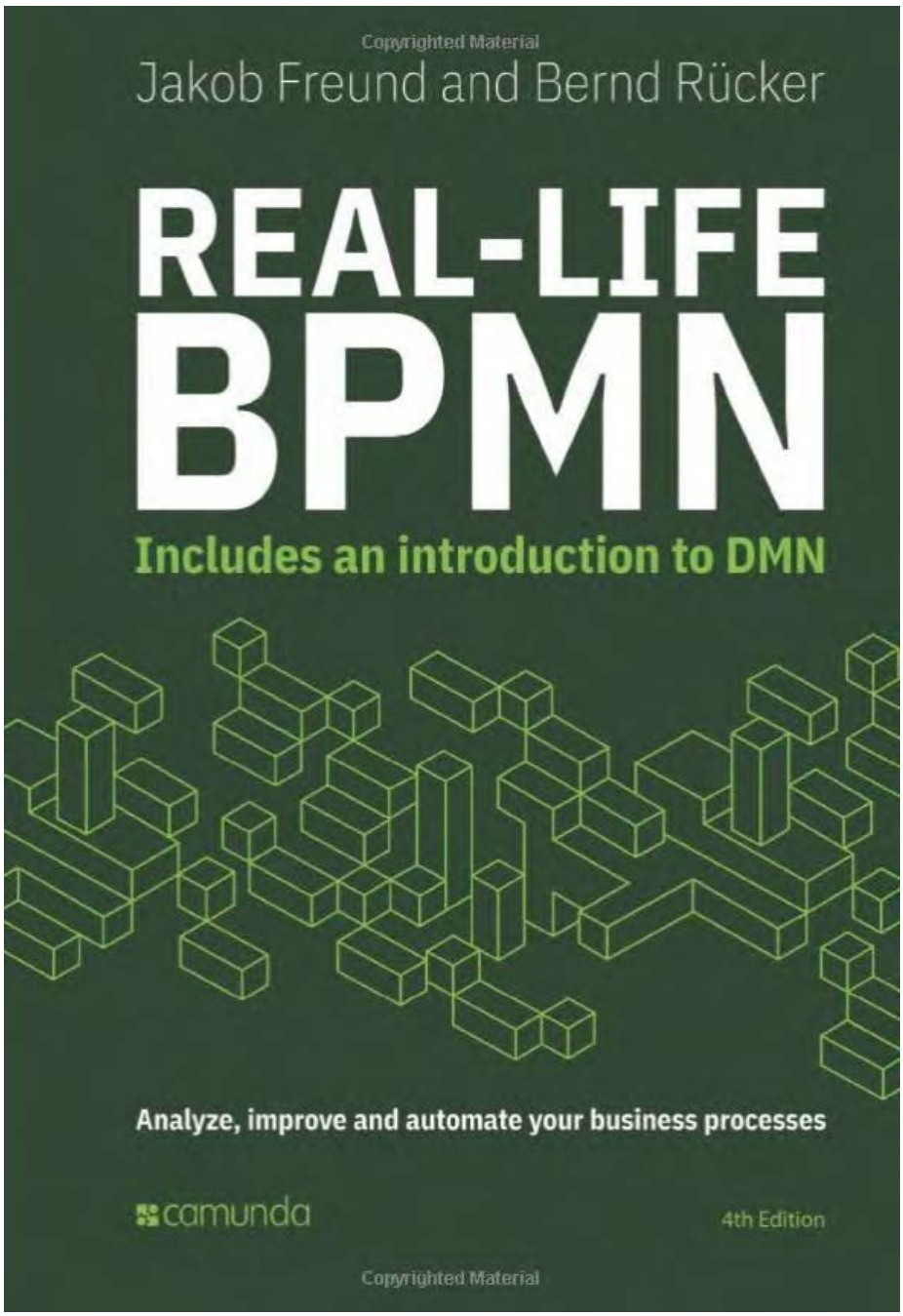
The following is a free excerpt from the book Real-Life BPMN. It can be used as study material in preparation for the training for certification according to OCEB.*



*Disclaimer: Please note that this excerpt does not represent the complete study material required in preparation for the training for certification according to OCEB.

Real-Life BPMN

4th edition

Jakob Freund
Bernd Rücker

Real-Life BPMN

4th edition

Using BPMN and DMN to analyze, improve,
and automate processes in your company

Jakob Freund, Bernd Rücker
Founders of Camunda
www.camunda.com

This fourth edition in English is based on the successful sixth German edition.
Also available in Spanish.

Editing for the English-language edition of this book was provided by James Venis of Lakewood, Colorado, USA, with assistance from Kristen Hannum (1st edition), Jalynn Venis (2nd edition) and James Simmonds (3rd edition).
www.jvenis.net

ISBN-13: 9781086302097

Contents

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, it is helpful to gain some understanding of 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 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 that need 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 for this view strongly: 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. 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. Likewise, 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 overly-simple ones contain only the most painfully obvious elements. They may be useful for marketing presentations, but not much else. On the other hand, overly complex models work so

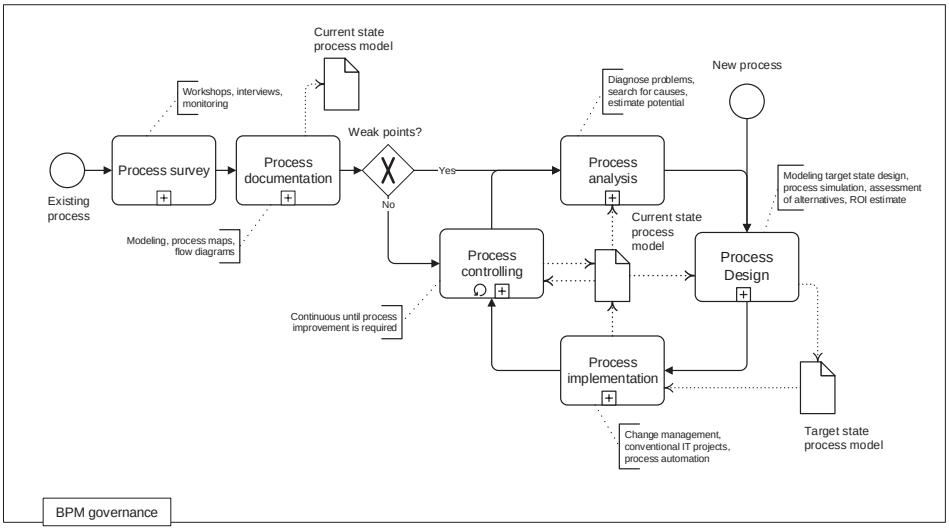


FIGURE 1.1 The Camunda BPM life cycle.

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 ourselves.

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 run through the life cycle 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 e.g. workshops or 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. Applying it requires coordination of the *triad*: 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 creditworthiness 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 known as a *workflow 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 workflow engine accesses the credit rating agency, transfers the pertinent details, and receives the report. Since the report is positive, the 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 *workflow engine*, which executes an executable process model.
- The workflow 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 workflow 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 following page illustrates these principles.

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

- The workflow engine specializes in representing process logic. The services it provides would have required extensive programming in the past; using a workflow 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 workflow 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 workflow engine to make the whole more versatile.
- As the workflow 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 work-

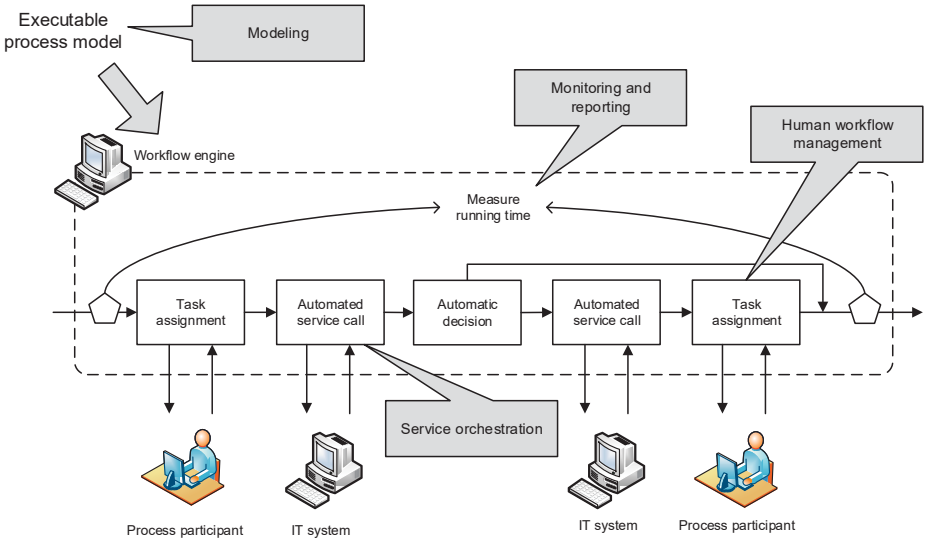


FIGURE 1.2 Process automation with a workflow engine.

flow 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 using a workflow engine, but there is a fourth justification: The workflow 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.

