

XPDL and BPMN

Stephen A. White, SeeBeyond, United States

ABSTRACT

The Business Process Modeling Notation (BPMN) specification provides a graphical notation for expressing business processes in a Business Process Diagram (BPD). The objective of BPMN is to support process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation to underlying the constructs of execution languages, such as BPEL4WS and Business Process Management Language (BPML). As the WfMC works with BPMI, a mapping from BPMN to XML Process Definition Language (XPDL) will also be created.

INTRODUCTION

The Workflow Management Coalition has been developing workflow specifications for many years. These specifications are designed for the developers of workflow software products to implement solutions that are consistent, complete, and interoperable with other systems.

The main focus of this paper is the specification for Interface 1, the recently completed XPDL 1.0. This specification defines how business processes are defined through modeling tools in order to be executed by a workflow engine. This means that if a business process is defined in XPDL through a modeling tool, then a workflow engine can execute that business process if both software tools comply with the XPDL specification. In addition, a third tool designed for monitoring business processes can also track the same business process when the XPDL definition is used in conjunction with the WfMC Interface 5 specification. Thus, XPDL and other WfMC specifications provide for the interoperability of workflow oriented software systems.

Although the XPDL specification defines the process activities, how they are performed, and the sequence in which they occur, the specification does not define how the process should be visualized. Thus, the individuals who are responsible for the development, maintenance, and management of business processes may see a different representation of the same business process as the move between different software tools and systems. Large organizations may have multiple workflow environments, particularly if the organization has grown through acquisition of other organizations. Thus, it is possible the same individual may see multiple business process representations in the course of their responsibilities. This means that there is a lack of software “interoperability” from the point of view of these individuals.

Interoperability, in this sense, means that those individuals responsible for the development, maintenance, and management of business processes can see a representation of business processes in any workflow environment and instantly recognize and understand the business process without any

mental translation between notations. Such interoperability will reduce training and will reduce potential errors in the development and management of the processes.

A STANDARD BUSINESS PROCESS NOTATION

The need for the interoperation of business processes at the human level, in addition to the software level, can be solved with standardization of the Business Process Modeling Notation (BPMN) being developed by the Business Process Management Initiative¹ (BPMI). BPMN is a flow chart modeling notation designed for use by the people who design and manage business processes. BPMN also provides a formal mapping to execution languages of BPM Systems, such as BPEL4WS and BPML. Thus, BPMN would provide a standard visualization mechanism for business processes defined in an execution optimized business process language.

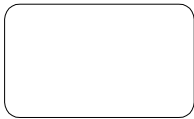
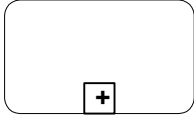
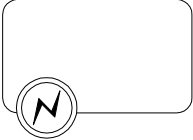
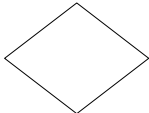
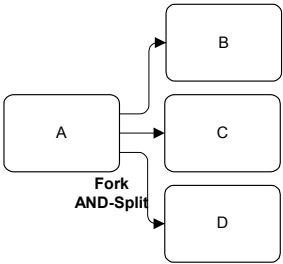
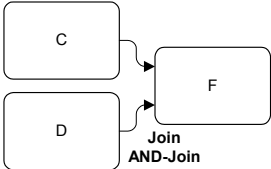
In 2002, BPMI and WfMC agreed to work together. As part of this agreement, the WfMC has accepted BPMN as a notation for XPD L and will work with BPMI to enhance BPMN to manage workflow issues that are not inherently handled by BPEL4WS or BPML. A mapping between BPMN and XPD L will also be created. This paper is the *first look* at the relationship between XPD L and BPMN and the mapping between them.

MAPPING FROM BPMN TO XPD L

Structurally, BPMN and XPD L are very similar; both being flow-chart structures. In fact, the mapping between BPMN and XPD L is much more straightforward than the mapping between BPMN and BPEL4WS or BPML. In general, we shall see the following mappings from BPMN to XPD L (see Table 1):

BPMN Graphical Object	Mapping to XPD L
<p>The details of a Pool or an Expanded Sub-Process</p>	<pre><WorkflowProcess/></pre>
<p>Start Event</p>	<pre><Activity> <Route/> </Activity></pre>
	<pre><Transition/></pre>

¹ Business Process Management Initiative: www.bpmi.org

Sequence Flow	
 <p>Task</p>	<pre> <Activity> <Implementation> <Tool/> <Performer/> </Implementation> </Activities> </pre>
 <p>Sub-Process</p>	<pre> <Activity> <Implementation> <SubFlow/> </Implementation> </Activities> </pre>
 <p>Intermediate Event attached to activity boundary</p>	<pre> <Activity> <Implementation/> <TransitionRestriction> <Split Type="XOR"/> </TransitionRestriction> </Activities> Combined with a: <Transition> <Condition Type="EXCEPTION"/> </Transition> </pre>
 <p>Decision</p>	<pre> <Activity> <Route/> <TransitionRestriction> <Split Type="XOR"/> </TransitionRestriction> </Activities> Combined with a: <Transition> <Condition/> </Transition> </pre>
 <p>Fork AND-Split</p>	<pre> <Activity> <Implementation/> <TransitionRestriction> <Split Type="AND"/> </TransitionRestriction> </Activities> </pre>
 <p>Join AND-Join</p>	<pre> <Activity> <Implementation/> <TransitionRestriction> <Join Type="AND"/> </TransitionRestriction> </Activities> </pre>

