

BPMN/DMN

TOM DEBEVOISE
JAMES TAYLOR

Tom Debevoise
James Taylor

The MicroGuide to Process and Decision Modeling in BPMN/DMN

*Building More Effective Processes
by Integrating Process Modeling
with Decision Modeling*

This book sample contains:

Table of Contents

Chapter 3: Building on the Basics

Appendix A: Commercial
Information

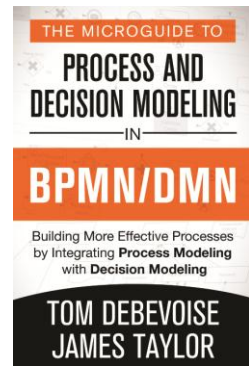
“With this third edition Tom and his coauthor James Taylor bring a new, much needed and appreciated additional perspective to process discovery and design - that of decision modeling.”
George Barlow EVP, AgilePoint Software.

The MicroGuide offers extensive coverage of the important topics related to business process and decision modeling:

- Quick guides to BPMN 2.0 & DMN 1.0
- A comprehensive framework for integrating decision, event and process modeling
- Process and decision discovery through effective requirements gathering techniques

“Net; Net: This book will stand the test of time in that it will be germane to most processes.”

Jim Sinur, Distinguished Industry Analyst.



ISBN: 1502789647

About the Authors

Tom Debevoise is a master solution designer, architect and systems integrator. Tom has process, decision and event modeling expertise in many areas including recent work in IOT, Next Generation Energy and Security C2.

www.tomdebevoise.com

James Taylor is an expert on how to use decision management, business rules & predictive analytics to improve operational performance.

James is also a contributor to V3 of the BABOK® Guide and a member of the submission team for the new Decision Model Notation (DMN) standard.

www.jtonedm.com

Now Available
on Amazon

THE MICROGUIDE TO PROCESS AND DECISION MODELING IN BPMN/DMN

**TOM DEBEVOISE
BLACK PEARL DEVELOPMENT, INCORPORATED**

AND

**JAMES TAYLOR
DECISION MANAGEMENT SOLUTIONS**

**CONTRIBUTIONS BY RICK GENEVA
FOREWORD BY JAMES SINUR**

**ADVANCED COMPONENT RESEARCH, INC.
LEXINGTON, VIRGINIA**

The MicroGuide to Process and Decision Modeling in BPMN/DMN

Copyright © 2014 Tom Debevoise and James Taylor

These books are widely used by corporations and government agencies for training. The publisher offers discounts on this book when ordered in bulk quantities. For more information contact booksurge publications (www.booksurge.com) and query on title or ISBN.

All rights reserved. No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

ISBN: 978-1-502-78964-8

Library of Congress Control Number: 2008902478

Editing by Christine Parizo, Christine Parizo Communications

Composition and book design by TIPS Technical Publishing, Inc.

Cover design by 2Faced Design

CONTENTS

Foreword xv

Preface from Tom Devevoise xvii

Preface from James Taylor xix

Introduction xxi

I	Definitions..... I
	Business Process Definitions I
	Business Process Management 3
	Business Process Modeling Patterns 3
	Decision Definitions 5
	Decision Management 5
	Business Decision Modeling Patterns 6
	Business Rules and Decisions 7
	Business Event Definitions 10
	Business Event Management 10
	Business Event Patterns 10
	Business Process, Business Decisions, and Business Events 12
	Summary 12

2	Modeling Basics.....	15
	BPMN/DMN Concepts	16
	Process Instance	17
	Scope Context	17
	Data	17
	Basic BPMN and DMN Subset	18
	Basic BPMN Elements	19
	Basic DMN Elements	20
	BPMN Basic Details	21
	Activity	21
	Process Flow	23
	Process Flow with Gateways, Splits, and Merges	26
	Implicit Splits and Merge	27
	Data-Based Exclusive Gateways	29
	Parallel Gateways	31
	Data-Based Inclusive Gateway	34
	Gateway Labels	36
	Inclusive/Exclusive Gateway Best Practices	36
	Ad Hoc Subprocess	36
	Process Event	37
	Empty Events	38
	Terminate Event	40
	Participant Pool	41
	DMN Basic Details	43
	Decision	44
	Input Data	44
	Knowledge Source	45
	Business Knowledge Model	45
	Decision Requirements	46
	Information Requirements	46
	Authority Requirements	47
	Knowledge Requirements	47
	Decision Requirement Diagrams	47
	Decision Context	48
	Initial Decomposition	49
	Complete Requirements	50
	Subset Requirements	50

	Other Diagrams	51
	Using the Basic Shapes	51
	Summary	54
3	Building on the Basics	59
	More Activities and Tasks	60
	Specific Task Types	60
	Iterations and Multiplicity	64
	Advanced Events	68
	Message Event	68
	Throwing and Catching Events	69
	Timer Events	72
	Scope Cancellation and Non-Interrupting Events	74
	Conditional Events	74
	Signal Event	76
	Link Event	78
	Multiple Events and Multiple Parallel Events	80
	Event-Based Exclusive Gateway	82
	Event-Based Parallel Gateway	85
	BPMN Scenarios	87
	Coordinating Looping Processes	87
	Decision Logic	90
	Decision Logic Representation	90
	Decision Tables	91
	Business Knowledge Models	93
	BPMN and DMN Documentation	94
	BPMN Documentation	94
	DMN Documentation	97
	Combining Signals and other Events	98
	Summary	100
4	Handling Complexity.....	103
	Complex Gateway	104
	Complex Split	104
	Complex Merge	105
	Events and Error Handling	106

Event Subprocess	107
Error and Escalation Events	108
Modeling Error Handling	111
BPMN Transaction Handling Mechanism	119
Compensation Events	119
Cancel Events	121
Complex Scenarios in BPMN	123
Decision Logic Implementation	126
FEEL	126
Advanced Business Knowledge Models	127
Advanced Decision Tables	127
Other Decision Logic Representations	129
Advanced Decision Requirements Diagrams	130
Advanced Authority Requirements	130
Analytic Requirements	130
Advanced Information Requirements	131
Summary	131

5	Business Events and Business Event Modeling	135
	Introduction	135
	Business Event Processing	138
	Business Events Notation	138
	Event Processing Overview	139
	Event Analysis	141
	Types of Event Processing	142
	Event Processing and Business Process Models	142
	Event Modeling Framework (EMF)	143
	Event Decision Logic	143
	Event Channel	145
	Event Repository	146
	BPMN Shapes Relevant to Event-Centered Processes	148
	Business Events Usage Patterns	148
	Conclusion	152

6	Connecting Decisions in DMN to Processes in BPMN.....	153
	Introduction	153
	Process Discovery for Execution in DMN/BPMN	155
	Task Sequencing	157
	Participant Assignment	159
	Effect Sequencing	160
	Data Information	160
	Detection of Events	160
	Exploring a Decision Use Case	163
	Sources	163
	Transportation	164
	Insurance	164
	Reporting	164
	Decision Solutions in DMN	165
	Top Level Decision for Material Sourcing	166
	Source Decision	168
	Business Knowledge and Expressions for Contract Selection	168
	Business Knowledge and Expressions for Contract Actions	171
	Business Knowledge and Expressions for Contract Costs	171
	Business Knowledge and Expressions for Candidate Vendor	171
	Business Knowledge and Expressions for Commercial Purchase	172
	Business Knowledge and Expressions for Internal Inventory	172
	Business Knowledge and Expressions for Sourcing Action	172
	Business Knowledge and Expressions for Purchasing Policies	173
	Process Responses	174
	Transportation Decision	174
	Business Knowledge and Expressions for	

Transportation Mode	176
Business Knowledge and Expressions for Local Transportation	176
Summary Material Sourcing Processes	177
Conclusion	177

7 Execution Semantics181

Introduction	181
Completing the Process Model for Execution	182
Tasks for Model Execution	185
Synchronous and Asynchronous Service Calls	187
Data Modeling	188
Data Object	189
Data Stores	191
Message	192
Execution Semantics	193
Implicit Splits and Merges	194
Understanding the Execution Semantics of the Gateways	195
Understanding the Life of the Instance	197
Event Subprocess	198
Execution Semantics for Subprocess	198
Compensations	201
Summary	202

8 Conclusions.....203

A Commercial Information207

Introductions	207
Black Pearl Development	207
Solutions for the Internet of Things	208
Contact Us	208
Decision Management Solutions	209
Consulting	209
Training and Workshops	210
Decision Modeling Software—DecisionsFirst	

Modeler	210
What Our Clients Are Saying	211
Vendor Services	211
Speaking and Keynotes	212
Contact Us	212
BPMInstitute.org	213
Industry experts	213
The knowledge you need	213
Earn your BPMP—or just take a course or two	214
Certify your BPM proficiency—become a CBPMP	214
Contact Us	214
Salient Process	215
Contact Us	215
Object Management Group® (OMG®)	216
Index	217

BUILDING ON THE BASICS

In Chapter 2, the basics of BPMN/DMN were discussed. In addition to modeling decisions and the paths a process travels, the chapter also detailed that process modeling in BPMN has a dynamic side that describes a time-dependent passing of data along sequences represented by tokens. A particular set of tokens must arrive in order for gateway and activity shapes to be activated. Once activated, a gateway or activity follows a predictable pattern that is denoted by the shape's decoration. In this chapter we will build on this with more BPMN elements.

Chapter 2 also discussed DMN and how to model decisions. This chapter introduces the addition of business logic to decision models, as well as defining some additional properties that are worth capturing to document decisions and decision models.

This chapter also covers the event-oriented features of BPMN, especially message events. Messaging is a prominent feature of BPMN models, and messaging is implemented as events and tasks. Messages are the mechanism for pools and processes to communicate. The adoption of Event-Driven Architecture (EDA) has evolved the practice of process modeling (see Chapter 5), and BPMN is especially designed to be event-driven and therefore respond to EDA.

A business process is dynamic and must allow for and perhaps recover from unpredictable circumstances. As such, BPMN allows the process flow to be triggered or altered by events both inside and outside the organization. Internally, events might be the reassignment of an employee, the change of budget items, or the issue of new guidelines and directives. Externally, these events can be transactions, such as notices and communications with customers, partners, and government entities. They can also include environmental, political, or economic data such as news, weather, water quality, or commodity prices.

As the focus of process modeling dives into the details of a process, models will need the ability to loop through records, time, and conditions.

BPMN elements build complexity by adding capabilities to base elements. That turns the simple event into a timer event by virtue of the addition of an interior clock. For most elements, events, gateways, and activities, the attributes of the inner element inherit the attributes of the outer elements, becoming a composite of all the attributes. Figure 3.1 reflects this idea in a timer event.

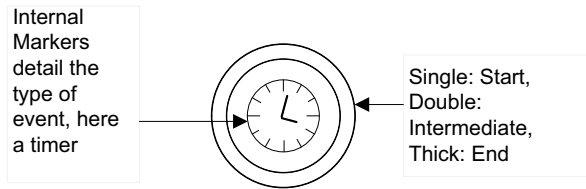


Figure 3.1 *BPMN event syntax.*

The graphical elements suggest functionality, and data attributes control the behavior of the elements. This chapter will review a number of these composites, including composites of activity types and gateways.

An effective model is both descriptive and concise, using detailed words and clear elements. Yet drawing out every detail in perfect BPMN syntax might not always be sufficient. BPMN includes features that are useful for documentation. This chapter will also introduce other shapes and modeling styles that might be helpful to model more details about processes and decisions.

MORE ACTIVITIES AND TASKS








As process modeling delves into deeper details, business and operational particulars will become clearer. The following activities and tasks model these deeper requirements.

Specific Task Types

The BPMN 2.0 specification adds six subtypes to tasks. The composite variations generally add metadata that relate to the nominal characteristics of the task. They also provide more detail on the execution characteristics of the process.

Often, in the early phases of modeling a process, many details are unknown, including participants and the characteristics of the system architecture that will execute the task. A good practice is to describe the happy path activity flow for the entire process first before attempting to identify who or what performs the tasks. Any involved activities should be defined before identifying who or what completes the activity. Subsequently, these specific activity types are useful for refining process discovery and documentation.