1.2 The BPM standards

Our focus in this book is on BPMN as a standard for modeling and automating processes. But there is at least one standard that relates closely to BPMN, and complement BPMN well. This is Decision Model and Notation (DMN) for managing decisions.

In this section, we provide an overview of the standards, and then we describe how they can be used in combination.

1.2.1 Workflows with BPMN

Initially, BPMN stood for *Business Process Modeling Notation*. The first version was developed predominantly by Stephen A. White from IBM before it was published in 2004 by Business Process Management Initiative (BPMI). From the outset, the aim was to provide a standardized graphical process notation that also could be used for process automation.

In 2005, Object Management Group (OMG) took over BPMI along with the further development of BPMN. OMG is an important institution in the world of IT. It is known especially for its Unified Modeling Language (UML), a modeling standard for software design. The merger of BPMI with OMG was also the beginning of a global triumph for BPMN, as it provided incentive for many companies to switch.

In February 2011, OMG released the current version, BPMN version 2.0. We were able to play a part in that. Version 2.0 came with a new definition of BPMN: *Business Process Model and Notation*, because not only did version 2.0 define the notation but also the so-called *formal metamodel*. Then in September 2013, BPMN was published as an ISO standard by the International Organization for Standardization (ISO) under ISO/IEC 19510:2013. Since then, the notation has been deliberately kept stable, because a proliferation of versions would destroy many of the advantages, because then, for example, each tool would support a different version, or books would have to take account of many version differences.

By now you may be wondering what this mysterious BPMN is in a material sense. BPMN is a specification. It exists in the form of a PDF document that you can download free from the OMG [Obj09] website. Whereas the specification document for BPMN version 1.2 was about 320 pages, version 2.0 has expanded to 500 pages. The documents define all the BPMN symbols, their meanings, and the rules on combining them.

With version 1.2, BPMN had not yet defined all the technical attributes necessary for direct execution of BPMN models in workflow engines. This led to several unfortunate attempts to convert ("map") BPMN models to BPEL models (see section 6.2.3 on page 172). BPMN version 2.0, however, made direct execution possible. That's an important factor in terms of the use of BPMN models. Another important factor is standardization, which offers the following advantages:

- You become more independent from certain BPM tools when you do not have to learn a new notation every time you change tools. Today more than 100 BPMN tools exist; many of them are free.
- There's a good chance that your partners in other companies (customers, suppliers, consultants, and so on) are familiar with BPMN and can therefore understand your process models quickly.
- When hiring new staff, it's likelier that more of them already can read or generate your BPMN process models.
- When universities and private companies invest time and money to develop additional solutions based on BPMN, this is to your benefit as well. Our BPMN framework, which we present later, is an example of this commitment —we never would have developed it if BPMN were not a standard.

1.2.2 DMN for rule-based decisions

DMN is short for *Decision Model and Notation*. Like BPMN, it is administered by OMG. DMN is the newest of the three standards. Version 1.0 was released in September 2015. Version 1.2 is the current version when releasing this edition of the book.

A *decision* in the DMN sense means deriving a result (*output*) from given facts (*input*) on the basis of defined logic (*decision logic*).

Unlike BPMN, DMN is not about activities or processes. DMN works in an operationally similar fashion: decisions can be modeled by a business user and then executed by a decision engine. Another similarity to BPMN is that the DMN standard specification contains both a written description of the notation and an XML-based formal metamodel.

The DMN standard offers different ways to model decisions. The most popular way is the *decision table* described in section 5.2.1 on page 151. Within decision tables you must define the specific conditions needed to determine a result. The definition has to be understandable and implementable on a technical level —BPMN users will recognize how this corresponds to BPMN —and it is why we use a formal language called Friendly Enough Expression Language (FEEL). FEEL is part of the DMN standard, and we introduce it in section 5.2.4 on page 159.

Often, complex decisions are made up of comparatively simple decisions. The Decision Requirements Diagrams (DRDs) described in section 5.2.5 on page 162 help us to dissect complex decisions into their components and make them clearer.

Similar to BPMN, the value of DMN peaks when modeled decisions are executed by a compatible decision engine. This offers the following advantages:

- **Transparency:** Everyone can easily understand how decisions are being made. This knowledge is no longer buried either in the heads of certain employees nor in barely intelligible application source code.
- **Traceability:** Every decision can be logged automatically by the decision engine. It is possible to trace why certain decisions were made.
- **Flexibility:** The decision logic can be adapted more readily. It does not have to be rolled out accompanied by lengthy training or documentation; it can just be deployed. In this regard, DMN is slightly superior to BPMN because changing BPMN diagrams intended for execution by a process engine can be too risky for a non-programmer. (This may be hard to appreciate —after all, how hard can it be to add, move, or delete a few symbols? True, but the technical process is only one part of an entire application architecture that can be affected by the unintended consequences of small changes.) Something similar can happen with a DMN decision table, but the consequences are more easily recognizable and, unlike in BPMN, there are no technical attributes behind the symbols that have to be maintained. It is thus more easily possible for the business department to design or adapt software solutions independently of IT.

Activities and decisions are closely entwined in business processes. BPMN version 2.0 defined a *business rule task* more than four years before the first version of DMN. Even then it was assumed that when completing processes, rules would be assessed constantly as part of making decisions. The term *decision management* was not common at that time, however; we spoke instead of *business rule management*, which explains the description of that task type in BPMN.

1.2.3 Structured vs. unstructured workflows

BPMN focuses on business processes, but there is an important limitation: There are some processes that are poorly suited to modeling or automation in BPMN. These are the *unstructured processes* —processes that do not always take place in a predictable and repeat-

able way. An example of an unstructured process is that of a doctor coming upon the scene of an accident with injuries. She is unlikely to work through a BPMN diagram but instead will quickly plunge in, making decisions based on her knowledge and experience, of course, but also in reaction to the chaos of the scene. We could draw other examples from practically every sector or industry, though many are less obvious.

This is why the CMMN standard was invented alongside BPMN. CMMN is short for *Case Management Model and Notation*. OMG published CMMN version 1.0 in March 2014 and the current version when writing this edition is 1.1. We introduced that notation in some detail in the 3rd edition of this book, used it a lot when working with customers and also added support for it in our software platform. We gave CMMN two years to take off, but, within that time, we experienced limited value from CMMN in our projects, especially if you compare it to BPMN or DMN. One observation was, for example, that most logic in CMMN models was hidden in complex rule-sets if certain activities are possible, impossible, mandatory or unnecessary. These rules were often expressed elsewhere and also not represented graphically. Exaggerating a bit, the CMMN model becomes kind of a graphical bullet point list. So we decided to remove CMMN from this book to not confuse anybody that just embarks on their BPM journey. Instead we want to emphasis how to tackle unstructured processes with BPMN in section 4.5.5 on page 142 and point out the limits of this approach.

Close your eyes (metaphorically speaking) and imagine that you are hosting a workshop to design a business process. You have a room full of people who have a stake in the process, and your mutual goal is to come up with a BPMN process model. You start with a manageable circle of participants, and you ask them what the first task should be.

The answer to your question depends, they tell you, and they proceed to pepper you with an entire list of conditions. It seems that you will have to model the evaluation of conditions first, and you'll use a gateway with many possible paths leading out of it.

During the course of the meeting, participants also point out the frequent need to jump back within the process and to repeat a previous task. While it is easy enough to represent such jumps in BPMN, if they have to be represented for more than half of the tasks, your model quickly starts to resemble a bowl of spaghetti. There are two ways out of this mess:

1. You explain that they will have to start working in a more structured manner, with fewer exceptions, deviations, backtracks and the like. This will limit their flexibility when acting within the process, which may frustrate employees and customers alike. On the other hand, the process will become predictable, repeatable, and less dependent on the implicit knowledge of the humans controlling the process.
2. You accept that every case may be different, and that this process cannot be structured homogeneously. You need to ensure that the people working on cases have enough latitude to make use of all their knowledge and experience. BPMN might be at its limit here and you need to find another way of approaching a solution. CMMN might be an option, but proprietary products might be as well. Very often these problems boil down to individual software, groupware, Trello- or IFTTT-alike tools or case management solutions.

BPMN assumes a clear order, a basic sequence in which the tasks are expected to be carried out. But of course there are possibilities to define branches, backflows, and reactions to events. In our projects, these possibilities are often sufficient to properly model partly

unstructured models. We will further examine this in section 4.5.5 on page 142. In the real world, an entire process seldom fits a completely structured or unstructured pattern. More commonly, there are some structured parts within a process —and these can be captured in BPMN —as well as some unstructured parts for which you will need other possibilities.