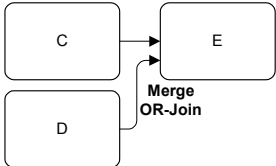

	<pre> <Activity> <Implementation/> <TransitionRestriction> <Join Type="XOR"/> </TransitionRestriction> </Activities> </pre>
 <p>End Event</p>	<pre> <Activity> <Route/> </Activity> </pre>

Table 1: BPMN Objects and their Mappings to XPDL

Applications assigned to the Tasks will fit into the *tool (implementation)* element of an *activity*. These applications and their parameters are not graphically depicted within BPMN, but are properties of the Tasks in the diagram. These and other properties can be captured by a modeling tool and then used in mapping to XPDL to generate the appropriate XML elements. End-Users assigned to the Tasks will fit into the *performer* element of the activity.

BPMN WORKFLOW EXAMPLE

To make the relationships between BPMN and XPDL clearer, an example of a business process that was modeled with BPMN will be analyzed and mapped to XPDL. The process that will be described is a process that the WfMC has been using for an example XPDL.

The BPMN Business Process Diagram (BPD) was derived from a pre-existing XPDL, rather than the other way around. However, BPMN handles some flow elements differently than the model from which the XPDL file was originally derived. Because of this, the XPDL that would be created from a mapping from the BPD will be slightly different than the original XPDL. Thus, the XPDL examples presented in this paper will not be exactly the same as the sample XPDL provided by the WfMC.

The BPD presented is a process for process a customer order electronically (see Figure 1). This Process will provide examples for many of the features of BPMN and how they map to XPDL. The sections below will highlight these features as we describe how the Process works.

Note: The mapping of BPMN to executable XML languages is still a work-in-progress. Thus, the mapping and language relationships defined in this paper may change in the future. The intent of these examples is to make the reader familiar with the general concepts of BPMN elements and their relationship to XPDL elements.

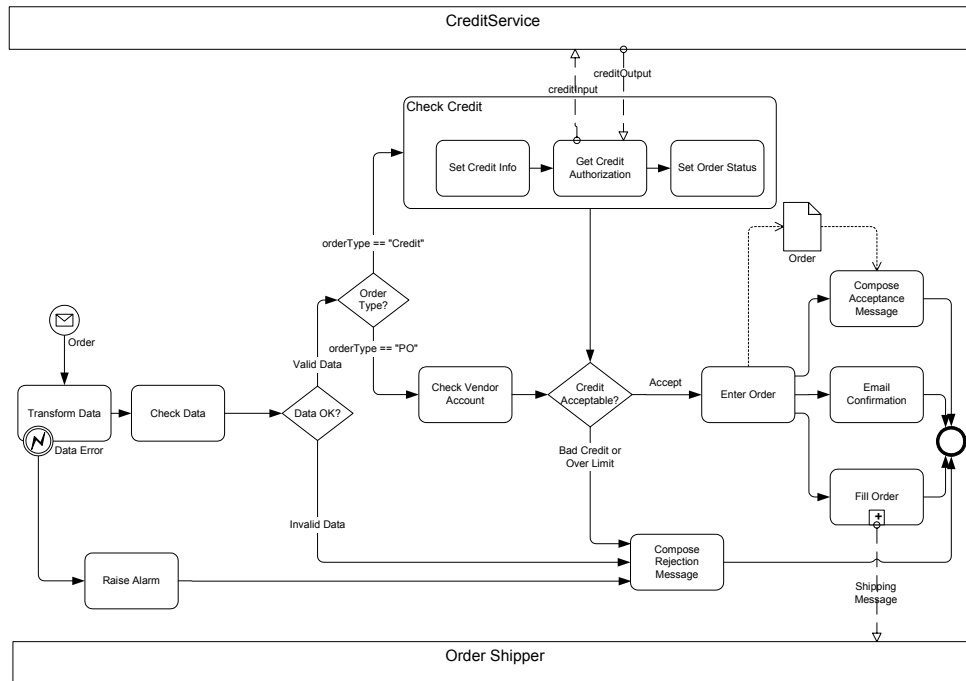


Figure 1: E-Order Process

The Process has a point of view that is from the perspective of the company processing the order. From that point of view, the credit service and the order shipper are considered as external Participants (shown as BPMN Pools) who will be communicated with by messages (shown as Message Flow).

The sections below will isolate different sections of the overall Process. Each of the process segments will be described and then mapped to XPDL.

THE BEGINNING OF THE PROCESS

The Process starts with an order being received (see Figure 2).

