Date: June 2010

BPMN 2.0 by Example

Version Alpha 8 (non-normative)

OMG Document Number: dtc/2010-06-xx

Standard document URL: http://www.omg.org/spec/acronym/1.0/PDF

Associated File(s)*: http://www.omg.org/spec/acronym/200xxxxx

http://www.omg.org/spec/acronym/200xxxxx

Source document: Title (document number)

* Original file(s): Title (document number)

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c) (1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.)

Table of Contents

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format,

may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

1 Scope

This is not a specification, but a non-normative document. It is published by the **Object Management Group (OMG)** to aid in understanding and implementing the OMG specification **Business Process Model and Notation (BPMN) Version 2.0**. The document presents examples that conform to the **Process Modeling Conformance** class as defined in the **BPMN 2.0** specification. The examples are provided in form of **Collaboration** diagrams, **Process** diagrams, and **Choreography** diagrams as well as machine-readable files using the **Extensible Markup Language (XML)**.

2 Conformance

As this is a non-normative document, an implementation, which claims conformance to any of the conformance classes defined in section 2 of the **BPMN 2.0** specification, is NOT REQUIRED to comply to statements made in this document. Furthermore, if there are any inconsistencies between the **BPMN 2.0** specification and this document, the statements of the **BPMN 2.0** specification are considered to be correct.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

Business Process Model and Notation (BPMN) Version 2.0

- OMG, May 2010
  http://www.omg.org/spec/BPMN/2.0

RFC-2119

- Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, IETF RFC 2119, March 1997
  http://www.ietf.org/rfc/rfc2119.txt

4 Additional Information

4.1 Changes to Adopted OMG Specifications

If there are any inconsistencies between the **BPMN 2.0** specification and this document, the statements of the **BPMN 2.0** specification are considered to be correct.

4.2 Acknowledgements

The following companies submitted this document:

- camunda services GmbH
- PNA Group
- SAP AG
- Trisotech, Inc.

The following persons were members of the core teams that contributed to the content of this document:

- John Bulles (PNA Group)
- Jakob Freund (camunda services GmbH)
- Denis Gagné (Trisotech, Inc.)
- Falko Menge (camunda services GmbH)
- Sjir Nijssen (PNA Group)
- Gerardo Navarro-Suarez (camunda services GmbH)
- Ivana Trickovic (SAP AG)

In addition, the following persons contributed valuable ideas and feedback that improved the content and the quality of this document:

- Mariano Benitez (Oracle)
- Conrad Bock (NIST)
- John Hall (Model Systems)
- Stephen A. White (International Business Machines)

5 BPMN in practice

Short introduction / clear scoping statements

This document is non normative and is only meant as support document in interpreting various aspects of BPMN 2.0.

All examples provided herein are non-executable BPMN 2.0 processes.

The serializations of the examples are provided into an accompanying machine-readable zip file.

6 Small Examples introducing Core Concepts

6.1 Shipment Process of a Hardware Retailer

AND, XOR, and OR gateways

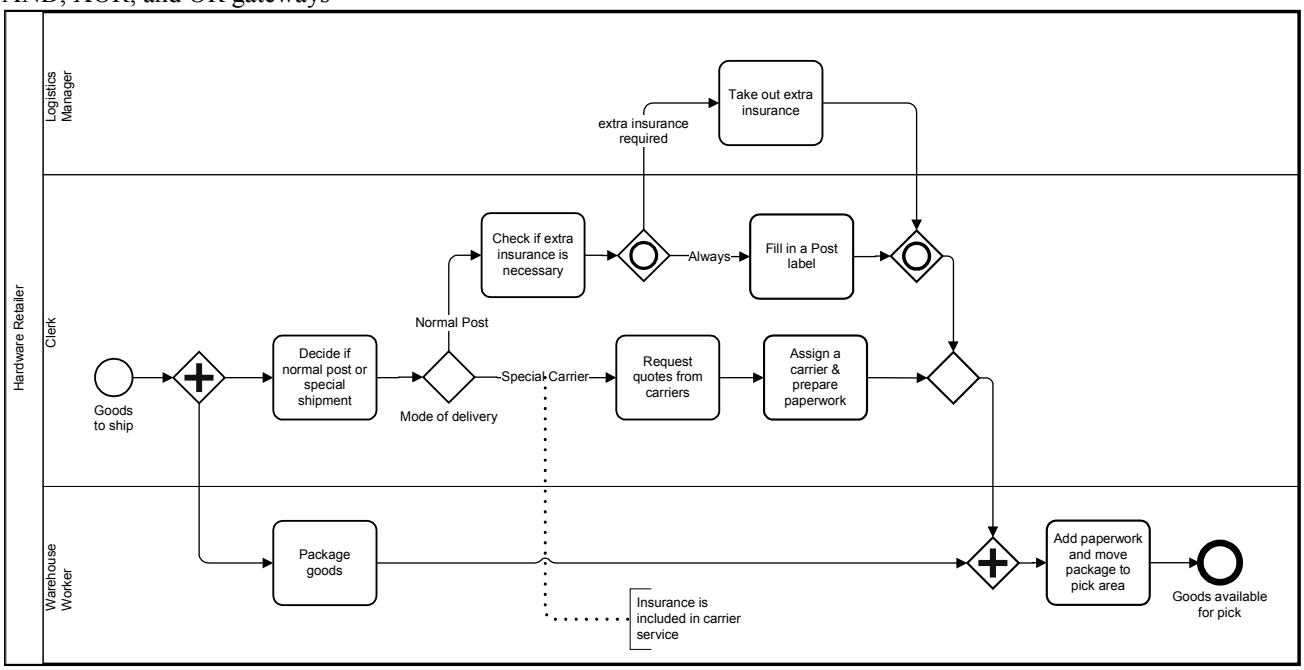


Figure 1: Shipment Process of a hardware retailer

6.2 The Pizza Collaboration

Events, Collaboration, and Message Flow

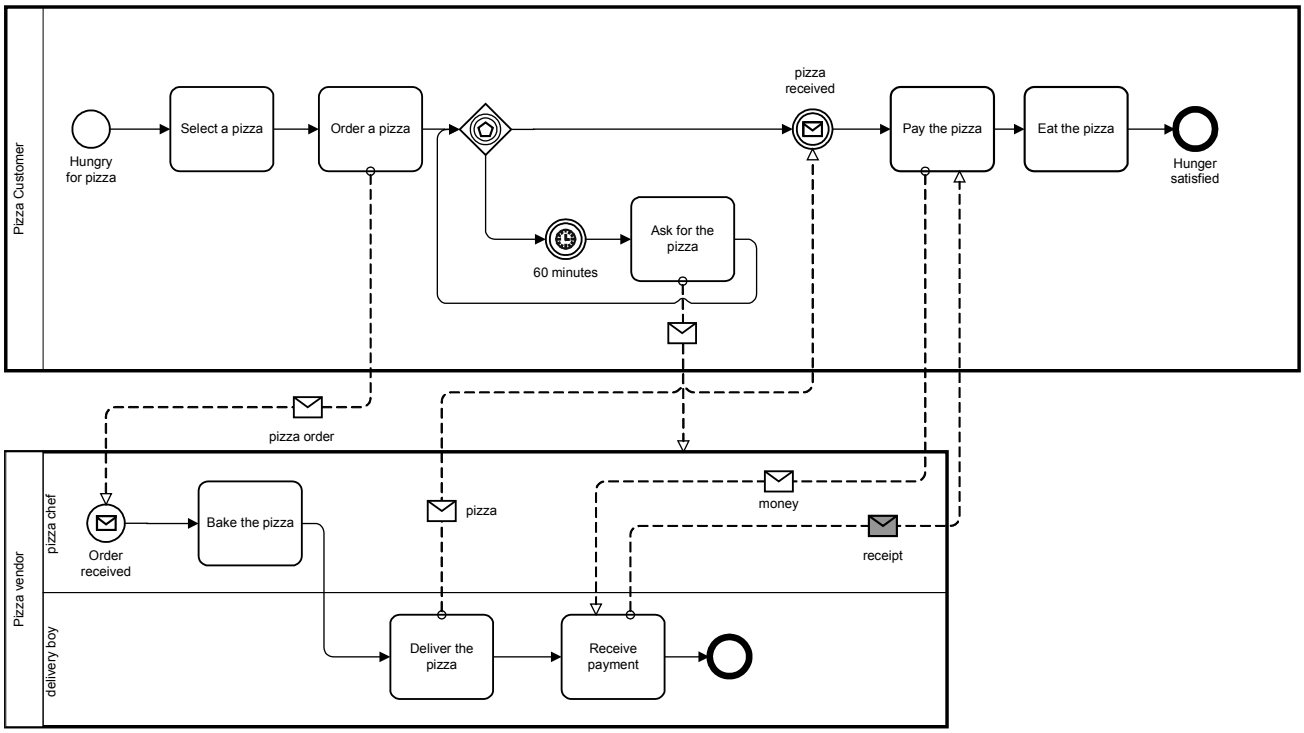


Figure 2: ordering and delivering pizza

6.3 Order Fulfillment and Procurement

Call Activity, Boundary Events, Exception, Escalation, Non-Interrupting Intermediate Events