So let's get going and look at our first example, including some flexibility for human intervention.

1.3 First example

Our scenario comes from the insurance industry. It is simplified, but it represents the kinds of real-life situations we have encountered repeatedly. Note that the models used here are more than theoretical constructs or documentation; they are executable by engines, one of which is our own product, Camunda BPM. Camunda BPM allows you both to model and execute models in BPMN as well as DMN.

If you have no experience with BPMN, or DMN, the following may seem like a forced march through notational language you don't yet know. To help, we have added cross references to the sections of the book in which we detail the notational elements.

Let's get started.

Suppose you want to take out car insurance. Nowadays your first stop is the Internet, where you compare offers and decide on a provider. You visit the website of your chosen insurance company —in this case, the fictitious *Camundanzia Insurance*. You complete an application form (see figure 1.3) with the following data:

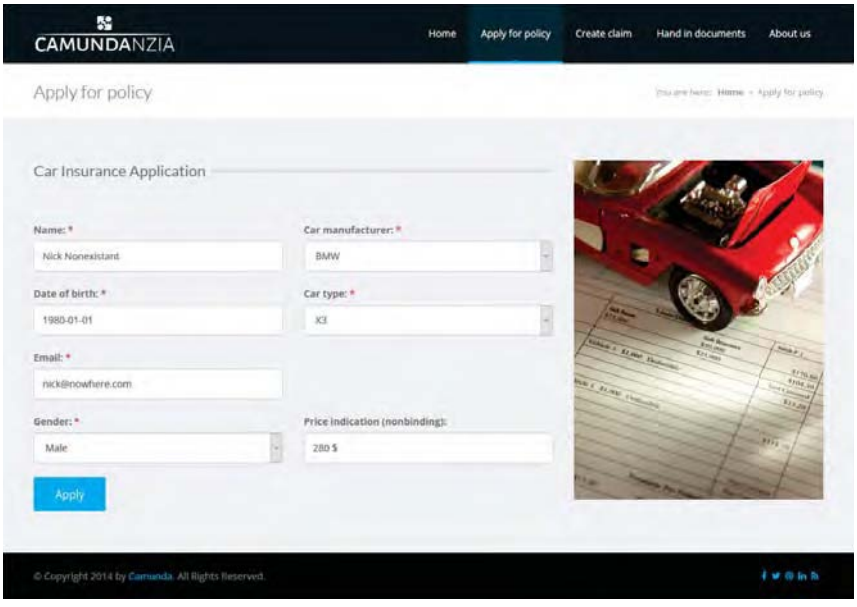


FIGURE 1.3 The Camundanzia website.

- Your date of birth is January 1, 1980. At the time we wrote this, you were 39.
- The manufacturer of your vehicle is BMW.
- The vehicle model is an X3.

You click to submit the form and lean back, eagerly awaiting your insurance policy.

The data from the form immediately creates an instance of a business process at Camundanzia, which has been modeled in BPMN and is technically implemented in Camunda BPM (see figure 1.4). We can tell this from looking at the start event *application received*. The BPMN-compatible workflow engine first mentioned in section 1.1.4 on page 5 now goes to work.

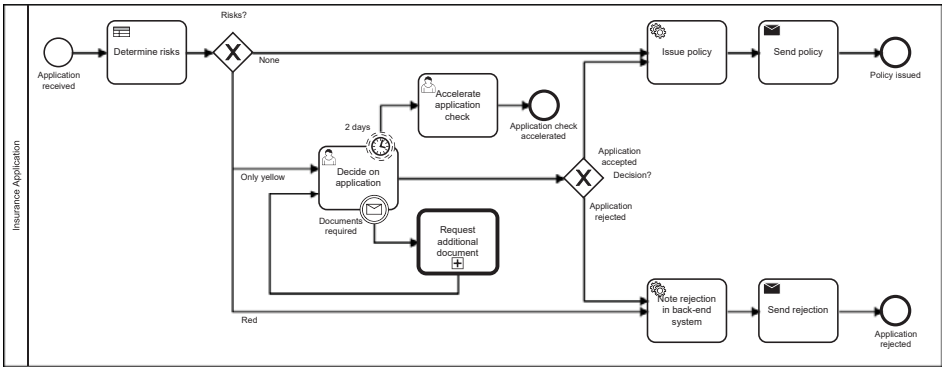


FIGURE 1.4 Request processing in BPMN.

The process starts with the *determine risks* business rule task. On the model side, this task is linked to the DMN decision table *risk assessment* (see figure 1.5) that is now carried out by the decision engine. The decision engine evaluates the input values for applicant age, vehicle manufacturer, and car model. The workflow model transferred these values to the decision engine when the business rule task was carried out.

Risk assessment					
C	*Age*	*Vehicle manufacturer*	*Vehicle type*	*Risk*	*Risk rating*
1	<= 21	-	-	"Beginner"	"yellow"
2	<= 25	"Porsche"	"911"	"Too young and fast"	"red"
3	<= 30	"BMW"	-	"Young and fast"	"yellow"
4	-	"Porsche"	"911"	"Pointless speeding"	"yellow"
5	-	"BMW"	"X3"	"High value vehicle"	"yellow"

FIGURE 1.5 Risk assessment in DMN.

Because you said you drive a BMW X3, rule number 5 applies regardless of your age. This rule states that any driver of a high-value vehicle gets a *yellow* risk rating.

The decision engine feeds two output values —one for the vehicle and one for the risk rating —back to the workflow engine, which continues to execute the process. In the following step we encounter an XOR gateway (see section 2.3.1 on page 28), which decides how the process should continue based on the risk assessment.

If no risks had been recognized, the gateway would have selected the path labeled *none*. This would have led to the *issue policy* service task (see section 2.7 on page 61). The workflow engine would have called Camundanzia's backend system through an interface, and the backend system would have generated a document. In turn, the document would have been fed back to the workflow engine. The next step would have been the *send policy* task, which would have forwarded the document to you.

Your risk, however, fell into the *yellow* category because you drive a BMW X3. The XOR gateway activates the *Decide on application* user task (see section 2.7 on page 61) and waits for some clerk to decide manually, because he got some task in his inbox. The workflow engine is patient, but because of the attached time event (see section 2.6.3 on page 49), after two days of waiting it will launch an escalation. It will initiate another user task called *accelerate application assessment*. The user task could be assigned, for instance, to the team leader of the worker who was responsible for this application assessment.

Let's assume that the office clerk with the expert knowledge processes the application immediately. He asks himself: "Should this applicant be insured despite the high value of this vehicle?" Maybe he needs to be flexible in handling the application, so in our example he can *request further documents*. This leads to a call activity (see section 2.8.2 on page 69) which is linked to a separate BPMN process model.

Requesting additional documents (or other activities you model accordingly) above can be carried out or skipped, that decision is up to the office clerk. In figure 1.6, we see what the clerk's task form looks like.

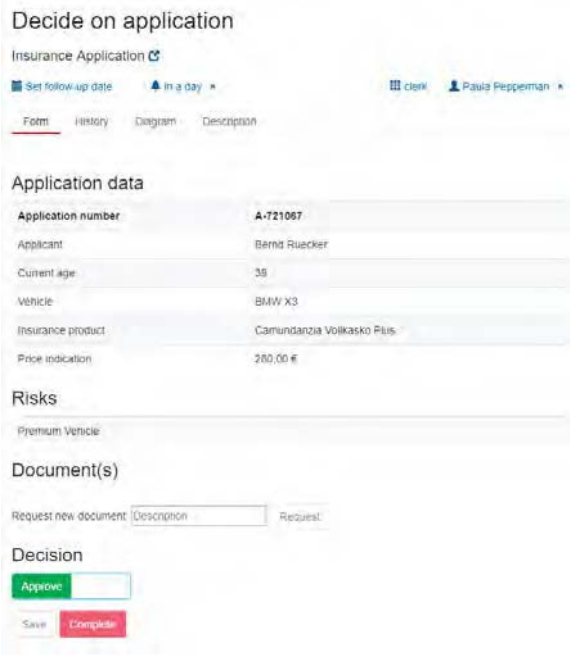


FIGURE 1.6 Office clerk's task form when deciding on the application.

If the end result is to accept the application, the BPMN process continues, that is, the workflow engine reaches the XOR gateway *decision* and selects the path *application accepted*. Now the possibility described above becomes reality —the service task retrieves the insurance policy from the back-end system, and the send task sends it to the applicant by email.