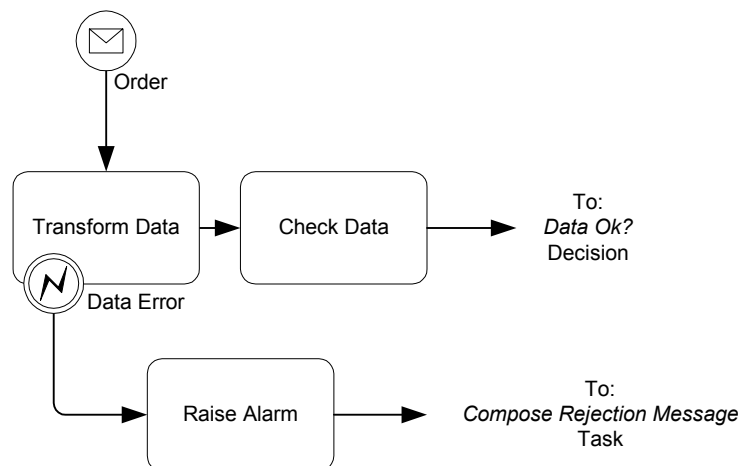


Figure 2: The Start of the E-Order Process

The order data is then sent through an application to transform the data within the through a “Transform Data” Task. Order data is passed to the application and result information is returned. Through the application, this Task may generate a Process Error, as shown by the Intermediate Event attached to the boundary of the Task, if there is a problem with the data. A Task to raise an alarm follows the Process Error Intermediate Event. If the data is transformed properly then a Task that uses an application to check the data is performed. Order information is passed to the application and status information is returned. After the “Check Data” Task, the Process continues to a Decision and follow-on Tasks. These details will be shown in the next section.

Mapping to XPD L

Figure 3 shows how the BPD diagram objects for the beginning of the “E-Order” Process will map to XPD L.

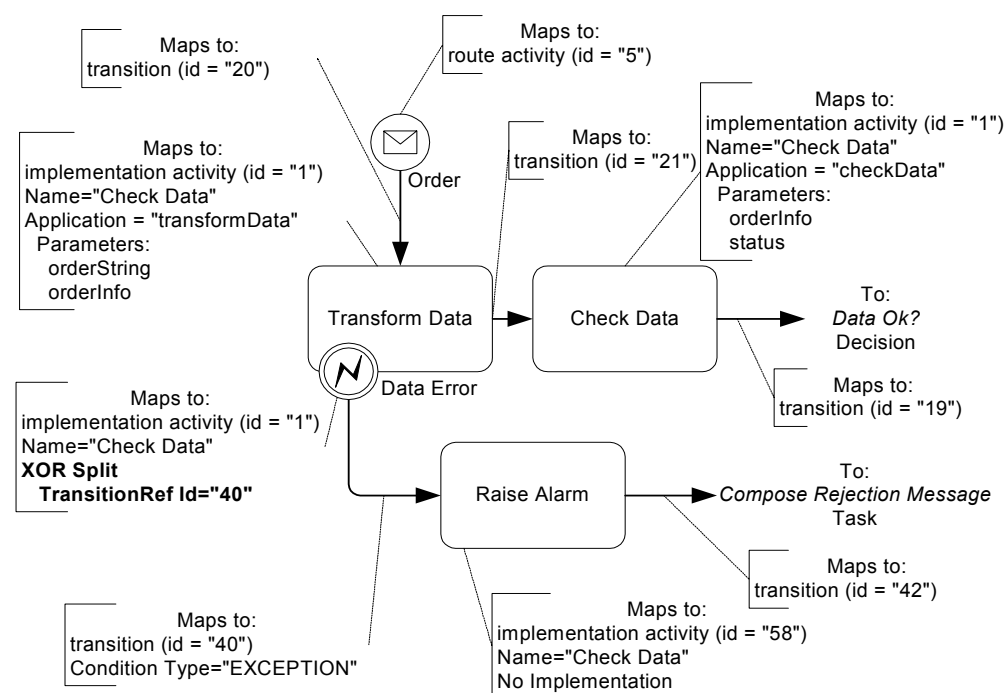


Figure 3: The Start of the E-Order Process and the Object Mappings to XPD L

The BPD objects are straightforwardly mapped to XPD L as shown in the figure. The most complex mapping in this section of the Process appears in the “Transform Data” Task, the “Data Error” Intermediate Event attached to its boundary, and the attached Sequence Flow. These map to an *implementation activity* (*id = “17”*) and a *transition* (*id = “40”*). The Task maps to the *implementation activity* and its *name* and *implementation* elements. The Intermediate Event attached to the boundary of the Task will result in an *XOR Split TransitionRestriction* within the *implementation activity*. *TransitionRestrictions* should only be used in *implementation activities* for excep-

tion and compensation handling. The *TransitionRestrictions* should reference only one *transition* for the normal flow and one or more *transitions* for the exception or compensation flow. In this example, the referenced *transition* (id = 21) defines the normal flow of the process and the other referenced *transition* (id = "40") defines the exception flow. The exception flow *transition* will have a Condition of type "EXCEPTION."