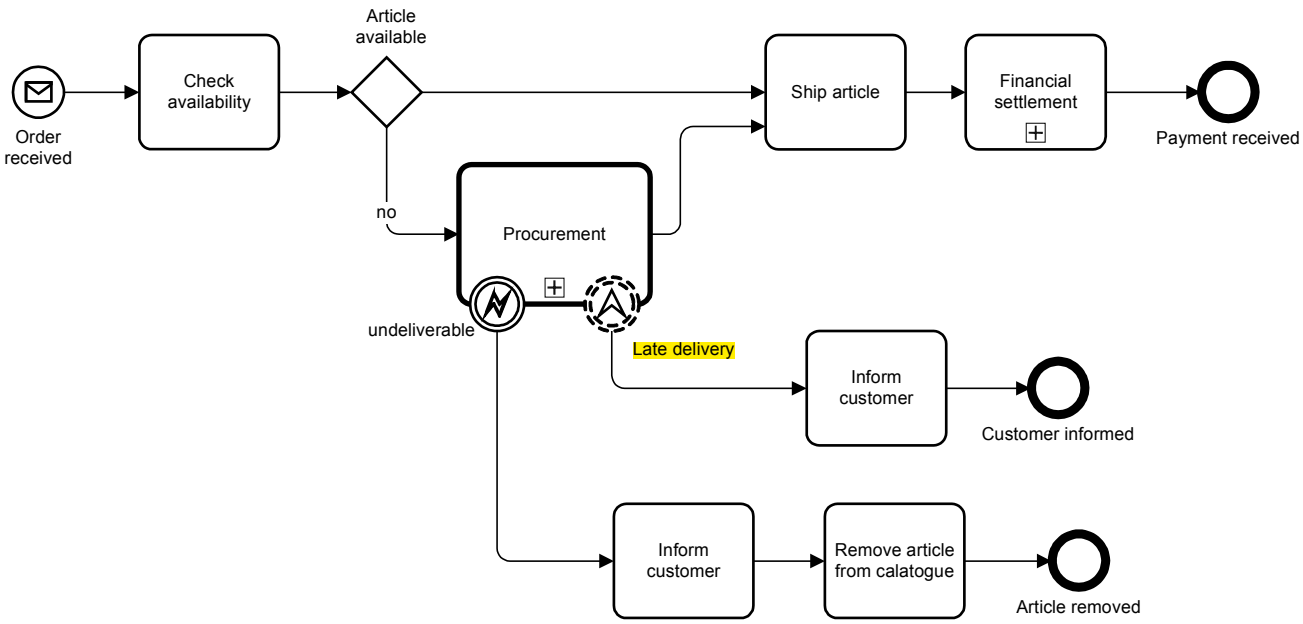


Figure 3: Order Fulfillment

Let us begin with the Order Fulfillment in figure 4. The process starts after receiving an order message and continues to check whether the article is available or not. An available article is shipped to the customer followed by a financial settlement. In case that an article is not available, it has to be procured by calling the procurement sub-process.

The check activity which is followed by the xor gateway is a good example for clarifying the recommended usage of the exclusive gateway. The gateway is not responsible for the determining of the product's availability. Instead, the calculation of the availability is done in the activity before. So, the question if the article is available or not, is answered in that check activity. The gateway works as a router which is based on the result of the previous activity and provides alternative paths.

Notice that the shape of the collapsed sub-process is thickly bordered which means that it is a call activity. It is like a wrapper for a task or a globally defined sub-process.

Another characteristic of the procurement sub-process are the two attached events. By using attached events it is possible to handle events that can spontaneously occur during the execution of an activity or sub-process. Thereby we have to distinguish between interrupting and non-interrupting attached events. Both of them catch and handle the incoming events, but only the non-interrupting type (here it is an escalation event) continues the execution of the activity or sub-process. When the interrupting event type triggers, the execution of the current activity is interrupted and stops.

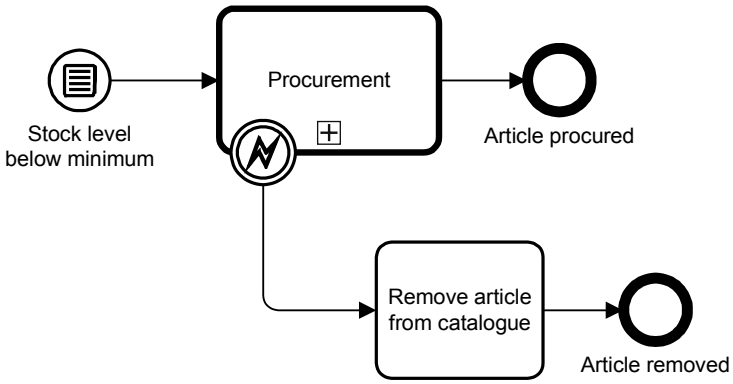


Figure 4: stock maintenance process

The Order Fulfillment process in figure 4 starts after receiving a new order message. The process for the Stock Maintenance (figure 5) is triggered by a conditional start event. It means that the process is instantiated in case that the condition is true, so in this example when the stock level goes below a certain minimum. In order to increase the stock level some products have to be procured. Therefore we use the Procurement process in figure 6 and refer to it by the call activity "Procurement" indicated by the thick border. Similar to the order fulfillment process this process handles the error exception by removing the article from the catalogue. But in this Stock Maintenance process there appears to be no need for the handling of a "late delivery" escalation event. That's why it is left out and not handled. After the procurement sub-process finishes, the stock level is above minimum and the Stock Maintenance process ends.

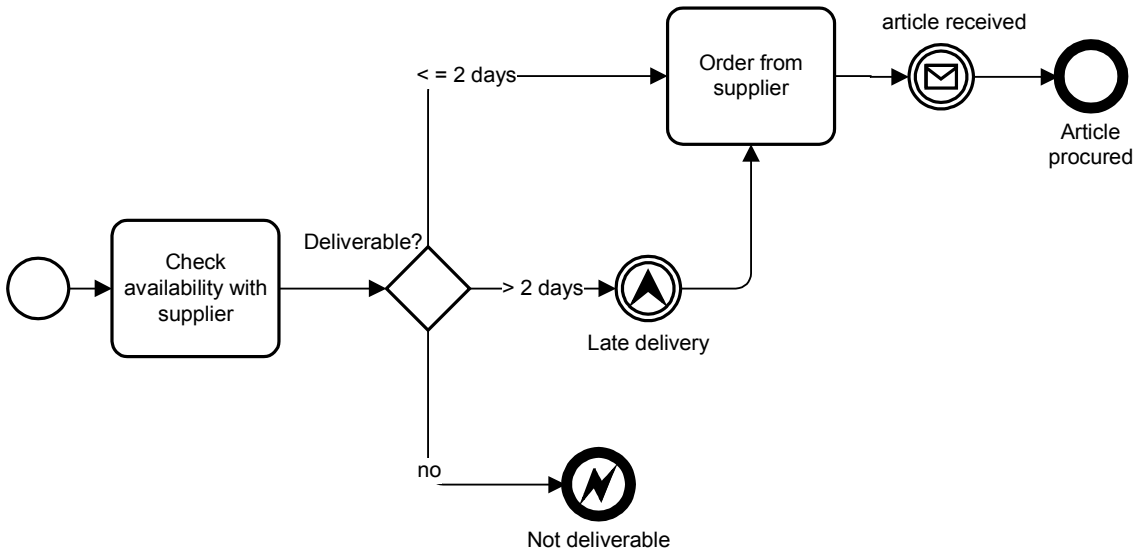


Figure 5: Procurement sub-process

After the Procurement process in figure 6 has started, it continues by asking whether the article is deliverable or not. A non deliverable article leads to an error event. The Procurement process ends at this point by throwing an error event with the errorCode undeliverable. If there were other active threads in the Procurement process, they would be terminated as a result of the error event. But where will the event be caught? The error event does not disappear. Instead it can be caught by an error event that is attached to the nearest parent activity. But the catching error event must have the same errorCode. For example the error event attached to the Order Fulfillment process (figure 4) has the same errorCode and so it handles the undeliverable exception by informing the customer and removing the article from the catalogue. Because of the interrupting type the token does not continue to the activity ship article, instead the token follows the exception handling path, is consumed at the end and the Procurement process ends there.