Perhaps only half an hour has elapsed since you sent your application for car insurance. You spent it idly snoozing at your desk, didn't you? But your reverie ends when a new email arrives. You are thrilled at the speed with which Camundanzia was able to issue your insurance policy.

This concludes our example. We hope it was enlightening.

As you can see, each of the BPM standards has a role to play, but they also overlap. A question we get asked is why we need decision tables if business rules can also be represented in BPMN by gateways? We answer this question in section 4.5.6 on page 144.

1.4 Can BPMN bridge the gap?

1.4.1 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 aimed 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 workflow 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 the 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 modeling objective and the expected consumers.

1.4.2 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 1.7) 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 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 from back to front, 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 although, to be honest, 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 workflow 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.

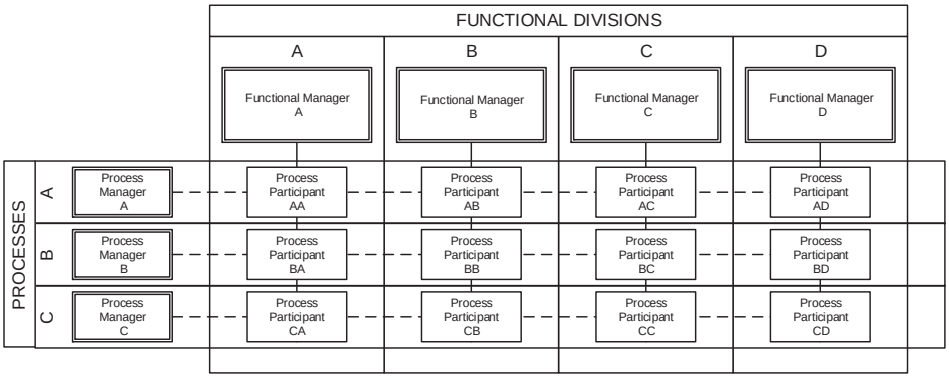


FIGURE 1.7 The process matrix organization.

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.

1.5 A method framework for BPMN

In our consulting projects and workshops, we have introduced a great many people from all kinds of enterprises to BPMN. From that collected experience, we have developed a practical framework for applying BPMN.

This 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 in which 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.

For this edition of the book, we revamped the way we visualize the framework. The following section introduces the new visualization, and then we explain why we changed it. Basically, we now find fault with a widespread approach to process-focused IT projects, and we want to present an alternative that our experience suggests is better.

1.5.1 The Camunda house

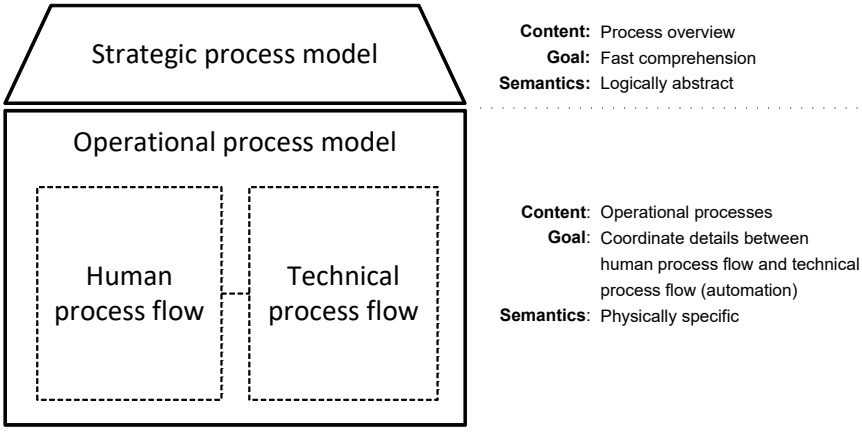


FIGURE 1.8 Camunda BPMN framework.

The *Camunda BPMN framework* in figure 1.8, or *Camunda house* for short, distinguishes between strategic and operational process models:

- **Strategic process model:** The primary target group for strategic process models are process owners and process managers. A secondary group early in the project may include process participants and process analysts. We provide the strategic model as a general,

results-oriented representation of the process because we want to create the quickest possible understanding for an audience that has no special BPMN knowledge. We sketch the process in a few steps, but we don't show errors or variations. See chapter 3 for more detailed information on creating strategic process models.

- **Operational process model:** At this level, we investigate operational details of the actual process. It may contain human or technical process flows, and we model them accordingly. A human flow is handled by a participant while a technical flow is handled by software —preferably a workflow engine. Of course, the human and technical flows can interact. A human may trigger a technical flow in the course of doing his or her work, as in the case of calling a software function. Equally, a technical flow may require a participant to send an email, assign a task, and so on. The human flow thus is triggered by the technical flow. We handle developing human and technical process flows in chapter 4 and chapter 6.

The Camunda house is a purely methodological framework. In other words, it works independently of particular software tools, although certain tool functions may make it easier to apply. We deal with this in section 7.4.2 on page 204.

About half of this book is a detailed description of this 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 applying BPMN practically.

Either way, we look forward to your comments and feedback, not just on this book, but also on the framework itself. By nature it's not a *perfect* approach, and it is subject to constant change and development. With your help, perhaps we can make it better for everyone!

Tooling

We developed the Camunda house specifically to represent projects involving a lone process or a manageable group of related processes. For now, we won't deal with modeling entire process landscapes. BPMN's portfolio does not encompass process landscapes. We *have* modeled process landscapes at a customer's request (we primarily used collapsed pools and message flows as described in section 2.9 on page 78), but we cannot recommend it. If you want a process landscape, you should use a more appropriate tool —perhaps a proprietary one that uses block arrows and rectangles and lots of colors. Of course, you can refine a process landscape with BPMN diagrams by linking the individual elements with flow charts.

1.5.2 The great misunderstanding

This is a confession. We declare ourselves guilty of spreading a deceptive image. The *Camunda BPMN framework* shown in figure 1.9 on the following page was used in a previous edition of this book. Released in German in 2009 and in English in 2012, it was a huge success. Hundreds of BPMN projects used the pyramid depiction of the framework for orientation. A large, international software vendor even included the pyramid in its marketing material. Unfortunately, it resulted in some misunderstandings.

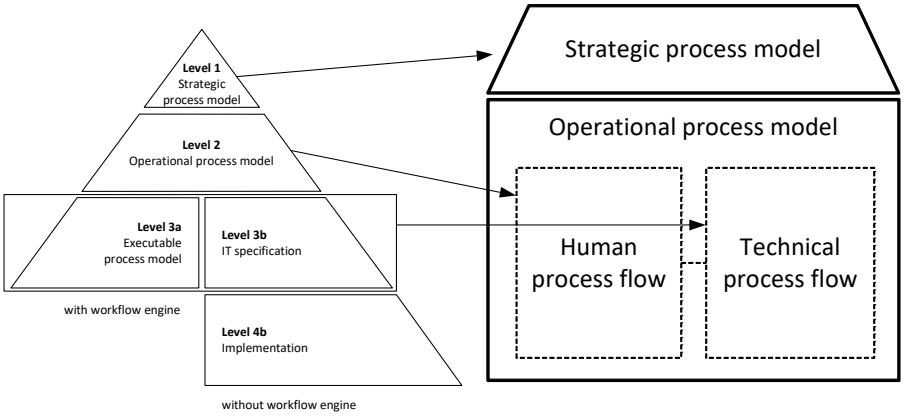


FIGURE 1.9 From old to new: The Camunda BPMN framework.

In the pyramid, we distinguished between strategic, operational, and technical levels. It seems similar at first to the Camunda house, but the Camunda house defines the technical level as a component called *technical process flows* within the *operational process model*, and not as a level of its own. The pyramid put the *operational level* in a position equivalent to what we now call *human process flows*.

This change was necessary because people too often assumed that the technical level was a refinement of the operational level, in other words, that the technical level merely added more detail. In reality, operational-level models (in the sense of the earlier framework) are often *more* detailed than their corresponding technical-level models. For example, think of a simple technical process flow —that triggers a complex manual task —which then requires a complex manual process.

Two related misunderstandings came up.

The first was a perception that the modeling on three levels had to take place in a fixed sequence, that the target-state process must be created first on the strategic level, and then on the operational level, and finally on the technical level. There's no need for that. It often makes more sense to create the operational or technical model first. Doing it this way allows you to develop a clearer understanding of the way process participants will have to do their work before you attempt to summarize or abstract it into a strategic process model. It is, in fact, common practice to conceive the technical and human flows of a process model *concurrently*, for example, in a workshop.

The second misunderstanding related to a strict separation of responsibilities. It was assumed that only the business side would define the strategic and operational levels while only the IT Department would define the technical level. We found this assumption most frequently in enterprises with difficult political situations, where cooperation between IT, operations, and business departments was less than ideal.