Table 3.1 Task Shapes Designated for a Specific Purpose

Receive Message Task Task involves asynchronous interaction with another participant.	The white envelope denotes that the task will not be complete until a message is received. A received message should be attached.	 Message Task
Send Message Task Task involves interaction with another participant.	The dark envelope denotes that the task immediately sends a message. A sent message should be attached.	 Message Task
Manual Task Task is performed manually outside the scope of a BPM System (BPMS) or software application such as a Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP) system.	Manual tasks are not managed by the engine and have no effect on execution. They immediately transition to completion and have no sent or received messages.	 Manual Task
Human Task Task is performed by a person. This is typically used in a BPMS to differentiate form-based and system tasks.	The human task generally works with a task list that manages the visibility of the task according to the role of the user.	 Human Task
Service Task Task is synchronously performed by a system service.	For legacy reasons, these can be visualized as an interaction with a web service. Sent and received messages can be attached to the service task.	 Service Task
Script Task Represents a software script that runs automatically when the task is activated.	This changes the values of input data objects in a programming language such as Xpath, Jscript, or Visual Basic(.net) that the engine understands.	 Script Task
Rule Task Represents an activity where business rules will determine the activity outcome.	This is a placeholder for (business rule-driven) decisions and is the natural placeholder for a DMN decision task.	 Rule Task

To expand on a DMN integration example, Figure 3.2 shows an example with the message, rule, and script shapes and gateways. Subsequent chapters will delve deeply into this.

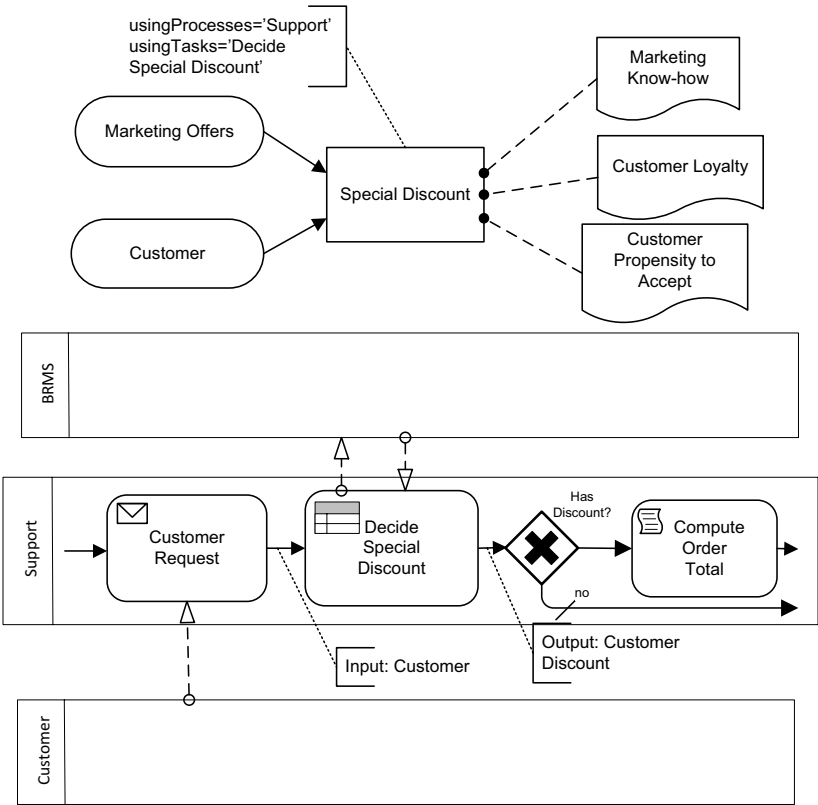


Figure 3.2 DMN connections to a BPMN process.

The customer request message moves customer data as input to Decide Special Discount. This rule task executes logic associated with the business knowledge pertaining to the special discount decision modeling in DMN. The business logic is implemented in a BRMS. The logic in the BRMS returns the discount, and a script applies the discount.

In DMN 1.0, a process is connected to a decision by metadata. Decisions can be associated with one or more business processes to indicate that the decision is made during those processes. Decisions can also be associated with one or more specific tasks, usually business rules tasks, within those processes to indicate that the tasks involve making the decision at that specific point in the process.

Figure 3.2 shows a public process with a customer participant. Either message is attached to the boundary, or only those activities that are used to communicate to the other participant are included in the public process. Process models do not always need to show the interaction details of participants. The focus of the diagram might be a single participant; however, the model might detail the points of interaction. The pool can define the other participant involvement without details of the activities. This is commonly used for an external participant, such as a vendor, supplier, or other third-party organization.

Note that there is no notation to show how a DMN decision requirements diagram is linked to a BPMN process diagram. Text annotations are typically used for this purpose, as shown.

Figure 3.3 illustrates a sample process that shows the proper usage of the send, receive, script, and human tasks.

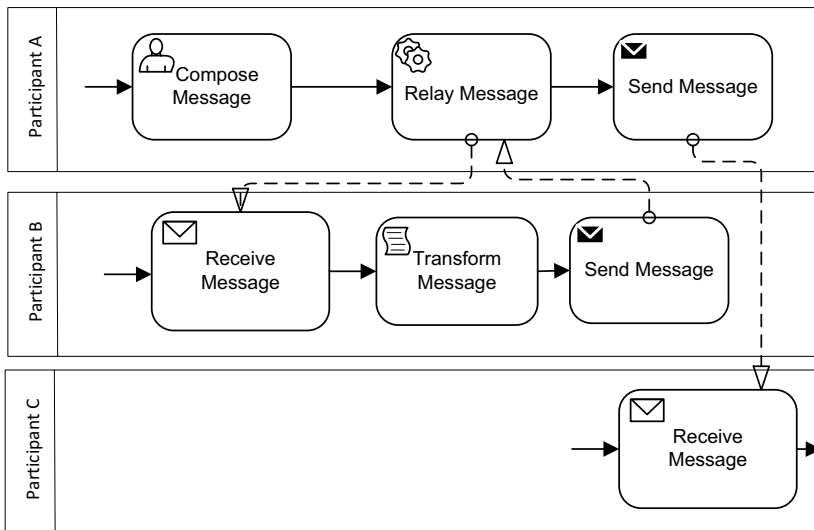


Figure 3.3 *An abstract use case to remember the role of activity types.*

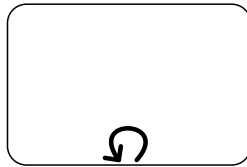
Participant A uses a form in Compose a Message and forwards this to the Relay Message service task, which in turn sends a message to participant B. The Transform Message script task uses code, such as XPath, to change the message. The message is returned to participant A. Subsequently, the send message task forwards the transformed message to participant C.

As the notation implies, a message task cannot simultaneously send and receive a message. Sending and receiving a message can be done with a single service, two message tasks, or two intermediate message events. The event section of this chapter discusses the message event further.

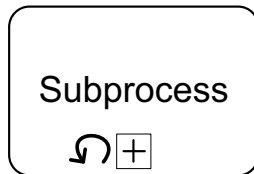
Iterations and Multiplicity

As modeling focuses on the details of a process, models will need the ability to loop through records, time, and conditions.

An activity that repeats for multiple iterations is called a *looping activity*. Looping activities can be either a task or a subprocess. The marker for the looping task or subprocess is a pointed arc. This means that the activity has the looping characteristic.



Looping Task



Looping Subprocess

The looping task is a basic activity performed repeatedly. For example, a procedure repeats until a condition is met. More often, looping activities are subprocesses. Because it is a compound, multi-step activity, a looping subprocess can apply a subprocess to a set of records. A purchase order might have multiple products to be sourced and delivered. When the loop ends, the activity is completed, and the process continues to the next activity.

The loop is dependent on a condition. For example, the condition could be, “Continue looping until the document is approved or rejected,” as in Figure 3.4. The three conditions or characteristics of the loop are:

1. Loop for a number of times, called the loop counter.
2. Loop while a condition exists.
3. Loop until the condition exits.

There is nothing that explicitly shows the loop characteristic; however, it is part of the BPMN data model. A text annotation might be used to describe the loop stopping condition. Annotations will be covered in a later chapter.

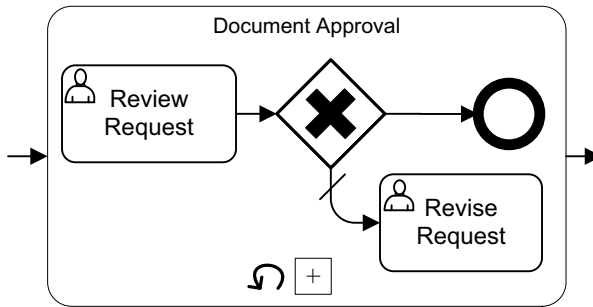


Figure 3.4 *Looping subprocess (expanded).*

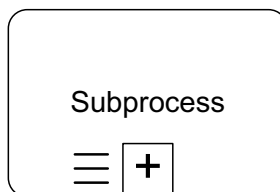
In Figure 3.4, a manager has a set of documents to review. The loop will iterate until the list of documents is completed.

Multiple-Instance Subprocess

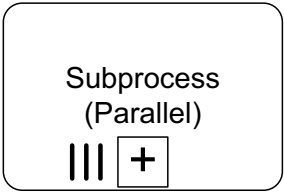
Another looping activity type is the *multiple-instance subprocess*. The purpose of this is to create a number of instances to process an input data source with the process flow located within the subprocess. The instances may execute in parallel or may be sequential. The multi-instance subprocess can manage four scenarios:

1. The activities in the subprocess should execute in a fixed number of parallel instances. The number can always be set programmatically.
2. The activities in the subprocess should execute in a fixed number of sequential instances.
3. The activities in the subprocess should execute across a collection in parallel. The number of parallel instances is determined by the collection count.
4. The activities in the subprocess should execute across a collection in sequence. The number of sequential instances is determined by the collection count.

The number of instances in the first two cases can be calculated by an expression assigned to the count by the process. This type of loop can be configured to control the tokens produced..



Multiple-instance subprocess: Serial execution



Multiple-instance subprocess: Parallel execution

In the first two scenarios above, the subprocess would take advantage of a multi-processor environment to apply computing resources to the procedure. When concurrent execution is faster or more efficient among multiple participants, parallelism can optimize a process. A listener that is handling queues of requests can start multiple instances.

In the second case, the subprocess can manage a collection or list of business objects including shipments, receipts, change requests, and chain of custody reports. Here, the count of the collection would drive the instance count. For example, a multiple-instance activity could be used for an order sourcing process with a collection of 10 order items that must be filled.

In contrast to these four scenarios, the looping subprocess might denote an indeterminate continuation. If the condition never arrives, the cycles would execute indefinitely.

The parallel version of the multiple-instance subprocess is used when all activities should be completed in parallel. Figure 3.5 shows a process with a parallel multiple-instance simultaneously sending and processing a commercial offering from each person out of a group of people.

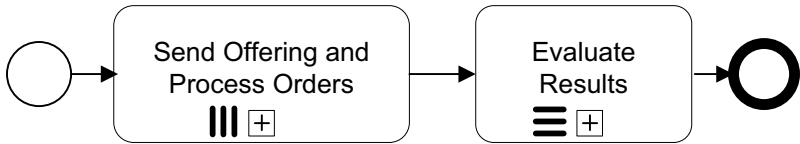


Figure 3.5 A contrast of serial and parallel multiple-instance.

Requirements such as iterating through a stack of documents might call for a serial, multiple-instance subprocess. For example, a participant receives five documents that must be signed. The documents can only be signed one at a time (sequentially) and must be kept in order. Therefore, this would impose serial execution. The discerning factor is whether or not a sequence is needed for iterations. Figure 3.5 shows the offer being evaluated with a serial multiple-instance. After all orders have been processed, a single participant will evaluate the order, once for each order received.

Consider the requirement, “For each customer in the Southeast, send a promotional email.” When the number of customers in the criteria is fixed, use the serial multiple-instances, instead of the looping subprocess. In contrast, the looping subprocess is more likely a do-while or do-until loop type. Do-while and do-until loop types typically lack a predetermined iteration count. An approval process continues to ask for revisions until the item is either approved or rejected (do-until).

Looping without the Looping Subprocess

There are ways to illustrate looping processes in BPMN without one of the looping subprocesses. The most common is called *upstream flow*. Here the flow routes back (upstream) or to a previously executed activity. Figure 3.6 shows an example of upstream flow. This style of modeling can be used with multiple lanes inside a single pool.