However, in case that the delivery in the Procurement process lasts more than 2 days an escalation event is thrown saying that the delivery will be late. Similar to the error event, the escalation event has also an escalationCode which is necessary for the connection between throwing and catching escalation events. Contrary to the throwing error event, currently active threads are neither terminated nor affected by the throwing intermediate escalation event. Furthermore, the Procurement process continues its execution by waiting for the delivery. But the thrown event is handled by the nearest parent activity with an attached intermediate escalation event which has the same escalationCode as the thrown escalation event. In Figure 4 the "late delivery" escalation event attached to the Procurement sub-process catches the thrown "late delivery" event. But now, the event is a non-interrupting event. Because of that a new token is produced, follows the path of the escalation handling and informs the customer that the ordered article will be shipped later. When the procurement sub-process finishes, the Order Fulfillment process continues with the shipment of the article and the financial settlement.

7 End-to-end Example: Incident management

starts with a high level model, which is then refined into operational and technical models using Choreography, Collaboration, different types of Tasks & Events-

7.1 High Level Model for End-to-end Process

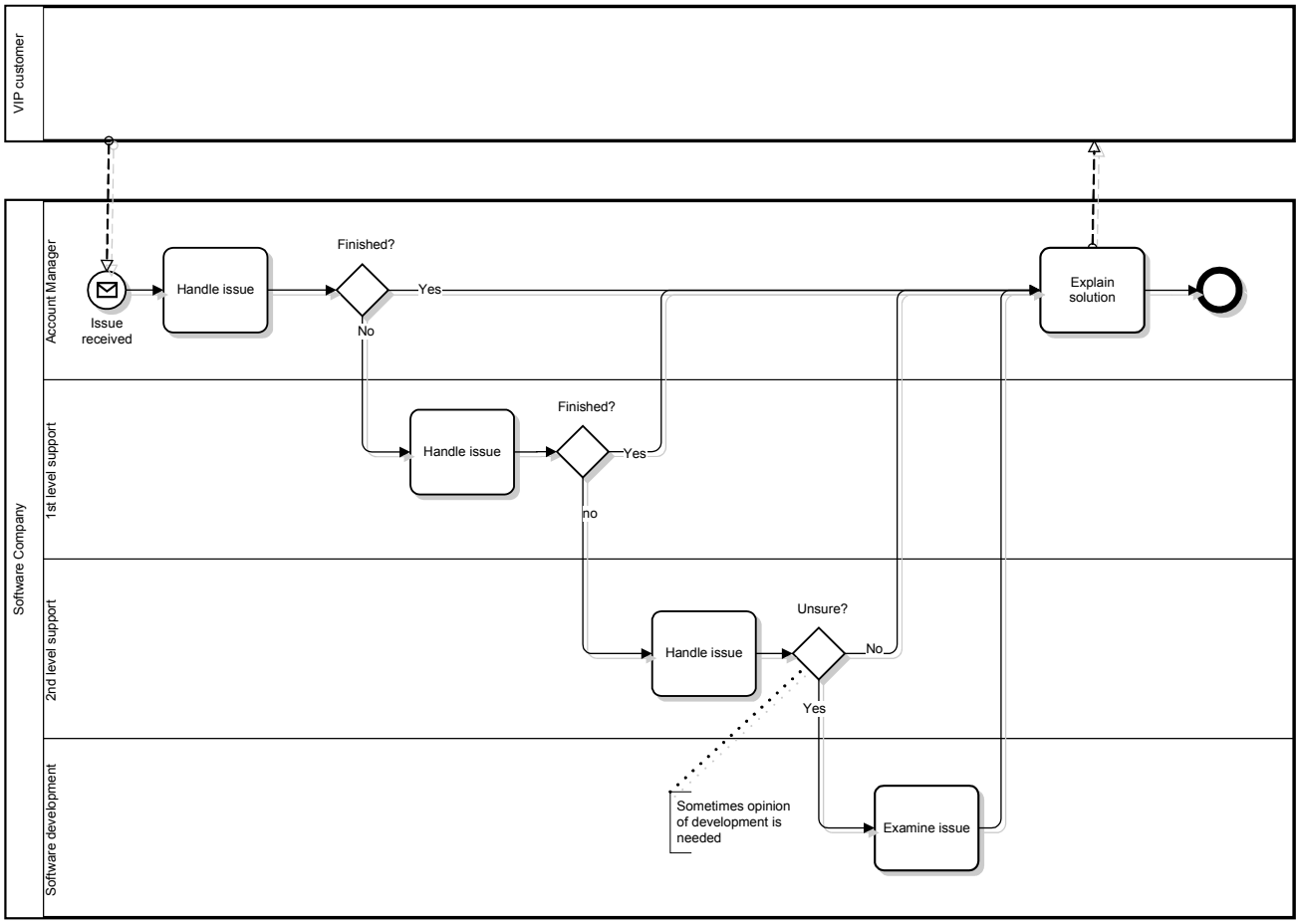


Figure 6: incident management from high level point of view

7.2 Detailed Collaboration and Choreography

Collaboration:

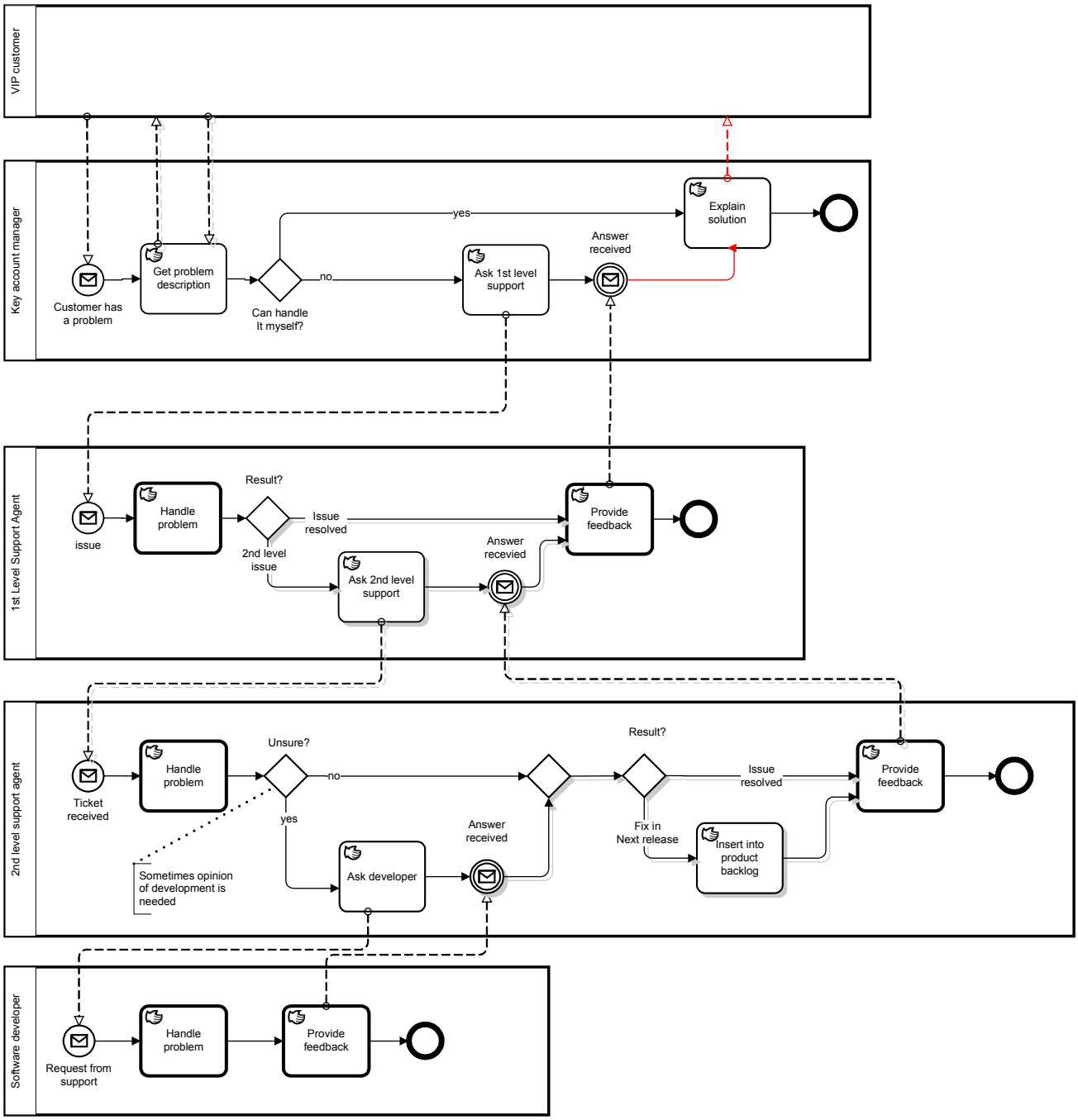


Figure 7: Incident Management as detailed collaboration

Choreography:

7.3 Human-driven vs. system-driven Control Flows

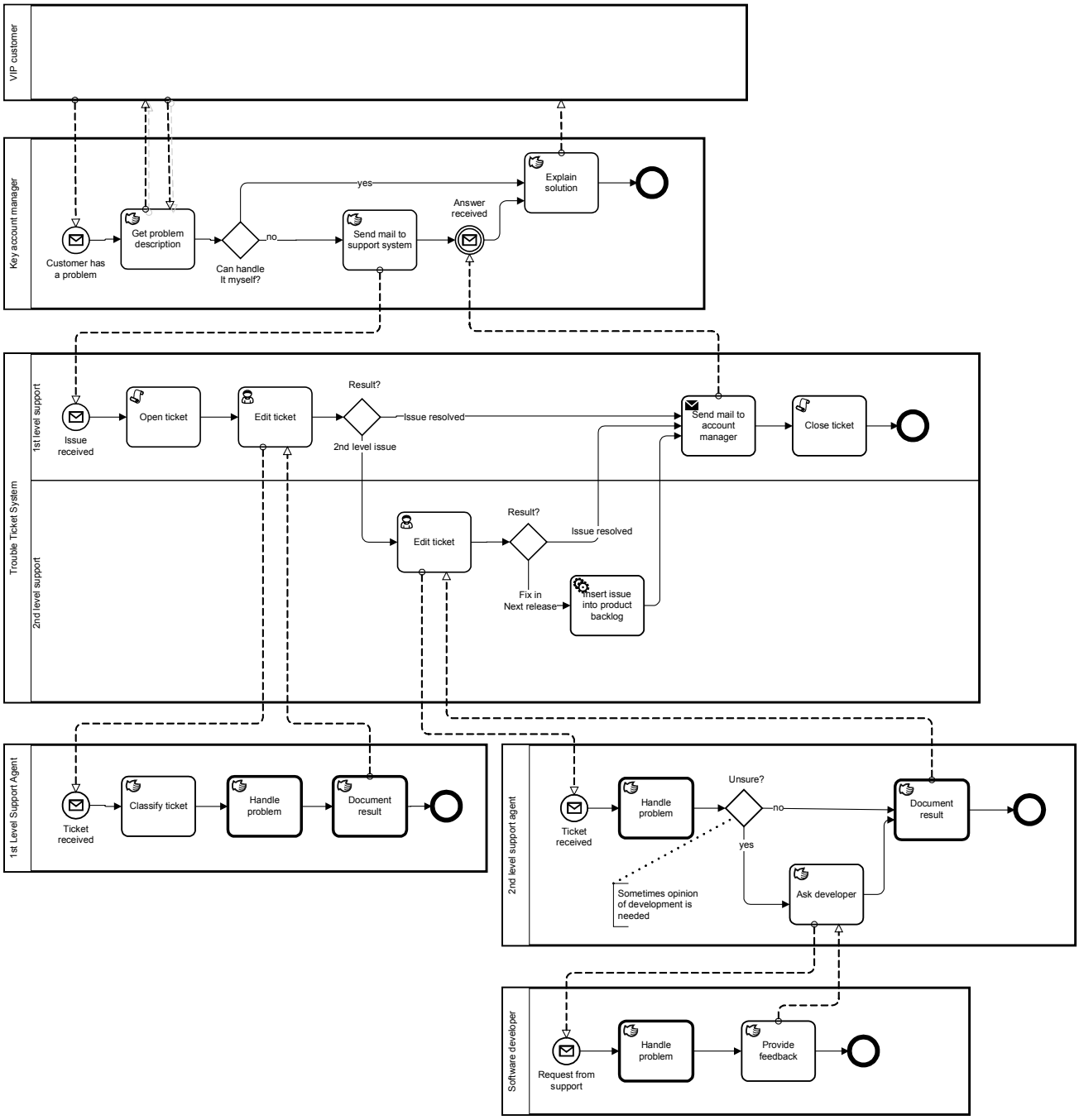


Figure 8: Incident Management with human driven and system driven pools

8 Nobel Prize Example

8.1 The Nobel Prize Process Scenario

The selection of a Nobel Prize Laureate is a lengthy and carefully executed process. The processes slightly differ for each of the six prizes; the results are the same for each of the six categories.
Following is the description for the Nobel Prize in Medicine. The main actors in the processes for Nomination, Selection and Accepting and Receiving the award are the:

- Nobel Committee for Medicine,
- Nominators,
- Specially appointed experts,
- Nobel Assembly and
- Nobel Laureates.

Each year in September, in the year preceding the year the Prize is awarded, around 3000 invitations or confidential nomination forms are sent out by the Nobel Committee for Medicine to selected Nominators.

The Nominators are given the opportunity to nominate one or more Nominees. The completed forms must be made available to the Nobel Committee for Medicine for the selection of the preliminary candidates.

The Nobel Committee for Medicine performs a first screening and selects the preliminary candidates.

Following this selection, the Nobel Committee for Medicine may request the assistance of experts. If so, it sends the list with the preliminary candidates to these specially appointed experts with the request to assess the preliminary candidates' work.

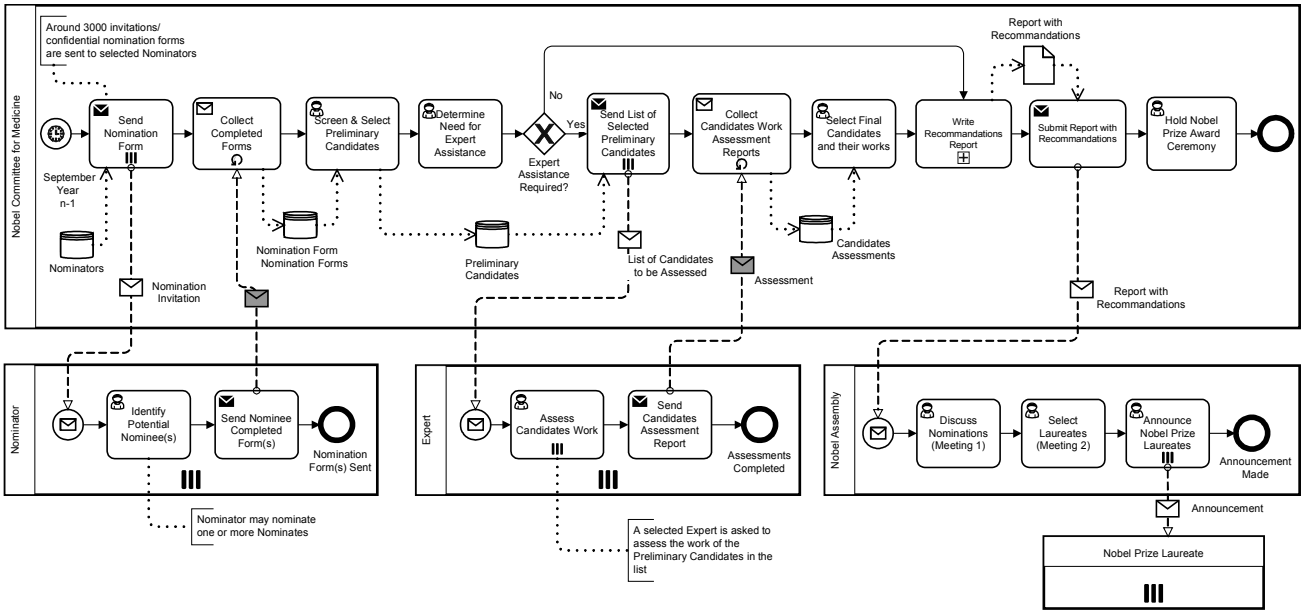
From this, the recommended final candidate laureates and associated recommended final works are selected and the Nobel Committee for Medicine writes the reports with recommendations.

The Nobel Committee for Medicine submits the report with recommendations to the Nobel Assembly. This report contains the list of final candidates and associated works.

The Nobel Assembly chooses the Nobel Laureates in Medicine and associated through a majority vote and the names of the Nobel Laureates and associated works are announced. The Nobel Assembly meets twice for this selection. In the first meeting of the Nobel Assembly the report is discussed. In the second meeting the Nobel Laureates in Medicine and associated works are chosen.

The Nobel Prize Award Ceremony is held, in Stockholm.

8.2 The Nobel Prize Process Diagram



9 Models and Diagrams

The purpose of this chapter is to demonstrate via examples some of the interrelations between models and diagrams. We explore how different BPMN diagrams of the same scenario lead to different serializations of the model.