Example 1 displays sample XPD L code that reflects the portion of the Process that is shown in Figure 3.

```
<WorkflowProcess Id="1" Name="EOrder" AccessLevel="PUBLIC">
  <!-- The Process parameters, data files, and applications
        are defined first-->
  <activities>
    <Activity Id="1" Name="Check Data">
      <Implementation .../>
    </Activity>
    <Activity Id="5">
      <!-- This starts out the process -->
      <Route/>
    </Activity>
    <Activity Id="17" Name="Transform Data">
      <Implementation>
        <Tool Id="transformData" Type="APPLICATION">
          <ActualParameters>
            <ActualParameter>orderString
          </ActualParameter>
            <ActualParameter>orderInfo
          </ActualParameter>
          </ActualParameters>
        </Tool>
      </Implementation>
      <TransitionRestrictions>
        <TransitionRestriction>
          <Split Type="XOR">
            <TransitionRefs>
              <TransitionRef Id="40"/>
              <TransitionRef Id="21"/>
            </TransitionRefs>
          </Split>
        </TransitionRestriction>
      </TransitionRestrictions>
    </Activity>
    <Activity Id="58" Name="Raise Alarm">
      <Implementation>No</Implementation>
    </Activity>
    <!-- more activity(ies) -->
  </activities>
  <transitions>
    <Transition Id="19" From="1" To="2"/>
```

```

<Transition Id="20" From="5" To="17"/>
<Transition Id="21" From="17" To="1"/>
<Transition Id="40" From="17" To="58">
  <Condition Type="EXCEPTION"/>
</Transition>
<Transition Id="42" From="58" To="39"/>
<!-- more transition(s)-->
</transitions>
</WorkflowProcess>

```

Example 1: XPDL Sample for Beginning of E-Mail Voting Process

THE MIDDLE OF THE PROCESS

Figure 4 shows the activities and Decisions in the middle of the “E-Order” Process. For the Purposes of this section, the “Check Credit” Sub-Process will be collapsed. The next section will take another look at the Sub-Process as an Expanded Sub-Process within the Process.

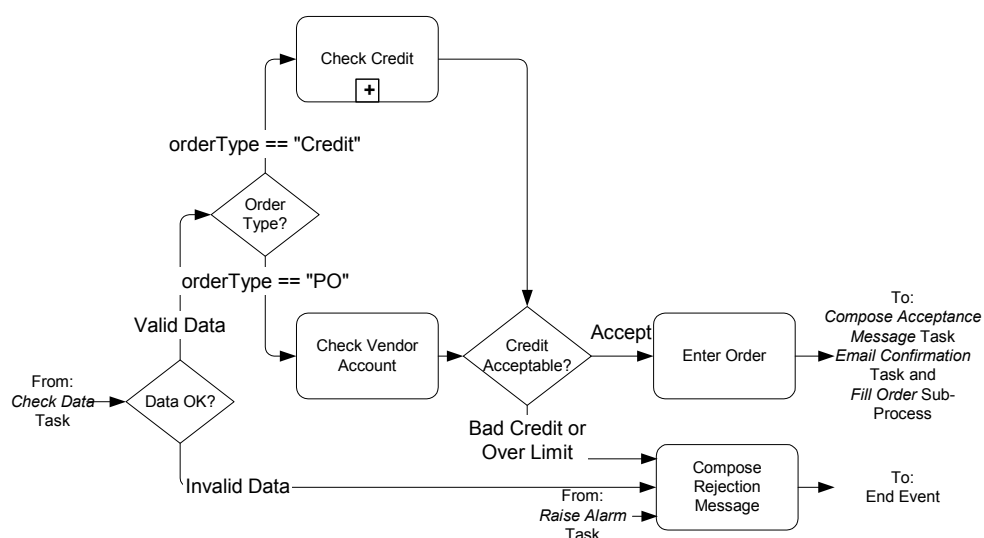


Figure 4: The Middle of the Process

This section starts out with a Decision that considers the results of the previous Task (“Check Data”). If the data is invalid then the process will end after a rejection message is created. If the data is OK, then a second Decision is made that checks the type of order. If the order is based on credit, then a credit check is performed through the “Credit Check” Sub-Process. If the order is based on a PO, then the vendor account is checked. In either case, if the credit is bad or the amount over the credit limit, then the process will end after a rejection message is created. If the credit is ok, then the process will continue with the order data being entered into the system.

Mapping to XPDL

Figure 5 shows how the BPD diagram objects for the middle of the “E-Order” Process will map to XPDL.