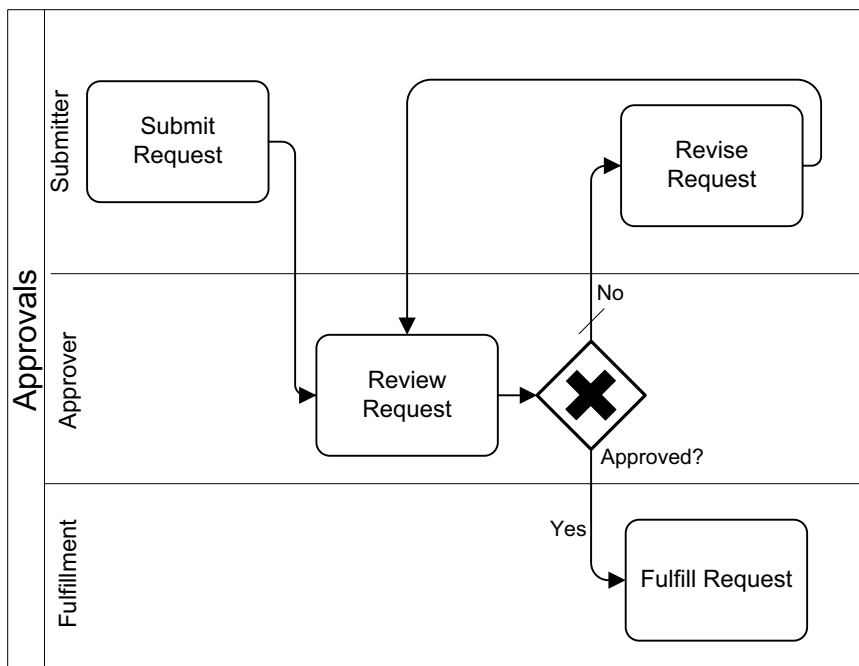


Figure 3.6 Looping using upstream sequence flow.

Often, a looping subprocess is a better choice because it shows a repeating behavior. There are advantages and disadvantages to using the upstream flow method of looping (Figure 3.19). While drawing lines in various directions can be difficult to follow, this simplified notation can be more understandable. Often, a







looping subprocess is a better choice because it shows a repeating behavior instead of a repetitive redoing of the results.

ADVANCED EVENTS

Message Event

There are five basic types of message events: start, non-interrupting start, intermediate, non-interrupting intermediate, and end. Boundary events can act on the border of a subprocess. Like the empty start, intermediate, and end shapes, the thin line, double line, and thick line mark where different message events can be used within a process flow. Dashed lines for the non-interrupting process are used on a subprocess boundary. .

Table 3.2 The six basic BPMN message events

Event Type	BPMN
Start	
Non-Interrupting Start	
Intermediate Receiving	
Non-Interrupting Intermediate	
Intermediate Sending	
Ending Sending	

The message shapes display an envelope icon or marker in the center. Each inherits the rules of the corresponding empty start, intermediate, and end event shapes. The start shape event is used at the beginning of a process diagram and has no sequence flow lines entering it. The end event shape, by definition, cannot have any sequence flow lines leaving.

Message start events (white envelope) receive messages, denoted by the sharp arrowhead pointing at the shape. Message end events (dark envelope) send messages, with the blunt end of the message line attached to the shape. In BPMN, white shapes are catching events, and dark shapes are throwing events.

Throwing and Catching Events

The term *throwing* means *sending*, and *catching* means *receiving*. There are a number of event shapes that throw and catch an event. An event is thrown when the event condition is triggered. An event is caught by the shape designated to handle the thrown condition. Message events are displayed with a filled icon sending, and the message (and the unfilled icon) receiving.

The throwing and catching pattern of filled and empty exists with the other event shapes. Filled icons specify an event thrower (sends), and unfilled icons specify an event catcher (receives).

Intermediate events can be attached to the boundary of a subprocess. These are known intermediate boundary events. In the case of messages, something inside the subprocess throws a message, and it is caught on the boundary of the process. Using an end message prevents any tokens from remaining behind.

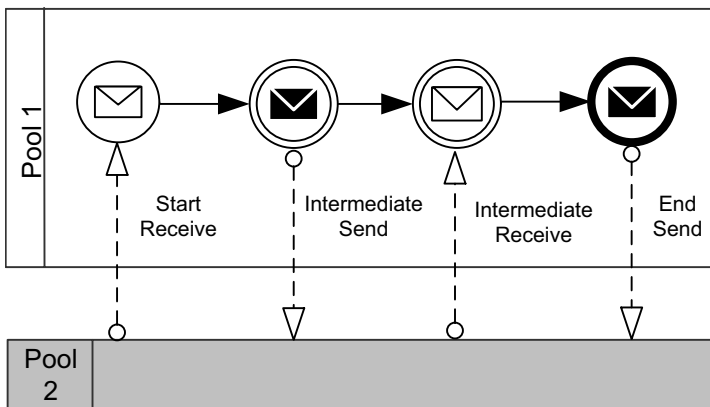


Figure 3.7 Examples of the four types of messages interacting with a public pool.

Figure 3.8 shows two pools representing two participants. First, the empty start event shape in the pool labeled Contract Officer shows the start of the process. A subprocess containing a human task prepares a message and throws the

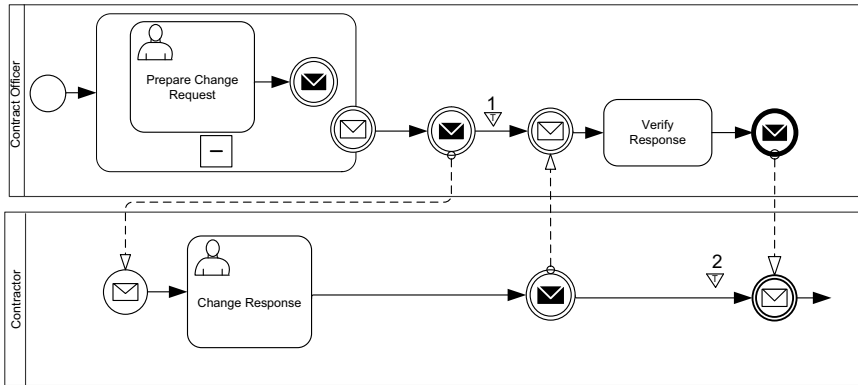


Figure 3.8 Use of message events between participants. The process is complete when the message in the top pool is sent.

message to a catching, intermediate subprocess event. From the Contractor perspective, a process instance starts when a message is received from the Contract Officer. Before this event, the Contractor is not involved in this process.

At Points 1 and 2 on the diagram, the token will advance to the intermediate receiving task and await the arrival of the message before activating the next sequence.

In general, a message start event starts when a message triggers the event. A token is generated and transitions to the next step (Change Response) in the example process. The message might be an unqualified trigger signal that starts the process, or it may include process data. In the context of this process, the message contains a requested contract change that the contractor reviews in a human task.

From the perspective of the participant represented by each pool, the intermediate message events indicate the wait state of an event on a sequence until receipt of messages from the other participant. A token would arrive at the intermediate event and await the arrival of the message. The ending for the Contract Officer participant in Figure 3.8 is a message end event.

There are important rules for modeling with message events:

- A start message event exclusively receives messages.
- An end message event exclusively sends a message.
- An intermediate message event can either send or receive a message.
- When a message is being received in an intermediate event, use the empty envelope (white). When a message is being sent, use the filled envelope (black).

Additionally, a process instance might wait for information from other participants to continue at multiple intermediate message events. Depending on the data received, there might be multiple outcomes or multiple end events.

Earlier, we discussed the use of the message events or an activity that sends a message. It can be difficult to determine if a task or an event should be used. To make the proper choice, the message interaction must be classified as synchronous or asynchronous.

Synchronous Messaging: Here the activity requests information from another participant and the message sender must wait for a response before continuing. This is also known as message blocking. Synchronous messaging is typically depicted in BPMN as a service task.

An example of this is when a courier delivers a package and requires a signature. The package will not be delivered unless there is an acknowledgement that the package was received. Another example is in system-to-system communication: as part of a transaction, the remote system sends an acknowledgement.

Asynchronous Messaging: The activity requests information from another participant. However, the message sender is not required to wait for an immediate response. Instead, the sender can do something else, and the recipient will eventually return a response. Asynchronous messaging is typically depicted in BPMN with two message events: one for sending and the other for receiving.

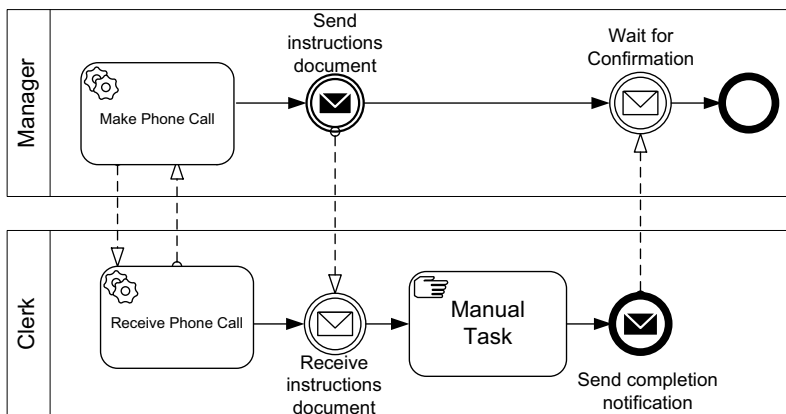


Figure 3.9 Usage of tasks and events for messaging. The *Make Phone Call* is a synchronous activity, and the service is used, while the events are not.

In Figure 3.9, the Manager first makes a phone call. This is actually a manual event; however, it is used as an analogy for synchronous messaging because it

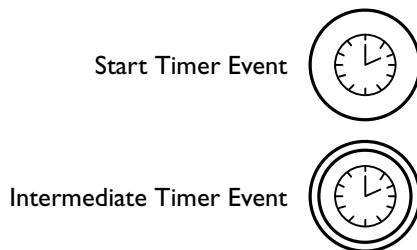
involves an interactive conversation. During the conversation, the manager cannot do anything else. Because the process is blocked from continuing until the task is complete, synchronous messaging is also referred to as blocking.

In the conversation, the manager informs the clerk of the task at hand and to expect a document and instructions. The conversation is complete, so the tasks for both participants are complete. Next, the manager sends the document. Between the phone conversation and the receipt of the document, the clerk might be doing something else. It is likely the clerk will react to the receipt of the document according to the priority set by the manager. Similarly, when the document is sent, the manager could be doing something else. But when the manager receives confirmation, his process is now active again.

The manual task is a placeholder for documenting an expected activity. However, it has no effect on the execution of the process. The token will pass through the manual event unimpeded.

Timer Events

The *timer* event is one of the versatile shapes in BPMN. It expresses a time interval in processing or a wait for a time, or it triggers actions on overdue events, activities, or other processes. The timer events include the start and intermediate events, but there is no timer end event. The start and intermediate events appear as shown.



In Figure 3.10, the Start Timer Event shows that a process starts in a given time period. For example, the system runs a report on the last day of each month.



Figure 3.10 A time-driven report.

A specified time could be a given day such as every Friday, the last day of the month, or the first day of each quarter. A time such as two hours or three days could also be specified. The time event has settable input attributes, which can be developed by upstream decisions. The decision would use multiple logic steps and data to decide optimal frequencies for processes activities or exceptions.

The intermediate timer event in Figure 3.11 expresses a process wait for a period before continuing. Like the timer start shape, the period may be expressed in an interval's duration, or it may be a calendar date.

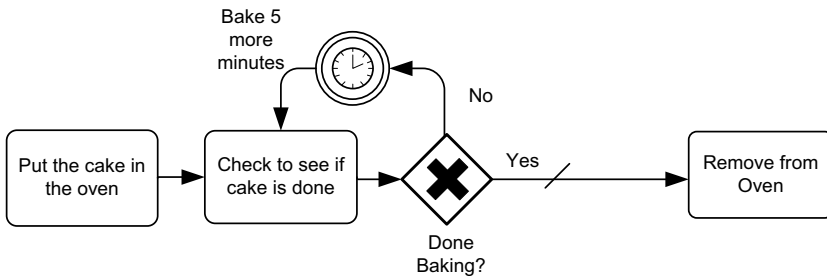


Figure 3.11 A timer event used as an intermediate event.

The intermediate timer event expresses actions taken when something is not done within a certain period, which means an alternate action such as an escalation path is desired.

Figure 3.12 shows an intermediate boundary timing event. In subprocesses, the intermediate event catches a timeout condition. This is another example of a subprocess shape with an intermediate event attached to the border of the subprocess. The alternate path starts in Figure 3.12 when the employee does not complete the work within the allotted duration

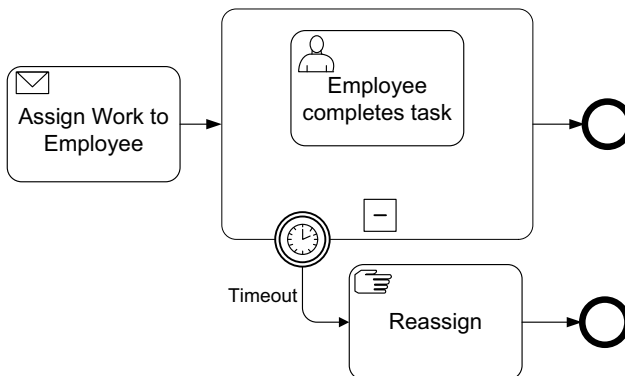


Figure 3.12 Timer event used as an intermediate event.