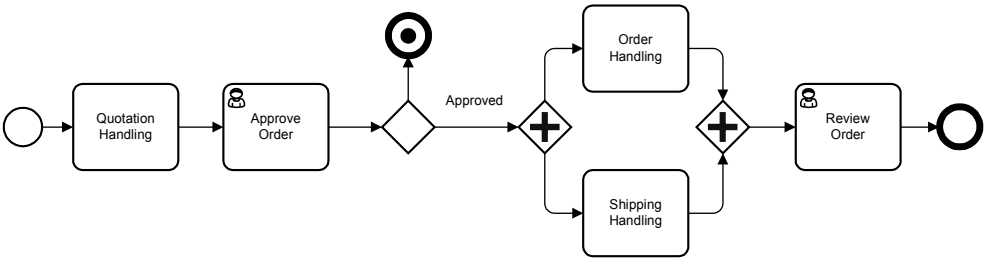
The process scenario used in the examples from this chapter is inspired from figure 10.24 of the BPMN 2.0 Specification document.

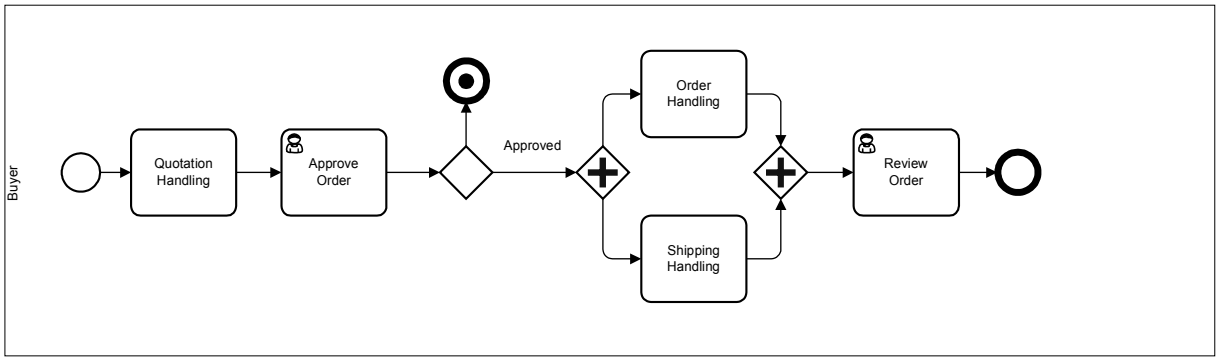
9.1 Lane and Pool

In this section, we explore the use of lanes and pools in a BPMN diagram and their corresponding serializations.

1.1.1 Lane

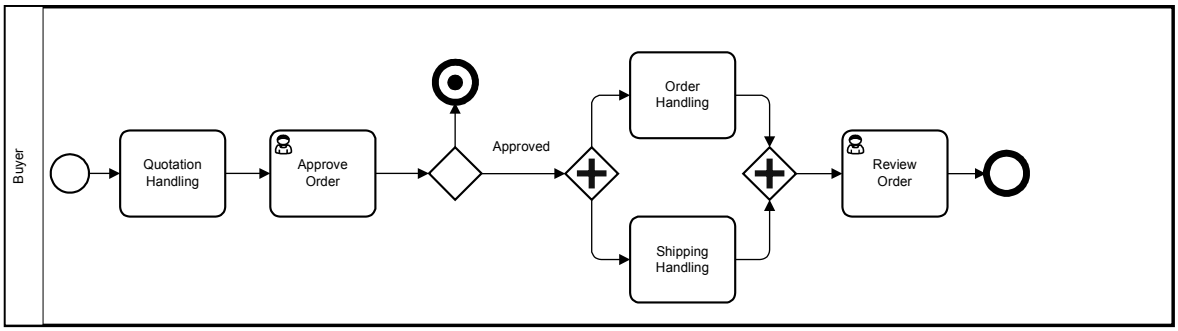
A process can be depicted in a Process Diagram with or without lanes. Both these depictions lead to one process in the model and one diagram of that process. The only difference in the two serializations is that one does not have a Laneset with a lane in it while the other does.





1.1.2 Pool

Pools are only present in Collaboration Diagrams (Collaborations, Choreographies, Conversations). Thus, when depicting the same scenario using a pool, we are in fact producing a Collaboration Diagram. The introduction of a pool in our depiction implies that we are producing a Collaboration Diagram. In fact, this is an incomplete Collaboration, as a Collaboration should be between two or more participants.

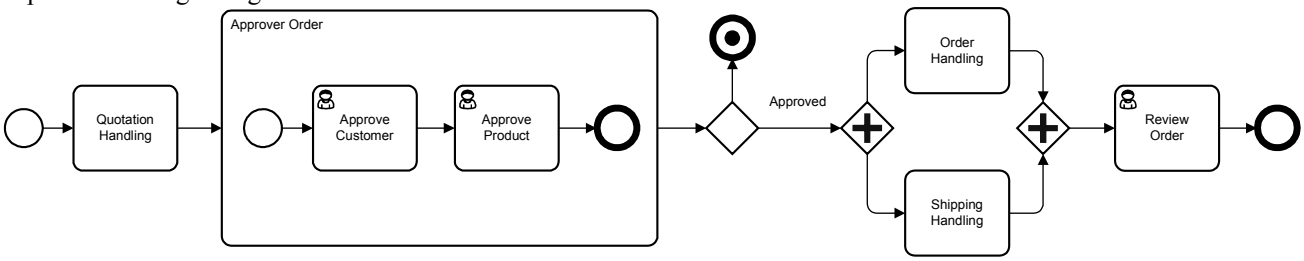


9.2 Sub Process and Call Activity

In this section, we explore the use of Sub Processes (expanded and collapsed) along with Call Activities and show how their content can be depicted in separate diagrams.

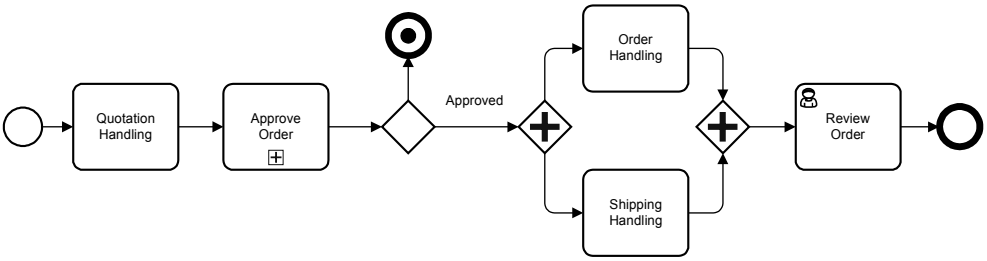
1.1.3 Expanded Sub Process Example

In this example our "Order Process" is depicted with an expanded "Approve Order" Sub Process. This is a single process depicted in a single diagram.

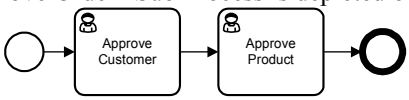


1.1.4 Collapsed Sub Process Example

In this example our "Order Process" is depicted with a collapsed "Approve Order" Sub Process.



While the content (or details) of the "Approve Order" Sub Process is depicted on a separate diagram.

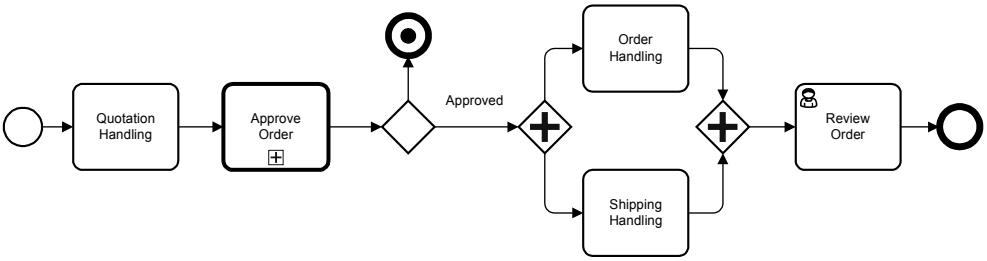


This is a single process depicted into two diagrams: one for the parent process and one for the sub process.

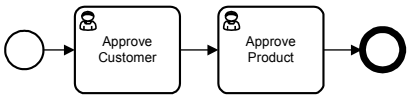
Note that both expanded and collapsed depictions are visual variations of the same single "Order Process".

1.1.5 Call Activity Example

In this example our "Order Process" is depicted with a collapsed Call Activity "Approve Order". This diagram is quite different than the previous example, as here we are introducing the notion of Process re-use. In this case, the "Approve Order" is not a Sub Process of "Order Process" but separate independent process that is called (re-used) within the "Order Process".