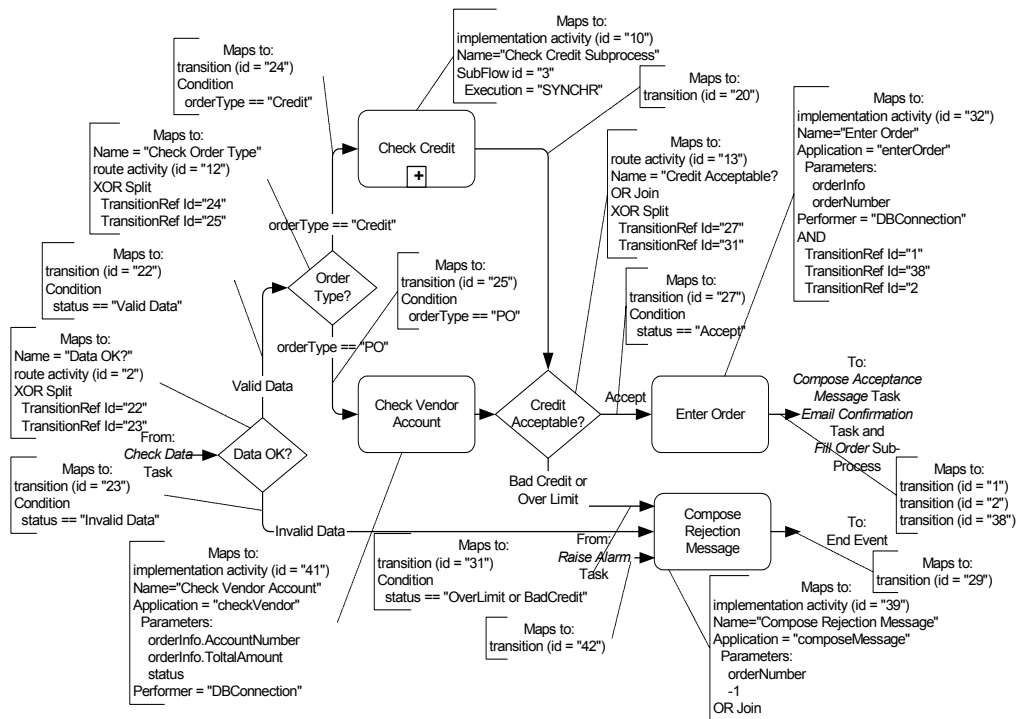


Figure 5: The Middle of the Process and the Object Mappings to XPDL

The Decisions in the Process map to XPDL *route activities* that have *split* elements of type "XOR." The *split* element references the *transitions* that follow the route activity. These *transitions* contain the conditions that determine which of the *transitions* will be taken.

Note: XPDL *implementation activities* may also have *split* elements that could be used for Decisions. However, the mapping from BPMN is much cleaner by using only *route activities* for Decisions and *split* elements in *implementation activities* only for exceptions and compensation.

Both the "Credit Acceptable?" Decision and the "Compose Rejection Message" Task have multiple alternative incoming Sequence Flows. This means that both of the activities to which these objects map will have a *join* element of type "OR."

Both the "Check Vendor Account" Task and the "Enter Order" Task define a performer for the Task and this maps to the *performer* element with the respective *implementation activities*.

The mapping of the "Enter Order" Task also must handle the three parallel outgoing Sequence Flows (as can be seen in Figure 1 and Figure 8). The set of parallel transitions is referenced through a *split* element of type "AND."

Example 2 displays some sample XPDL code that reflects the portion of the Process as described above and shown in Figure 5.

```

<activities>
  <Activity Id="2" Name="Data OK?">
    <Route/>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            <TransitionRef Id="22"/>
            <TransitionRef Id="23"/>
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
  </Activity>
  <Activity Id="10" Name="Check Credit Subprocess">
    <Implementation>
      <SubFlow Id="3" Execution="SYNCHR">
        <ActualParameters/>
      </SubFlow>
    </Implementation>
  </Activity>
  <Activity Id="12" Name="Check Order Type">
    <Route/>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            <TransitionRef Id="24"/>
            <TransitionRef Id="25"/>
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
  </Activity>
  <Activity Id="13" Name="Credit Acceptable?">
    <Route/>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            <TransitionRef Id="27"/>
            <TransitionRef Id="31"/>
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
  </Activity>

```

```

<Activity Id="32" Name="Enter Order">
  <Implementation .../>
  <Performer>DBConnection</Performer>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Split Type="AND">
        <TransitionRefs>
          <TransitionRef Id="1"/>
          <TransitionRef Id="38"/>
          <TransitionRef Id="2"/>
        </TransitionRefs>
      </Split>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes .../>
</Activity>
<Activity Id="39" Name="Compose Rejection Message">
  <Implementation .../>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
    </TransitionRestriction>
  </TransitionRestrictions>
</Activity>
<Activity Id="41" Name="Check Vendor Account">
  <Implementation .../>
  <Performer>DBConnection</Performer>
</Activity>
<!-- more activity(ies) -->
</activities>
<transitions>
  <Transition Id="1" From="32" To="8"/>
  <Transition Id="2" From="32" To="11"/>
  <Transition Id="19" From="1" To="2"/>
  <Transition Id="22" From="2" To="12">
    <Condition>status == "Valid Data"</Condition>
  </Transition>
  <Transition Id="23" From="2" To="39">
    <Condition>status == "Invalid Data"</Condition>
  </Transition>
  <Transition Id="24" From="12" To="10">
    <Condition>orderType == "Credit"</Condition>
  </Transition>
  <Transition Id="25" From="12" To="41">
    <Condition>orderType == "PO"</Condition>
  </Transition>
  <Transition Id="26" From="10" To="13"/>
  <Transition Id="27" From="13" To="32">
    <Condition>status == "Accept"</Condition>
  </Transition>

```

```

<Transition Id="29" From="39" To="33"/>
<Transition Id="30" From="41" To="13"/>
<Transition Id="31" From="13" To="39">
    <Condition>status == "OverLimit or BadCredit"</Condition>
</Transition>
<Transition Id="38" From="32" To="56"/>
<Transition Id="42" From="58" To="39"/>
<!-- more transition(s) -->
</transitions>

```

Example 2: XPDL Sample of “Discussion Cycle” Sub-Process Details

THE EXPANDED SUB-PROCESS

Figure 6 shows isolates the Expanded Sub-Process “Check Credit.”