We should all understand that even a technical flow represents a *business model*. After all, it describes business requirements. It differs from a classic request document only in that the technical flow anticipates the executable source code —a major advantage of BPMN. The risk in such a strict segregation of responsibilities is that the technical model, while

compliant with requirements, may become incomprehensible and unsatisfactory to the business.

It is a similarly serious matter not to involve IT sufficiently in the design of human processes. To believe that you can define a process purely from an operational perspective and only *then* align the technical implementation with it is . . . naive. Experience shows us repeatedly that operational decisions can and should be influenced by technological realities, either because what the business wants is technologically impossible (or perhaps infeasible for cost reasons), or because the technology can offer solutions that are not on the radar for the people defining operational requirements.

To summarize, you could say that the operational process model belongs both to the business and to IT. As a shared artifact, both parties should share in its development.

What does this thinking mean in terms of our approach to projects? Basically, it aligns with that of agile project organizations: The strict separation of concept from realization is as outmoded as the classic waterfall pattern of development. Most IT projects go better with iterative development, either in sprints within Scrum or otherwise, and it doesn't matter if the project is about process improvement or automation. The business and IT shouldn't work in isolation.

To be abundantly clear: Project participants may need to be shaken out of their comfort zones and motivated sufficiently to work honestly with "the other side." In our engagements during the last few years, the result of our strong encouragement for cooperation always has been the same: massive amazement at how productive a project can be. When IT and the business work side-by-side to define the target-state process at the strategic and operational levels, *including* technical flows, the technical flows can become executable within days or even hours.

As Thorsten Schramm of LVM Versicherung (a large insurance firm) put it during one of our workshops:

"It took only a few days to highly inspire the whole project team (consisting of people from both IT and business departments) for process modelling with BPMN, and right now the first improved processes are already emerging."

Thorsten distills our message nicely. Sometimes, the cooperation experienced within a workshop is just as meaningful as learning the BPMN methodology. BPMN thus can operate synergistically to produce positive change within the enterprise.

1.6 Domains, boundaries and the risk of BPMN monoliths

The microservice architectural style is currently on the rise. For example, in a 2018 survey ([Cam18]) 63% of participating companies had already adopted microservices architectures. The idea is that software is no longer built as large monolithic applications, but rather as a bunch of smaller microservices that focus on exactly one business capability each. Every services is owned by one team that cares about design, development, deployment, operations and maintenance. A microservice has a clear responsibility and API, and

only these are known to the rest of the company Without implementation details. This is contrary to horizontal teams you see so often, like the business analysts, the software developers, the database admins and operation folks. Instead, you will have the "customer onboarding" team, that bundles all these roles and can act on its own.

Splitting up logic into microservices influences business processes and their models in BPMN. There are only rare cases where a business process will be handled completely by one microservice, instead you will see microservices that need to collaborate to implement end-to-end business processes.

If you now think of the Camunda house, this means that several operational processes (in the respective microservices) interact to achieve the overall goal. If we wanted to drive the metaphor further, it would probably be a village consisting of different Camunda houses. The end-to-end process from the customers' point of view would probably be the gate in the city wall and... But let's drop that and look at a small example in figure 1.10.

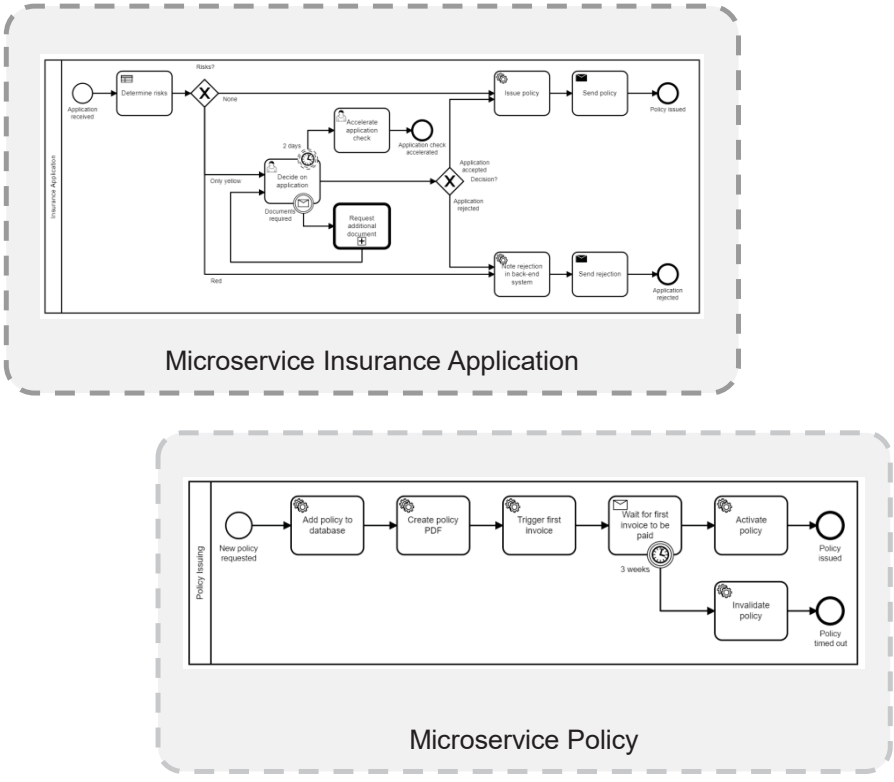


FIGURE 1.10 Several microservices need to collaborate to implement an end-to-end business process. Every microservice has its own local process.

The microservice inbound application is responsible for end-to-end processing applications for an insurance policy. Therefore it includes a BPMN process, which we've already looked at. Now, however, the policy itself is an independent task and will probably be dealt with in a separate microservice. This is perhaps only a facade in front of an existing legacy system. The two microservices must now work together to process a new application.

The challenge, by the way, is usually to determine the boundaries of the services and the exact responsibility of individual services. There is no right or wrong, only more or less suitable. In our example, there are different variants that can all make sense. For example, the policy microservice could send out documents itself to the customer, but that could also be part of the application microservice. However, there may also be a separate service for sending documents. It is important to make a conscious decision and then design the processes accordingly. For this topic we can recommend the literature around domain-driven design. And if you are in need of a distraction anyway, please take a look on the Internet and search for the "bounded context".

At this point we just want to explicitly warn against, as we call it, "BPMN monoliths". A BPMN monolith is a process model that mixes details from different microservices, thus not respecting their responsibilities and boundaries. Such a model does not have a single person responsible for the process and is usually very cumbersome to coordinate because too many stakeholders want to participate. You cannot automate this model directly because it has to be distributed among different microservices.

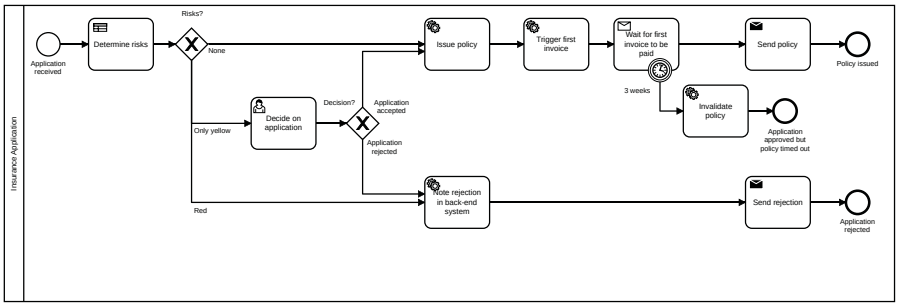


FIGURE 1.11 Antipattern BPMN monolith: This model includes details about responsibilities from other microservices.

Figure 1.11 shows an example of such a BPMN monolith. In addition to the processing of the application, there is business logic of the policy included, like the fact that policies only become valid if the first invoice is paid within a defined period. The application microservice should not know this detail - it only wants to know whether a policy was successful or not - and perhaps how long it has to wait at most.

We know from our own experience that in the heat of a successful modeling workshop these monoliths can be created very quickly, as many details bubble out of the participants very naturally at this moment. Often it is even helpful to understand the overall situation and to allow these models. However, they may not be continued, let alone automated, so they are clearly an intermediate step before the process is sliced into individual parts. When doing so, the microservices boundaries must be taken into account.

And, of course, it can still make sense to design a monolithic system. In this case you can model and execute a BPMN monolith accordingly.

In chapter 4.5.3 on page 140 we will follow-up on this topic again with another example.

expressions. While this is a deliberate *deviation* from the standard, we are not *violating* it because the standard permits extensions. Overall, we find that the extensions make it much easier for Java developers to deal with process definition. Success depends on the individual customer's situation. Most of our clients seem to think it is a worthwhile trade-off.