The "Approve Order" Process

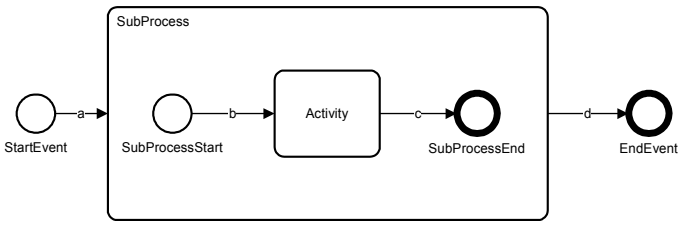


We thus have two processes each in their own diagrams (2 processes, 2 diagrams)

10 Examples from Diagram Interchange Chapter

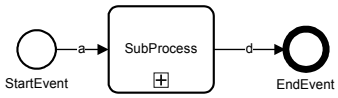
The purpose of this chapter is to provide a subset of the diagrams used into the Notation and Diagrams chapter of the BPMN 2.0 specification along with their serializations.

10.1 Expanded Sub Process Example



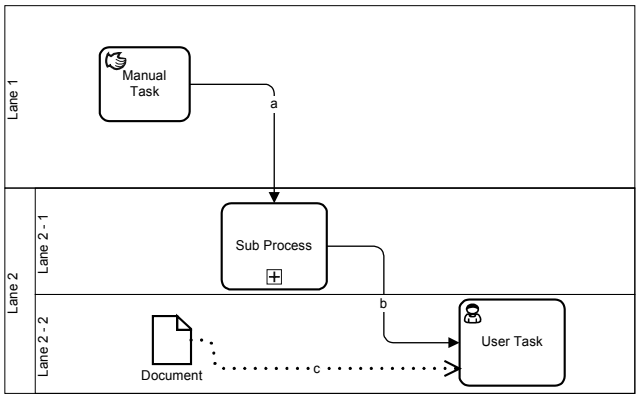
10.2 Collapsed Sub Process Example

1.1.6 Process Diagram

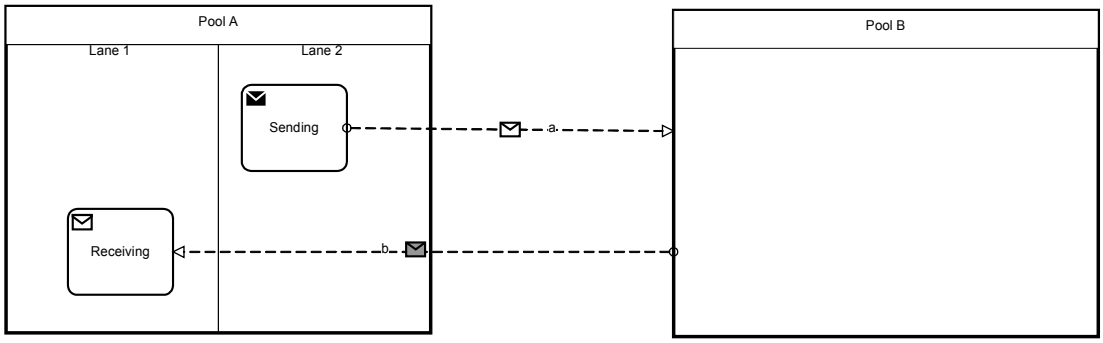


1.1.7 Sub Process Diagram

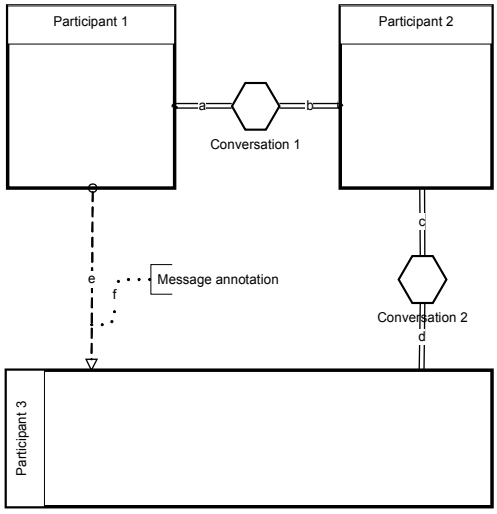
10.3 Multiple Lanes and Nested Lanes Example



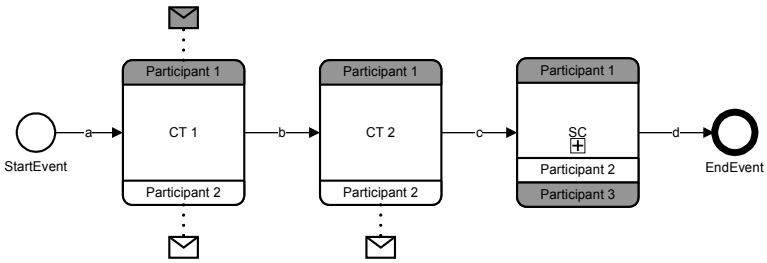
10.4 Vertical Collaboration Example



10.5 Conversation Example



10.6 Choreography Example



11 Travel Booking Example

The purpose of this chapter is to provide an example of in-line event handling via event sub-process constructs.

The process scenario is inspired from figure 10.100 of the BPMN 2.0 Specification document.

11.1 The Travel Booking Scenario

The Travel Agency receives a travel reservation request, including airline transportation and hotel reservation, from a Client.

Following research and evaluation of both flights' and hotel rooms' availability, selected alternatives are packaged and offered to the Client.

The Client has 24 hours to either select a proposed alternative or cancel the request. In case of a cancellation, or after this delay, the Agency updates the Client record to reflect the request cancellation and the Client is notified.

When a selection is made, the Client is asked to provide the Credit Card information. Again, the Client has 24 hours to provide this information or the request is cancelled via the same activities stated before (update and notification).

Having received the Credit Card information, the booking activities take place:

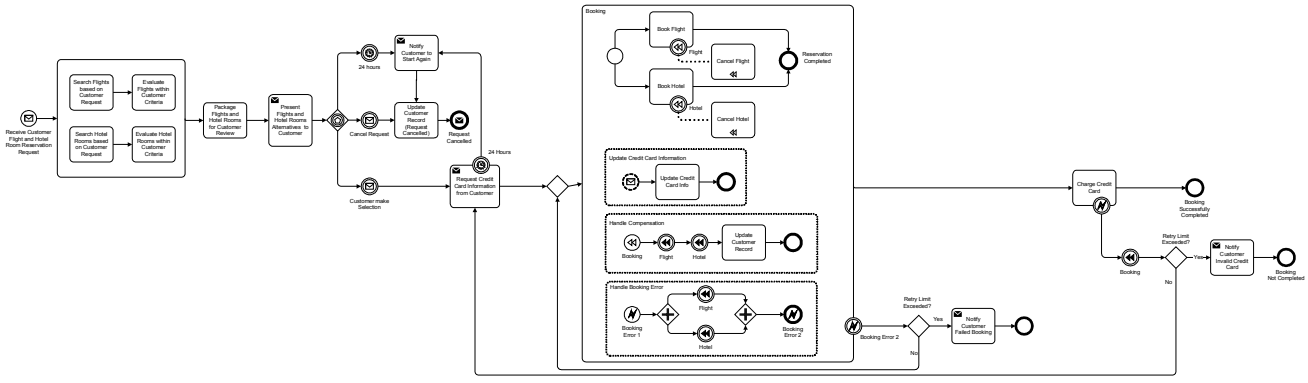
The flight and the hotel room are booked. Measures are taken to insure reservations reversals if problems occur in the booking and payment activities. The Client is also entitled to provide the Agency with Credit Card Information modifications before the booking is completed. Such information will be saved in its record.

If an error arises during the booking activities, the flight and hotel room reservations are reversed and the Client record is updated. The booking is tried again as long as the booking retry limit is not exceeded.

Following successful booking the Reservations are charged on the Client's Credit Card and the process stops following successful confirmation. If an error occurs during this activity the flight and hotel room reservation are reversed. The Client is asked again for the Credit Card Information and the booking is tried again as long as the payment processing retry limit is not exceeded.

In both cases, following the error, when the retry limit is exceeded, the Client is notified and the process stops.