Scope Cancellation and Non-Interrupting Events

Some modeling scenarios might call for a flow of activities that do not interrupt the sequence flow within the parent subprocess. For example, after a period of inactivity, the process requests a status check without cancelling the current activity. Figure 3.13 introduces a timer event that does not interrupt the subprocess flow.

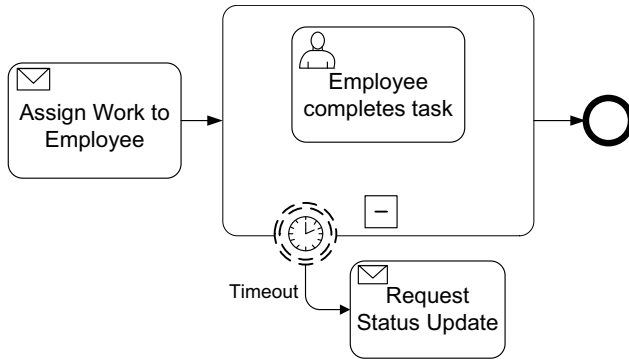


Figure 3.13 Non-interrupting timer event.

The timer event in Figure 3.13 is intermediate, but with one variation. The dashed lines on the intermediate event indicate events that do not interrupt the parent subprocess. Non-interrupting events can only be used on a subprocess border.

Multiple events can be attached to the border of a subprocess. As the diagram for the previous example develops, it might be determined that a timer is inadequate. For instance, when a message is received from an external source, the subprocess might need to be explicitly cancelled.

In Figure 3.14, the other intermediate message event cancels the parent subprocess. The timer does not. The timer is triggered after a period and prompts the manager to check the status of the employee activity. If the cancellation message is received, the employee's activity is discontinued. Because the employee task was delegated, a notification is sent, informing the employee to stop working on this process instance.

Conditional Events

The *conditional event* is a bit more sophisticated than the timer and message events. Timers are triggered by time, and messages are triggered once a participant sends a message. A conditional event is somewhat more automated and

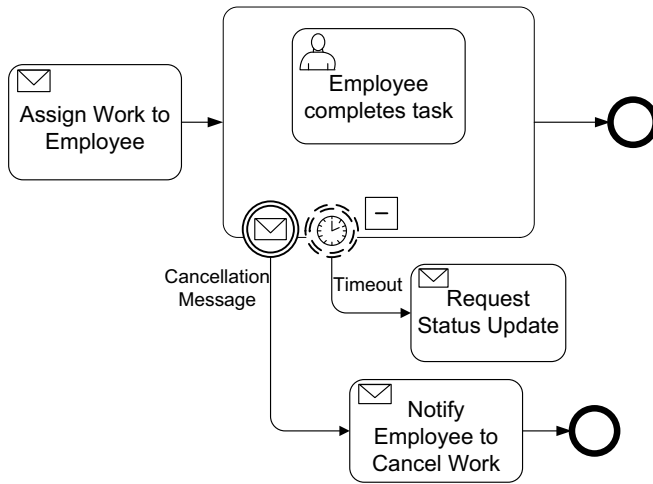


Figure 3.14 Multiple events on the subprocess border.

assumes an active system mechanism, such as an event processing platform, to trigger the condition.



Here, three of the conditional event varieties will be described: start, intermediate, and intermediate non-interrupting. The non-interrupting start will be covered in the next chapter. Figure 3.15 shows an example of the condition start event in use.

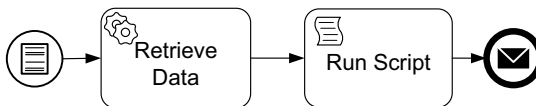


Figure 3.15 Condition start event triggers an automated process.

Scenarios employing the conditional event often employ decisions or can be replaced with an event timer and a rule/BRMS-based decision. When the condition for a conditional event gets very complex, it might be better to trigger a decision with a timer, decide if the condition has occurred (using a true/false decision model), and then conditionally generate a message.

It is possible to use the conditional event in human-centric processes. A condition event can be placed on the activity border when the activity has not completed and an interrupting condition arises or is discovered (Figure 3.16).

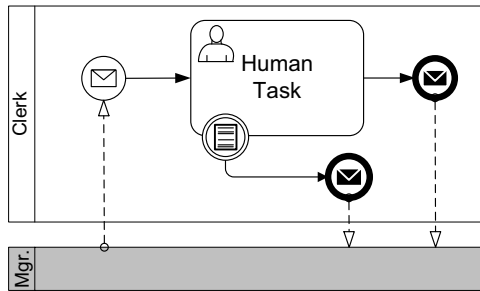


Figure 3.16 Condition event interrupting a human task.

Without driving data or triggering events, condition events do not arise. Also, a participant must subsequently evaluate the condition. In Figure 3.16, a system facility likely monitors activity and triggers the start.

Alternately, the condition event can suggest a hidden or off-diagram participant. In Figure 3.16, if the participant were a person, how does the participant receive the data and events to evaluate the condition? This point is not evident by assessing the diagram. Therefore, in some cases, another event shape, such as the signal event, should be used instead.

Signal Event

Signal events are depicted by triangles inside circles. There are six types of signal events and corresponding symbols. Five are shown in Table 3.3.

The sixth signal, the non-interrupting start signal, will be covered in the next chapter.






The signal pattern is like a radio broadcast. Ordinary message shapes send a message from one participant to another. Yet the process might need to send messages to a group of listening participants. The receiving group might even be uncertain. The signal message continuously broadcasts, and those who want to listen to the broadcast tune in or subscribe.

The signal event can broadcast to all processes simultaneously. The signal intermediate (throwing) and signal end shapes denote this broadcast. The signal start and signal intermediate (catching) events allow an activity to receive a broadcast.

Signal events can also be used for synchronization between parallel branches. In Figure 3.17, activities A1 and B1 begin at the same time. However, B2 cannot begin before A1 is complete. Messages cannot be sent to the same participant in the same pool. Because all participants can potentially receive signals, they can be used in the same pool.

A process might make another broadcast after A2 in Figure 3.17. The signal after B1 could receive either signal from the A branch. Again, the process could be stopped because B1 might take longer to complete than A1 and A2 combined.

Table 3.3 Five types of signal events

Event Type	BPMN
Start Signal Event	
Intermediate Signal Event	
Intermediate Signal Event (catching, non-interrupting)	
Intermediate Signal Event (throwing)	
End Signal Event	

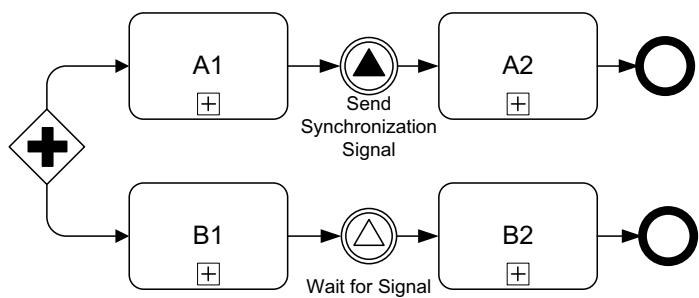


Figure 3.17 Use of signals for synchronization.

The addition of the parallel split and merge in Figure 3.18 ensures that A1, B1, and A2, B2 are kept in synch.

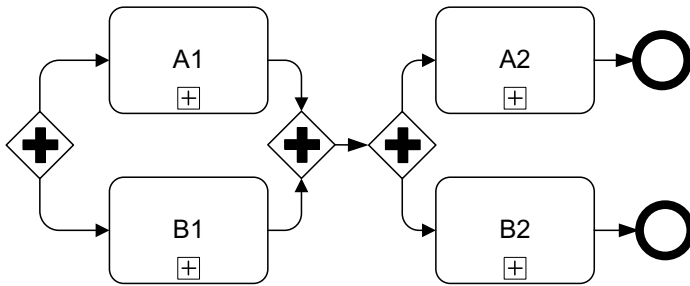


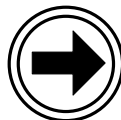
Figure 3.18 Using parallel merge instead of signals.

Here is a comparison of the different types of events:

- Messages are used for point-to-point communication between participants when it is desired to show where interactions exist. Guaranteed message receipt is desired.
- Signals are used for communications where a relevant participant might or might not exist. The signal sender is not necessarily aware of any specific recipients. Signals are typically used to notify interested participants of changing data or processing states. Message receipt confirmation is not desired because it would generate too much overhead.
- Conditions are used to detect a possible combination of criteria from multiple sources. A participant is responsible for detecting its own condition criteria because no notification of changing criteria exists.

Link Event

The link event is a bit of a misnomer. The *link event* is a diagramming mechanism for breaking and continuing sequence flows. This might also be thought of as a go-to for a process model. There are two types of link events, intermediate catching and intermediate throwing, as shown here.



The link denotes a break in a sequence flow. Therefore, there is no start or end event. If a diagram page is to start with a link event, then the catching (unfilled) intermediate link event should be used.

Because it is a break in a sequence, the link shape marks a continuation point of a process. Therefore, only the intermediate link shapes make sense.

Figure 3.19 shows the proper usage of the link events. The process can be started by any start event. Alternately, the process can continue from this point with the intermediate link event (catching). The gateway makes a decision to either continue in this diagram or continue on another diagram with the intermediate link event (throwing).

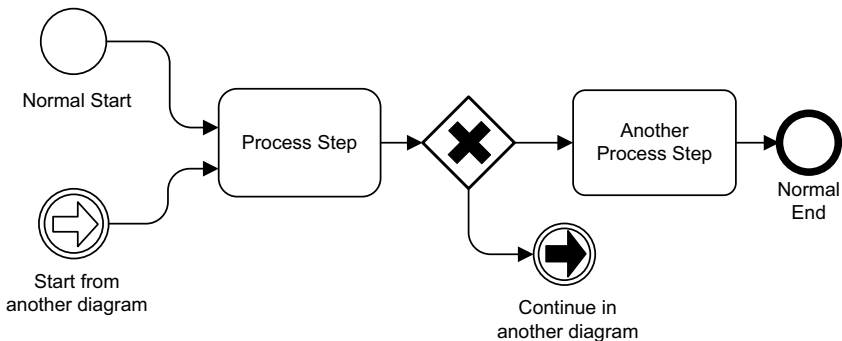


Figure 3.19 *Link event usage.*

The link event can be used to jump to another point in a diagram where it might be difficult to follow the thin line of an interaction. Therefore, links are often used instead of drawing lines because it creates a much cleaner diagram.

Figure 3.20 illustrates a hiring process which finds 10 candidates and does initial telephone screenings. If the candidates have the necessary skills, they move to the interview subprocess. Sometimes, after speaking with the first 10 candidates, not enough are found to start interviewing. If this is the case, the gateway “Enough to interview?” routes to “No,” leading to the link event. The link points back to point A.

The candidate search is conducted in parallel with multiple recruiters (note the parallel marker on the multi-instance subprocess). However, the actual interview process occurs in a series, one after the other. The number of candidates is predetermined before the subprocess starts, so the multiple-instance serial sub-

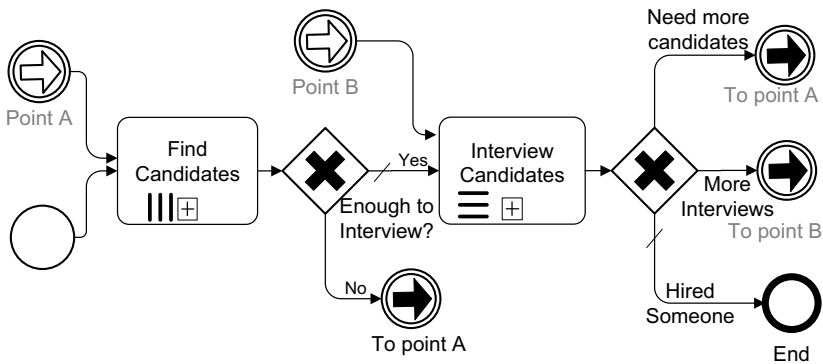


Figure 3.20 Using link events instead of lines.

process is used. After the first round of interviews, there could be one of three different outcomes:

1. All candidates might have been eliminated. In Figure 3.20, this would cause the second gateway to route to the link event, directing back to point A.
2. There might be a protocol to interview candidates more than once to help with the decision process. The second gateway in Figure 3.20 would route to the link event that routes to point B.
3. In the default outcome, someone was hired, and the process is complete.

