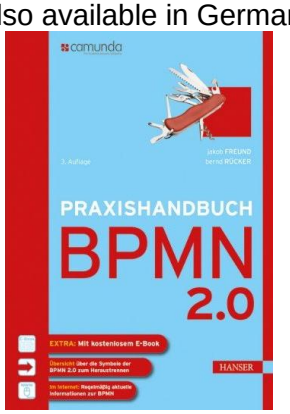
This is a sample of the book

Real-Life BPMN

Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company



Also available in German



and Spanish.

Buy it online: http://www.amazon.com/Real-Life-BPMN-Analyze-Automate-Processes/dp/1480034983/

Sample Review on amazon.com:

★★★★★ **Great Introduction to BPMN and practical examples how to use** October 24, 2012

By Joerg

Format: Paperback

I am happy to see the book now being available in English as well. The German version - 3rd Edition already - which I am referring to is the most comprehensive and practical BPMN introduction I am aware of. It provides guidelines for all 4 levels of BPMN from 1 Strategic (process overview) to 4 Implementation (technical implementation).

I've not counted the number of examples but there seem to be hundreds, and what I like especially are the tables with recommended vs. not recommended modelling.
The diagram related FAQ are really great too, example no merging parallel gateway how long will the process instance live etc.

What you clearly can see from the book is that the authors know their stuff not just theoretically but practically as well and that the book contains many lessons learned from their work consulting customers on process analysis and implementation. Camunda being part of the Activiti BPM core team and branching there own Camunda Fox engine driving the BPMN feature development is being reflected in the content as well.

Focus on real life examples and needs not on the ivory tower.

Working in a multinational non German environment I am more than happy to be able to provide some good BPMN reading for our business analysts, system analysts and BPM developers.

7 Tips to get started

7.1 Develop your own style

We have explained BPMN and illustrated its hands-on application based on our framework. Now it's your turn. You have to consider what you want to do with BPMN and develop your own procedures and associated conventions. You can resort to our framework, which —deliberately —allows enough room for creativity. So familiarize yourself with BPMN and then decide when you want to apply which symbols and constructs.

It is best to develop your BPMN style not in an abstract way, but rather by working with it, with actual processes from your company. Start with processes that are relatively straightforward, for example:

- Making a vacation request
- Receiving invoices, including verification and release
- Ordering office supplies

Yes, you could start by jumping on your core processes, trying to survey and document them completely. These are wonderfully suitable as long-term undertakings, and maybe you could benefit from them in your next life. For starting out, however, we cannot recommend these as BPMN projects.

For starting out, you should prefer a compact and easily manageable support process, and you should model it at level 1, that is, with a focus on operations and results. When ordering office supplies, for instance, an employee urgently needs something. She reports this need, the purchasing department procures the item, and the employee receives it and is happy. Now proceed to level 2, where you can go into the detailed operational handling, perhaps taking into account that the purchasing department won't order the items immediately. Instead, purchasing may accumulate all office product requests into a larger order. Then, from your level 2 model, you can derive a simple work flow for level 3 and even implement it. Voilà, you have just automated a process that is transparent, efficient, and agile. You're ready to tackle a more difficult process, like those invoices.

The devil is always in the details, even with relatively simple processes. And you have to be aware that these processes often do not contain all the possible problems you will encounter in your core processes. The bottom line is still what we explained in the introduction: BPM works best step-by-step, and when you have a map and compass.

7.2 Find fellow sufferers

You are not alone. Many people in many organizations already have experience in BPMN. Find and contact them. Exchange information with them. There are some interesting platforms for this purpose:

- **BPMN.info**: Our knowledge portal provides compact information on the pros of BPMN, the organizations working with BPMN, tutorials for beginners, and an overview of available BPMN tools.
- **camunda workshops**: We regularly hold practical workshops on selected BPM issues with customers and other interested parties. They are usually free to attend, because they provide us with the opportunity to strengthen connections with our customers. If you are interested in attending such a workshop, send us an email: info@camunda.com.

If you become a member of a network like the ones listed above, do yourself and everyone else a favor:

Be generous!

A community is not just a chance to extract knowledge from others for free. If all you do is ask questions without providing any answers, or criticize without offering ideas for improvement, eventually no one will want to talk to you. Benefit from the ideas and experience of others, of course, but share your ideas and experience as well. To give is not only more blessed than to receive, it also creates more success. Does that sound esoteric? Maybe, but it works!

7.3 Get started

Thank you for reading our book. We hope it will help you to improve the processes in your organization. Ideally, good processes free everyone to focus on the things that truly create value. If our book helps you to do that, then we have achieved our goal.

Do you have feedback about the book? Do you have ideas for improving our BPMN framework? We are eager to hear from you. Please email us at bpmn@camunda.com. Any maybe we'll meet up in some of our BPMN classroom trainings (see http://www.camunda.com/bpmn/)!

We've kept you long enough. Go get started!

Bibliography

[DM08] DECKER, Gero ; MENDLING, Jan: Process Instantiation. In: *Data and Knowledge Engineering (DKE). Volume 68* (2008)

[Eur09] EUROPEAN ASSOCIATION OF BPM: *Common Body of Knowledge for BPM*. Schmidt (Götz), Wettenberg, 2009

[Kön07] KÖNIG, Dieter: *Web Services Business Process Execution Language (WS-BPEL 2.0). http://events.oasis-open.org/home/sites/events.oasis-open.org.home/files/Koenig.ppt*, 2007

[Obj09] OBJECT MANAGEMENT GROUP: *Business Process Modeling Notation (BPMN) Version 1.2. http://www.omg.org/spec/BPMN/1.2/PDF*, 2009

[ODHA06] OUYANG, Chun ; DUMAS, Marlon ; HOFSTEDE, Arthur H. ; AALST, Wil M. d.: *From BPMN Process Models to BPEL Web Services. http://bpt.hpi.uni-potsdam.de/pub/Public/MatthiasWeidlich/bpel2bpmn.pdf*, 2006

[WDGW08] WEIDLICH, Matthias ; DECKER, Gero ; GROSSKOPF, Alexander ; WESKE, Mathias: *BPEL to BPMN: The Myth of a Straight-Forward Mapping. http://bpt.hpi.uni-potsdam.de/pub/Public/MatthiasWeidlich/bpel2bpmn.pdf*, 2008