User tasks

Human interaction takes place in the form of a user task. The existence of user tasks in the process leads to tasks being placed in task management and tasks ending up on a user's task list. Only after the user completes the task does the process continue.

The exact technological connection is often a detail in the implementation of the particular workflow engine, making seamless integration possible. If it is essential for you to be independent of a specific engine, there is generally a standardized way using web services: *WS-HumanTask* (WS-HT). This comprehensive specification defines user tasks in a detailed and powerful way. Aspects such as responsibilities, delegation, escalation, or even meta information for the display can be defined. In real life, however, WS-HT is often too complex to be used easily.

Forms for tasks or other interface components, by the way, are completely left out by BPMN. This is where workflow engine vendors go off in different directions.

6.4 Practical tips

6.4.1 Embedded and decentralized workflow engines

If you want to run your BPMN models on a workflow engine, the question immediately arises: how to run this engine? In the past, the image of the central workflow engine or BPM suite dominated. Various operational processes were operated on this central platform. The main advantage was that the platform only had to be operated once, so that fewer people had to be familiar with it and perhaps licensing costs could be saved.

However, this view does not fit into modern architectures around microservices. In section 1.6 on page 19 and section 4.5.3 on page 140, we talked about BPMN monoliths and showed that an operative business process also has to consider system boundaries.

This also runs through the technical infrastructure, because one wants to grant the individual microservice teams as much autonomy as possible. A microservice is actually defined only by its responsibility and the API. The use of a workflow engine and the selection of the concrete product is an implementation detail that should be left to the team itself. Of course, autonomy in practice usually has its limits, since companies want to avoid a proliferation of technologies and often form competence centers for key technologies such as workflow engines.

But nevertheless the workflow engine is logically part of a microservice as shown in figure 6.6. It is not a central infrastructure or a microservice in itself. This view becomes possible because lightweight workflow engines today, unlike complex BPM suites in the past, can be

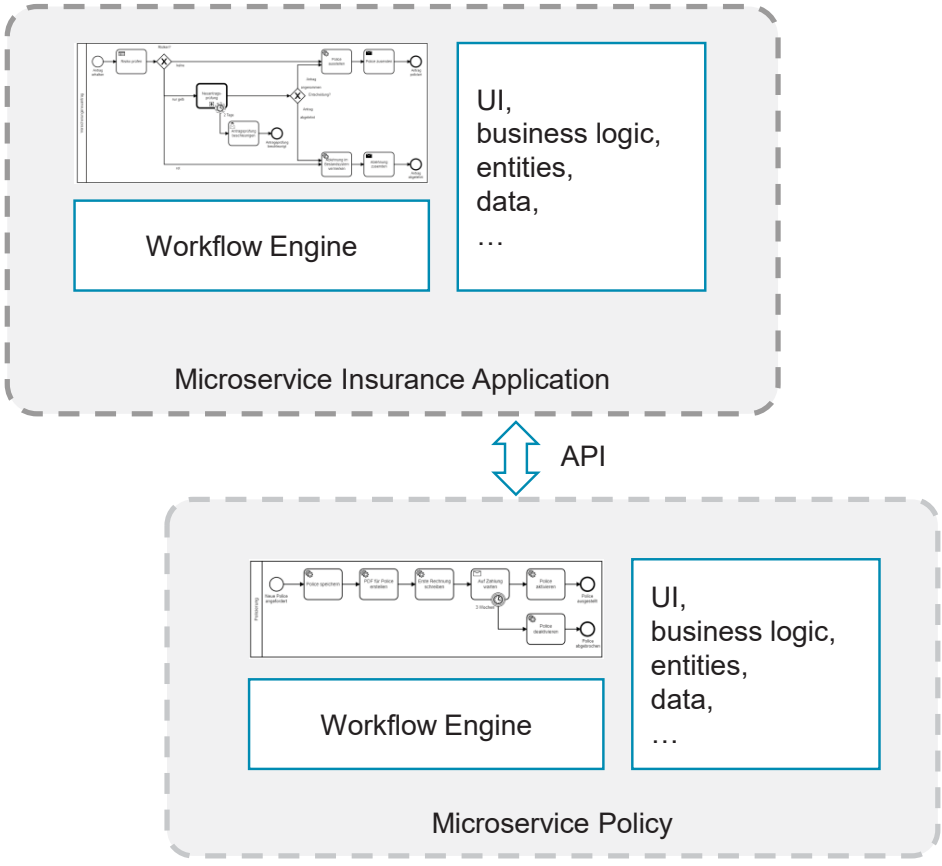


FIGURE 6.6 In microservice architectures, the workflow engine is usually an implementation detail and not a central infrastructure.

operated very easily. In this book we don't want to go further into these details and instead refer to relevant tutorials and how-tos on the Internet, for example about our open source platform Camunda BPM.

This view is essential in microservice architectures. We often find that BPMN tools are rejected by architects or developers because they automatically think of centralized or monolithic BPM systems. And in the world of microservices, which is characterized by decentralization and autonomy, such a tool has no place! So it is all the more important for us to spread the knowledge that a workflow engine can also be used decentrally *in* a microservice. Then it quickly becomes an essential building block in harmony with your microservices architecture.

6.4.2 The low-code trap

Once people get familiar with the idea of a workflow engine, we often run into a problem: the expectation that the magical BPM suite will solve everything. Ideally, we feed the suite

on models developed by the business before IT systems are automatically integrated and human workflow management just works magically. Finally, we create a dashboard of key performance indicators. These enable the business to recognize process problems in real time and to resolve problems for themselves.

This scenario sounds too good to be true, and in practice, it is. Developing process applications is always a form of software development. The promise that this can be taken on by business users in the future is appealing, but it is a promise that cannot be kept. We have seen this over and over. At the end of the day, you need comprehensive wizards and forms to develop process applications in a model-driven way, and the wizards and forms are so complex that they overwhelm the average business user.

What happens then? The company's own IT department is called in to take over development. Unfortunately, the first thing the company's software engineers have to do is figure out the BPM suite. After all, they can't just apply their prior knowledge of programming languages because the technology has been hidden behind the wizards and forms. Ironically, the aim of making development faster and easier is completely thwarted.

During our many years of BPM project experience, we have come to realize that this is exactly where the core problem with the classic BPM suite lies. It is especially pronounced in companies that already have been carrying out software development —in Java, for instance. The following disadvantages arise:

- **High programming effort**: Because software development is vendor specific, in-house developers need to learn and practice the vendor's specific platform. The related expense is not a one-off but instead continuous, with retraining required to maintain knowledge. Existing knowledge (in Java, for example) cannot be applied. In addition, existing tools, techniques, and best practices of software development (unit testing, for example) can be applied partially or not at all. This severely limits the developers' productivity, and as a result, technical implementation is much more complex than it at first appears.
- **Inability to model distinctive parts of a process**: Because the development approach is model-driven, the possibilities for technical implementation are limited. Compare this to a painter: On a blank canvas, an artist can paint a picture exactly the way she imagines. Another artist paints by numbers. He can create stunning images, but only by daubing predetermined colors onto predetermined places. This second artist can create only what was predesigned.

  Painting by numbers in BPM suites is like using off-the-shelf application software. Often it is sufficiently flexible for standard support processes (vacation requests, for example, or invoicing), but the limited possibilities for technical implementation remain insufficient for capturing and implementing core business processes.
- **Inability to integrate into existing IT**: On an operational level, the drawbacks of off-the-shelf applications also apply to traditional BPM suites: they cannot be incorporated easily into existing IT structures.
- **Specialized developers needed**: As we mentioned, a model-driven development approach is inevitably vendor specific, so there's no escaping the fact that you will need developers trained in a particular BPM suite. Such developers are scarce, and if they are not available, it is much easier to find developers for popular programming languages such as Java.

- **Vendor lock-in**: Consequently, there is a strong vendor dependency as the vendor and its partners are usually the only ones with developers with the required level of expertise. This is acceptable in the context of support processes (invoicing, vacation requests, and so on), but it represents unacceptable risk when capturing and implementing core business processes.

To us, it seems that traditional BPM suites are . . . stuck in the middle . . . neither fish nor fowl. They are as unsuitable, compared to existing software development, as off-the-shelf application products, and they don't even offer an out-of-the-box solution for process automation. This dilemma has resulted from an unsuccessful search for compromise between the two extremes and also, to a large extent, to a more academic flow during the last decade: model-driven software development. The modest growth of BPM suite vendors during the same decade seems to confirm our assessment.

Does this mean that using BPM software for process automation is a bad idea? Of course not, but it does mean that the right approach is not as simple as we might wish. In practice, it is a hybrid approach that proves best, where certain parts —the process itself —are model-driven while other parts —complex user interfaces for instance —are developed through classic programming. You therefore have to accept that software development will require software developers in the future. Sounds fairly logical, doesn't it?