Multiple Events and Multiple Parallel Events









There are a number of scenarios where one of a number or a combination of events will start or complete an intermediate point in a process. For example, a process might start with a phone call or an online application. More typically, different channels or services, having different forms, will trigger a single event. For this scenario, BPMN has included the multiple and parallel events.

A multiple event is started or activated when one of the triggering events occurs, in the case of starting or intermediate, or when there are multiple outcomes for the end of a process. Because a single event triggers the event, these are exclusive forms. Parallel events are similar; however, all the events in the sequence must occur for the event to activate.

Intermediate tasks can be added to the boundary of subprocesses. There are non-interrupting starting forms of these that will be covered in the next chapter.

A multiple event is a mechanism for controlling a process with multiple triggering events. Triggers could include messages, timers, conditions, signals, escalations, and other event types. Escalations and signals are covered later in this book. First, the general purpose of the multiple event is introduced, and then some of

Table 3.4 Multiple Event and Multiple Parallel Event Types

Event Type	Activation Requirements	Multiple Event	Multiple Parallel Event
Start	For multiple, one event activates; for parallel, all must occur to start.		
Intermediate	For multiple, one event activates or catches the trigger; for parallel, all start.		
Intermediate	Throwing, all events are thrown, no parallel equivalent.		N/A
Non-Interrupting	One event activates or catches the trigger; for parallel, all events must catch the trigger.		
End Event	All events are thrown, no parallel equivalent		N/A

the complex use cases are examined. Figure 3.25 illustrates an example of using multiple events.

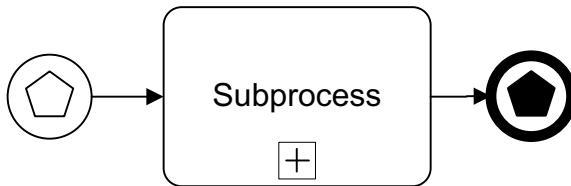


Figure 3.21 Drawing generic events with multiple shapes.

Importantly, any of the possible events could trigger the start event. Since one event triggers the transition, this is “exclusive” behavior.

The end event in Figure 3.25 could throw multiple events, and it throws all events that apply.

Event-Based Exclusive Gateway

In Chapter 2, data-based gateways were introduced. Data-based gateways evaluate a data condition to select a sequence flow path. For example, “Hire the candidate?” could have answers such as “yes,” “no,” and “maybe.” With event-based gateways, the selection is based on an event, often arriving in a message. Figure 3.22 provides an example of handling multiple events from one gateway.

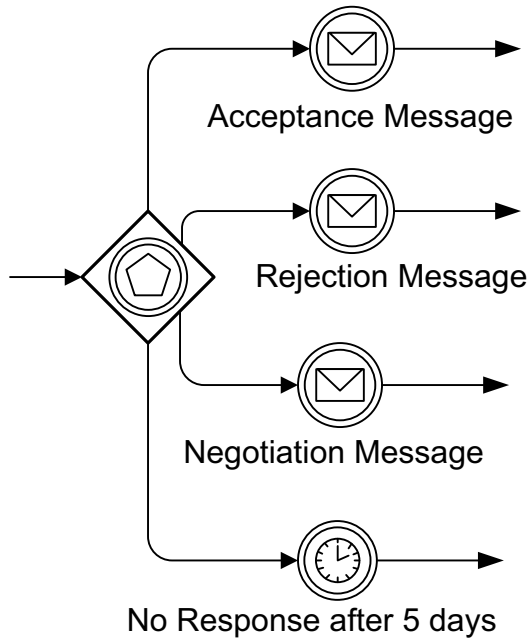


Figure 3.22 Multiple events on an intermediate, event-based exclusive gateway.

If a process is to start upon receiving one of multiple starting messages, perhaps with different data types, then the event-driven exclusive gateway with intermediate message events should be used.

As shown in Figures 3.23 and 3.24, the event-driven exclusive gateway is a diamond shape with either an inner start (single line) or an inner intermediate event shape (double thin line). As the shape implies, the event-driven gateway shape is a composite of both the data-based exclusive gateway and the multiple event start.

The event-driven exclusive gateway awaits a single message or event. Events include regular messages, signals, and timers. Usually, messages and events occur before process gateways, so the notation may seem backwards. However, event shapes are drawn on the right side of the event-driven gateway or downstream.

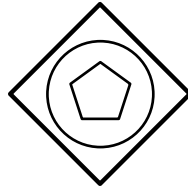


Figure 3.23 *Event-driven gateway symbol (start).*

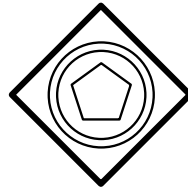


Figure 3.24 *Event-driven gateway symbol (intermediate).*

Placing the events on the left (upstream) side would be a series of events converging at the gateway.

The event-driven exclusive gateway can come in the start of a process or in a sequence as an intermediate shape. When the gateway is at the start of the process, the event shape inside the diamond is the multiple start event, a single thin line. Figure 3.25 shows the use of the event-driven gateway at the start of a process.

The gateway shape is a substitute for a start event. It awaits the activation of the downstream event shapes. In most process scenarios, one defined starting point exists, but with this gateway, a process can start in multiple ways.

When used as the start event, the event-based exclusive gateway indicates a new process instance for each event. In Figure 3.25, the receipt of a fax will create one process instance. A subsequent receipt of an email on the gateway would start a different process instance, separate from the instance started by the fax message.

In Figure 3.25, three different event shapes can activate the merging gateway. One option is for an email message event to start a process. There is a corresponding activity, Transfer Data. Once an intermediate message is received and the process instance starts, it merges back into a people or systems process. Through one of three possible message types, the process initializes. Each message type requires different processing activities.

Another use case can also demonstrate the power of the event driven gateway. The first activity in Figure 3.26, Call External System, awaits an answer from an external system. A message gateway awaits the system's response. The external system may be owned by another organization and controlled by its resources. There is rationale for separating system responses. A delay period occurs between sending and receiving the request. This is also an asynchronous interaction.

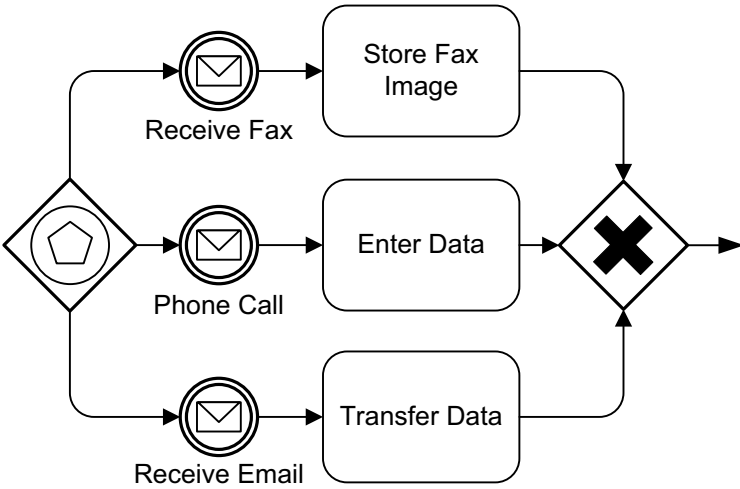


Figure 3.25 Event-driven exclusive gateway used as a start event.

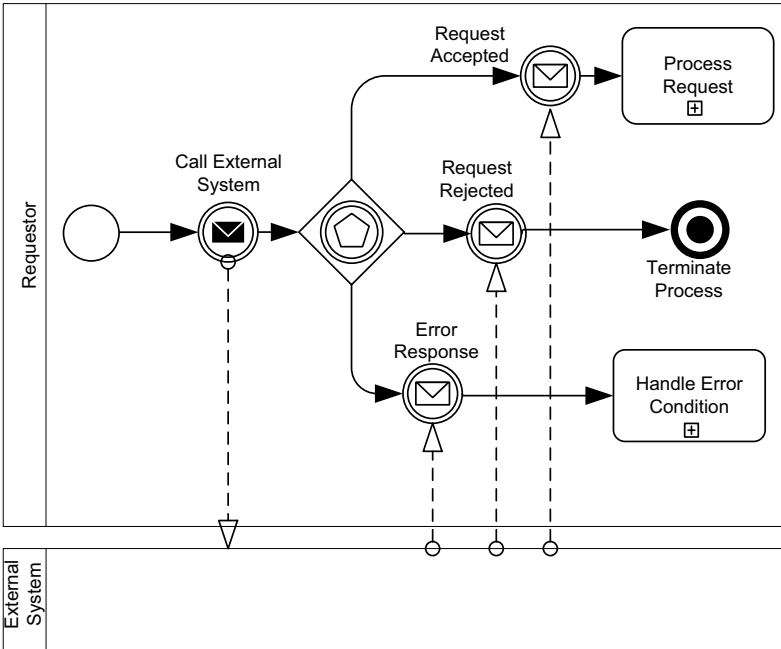


Figure 3.26 Event-driven gateway used as an intermediate event.

Processes use different data types to start the same process. In the example illustrated in Figure 3.26, the external system responds with one of three different messages. Because the message format is different for each condition, three different message types can be received. This pattern uses a single message event, followed by an exclusive gateway. In many scenarios, different message types arrive from other participants and require mapping to the needed message types.

This process pattern is useful when there are multiple suppliers or partner organizations (external participants). Often, the external participant has varying degrees of system automation capabilities. Some external participants might support a web services interaction, while others support file uploads. Still others need support for a manual process. Each method of receiving input needs different incoming message processing. Therefore, the event-based exclusive gateway is an excellent choice for diagramming this pattern.

The event-based gateway can use a message task instead of a message event. This notation can be used with other events, such as a timer.

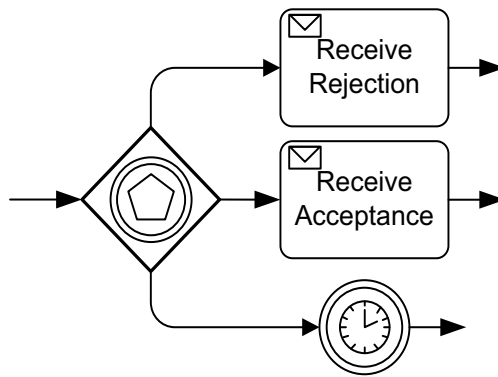


Figure 3.27 Event-based gateway with message task and timer event.

Figure 3.27 shows an example of message tasks used with a timer event. The process will either receive a rejection letter, an acceptance letter, or after the specified time has passed, the process will continue without either message.

Event-Based Parallel Gateway

A process might require receipt of multiple events before proceeding. This is the purpose of the parallel version of the event-driven gateway, the *event-driven parallel gateway*. Its shape is a combination gateway and parallel event, and it inherits the characteristics of both elements. Start or intermediate forms are inside a diamond.

The pattern for the parallel event-driven gateway includes a parallel gateway for the merge. Alone, the event-driven gateway is never used for a merge. Instead,



Figure 3.28 *Event-based inclusive gateway (parallel, start).*



Figure 3.29 *Event-based inclusive gateway (parallel, intermediate).*

a standard parallel gateway is used. It is a best practice to always include the parallel merge immediately following the connected events.

Figure 3.30 shows that more than one message is to be received, and a condition event is satisfied. If any of the events are not yet triggered, the process will not proceed. For example, the pattern could include an approval process with mandatory responses from two reviewers, as well as a business day constraint on the activity. When the second message was received on a weekend, the business day condition would not be true until Monday morning. Then it would trigger the condition, causing the process to continue.

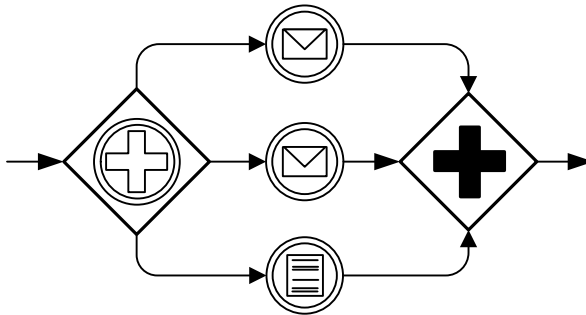


Figure 3.30 *Usage pattern for the event-based inclusive gateway.*

When used in an intermediate context, there is little difference between an event-driven parallel gateway and a standard parallel gateway. However, an event-based gateway models a process that does not start or continue until all the events are received. The ordinary parallel gateway signifies that the process has already begun and must now wait for the events. An instance of the process would not

emerge until all the events occur. With a parallel gateway, the process instance will be running.