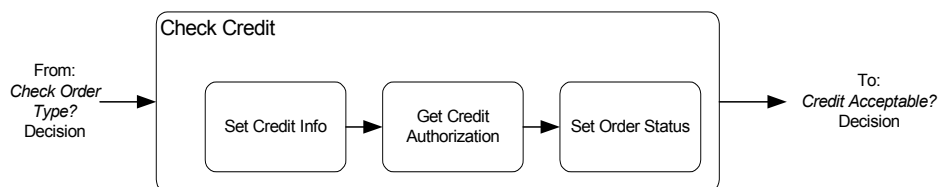


Figure 6: “Check Credit” Sub-Process

Once within the Sub-Process, a Task is set to prepare the credit information for transfer to the credit service, then a Web service is used to get the results from the credit service, and then a Task is used to process the results from the credit service. The Process then moves back up to the top level and continues. The details of the rest of the Process are shown in the previous section and the next section.

The “Get Credit Authorization” Task has communications, through a Web service, with an external company. The communication is shown through Message Flows to and from the “Credit Service” Pool in Figure 1.

Mapping to XPDL

Figure 7 shows how the BPD diagram objects for the Expanded Sub-Process will map to XPDL.

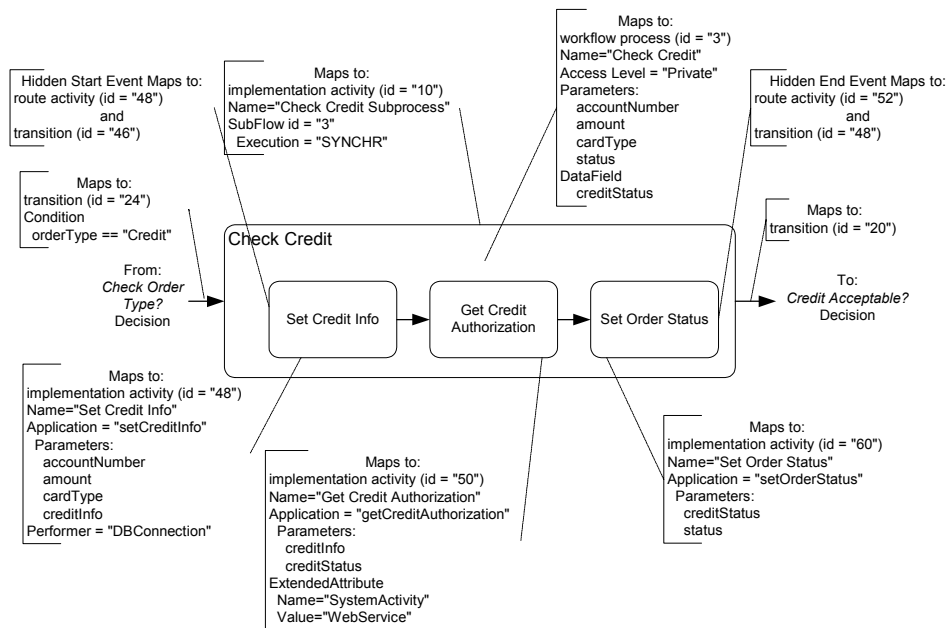


Figure 7: “Check Credit” Sub-Process and the Object Mappings to XPD L

As seen in the last section, the “Check Credit” Sub-Process maps to an implementation activity that has a *SubFlow* element. The details within the Expanded Sub-Process are defined in a separate *WorkflowProcess* element (id = “3”).

The mapping for the details of the Expanded Sub-Process is straightforward except that Start Event and End Event for the Sub-Process are not displayed. Thus, the *route activities* that start and end the XPD L *process* and the transitions that connect them are created through the implicit Start and End Events, rather than any actual graphical objects.

Example 3 displays some sample XPD L code that reflects the portion of the Process as described above and shown in Figure 7.

```
<WorkflowProcesses>
  <WorkflowProcess Id="1" Name="EOrder" AccessLevel="PUBLIC">
    <activities>
      <Activity Id="10" Name="Check Credit Subprocess">
        <Implementation>
          <SubFlow Id="3" Execution="SYNCHR">
            <ActualParameters/>
          </SubFlow>
        </Implementation>
      </Activity>
      <!-- more activity(ies) -->
    </activities>
    <Transitions>
      <Transition Id="24" From="12" To="10">
        <Condition>orderType == "Credit"</Condition>
      </Transition>
    </Transitions>
  </WorkflowProcess>
</WorkflowProcesses>
```

```

        <Transition Id="26" From="10" To="13"/>
        <!-- more transition(s) -->
    </Transitions>
</WorkflowProcess>
<WorkflowProcess Id="3" Name="CreditCheck" AccessLevel="PRIVATE">
    <Activities>
        <Activity Id="48">
            <Route />
            <ExtendedAttributes ... />
        </Activity>
        <Activity Id="49" Name="Set Credit Info">
            <Implementation ... />
            <Performer>DBConnection</Performer>
            <ExtendedAttributes ... />
        </Activity>
        <Activity Id="50" Name="Get Credit Authorization">
            <Implementation ... />
            <ExtendedAttributes>
                <ExtendedAttribute Name="SystemActivity"
                    Value="WebService" />
            </ExtendedAttributes>
        </Activity>
        <Activity Id="52">
            <Route />
            <ExtendedAttributes ... />
        </Activity>
        <Activity Id="62" Name="Set Order Status">
            <Implementation ... />
            <ExtendedAttributes ... />
        </Activity>
        <!-- more activity(ies) -->
    </Activities>
    <Transitions>
        <Transition Id="35" From="50" To="62"/>
        <Transition Id="46" From="48" To="49"/>
        <Transition Id="47" From="49" To="50"/>
        <Transition Id="48" From="62" To="52"/>
        <!-- more transition(s) -->
    </Transitions>
</WorkflowProcess>
<!-- more WorkflowProcess(es) -->
</WorkflowProcesses>