In section 7.4.3 on page 204, we present the Camunda BPM platform as an example that supports the hybrid approach and is available as an open source project.

6.4.3 The myth of engine interchangeability

We have indicated that some aspects may be solved differently in different products. The flexibility of the standards gives vendors latitude in selecting technology. Can a process model then be executable on different engines? Usually not. That is, we have not yet seen a model that covers real requirements and was able to do it completely without vendor extensions.

We think that BPMN engine interchangeability is of limited concern. Consider how much time the SQL standard for databases has had to mature. Still, you often want to fall back on features of a product that you know. We find that legitimate. Any requirement for swapping around the workflow engine without having to touch your process solution should not be given too much weight.

Most projects end up buying a clearly simplified process execution, such as using Java shortcuts, while sacrificing interchangeability. This is further catalyzed by the fact that Webservices are not exactly touted as the technology of the future, and many projects prefer other technologies.

In short: Don't make complete interchangeability a sticking point. Just trust that the notation, meaning the BPMN model structures agreed upon with the business (without execution attributes), can survive a switch in engines.

6.4.4 Modeling or programming

Okay, so you're going to use a BPMN engine. The next questions are which aspects will be expressed through technical process models, and which requirements may it be better to continue addressing with classical software development? As is so often the case, there is no generally applicable answer to this question. Even if some of the following factors sound trivial to you, our experience shows that you will do well not to overlook:

- **Technology and architecture**: Depending on the workflow engine and the overall architecture to be used, it can be either simple or difficult to execute certain requirements within a process. Some engines, for example, make it possible directly to integrate Java code, connectors, or script language. Others restrict the possibilities to web services.
- **Existing infrastructure**: Few projects start from scratch. Existing systems and services should be reused or integrated. Processes can be triggered using an existing scheduler, for example, and not the workflow engine itself. These peripheral conditions need to be taken into account.
- **Roles within the project**: It is important to account for existing roles and know-how within the project. Often, there are developers involved who can implement certain functionalities quickly with classical programming but who will need a long time to do so with the workflow engine. On the other hand, there are also projects involving trained process engineers, and they work better with process models than with programming languages.

We often see that once a workflow engine has been procured, there is impetus to use it come hell or high water, and process models quickly follow that are so detailed that you can't see the forest for all the trees. These models do not help when communicating with the business, and they are no easier to maintain than classical programming code. Besides, the IT department will hate the more detailed models, and that's no good to anyone. Success is all about finding the right granularity. Modeled processes are a piece of the puzzle, but they are only one piece.

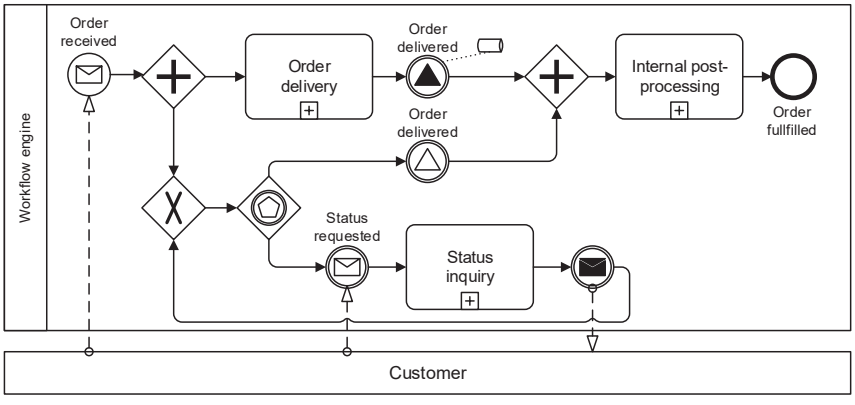


FIGURE 6.7 A bad example of modeling. It models too many aspects of the process.

Figure 6.7 shows an example of going too far. The model shows a customer status inquiry explicitly. At the moment, it doesn't matter if we're modeling this process with a signal

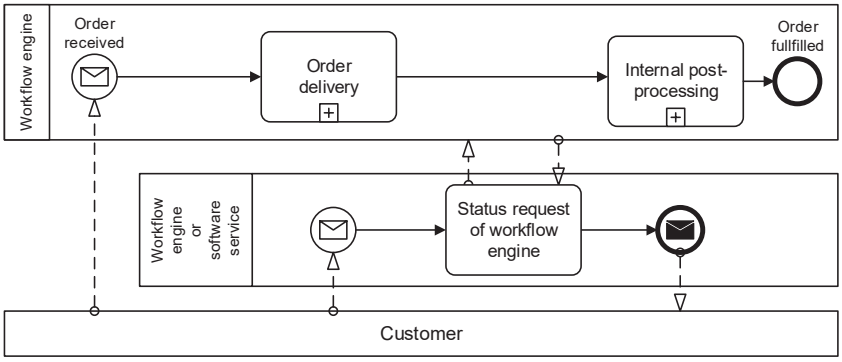


FIGURE 6.8 The model improved by removing status inquiry from the process.

event (as shown), or with a condition event, or perhaps even with a terminating end event; you can see how complicated the process becomes. It probably isn't such a good idea to integrate the inquiry into the actual order process. It would be better to model it in its own process or to use a simple service to query the state of the workflow engine's instance. The only requirement on the engine is that the status must be retrievable. Now look at the modeling in figure 6.8. Whether the inquiry is carried out as a process or as a simple service depends on the architecture.

Hint: Business-IT-alignment

Business-IT alignment doesn't mean that software can no longer be developed conventionally. It does mean introducing the workflow engine and graphical views of technical processes as additional tools. But beware of process models that are too finely granular! Use diagrams in a way that enables business users to understand the technically executable models.

6.4.5 Overcoming technical challenges

In automation projects based around BPMN and DMN, there are some patterns and pitfalls that we would call typical. They depend on the technical environments and the tools used. Vendors address many problems with their features and extensions, so it is hardly worth going into too much detail in this book. Instead, let us give you some examples that will support your understanding of the kinds of problems you may encounter.

Data in the process

An exciting discussion always takes place in a project: How much data do we want to store in the process? Surely we'll need all order data for an order process, won't we? Our recommendation would be the opposite: Store as little as possible —but as much as necessary. It is usually a good idea to store order data in the preceding system for orders. Then, for the process itself, just reference the order number. If the process needs more information to

support a decision at a gateway, for example, it can load the order data using a service call. This has advantages:

- The data is always up-to-date and there is no risk of divergence.
- You do not need to store data redundantly in the process where it may persist in different versions —meaning multiple times.

Exceptions confirm the rule, however, and our experience shows that there are times to keep data in the process:

- The preceding system is too slow. Constant loading leads to a performance issue. In this case, you could use a cache, or you could abuse the engine as a cache.
- You want to keep a copy of the data as it existed at a particular moment in the process and to use it for the duration of the process run.
- Your BPM platform does not make it possible to load data in the background. If this means you have to model a service task over and over just to load the order, your model quickly will become unreadable. As we said, it's about finding the right balance between graphical modeling and programming behind the scene.
- You only store data needed for controlling the process flow, for example, decisions in user tasks not available from other systems.
- Often it may be desirable to store messages in an as transmitted form. This can make it easier to repeat service calls or to identify errors.

Being *ready-to-receive* to receive events

From an automation point of view, intermediate events deserve a closer look. First, consider the semantics of the *ready-to-receive* state we mentioned in section 2.6 on page 43. Remember that, strictly speaking, incoming events are lost if no process instance is ready

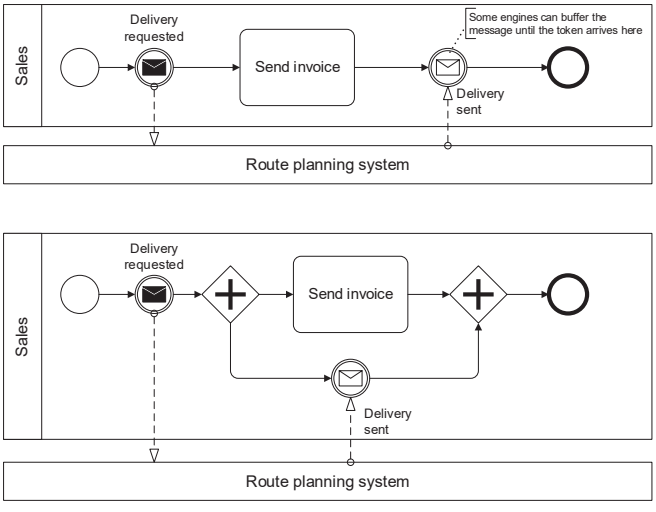


FIGURE 6.9 Never miss a message, even if the process is not ready to receive. A feature of the workflow engine or explicit modeling?

to receive them. Looking at the example in the upper part of figure 6.9 on the facing page, a delivery order is sent to route planning, then the invoice goes out. If *send invoice* is a human task, it may take some time. The *delivery sent* message may come in before the invoice is ready, but we don't want to lose the message.