The intermediate version of the parallel event-based gateway can be used for documentation and to limit the scope of the subsequent flow to include events. Importantly, processes change over time, and BPMN includes several ways of expressing the exact process behavior.

Figure 3.31 shows a pattern that starts a service order process. This particular process begins in a unique way. Both a purchase order and a signed statement of work are required before the process can start (messages received). Additionally, the account must be in good standing (a condition). If all three of these conditions are not met, the process will not start.

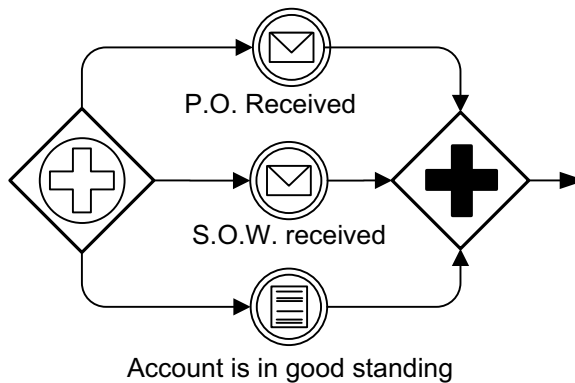


Figure 3.31 *Parallel event-driven gateway as a start event.*

Without the event-driven parallel gateway, the example in Figure 3.31 would be more complicated to depict in BPMN. It is not known before the process starts which message will be received first. It is also not known whether or not the account is in good standing, which might change before or after either of the messages is received.

BPMN SCENARIOS

This section will review a number of BPMN examples that use what has already been covered in this chapter.

Coordinating Looping Processes

A common looping pattern in process modeling is to use a coordinator participant that makes decisions and dispatches requests to the proper role. The coordinator participant is a valuable abstraction. It can coordinate decisions for an entire organization. The coordinator is a participant that is aware of all activity. It

decides and coordinates events across all other participants. Usually, a coordinator does not perform any of the tasks. Instead, the coordinator applies rules and dispatches tasks to other participants to complete. Figure 3.32 demonstrates a looping subprocess with a coordinator participant pool.

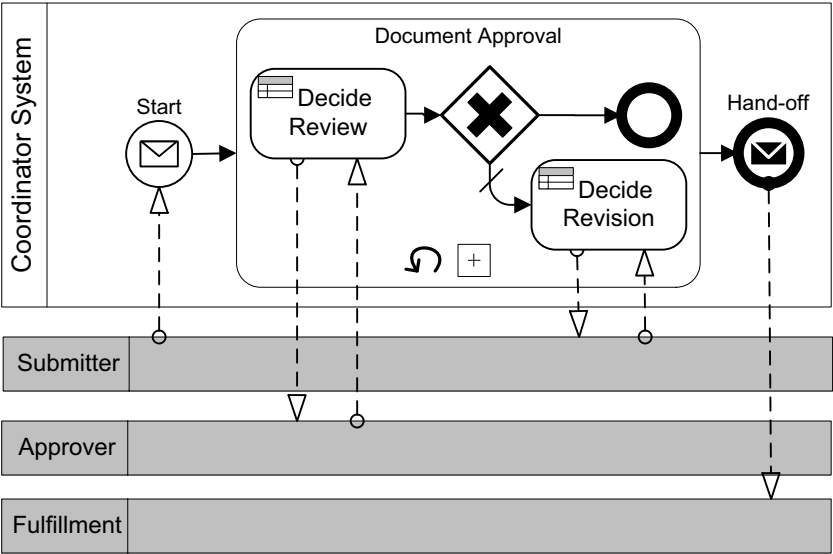


Figure 3.32 Looping subprocess using a coordinator pool.

Figure 3.32 is yet another pattern where decisions can direct processes. In this case, the decisions use the data in the request to select the participant—and probably the procedure—to be executed.

Figure 3.32 shows the participants involved in the process in a public process notation, and it also details how the participants interact. Figure 3.32 does not show the means of communication. For some process models, the interaction details might be important for synchronization purposes. For example, it might be important to show that a document approver is a person who is not normally involved in the full process. The approver must be alerted to do something. An alert implies an event, and a message interaction is the appropriate method of triggering an alerted action.

Figure 3.32 provides a decision-driven process model and is simpler to create. When the decisions coordinator is designed first, as in Figure 3.32, the overall design is faster and more complete. Additional participants are filled in after the decisions are modeled and process objectives have been properly diagrammed. The focus should be on process decisions first, rather than participants (swimlanes). This is because an analyst who draws a swimlane (pool or lane) might assign activity to a participant. In contrast, an analyst who models the deci-

sions first will carefully consider the inclusion of each participant, as necessary, to support the process goals and objectives.

When used with different pools, the multiple-instance subprocess has some unique capabilities. The pattern shown in Figure 3.33 shows a voting process. First, a request is made for a vote from committee members. For each committee member, a vote request will be sent, and the process cannot continue until all members have voted.

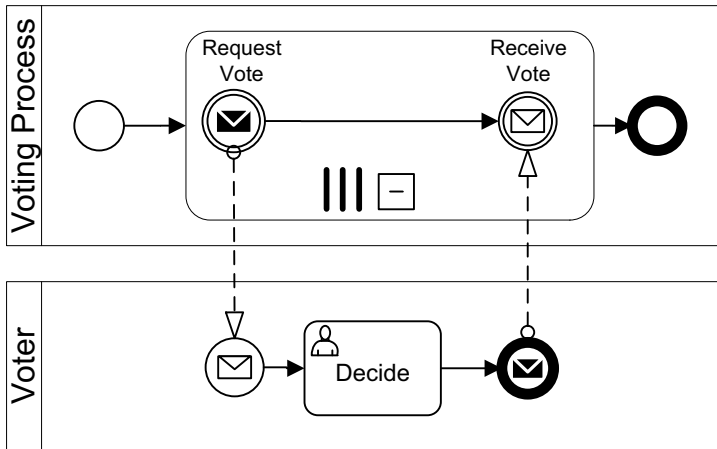


Figure 3.33 *Multiple-instance subprocess interaction.*

The messaging pattern shown in Figure 3.33 is often called asynchronous, meaning that a request will be sent, and separately, a reply will be made on another event. In contrast, Figure 3.34 shows a synchronous pattern which requires the process to wait for a reply before doing anything else. With asynchronous communication, the participant can do other activities between Request Vote and Receive Vote.

The parallel multiple-instance subprocess pattern in Figure 3.33 is common and particularly useful. Depicting this process behavior with a looping subprocess adds more complexity. First, a number of requests must be sent, and then later another subprocess that receives the same number of replies is sent. Figure 3.34 shows a looping subprocess pattern solution.

Because the looping subprocess is sequential, it can only interact with a single voter. By detaching the sending subprocess from the Receive Replies subprocess, Figure 3.34 mimics a parallel solution. This adds complexity for both the reviewer and the implementer.

The pattern in Figure 3.33 is simpler and better coordinates the voter response. In Figure 3.34, if a voter replies before all requests are sent, the reply would be lost. Because each instance of the subprocess is running independently,

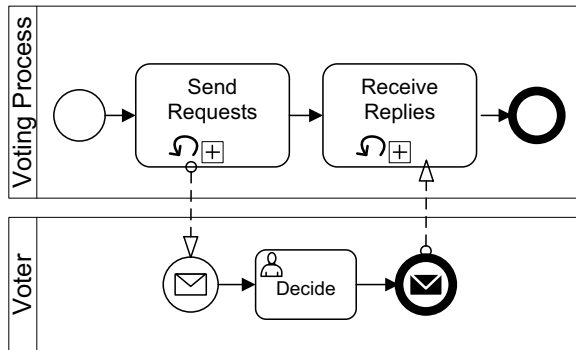


Figure 3.34 *Voting process with looping subprocesses.*

in parallel, the parallel multiple-instance in Figure 3.33 fixes this problem. Here, the send and receive can easily be managed within a single scope and clearly correlated.

DECISION LOGIC

There is tremendous value to a complete or even partial decision requirements diagram. The decomposition of decision making, the linking of required input data, and the identification of knowledge sources all clarify the exact nature of a decision in a business process. What they do not do is define the actual decision making logic that must be executed to make the decision.

DMN allows for a decision requirements diagram to be enhanced or extended with decision logic, optionally all the way to a completely executable specification of the decision making. To this end, decision logic can be specified for any decision or business knowledge model in a decision model. When adding decision logic to a decision requirements diagram, it is often not necessary to add any objects to the diagram; all the decision logic may be directly associated with the decisions already in the diagram. As the decision logic is specified, however, it may become clear that additional decomposition of the decision making or additional reusable business knowledge models make sense and should be added.

Decision Logic Representation

The decision logic required can be represented in three main ways in DMN:

- Literal expressions, which is text that describes how to derive a decision's result from its inputs.
- Decision tables which are tabular representations of decision logic, sometimes also known as rule sheets.

- Invocations which are demonstrated below in an invocation of a business knowledge model.

Literal expressions are the most flexible and can be purely descriptive text, such as a natural language description. More executable formats can include formal logic, a programming language such as Java, or rule syntax supported by a BRMS and Friendly Enough Expression Language (FEEL), which is part of DMN and described further in Chapter 4. Literal expressions could also be based on other standards such as Predictive Model Markup Language (PMML), which would allow the decision making logic to be defined in terms of an analytical model.

Decision tables are a very popular representation format for business rules and decision logic, and DMN supports a number of different layouts and approaches, described below.

The specification of an invocation allows a decision to be linked to a single business knowledge model so that the decision can invoke, or call, the decision logic defined in the business knowledge model, therefore reusing the decision logic specified in the business knowledge model.

Literal expressions can also include invocations of one or more business knowledge models. If there is decision logic required that is not in the invoked business knowledge model, or if there is more than one business knowledge model to be invoked for a specific decision, then a modeled invocation cannot be used, and a literal expression must be written that contains the invocation(s), as well as the additional logic.

Whatever the format, the decision logic consumes variables that represent the information requirements of the decision; each requirement has a variable that represents it. For literal expressions and decision tables, these requirements are mapped directly into the logic, while an invocation maps these to the variables defined for the business knowledge model.

Finally, the output of a decision's expression can be returned to a process model so that the action or actions suggested by the decision can be carried out. For instance, if the logic of a decision model decides that a particular offer is to be made to a customer, then the process can use the output of the decision to retrieve and actually present the offer to the customer.

Decision Tables

Three types of decision tables are defined in DMN:

1. Cross-tab or classic decision tables have dimensions that represent input variables, with each row and column in the table representing a specific condition based on the relevant input variable. Selecting a row and column based on specific values for those variables identifies a single cell

in the table. This cell contains the value to be assigned to the output of the table.

Insurance Coverage Requirement					
Hazardous Material		Transportation Mode			
		Truck	Rail	Barge	Package
HAZMAT Product Category	1	\$1M	\$4M	\$6M	\$500K
	2	\$6M		\$4M	\$100K
	3	\$8M		\$12M	\$500K

Figure 3.35 A cross-tab decision table.

- 2. Decision tables with rules as rows. These decision tables are sometimes known as rule sheets, and in Figure 3.36, each column represents an input variable or an output variable. Each row contains values in zero, one or many of the input variable columns, as well as in one or more of the output columns. If the values in the input variables match all the columns in a row, then the rule is true, and that selects the appropriate output value(s).

Insurance Coverage Requirement			
UC	Hazardous Material Category	Transportation Mode	Coverage
	Class 1	Truck	\$1M
		Rail	\$4M
		Barge	\$6M
		Package	\$500K
	Class 2	Truck	\$6M
		Rail	\$6M
		Barge	\$4M
		Package	\$100K
	Class 3	Truck	\$8M
		Rail	\$8M
		Barge	\$12M
		Package	\$500K

Figure 3.36 Rules as rows decision table.

- 3. Decision table with rules as columns. These decision tables are also sometimes known as rule sheets and are a transposition of the rules as rows type. Each row is a variable, and each column is a rule.

Marketing Offer Selection			
Product	Product 1		Product 2
Offer Value	Low	High	
Marketing Offer	Offer A	Offer C	Offer B
UC	1	2	3

Figure 3.37 Rules as columns decision table.

It should be noted that compound outputs are allowed because each rule can set more than one output value, and the syntax allows merged cells as shown in Figure 3.35 for ease of reading.