11.2 The Travel Booking Diagram



12 Correlation Examples

12.1 Key-Based Correlation

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions id="def"
      targetNamespace="http://www.example.org/Processes/sellerProcess"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      expressionLanguage="http://www.w3.org/1999/XPath"
      xsi:schemaLocation="http://www.omg.org/bpmn20 schemas/bpmn20.xsd"
      xmlns="http://www.omg.org/bpmn20"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:myData="http://www.example.org/Messages"
      xmlns:tns="http://www.example.org/Processes/sellerProcess">
   <!--    Structures and Messages -->
   <import importType="http://www.w3.org/2001/XMLSchema"
      location="DataDefinitions.xsd"
      namespace="http://www.example.org/Messages"/>
   <import importType="http://schemas.xmlsoap.org/wsdl/"
      location="Interfaces.wsdl"
      namespace="http://www.example.org/Messages"/>
   <itemDefinition id="itemRFQMessage" structureRef="myData:rfqRequest">
      <!-- Single part message -->
   </itemDefinition>
   <itemDefinition id="itemQuoteMessage" structureRef="myData:rfqResponse">
      <!-- Single part message -->
   </itemDefinition>
   <itemDefinition id="itemFaultMessage" structureRef="myData:rfqFault">
      <!-- Single part message -->
   </itemDefinition>
   <itemDefinition id="itemOrderRequest" structureRef="myData:orderRequest">
      <!-- Multi part message -->
   </itemDefinition>
   <itemDefinition id="itemOrderResponse" structureRef="myData:orderResponse">
      <!-- Multi part message -->
   </itemDefinition>
   <message id="msgRFQ" name="RFQ Message" structureRef="tns:itemRFQMessage"/>
   <message id="msgQuote" name="Quote Message"
      structureRef="tns:itemQuoteMessage"/>
   <message id="msgFault" name="Fault Message"
      structureRef="tns:itemFaultMessage"/>
   <message id="msgOrderData" name="Order Data Message"
      structureRef="tns:itemOrderRequest"/>
   <message id="msgOrderConfirmation" name="Order Confirmation Message"
      structureRef="tns:itemOrderResponse"/>
   <message id="msgShippingData" name="Shipping Data Message"
      structureRef="tbd"/>
   <message id="msgShippingConfirmation" name="Shipping Confirmation Message"
      structureRef="tbd"/>
   <!-- Collaboration: Seller entity ("concrete" participant)
      and Buyer/Shipper role ("abstract"/prototypical participants) -->
   <partnerEntity id="theSeller" name="The Seller"/>
   <partnerRole id="aBuyer" name="A Buyer"/>
   <partnerRole id="aShipper" name="A Shipper"/>
```

```
<correlationProperty id="propQuoteID" name="Property Quote ID"
   type="xsd:string">
   <correlationPropertyRetrievalExpression messageRef="tns:msgRFQ">
      <messagePath>/request/quoteID</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression messageRef="tns:msgQuote">
      <messagePath>/response/quoteID</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression messageRef="tns:msgFault">
      <messagePath>/fault/quoteID</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression messageRef="tns:msgOrderData">
      <messagePath>/priceQuotationRef</messagePath>
   </correlationPropertyRetrievalExpression>
</correlationProperty>
<correlationProperty id="propCustomerID" name="Property Customer ID"
   type="xsd:string">
   <correlationPropertyRetrievalExpression messageRef="tns:msgOrderData">
      <messagePath>/customer/id</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression
   messageRef="tns:msgOrderConfirmation">
      <messagePath>/customerID</messagePath>
   </correlationPropertyRetrievalExpression>
</correlationProperty>
<correlationProperty id="propOrderID" name="Property Order ID"
   type="xsd:string">
   <correlationPropertyRetrievalExpression messageRef="tns:msgOrderData">
      <messagePath>/order/orderID</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression
   messageRef="tns:msgOrderConfirmation">
      <messagePath>/order/orderID</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression messageRef="tns:msgShippingData">
      <messagePath>tbd</messagePath>
   </correlationPropertyRetrievalExpression>
   <correlationPropertyRetrievalExpression
   messageRef="tns:msgShippingConfirmation">
      <messagePath>tbd</messagePath>
   </correlationPropertyRetrievalExpression>
</correlationProperty>
<collaboration id="sellerCollab">
   <participant id="seller" name="Seller" partnerEntityRef="tns:theSeller"
   processRef="tns:sellerProcess">
      <interfaceRef>tns:sellerServiceInterface</interfaceRef>
   </participant>
   <participant id="buyer" name="Buyer" partnerRoleRef="tns:aBuyer"/>
   <participant id="shipper" name="Shipper" partnerRoleRef="tns:aShipper">
      <interfaceRef>tns:shipperServiceInterface</interfaceRef>
   </participant>
   <messageFlow id="mf1" messageRef="tns:msgRFQ" sourceRef="tns:aBuyer"
   targetRef="tns:receiveQuoteRequest"/>
   <messageFlow id="mf2" messageRef="tns:msgQuote" sourceRef="tns:sendQuote"
   targetRef="tns:aBuyer"/>
   <messageFlow id="mf3" messageRef="tns:msgFault" sourceRef="tns:sendFault"
   targetRef="tns:aBuyer"/>
```

```
    <messageFlow id="mf4" messageRef="tns:msgOrderData" sourceRef="tns:aBuyer"
    targetRef="tns:receiveOrderRequest"/>
    <messageFlow id="mf5" messageRef="tns:msgOrderConfirmation"
    sourceRef="tns:sendOrderResponse" targetRef="tns:aBuyer"/>
    <messageFlow id="mf6" messageRef="tns:msgShippingData"
    sourceRef="tns:sendShippingRequest" targetRef="tns:aShipper"/>
    <messageFlow id="mf7" messageRef="tns:msgShippingConfirmation"
    sourceRef="tns:aShipper" targetRef="tns:receiveShippingConfirmation"/>
    <!--   Conversations -->
    <conversation id="conversationQuoteRequest">
       <messageFlowRef>tns:mf1</messageFlowRef>
       <messageFlowRef>tns:mf2</messageFlowRef>
       <messageFlowRef>tns:mf3</messageFlowRef>
       <messageFlowRef>tns:mf4</messageFlowRef>
       <correlationKey id="correlQuote" name="Quote CorrelationKey">
          <correlationPropertyRef>tns:propQuoteID</correlationPropertyRef>
       </correlationKey>
    </conversation>
    <conversation id="conversationOrderHandling">
       <messageFlowRef>tns:mf4</messageFlowRef>
       <messageFlowRef>tns:mf5</messageFlowRef>
       <correlationKey id="correlOrder" name="Order Correlation Key">
          <correlationPropertyRef>tns:propCustomerID</correlationPropertyRef>
          <correlationPropertyRef>tns:propOrderID</correlationPropertyRef>
       </correlationKey>
    </conversation>
    <conversation id="conversationShipmentRequest">
       <messageFlowRef>tns:mf6</messageFlowRef>
       <messageFlowRef>tns:mf7</messageFlowRef>
       <correlationKey id="correlShipment" name="Shipment Correlation Key">
          <correlationPropertyRef>tns:propOrderID</correlationPropertyRef>
       </correlationKey>
    </conversation>
 </collaboration>
 <!--    Interfaces -->
 <!--The interface of the Seller Process-->
 <interface id="sellerServiceInterface" name="Seller Service Interface">
    <operation id="requestQuoteOp" name="Request Quote Operation">
       <inMessageRef>tns:msgRFQ</inMessageRef>
       <outMessageRef>tns:msgQuote</outMessageRef>
       <errorRef>tns:msgFault</errorRef>
    </operation>
    <operation id="orderOp" name="Order Operation">
       <inMessageRef>tns:msgOrderData</inMessageRef>
       <outMessageRef>tns:msgOrderConfirmation</outMessageRef>
    </operation>
 </interface>
 <interface id="shipperServiceInterface" name="Shipper Service Interface">
    <operation id="requestShippingOp" name="Request Shipping Operation">
       <inMessageRef>tns:msgShippingData</inMessageRef>
       <outMessageRef>tns:msgShippingConfirmation</outMessageRef>
    </operation>
 </interface>
 <!--    Correlation Keys and associated expressions -->
 <!--    Process Definition -->
 <process id="sellerProcess" name="Seller process"
    definitionalCollaborationRef="tns:sellerCollab">
```