In technical modeling, pragmatism usually gains the upper hand —and modeling that is completely correct can seem unclear. In principle, workflow engines can solve the issue of an out-of-sequence message by buffering the message. This is an extension of the BPMN standard. If you have this functionality at your disposal, we recommend that you make your approach visible in the model with an annotation.

By the way, buffering a message in an engine usually raises an immediate question: What happens if a process never pulls the event out of the queue? The message sender can't be informed. This means that sophisticated error handling needs to recognize when a process instance terminates the correlated event. As you can already tell, this is no mean feat.

You may have no choice but to restructure the model. You can wait for the event in parallel. As figure 6.9 on the preceding page shows in the lower part, a working alternative is not difficult. This restructuring shouldn't cause a problem for IT, should it? At the very least, the diagram has become more complicated, and that's a problem in terms of business-IT alignment. Even this modeling does not always have to be consistent because in some technical environments the answer may arrive before the process moves toward the receive event. Here, again, the devil is in the details.

Testing processes

Testing processes and decisions is crucial. Executable models equate to source code, after all. They need testing in the same way and, as is now common in software development, the tests should be automated. Automated testing has the great advantage of being repeatable at any time at no additional cost. If you change the process, you can repeat the testing to confirm that you haven't broken anything. Of course this approach means that when you change the model sufficiently, you also have to adapt the test cases. It may seem at first that testing comes at the price of agility, but our experience suggests that you have to take a longer view. Agility is preserved over time because otherwise, at some point, no changes will be possible. You will have become too worried about breaking something.

There are now exciting frameworks for automating tests that make the writing of tests easy to read or that allow test cases to use tables or languages readable by normal people. In addition, *mocks* make it possible for process tests that are not necessarily integration tests. In other words, during a test, services from the surrounding systems are not called up. This then makes *unit tests* possible that can be automated easily and without placing a burden on the environment —as long as the engine supports them.

Processes are typically tested with specific scenarios that illustrate a run-through of the BPMN model.

6.4.6 Acceptance criteria when introducing a BPM platform

If you want to introduce a BPM platform, there are more than technological challenges to overcome. Some aspects we have already addressed:

- The relevance of business-IT alignment through a suitable methodology as suggested in this book.
- The roundtrip, so that all stakeholders understand the executed models that represent the truth.
- The right granularity of models, so that they contain only business-motivated issues.

The approach taken when introducing a BPM platform also is important, in particular the *choice of tool* and the *first steps* undertaken with the platform. In our experience with projects, the approach shown in figure 6.10 has proved itself.

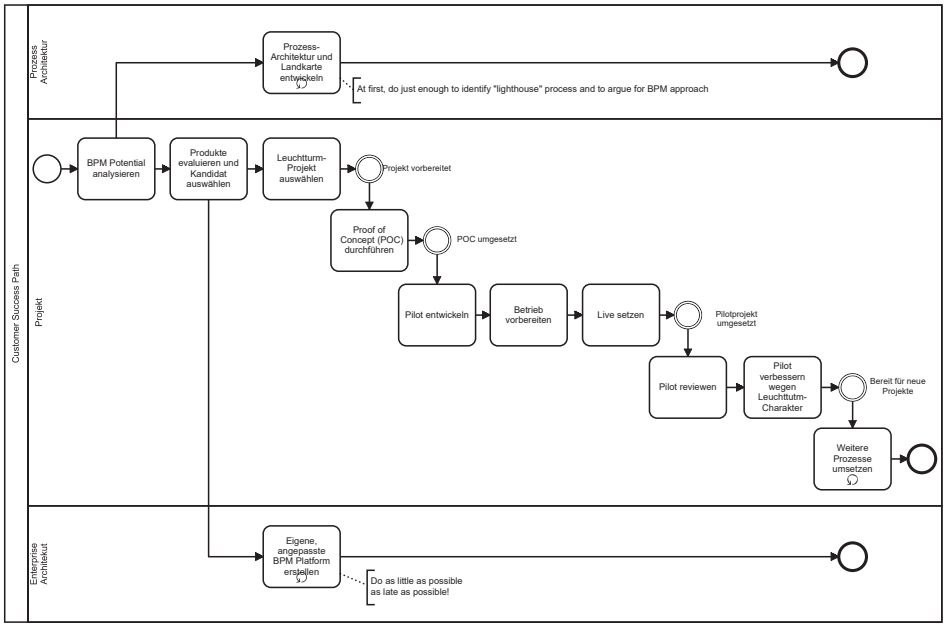


FIGURE 6.10 Best practice for introducing a BPM platform.

It's hard to carry out a sensible evaluation. We regularly see long tables with feature wishes sent to vendors of BPM platforms. The expected answers are *yes, no, maybe* —so that a score can be easily tallied. Of course, the following happens: The vendor who wants to obtain a high number of points replies *yes* as often as possible. That's how he expects to get his foot in the door during an early evaluation phase. Features may hurriedly be introduced just to be able to have more *yes* answers. Whether or not the feature is really a good solution for the fundamental requirements is sometimes secondary. We had a revealing experience of this when an employee of a potential client called us up secretly and said: "You said no at some places where your competitors said yes. I *know* that your software is better than the competition's in this area. Please don't be too honest, otherwise we will end up having to choose the wrong tool!"

As we keep saying, process automation projects are always software development projects. If you accept this, then you can see it is not necessary for *every* requirement to be covered by a zero-code tool. You will remain able to solve many requirements with conventional

software development, and you won't necessarily need to hear yes to every item on your wish list. A *maybe* may suffice when it means *it can be implemented in the project*.

What is appropriately decisive? We would ask: Does the platform offer extensions so that you can implement requirements yourself if needed? Questions of implementation effort remain, of course, and of course, prefabrication by the vendor remains desirable. But the worst thing is a boarded-up platform that leaves you at a dead end. Too rarely do prospective clients inspect the BPM vendor's *philosophy* or *vision* deeply enough to understand if there is consonance with their own.

Once you have determined a suitable candidate, we recommend a *proof of concept* (POC) that implements one of your processes in the planned target environment. By all means, let the vendor assist with this; it will be quicker. In any event, make certain you are present to truly experience how process applications are generated, what effort goes into them, and what kind of know-how is required. Also be alert to tricks that the vendor's consultants may have up their sleeves, tricks they may be applying secretly to hide shortcomings in their own products.

Prior to the POC, you have to be absolutely clear about your goals. Are you seeking to verify if the tool will fit into your architecture and is able, for example, to call up your specific services correctly? Do you have a whole catalog of questions you are seeking to put to the vendor? Are you looking to showcase a wide variety of choices as a way of selling BPM or a particular tool to decision makers within your own company? The POC will be designed differently for different goals. It's therefore essential for all parties to be clear on what the goal is.

Once you decide on a tool, you should move quickly to implement a suitable process. Put this process into live operation! Suitable processes should be:

- Relevant, preferably close to the core processes of the business.
- Not too small, otherwise it will seem too easy, and it will make it difficult to show off the project as a BPM success story.
- Not too big, or you risk taking too long to deliver results.
- Not a political minefield. Unfortunately, this often is the case with business processes.
- Suitable for displaying the advantages of BPM. Remember that the first project will be used to decide on how to proceed.
- Not too demanding on organizational change. Suggesting too much change only generates resistance that can be overcome only with a lot of willpower and endurance. It isn't helpful to expect much change while also introducing a new technological platform. On the other hand, organizational change cannot be completely avoided —and a desire for change is usually what triggers the introduction of the BPM platform in the first place.

In the context of that final point, by the way, it may be a good idea to keep an existing task list and to enter tasks for the new BPM platform into it. This implies that end users (process participants) do not need to use either a new interface nor two different task lists. If you are replacing an existing workflow management tool, the change could be transparent to end users.

In your first project, you should move forward as quickly as possible and not spend too much time on conventions, patterns, or setting up your own process-application blueprint. You will learn so much during the first project that you will be better off to plan time *after*

the project for analysis and documentation of lessons learned and best practices. Best of all will be to review your process once more after the fact. It will surely have a spotlight on it, and you can hope that many others will seek to copy what you did. To summarize: Revising your first project afterwards will cost you a lot less than trying to get it completely right the first time. In principle, this is us recommending an agile approach.

Armed with this, you can venture out confidently to meet your next projects. We hope you have lots of fun automating your business processes! Further information on BPM tools can be found in section 7.4.1 on page 202.