Decision tables can be single or multiple hit tables. In a single hit table, only one result is allowed, while in a multiple hit table, many hits are allowed and more than one rule may be acted on. While DMN defines a number of hit policies and insists that these are displayed as part of the table, the core decision table hit policies that generally analysts should be restricted to are:

- Unique single hit tables where it is only ever possible for a single rule to be true, as the rules involved are exclusive.
- Any single hit tables where there may be many rules that are true at the same time, but it does not matter as they all result in the same output (e.g., a lot of ways to reject something).
- Multiple hit tables aggregate their results and valid use cases include those that collect all the hits to, for example, build a list of all the reasons for a rejection, as well as those that sum up a score based on all the rules that fire.

Other options for decision table hit policies are discussed in Chapter 4.

Business Knowledge Models

Business knowledge models act as functions, allowing decision logic to be parameterized and reused across decisions. For example, the decision logic for checking that a US address is valid may need to be applied in a decision that has an invoice as input data, as well as in a decision that has a purchase order as input data. If the logic is defined as a business knowledge model, then it can be reused across the two decisions.

Business knowledge models are linked to the decisions that use them for logic and to other business knowledge models that call them using knowledge requirements. They can also be linked, using authority requirements, to knowledge sources that describe where the logic originated.

Business knowledge models can be any decision logic: a decision table or a literal expression that contains business rules in a decision tree, rules in some

other format, or non-rule formats such as predictive analytic models. Business knowledge models have parameters defined that are used in the decision logic and return a value or a business object containing values.

When a business knowledge model is invoked, whether using a modeled invocation or in a literal expression, the input values to the decision or business knowledge model doing the invoking are mapped to the business knowledge model's parameters, and the result of the business knowledge model is mapped to the output data. As implied in the example in Figure 3.1, when a decision is to be integrated in a process, these output data can drive conditions in gateways and values in activities.

While the decision logic for a decision requirements diagram may be added entirely to the decisions in the diagram, it is possible that some number of business knowledge models will be added either because the logic is reused (or intended to be reusable) or to provide easier management of certain pieces of business logic.

BPMN AND DMN DOCUMENTATION

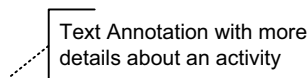
Modelers can add information to DMN and BPMN diagrams to create clarity and improve understanding. BPMN has several inherent elements, artifacts, and data items for documentation. There is no equivalent in DMN for these; however, the tacit understanding of the specification writers was that things like the text box would be available in DMN.

Documentation in DMN is more abstract and a matter of using the metadata properly.

BPMN Documentation

BPMN provides the capability to append additional information to diagrams through shapes and annotations. There are three diagram documenting shapes: text annotation, associations, and groups. The shapes can be associated with other shapes but do not have a sequence flow or message flow. The dotted association line is used to attach text and data shapes to other BPMN shapes. Association lines attached to a data object may have an open arrowhead to indicate input or output data for an activity.

Because they can provide more written context about an activity than the activity name, text annotations are useful.



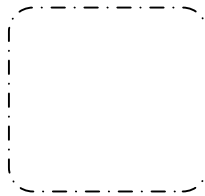
Activities, events, and gateways will normally have a short label. One goal of creating a BPMN diagram is to provide clarity with good descriptions. A good

BPMN diagram is also concise and can be improved with text annotations and well-formed BPMN labels.

While DMN does not include a similar text annotation approach today, it is likely to do so in the future. This kind of annotation can provide benefits to DMN models also and should be used as necessary to improve readability and understanding.

Groups

The group shape denotes a common purpose to a group of shapes. The group shape is drawn as a rounded corner rectangle with a dot-dash-dot-dash pattern, as shown here.



The group shape is permitted on the process diagram above pools and lanes. A group shape surrounds other shapes located anywhere in the diagram. The group illustrates related activities, even when they cross multiple participants (see Figure 3.38). As a non-executing alternative to a subprocess, the group shape might surround shapes inside the pool.

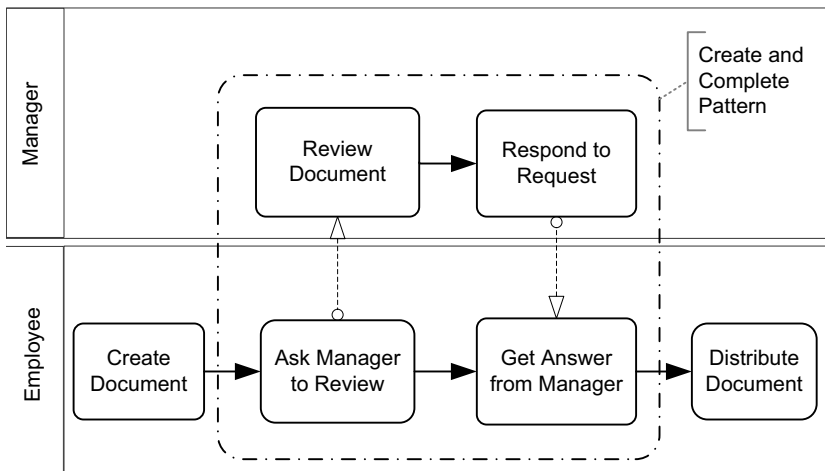
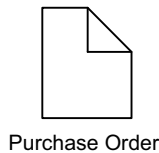


Figure 3.38 *The group shape encircles related activities in two participant pools.*

DMN does not include a group shape, but there is real value to being able to draw groups on DMN diagrams. Sets of decisions and business knowledge models might be grouped to show implementation boundaries or project phases, for example, while groups of input data might show which information objects are delivered to the system together.

Data Object

The data object can be used as a means to document a diagram and a mechanism for connecting a schema to a BPMN model. The data object is a rectangle with the upper right corner folded over, as shown here:



The text label for a data object is underneath the shape. Often, the current state of the data object is shown as an attribute placed in brackets under the text label. As the diagram progresses, the state of the data object can easily be read, as illustrated in Figure 3.39.

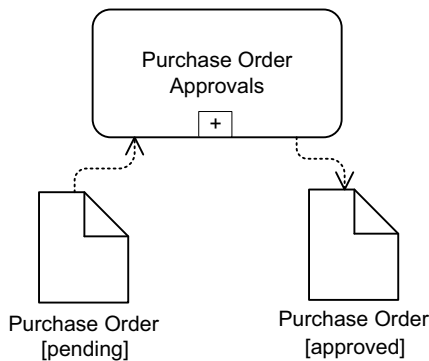


Figure 3.39 *Use of data shapes.*

As with the text annotation, the association line connects the data to another shape. Data shapes are often associated with tasks, gateways, events, sequence lines, or message lines. In message flow, data objects portray the payload or content of messages.

The central purpose of the data object is to add the details necessary to make a process model executable. Data objects also add more information about the behavior of the execution process. More details about data objects will be covered in the chapter on execution semantics.

In DMN, there is a formal object for data: the input data shape. It is extremely useful if the data objects in process models have a matching input data representation in related decision models. See also Information Items below.

DMN Documentation

There are no explicit graphical elements for documenting DMN. However, there are additional properties and associations worth tracking for DMN objects. These are defined in DMN as metadata properties of the various objects, but no specific notation is given for them. The power of questions and allowed answers has been discussed in defining decisions, as well as the importance of tracking which processes and tasks used a particular decision. The relationship of decisions to objectives, organizations, and other context also matter.

Objectives and Metrics

DMN allows for OMG's Business Motivation Model (BMM)¹ objectives and KPIs or metrics to be linked to the decisions that impact them. This mapping can be used to put each decision, or at least the top-level decisions that are not required by other decisions, into a business context. This involves assessing the KPIs and objectives in the business area and determining which KPIs or objectives are impacted by each decision. A decision impacts a KPI or objective if changing the way the decision is made could be reasonably expected to have an impact on the value recorded for the KPI or objective.

The BMM describes a way of documenting how objectives are achieved. An instance of a decision can be associated with multiple supported objectives and performance indicators.

Organizational Impact

Decision making is central to organizations, and the various roles played in decision making by an organization matters. Three particular roles can be played by organizations, organizational units, or roles, like those in BMM. For repeatable decisions, these relationships are not generally to an individual but to a team, department, or role within a team or department. It should also be noted that multiple organizations or roles can own, make, or care about a specific decision. DMN defines two such roles:

- **Who owns the decision?** Who decides how the decision should be made? Who is the approver of the approach taken? Who decides that we shall

1. Object Management Group, Inc.: Business Motivation Model,
<http://www.omg.org/spec/BMM/1.2/>

decide this way and not that way? Who will be maintaining the business rules for this decision?

- **Who makes the decision?** Often, one part or level of the organization owns a decision, but a different part or level makes the day-to-day decisions. Sales management might own the pricing decision, for example, but pricing decisions are made by the sales teams themselves. If decisions are automated or partially automated, then the organization that passes on the decision to a customer or supplier could be considered the decision maker as well.

It is often also worth documenting other stakeholders in the decision. While many decisions have owners and makers, other parts of the organization might have a stake or interest and expect their opinions to be taken into account.

Other Context

In addition to a decision's relationship to business processes and the tasks that invoke them, consider tracking the relationships of decisions to objects such as business events or information systems.

Information Items

Input data objects, the output of decisions, and the parameters and output of business knowledge models can be formally defined in DMN using information items. Information items are a basic hierarchical information structure such that each can be a single data item, a structure of other data items, or a collection of either of these. A basic set of DMN types can be used, or external types can be referenced. Restrictions on the allowed values can be specified using expressions.

When a DMN model is extended to the point of complete specification, using FEEL for example (see Chapter 4), the information manipulated by the decisions, business knowledge models, and decision logic must be fully specified. In other circumstances, some degree of documentation of the information items will be required to show what information is being used where.

COMBINING SIGNALS AND OTHER EVENTS

As the capabilities of BPMN are beginning to be visualized, more complex or subtle situations can be modeled.

Figure 3.40 shows a busy intersection from the perspective of three participants: a northbound driver, a westbound driver, and the traffic light that helps the two drivers avoid a collision. Drivers entering the intersection determine whether or not to proceed based on the color of the traffic light. The driver observes the state and decides. Because the traffic light occurs during normal driving, the traffic signal light is a driver's signal event. The stopped driver proceeds through

the intersection if there is a green signal. From the perspective of the traffic light, a green signal is sent to traffic waiting at a red light through the intersection.

Without participants, there is no explicit message interaction. Instead of using interaction lines, Figure 3.40 uses the group shape to clarify the relationship between the signal throw and catch events. The usage of the group shape is a documentation notation; it makes the diagram easier to read.

Figure 3.40 illustrates what happens after the signal change. The northbound driver is reckless and hits the accelerator pedal. The westbound driver is more cautious and watches for crossing traffic even though the traffic light signals green. When a speeding driver crosses with indifference to the red signal, an error handling subprocess is managed with this exception condition.

This is an example of how easy it is to represent a complex interaction using just a few BPMN shapes. In fact, if the whole scenario was written in words, there would probably be three or four pages. This succinct diagram shows the perspective of three participants, what they do, and how each action relates to the other participants.

The process in Figure 3.40 contrasts the condition and signal events. In the event gateway in the condition event shape, participants actively monitor for a condition. There is no guarantee that a signal light will be encountered while driving (the condition). The default event is the green condition, where the criterion is that the signal is recognized, observed, relevant to the driver's lane, and currently green. Otherwise the driver waits for a red signal to turn green (a broadcast).

With the signal event shape, participants observe a broadcast message. The driver subscribes to the signal event and waits for it. The signal event is received by anyone within view of the traffic light (the subscribers). The signal is used in this case because there is already a potential participant. In contrast, the condition event is used when it is not known if a related process exists or not. This is because the act of locating a traffic signal and responding to its current state is fairly complex. It requires a driver to pay attention to many objects as he drives down the road. If the traffic signal were instead an ambulance siren, a signal would be a better choice. With complex criteria for event recognition, a condition should be used instead of a signal.

Figure 3.41 shows a process where the sales team needs a contract reviewed by the legal team. The contract is not directly created by the sales team. Instead, the contract is generated by a system. First, the CRM data source is updated. Some additional information might be retrieved from the CRM system that automatically populates the contract documents. Contracts are generated, and the legal team is notified with a hyperlink that points to the document location on a network file share.