```
      <!--Receive quote request message from caller.-->
      <receiveTask id="receiveQuoteRequest" name="Receive Quote Request"
            instantiate="true"
            messageRef="tns:msgRFQ"
            operationRef="tns:requestQuoteOp"/>
      <sequenceFlow targetRef="decision1" sourceRef="receiveQuoteRequest"/>
      <!--Decide whether quote is available and can be returned, or not.
(The actual processing logic is omitted from the example.)-->
      <exclusiveGateway id="decision1" gatewayDirection="diverging"
            default="noQuote"/>
      <sequenceFlow id="quote" targetRef="sendQuote" sourceRef="decision1">
         <conditionExpression>Quote available and okay.</conditionExpression>
      </sequenceFlow>
      <sequenceFlow id="noQuote" targetRef="sendFault" sourceRef="decision1"/>
      <!-- Respond successful quote back to caller â€" this is a reply, so use
same service reference and operation as in associated receive.-->
      <sendTask id="sendQuote" name="Send Quote" messageRef="tns:msgQuote"
            operationRef="tns:requestQuoteOp"/>
      <sequenceFlow targetRef="eventWait" sourceRef="sendQuote"/>
      <!--This is a reply, so use same service reference and operation as in
associated receive.-->
      <sendTask id="sendFault" name="Send Fault" messageRef="tns:msgFault"
            operationRef="tns:requestQuoteOp"/>
      <!-- Respond error back to caller-->
      <sequenceFlow targetRef="eventWait" sourceRef="sendFault"/>
      <!--Wait for another quote request, an order, or a timeout-->
      <eventBasedGateway id="eventWait" gatewayDirection="mixed"/>
      <sequenceFlow targetRef="receiveQuoteRequest" sourceRef="eventWait"/>
      <sequenceFlow targetRef="receiveOrderRequest" sourceRef="eventWait"/>
      <sequenceFlow targetRef="timeout" sourceRef="eventWait"/>
      <!--Timeout and end-->
      <intermediateCatchEvent id="timeout">
         <timerEventDefinition>
            <timeDate>PD4h</timeDate>
         </timerEventDefinition>
      </intermediateCatchEvent>
      <sequenceFlow targetRef="end1" sourceRef="timeout"/>
      <endEvent id="end1"/>
      <!-- Receive an order message-->
      <receiveTask id="receiveOrderRequest" name="Receive Order Request"
            messageRef="tns:msgOrderData"/>
      <sequenceFlow targetRef="fork" sourceRef="receiveOrderRequest"/>
      <parallelGateway id="fork" gatewayDirection="diverging"/>
      <sequenceFlow targetRef="sendOrderResponse" sourceRef="fork"/>
      <sequenceFlow targetRef="sendShippingRequest" sourceRef="fork"/>
      <!-- Send order confirmation â€" this is a reply, so use same service
reference and operation as in associated receive.-->
      <sendTask id="sendOrderResponse" name="Send Order Response"
            messageRef="tns:msgOrderConfirmation"/>
      <sequenceFlow targetRef="join" sourceRef="sendOrderResponse"/>
      <!-- Trigger Shipping-->
      <sendTask id="sendShippingRequest" name="Send Shipping Request"
            messageRef="tns:msgShippingData"/>
      <sequenceFlow targetRef="receiveShippingConfirmation"
            sourceRef="sendShippingRequest"/>
      <!-- Receive Shipment Notification-->
      <receiveTask id="receiveShippingConfirmation"
```

```
            name="Receive Shipping Confirmation"
            messageRef="tns:msgShippingConfirmation"/>
      <sequenceFlow targetRef="join" sourceRef="receiveShippingConfirmation"/>
      <parallelGateway id="join" gatewayDirection="converging"/>
      <sequenceFlow targetRef="end2" sourceRef="join"/>
      <endEvent id="end2"/>
   </process>
</definitions>
```

12.2 Context-Based Correlation

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions id="def"
      targetNamespace="http://www.example.org/Processes/subscriberProcess"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      expressionLanguage="http://www.w3.org/1999/XPath"
      xsi:schemaLocation="http://www.omg.org/bpmn20 schemas/bpmn20.xsd"
      xmlns="http://www.omg.org/bpmn20"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:myData="http://www.example.org/Messages"
      xmlns:tns="http://www.example.org/Processes/subscriberProcess">
  <import importType="http://www.w3.org/2001/XMLSchema"
      location="DataDefinitions.xsd"
      namespace="http://www.example.org/Messages" />
  <import importType="http://schemas.xmlsoap.org/wsdl/"
      location="Interfaces.wsdl"
      namespace="http://www.example.org/Messages" />
  <itemDefinition id="topicMessage" structureRef="myData:topic">
      <!--  Single part message -->
  </itemDefinition>
  <message id="msgTopic" name="Topic Message" structureRef="tns:topicMessage" />
  <!-- Collaboration: Subscriber entity ("concrete" participant) and Provider
      role ("abstract" participant)  -->
  <partnerEntity id="theSubscriber" name="The Subscriber" />
  <partnerRole id="aProvider" name="A Provider" />
  <correlationProperty id="propTopicID" name="Property Topic ID" type="xsd:int">
      <correlationPropertyRetrievalExpression messageRef="tns:msgTopic">
            <messagePath>/request/topic/ID</messagePath>
      </correlationPropertyRetrievalExpression>
  </correlationProperty>
  <collaboration id="subscriberCollab">
      <participant id="subscriber" name="Subscriber"
            partnerEntityRef="tns:theSubscriber"
            processRef="tns:subscriberProcess">
            <interfaceRef>tns:subscriberServiceInterface</interfaceRef>
      </participant>
      <participant id="provider" name="Provider" partnerRoleRef="tns:aProvider"
/>
      <messageFlow id="mf1" messageRef="tns:msgTopic"
            sourceRef="tns:aProvider" targetRef="tns:receiveFirstTopic" />
      <messageFlow id="mf2" messageRef="tns:msgTopic"
            sourceRef="tns:aProvider" targetRef="tns:receiveNextTopic" />
  </collaboration>
  <!-- Conversations  -->
  <conversation id="conversationTopic">
```

```
      <messageFlowRef>tns:mf1</messageFlowRef>
      <messageFlowRef>tns:mf2</messageFlowRef>
      <correlationKey id="correlTopic" name="Topic CorrelationKey">
            <correlationPropertyRef>tns:propTopicID</correlationPropertyRef>
      </correlationKey>
  </conversation>
  <!-- Interfaces  -->
  <!-- The interface of the Seller Process   -->
  <interface id="subscriberServiceInterface" name="Subscriber Service Interface">
  <operation id="topicCallbackOp" name="Topic Callback Operation">
      <inMessageRef>tns:msgTopic</inMessageRef>
  </operation>
  </interface>
  <!-- Process Definition  -->
  <process id="subscriberProcess" name="Subscriber Process"
      definitionalCollaborationRef="tns:subscriberCollab">
  <!--  Correlation subscription definition -->
      <correlationSubscription id="correlTopicSubscription">
            <correlationKeyRef>tns:correlTopic</correlationKeyRef>
            <correlationPropertyBinding id="correlTopicSubscriptionBinding"
correlationPropertyRef="tns:propTopicID">
                  <dataPath>tns:nextTopicId</dataPath>
            </correlationPropertyBinding>
       </correlationSubscription>
  <!--  Definition of the topic ID and summary data objects
  -->
      <dataObject id="summary" isCollection="true" />
      <dataObject id="nextId" />
  <!--  Receive initial topic   -->
  <receiveTask id="receiveFirstTopic" name="Receive Initial Topic"
instantiate="true" messageRef="tns:msgTopic" operationRef="tns:postTopic">
       <ioSpecification id="receiveFirstTopicIO">
             <dataOutput name="topic" />
            <dataOutput name="ID" />
      </ioSpecification>
      <dataOutputAssociation from="receiveFirstTopicIO/topic" to="summary" />
      <dataOutputAssociation from="increment(receiveFirstTopicIO/ID, 1)"
to="nextId" />
  </receiveTask>
  <sequenceFlow targetRef="receiveNextTopic" sourceRef="receiveFirstTopic" />
  <!--  Collect 10 subsequent topics and append to summary data object
  -->
  <receiveTask id="receiveNextTopic" name="Receive Next Topic"
messageRef="tns:msgTopic" operationRef="tns:postTopic">
  <multiInstanceLoopCharacteristics isSequential="true" loopCardinality="10" />
  <ioSpecification id="receiveNextTopicIO">
      <dataOutput name="topic" />
      <dataOutput name="ID" />
  </ioSpecification>
  <dataOutputAssociation from="receiveNextTopicIO/topic" to="summary" />
  <dataOutputAssociation from="increment(receiveNextTopicIO/ID, 1)" to="nextId"
/>
  </receiveTask>
  <sequenceFlow targetRef="processSummary" sourceRef="receiveNextTopic" />
  <!--  Process results
  -->
  <task id="processSummary">
```

```
      <ioSpecification id="processSummaryIO">
            <dataInput name="summary" />
      </ioSpecification>
      <dataInputAssociation from="summary" to="processSummaryIO/summary" />
 </task>
 <sequenceFlow targetRef="end" sourceRef="processSummary" />
 <!--   Terminate process
 -->
 <endEvent id="end" />
 </process>
</definitions>
```

Annex A: XML Serializations for all presented Models

(informative)

A.1 Machine-readable XML Serializations

The XML serializations for all models are provided in machine-readable form as a separate zip file, which has the OMG Document Numer dtc/2010-05-xx.