```

Example 3: XPD L Sample

THE END OF THE PROCESS

Figure 8 shows the last section of the Process.

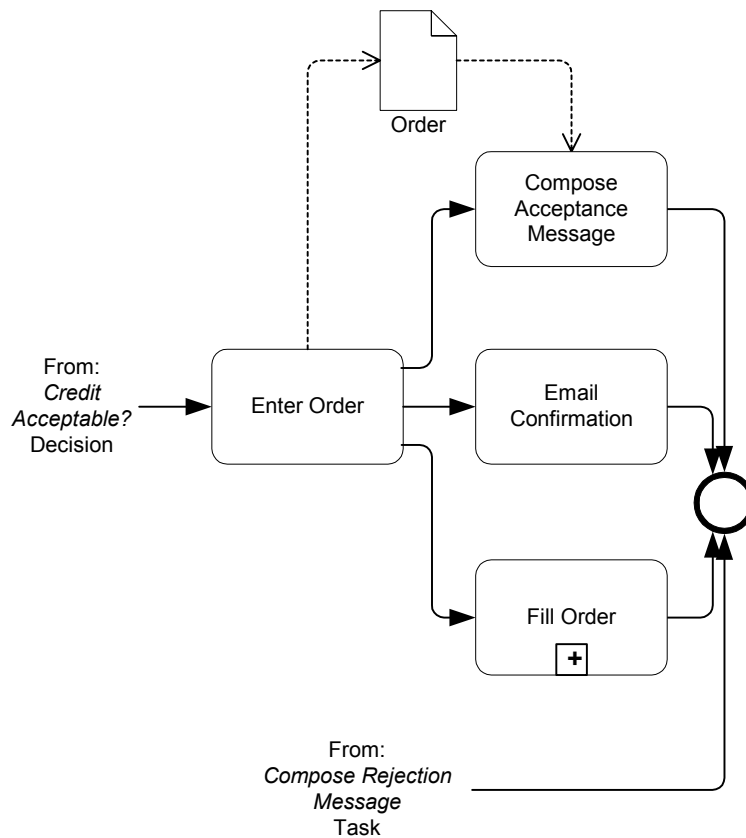


Figure 8: The last segment of the E-Order Process

This segment of the Process continues from where the last segment left off (as described in the section above). After the “Enter Order” Task there are three parallel activities: Two Tasks (“Compose Acceptance Message” and “Email Confirmation”) and a Sub-Process (“Fill Order”). The details of the “Fill Order” Sub-Process are not shown in this example. After these three activities have been completed, the entire Process is completed.

The “Fill Order” Sub-Process has communications with an external company. The communication is shown through a Message Flow to the “Order Shipper” Pool in Figure 1.