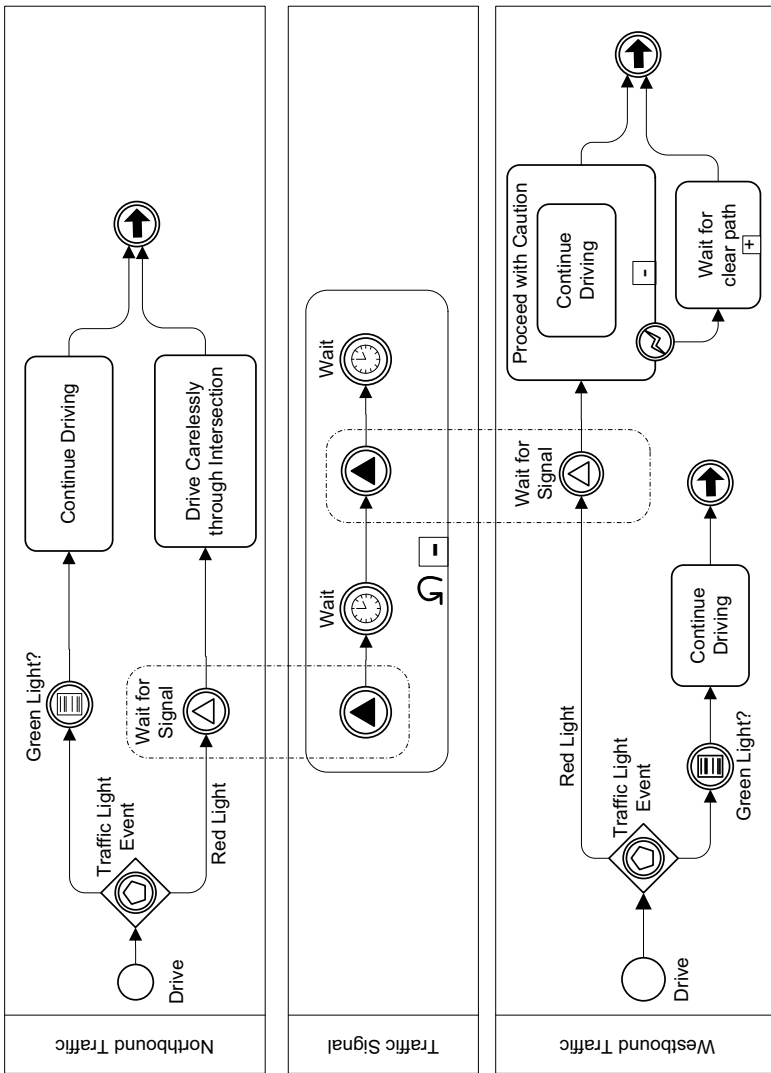


Figure 3.40 Traffic light diagram using the signal event shapes.

SUMMARY

This chapter covered several useful event types in BPMN. Modelers can develop fairly complete processes with the information in this (and the previous two) chapters. This chapter also covered some of the more powerful shapes in BPMN. These shapes are composites of the behaviors and notations described in the first BPMN chapter.

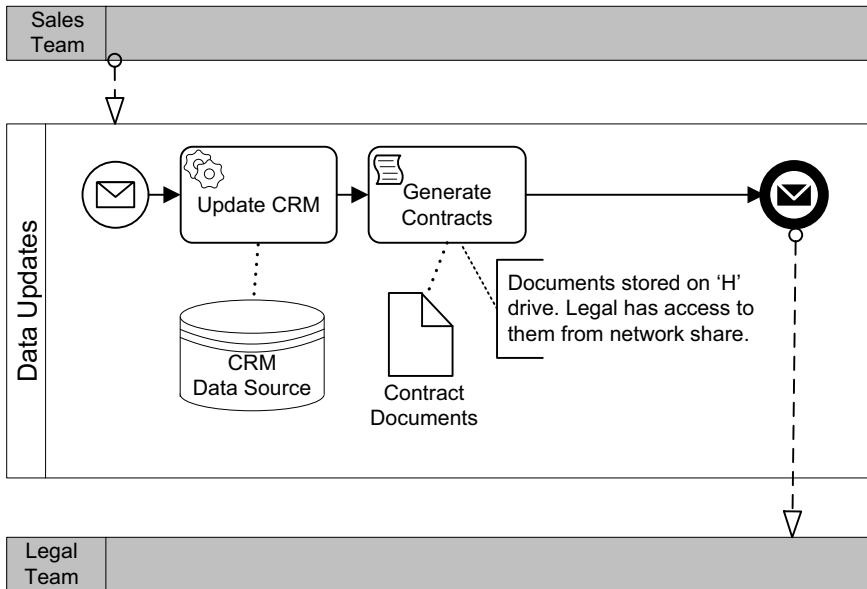


Figure 3.41 *Combination of multiple document shapes and annotations.*

The chapter also covered six additional tasks that are denoted with special markers. These were:

- Manual task (hand marker): A manual activity outside the scope of a BPMS.
- Human task (person marker): A task performed by a person, used to differentiate people and system tasks.
- Message task (envelope marker): A task involving interaction with another participant.
- Service task (gear marker): A task performed by a system service.
- Script task (script marker): A software script that runs automatically.
- Rule task (table marker): An activity run by business rules, often linked to DMN decision models.

Next, this chapter covered timer events and introduced the non-interrupting timer event. This provided a strong foundation for learning about events that are attached to the boundaries of subprocesses.

Next, the signal event was described.

Looping activities provide important capabilities for processing sets of records or executing until a condition is met.

BPMN provides the link event to simplify large diagrams. There are two types of link events, intermediate catching and throwing.

The event-based gateway solves a common, yet complex scenario. When a process should start upon the receipt of multiple starting messages with different data types, it should use the event-driven gateway with intermediate events.

These shapes address an important situation: multiple events that start or participate in a process. They are composites of the basic elements of gateways and events. In order to model more complex business processes, these are necessary knowledge. The next chapter will complete the tools needed to manage events.

This chapter also set out the logic elements of a decision model, especially the decision table. Logic elements can be combined into decision requirements diagrams to describe precisely how a decision can be executed. That is, diagrams can be extended with decision logic to form a complete definition of exactly how the user wishes to make a decision.

One purpose of using BPMN and DMN is to reduce text documentation that describes a complex process or decision. To accomplish this, as many details as necessary must be included to illustrate the intent of the model.

COMMERCIAL INFORMATION

INTRODUCTIONS

These are the companies that we recognize as leaders in the fields of Business Process Management and Decision Management. The information is provided for your use. If you are looking for training, solutions, integrations, or consulting, the companies and individuals will provide great assistance.

BLACK PEARL DEVELOPMENT



Black Pearl Development specializes in helping companies create process and decision-oriented solutions in complex engineering, energy, manufacturing, and financial domains. A comprehensive approach that starts with an architectural framework and includes Process, Event, and Decision modeling is the backbone of Black Pearl Development's approach for developing requirements. Our complete set of integration and development capabilities, including near shore resources, eases implementation and ensures rapid success.

Solutions for the Internet of Things

The Internet of Things will be a critical strategy for every business that must adapt to a newly transformed customer relationship, boost competitive advantage and generate new revenue streams. However, many factors go into creating compelling IoT offering—from product architecture, and wireless connectivity decisions, to security, analytics, billing, and go-to-market strategy—can make it a daunting undertaking. Today, many different means enable communication between the tiers of an IoT solution. Furthermore, existing solutions do not address the scalability requirements for a future Internet of Things; Black Pearl Development helps companies architect and build the IOT solutions that companies need.

Cloud computing has a large impact on IoT solutions and can provide for the “big data” capabilities demanded by those solutions. Traditionally, the ability to deal with very large amounts of data required a lot of computing power and has been very cost-prohibitive for businesses to access. This kind of processing power is needed in order to enable the type of business intelligence and decision analytics needed for IoT solutions, due to the sheer amount of data being generated on the edge devices. Because of the ability to allocate more resources instantaneously to an application, it is possible for a cloud solution to easily surpass the capabilities of a single organization’s IT infrastructure. The cloud also provides more agility because of the ability to bring up infrastructure and applications very quickly. With this increased agility, organizations can expand or adapt to changing market conditions, such as needing to address the evolving IoT world, with less risk and at less cost.

Contact Us

www.blackpearldevelopment.com

info@blackpearldevelopment.com

+1 (540) 817-4304

DECISION MANAGEMENT SOLUTIONS



DECISION MANAGEMENT SOLUTIONS

Decision Management Solutions specializes in helping companies build decision-centric, action-oriented systems and processes using decision management, business rules, and advanced analytic technologies. Decision modeling is the cornerstone for developing requirements for these next-generation systems and processes. Our consulting, training, and software eases implementation and ensures success.

Consulting

Decision Modeling

- Decision Modeling Service
- Requirements Conversion
- Existing Implementation Modeling
- Decision Inventory

Decision Management

- Decision Management System Design
- Vendor Selection
- Rapid Kickoff Service
- Decision Requirements Modeling Service

Business Rules

- Getting Started
- Vendor Selection
- Center of Excellence

- Implementation Best Practices
- Re-platforming strategy

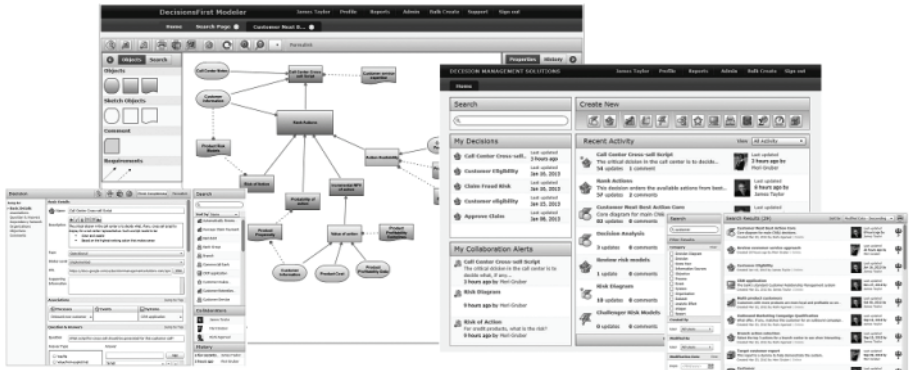
Predictive Analytics

- Getting Started
- Vendor Selection
- Business Understanding Definition
- From BI to Analytics Strategy and Roadmap

Training and Workshops

- Live online training in Decision Management and Decision Modeling (an IIBA Endorsed Course)
- Live online and on-site training with BPMInstitute.org
- Online course
- Defining Business Goals for Predictive Analytics - University of California, Irvine Extension
- On-site workshops in Decision Modeling and Decision Management

Decision Modeling Software—DecisionsFirst Modeler



Our collaborative decision modeling software, DecisionsFirst Modeler, is based on the new Decision Model Notation (DMN) standard.

Key Features

- Interactive decision diagrams allow each decision to be decomposed, showing exactly how each decision is made, how it can be improved, and what rules, analytics, and data are needed.
- A shared repository provides a framework for managing business rules, checking completeness, and managing reuse.
- Decisions requirements are a common language across business, IT, and analytic organizations improving collaboration, increasing reuse, and easing implementation.

More details on DecisionsFirst Modeler are available at:
<http://decisionmanagementsolutions.com/decisionsfirst-modeler>.

What Our Clients Are Saying

- “This is the critical path to monetizing advanced models.”—*Head of Analytics, North American Insurance Company*
- “Concrete, actionable recommendations.”—*Program Leader for Next Best Action Program at a top European Retail Bank*
- “What used to be one week of requirements work was done in a few hours with decision management.”—*Lead Business Analyst at a top Insurance Company*

Vendor Services

Decision Management Solutions also offers services to software and consulting firms:

- Strategic advice on product strategy, roadmap, and development.
- Independent validation of solutions and product reviews.
- Independent substantiation of the power of Decision Management to improve technology applications.
- Evangelism, independent research, and promotional services, such as white paper writing, to support adoption of Decision Management and its related technologies.

Speaking and Keynotes



Decision Management Solutions is led by leading decision management expert James Taylor. James is an experienced keynote presenter, speaking on analytics, decision management, business rules, and more. James is passionate about helping companies adopt Decision Management to develop agile, analytic, and adaptive business processes and systems. He has spent the last 20 years developing approaches, tools, and platforms that others can use to build more effective information systems.

James is also the author of “Decision Management Systems: A Practical Guide To Using Business Rules And Predictive Analytics” (IBM Press, 2011), “Smart (Enough) Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions” (Prentice Hall) with Neil Raden, and has contributed chapters on Decision Management to multiple books including “Applying Real-World BPM in an SAP Environment,” “The Decision Model,” “The Business Rules Revolution: Doing Business The Right Way,” and “Intelligent BPM Systems: Impact and Opportunity,” as well as many magazine articles.

Contact Us

www.decisionmanagementsolutions.com

info@decisionmanagementsolutions.com

+1 (650) 400-3029