Mapping to XPDL

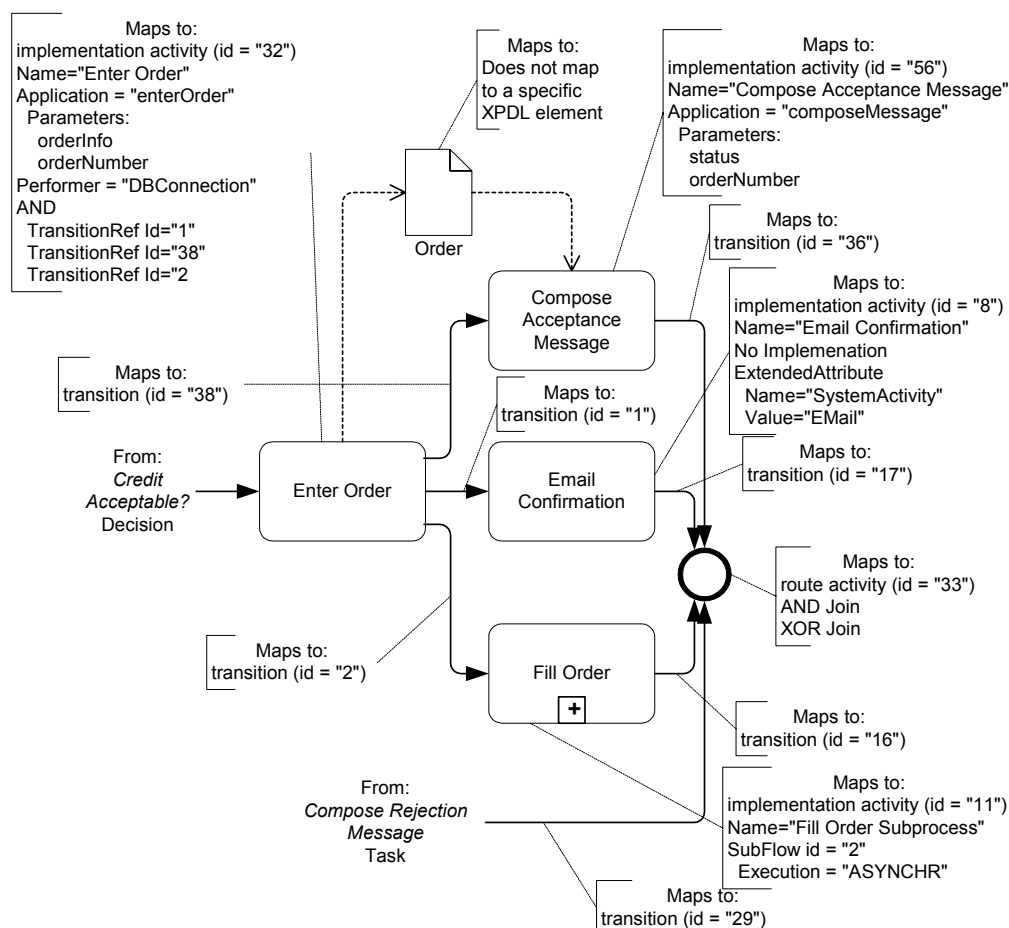


Figure 9: The last segment of the E-Order Process and Object Mappings to XPDL

The mapping is fairly straightforward in this section of the Process. The "Enter Order" Task mapping, with the parallel outgoing Sequence Flows, was described in a previous section. The only other point to note is the mapping of the End Event. This object has three parallel incoming Sequence Flows and a forth alternative incoming Sequence Flow. This results in a *join* element of type "AND" and a second *join* element of type "OR" within the *route activity* that concludes the process.

The "Order" Data Object does not map to a specific XPDL element, but is shown here to illustrate how Data Objects can be used by a modeler to track how business objects interact with the activities of a business process.

Example 4 displays some sample XPDL code that reflects the portion of the Process as described above and shown in Figure 9.


```

<activities>
  <Activity Id="8" Name="Email Confirmation">
    <Implementation>
      <No />
    </Implementation>
    <ExtendedAttributes>
      <ExtendedAttribute Name="SystemActivity" Value="Email" />
      <ExtendedAttribute Name="Email">
        <xyz:Email to="%%orderInfo.emailAddress"
          subject="Order %%orderNumber">
          <xyz:MessageText>Order number %%orderNumber
is being processed.Thank-you for ordering from PQR Products, Inc< /xyz:MessageText>
        </xyz:Email>
      </ExtendedAttribute>
    </ExtendedAttributes>
  </Activity>
  <Activity Id="11" Name="Fill Order Subprocess">
    <Implementation>
      <SubFlow Id="2" Execution="ASYNCHR">
        <ActualParameters>
          <ActualParameter>orderNumber
        </ActualParameter>
          <ActualParameter>orderInfo.orderType
        </ActualParameter>
          <ActualParameter>orderInfo.emailAddress
        </ActualParameter>
        </ActualParameters>
      </SubFlow>
    </Implementation>
    <ExtendedAttributes ... />
  </Activity>
  <Activity Id="32" Name="Enter Order">
    <Implementation ... />
    <Performer>DBConnection< /Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="AND">
          <TransitionRefs>
            <TransitionRef Id="1" />
            <TransitionRef Id="38" />
            <TransitionRef Id="2" />
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes ... />
  </Activity>
  <Activity Id="33">

```

```

    <Route />
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="AND" />
      </TransitionRestriction>
      <TransitionRestriction>
        <Join Type="XOR" />
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes ... />
  </Activity>
  <Activity Id="56" Name="Compose Acceptance Message">
    <Implementation ... />
    <ExtendedAttributes ... />
  </Activity>
  <!-- more activity(ies) -->
</activities>
<transitions>
  <Transition Id="1" From="32" To="8"/>
  <Transition Id="2" From="32" To="11"/>
  <Transition Id="16" From="11" To="33"/>
  <Transition Id="17" From="8" To="33"/>
  <Transition Id="29" From="39" To="33"/>
  <Transition Id="27" From="13" To="32">
    <Condition>status == "Accept"</Condition>
  </Transition>
  <Transition Id="38" From="32" To="56"/>
  <Transition Id="39" From="56" To="33"/>
  <!-- more transition(s) -->
</transitions>

```

Example 4: Sample XPD L code for the last section of the Process

SUMMARY

The Business Process Modeling Notation (BPMN) provides ease-of-use and interoperability for business people who will design and manage the business processes on a daily basis. XPD L provides a standard mechanism for defining and executing business processes, enabling interoperability between workflow environments. The WfMC and BP MI are actively developing a bridge between the two standards. This paper provides the first look at the relationships and mapping between XPD L and BPMN as shown through a sample business process.

*Extracted with permission from the **Workflow Handbook 2003** published by Future Strategies Inc., in collaboration with the WfMC.*

*Order the **Workflow Handbook 2003** at www.wfmc.org*