Business process modeling
Using IBM WebSphere Business Integration Modeler

Skill Level: Intermediate

Doug Tidwell (dtidwell@us.ibm.com)
XML Evangelist
IBM

18 Oct 2005

This tutorial shows you how to create a business process in WebSphere® Business Integration Modeler, which is an extremely powerful tool that lets business experts apply their expertise to process design. You start by defining the sequence of tasks that make up the process. As you build the model, you'll define the resources and costs associated with the process. Once everything is defined, you'll use the tool's simulation facility to run high-level test cases. Best of all, you can compare different simulations and different versions of the process to determine the best process design.

Section 1. Before you start

This tutorial takes you through three steps of defining a business process inside IBM WebSphere Business Integration Modeler. You'll start by defining the steps in your process and the resources required to complete them. Next, you'll define the order in which the tasks should be performed. Finally, you'll run simulations of the process to find the most efficient model.

Prerequisites

To get the most out of this tutorial, install WebSphere Business Integration Modeler on your machine. You can install the modeler by itself or as a plug-in for WebSphere Studio Application Developer, Integration Edition. The screen captures in this tutorial are from a plug-in installation. If you're using the modeler by itself, there will be small changes in the title bar, tool bar, and other parts of the surrounding screen. The modeling tools themselves are the same.
Start the product on your desktop; like any other Eclipse-based tool, you'll be asked to choose a workspace. The workspace is simply a directory that stores all the files for your projects, including the business process you'll define here.

To view the demos included in this tutorial, JavaScript must be enabled in your browser and Macromedia Flash Player 6 or higher must be installed. You can download the latest Flash Player at http://www.macromedia.com/go/getflashplayer/.

Objectives

After completing this tutorial, you will know how to model a business process and all the resources it requires. You will understand how to define branching and logic within the process, and how to simulate the process you defined. Finally, you will learn how to compare different result sets from different simulations to determine the cost and efficiency of different paths through the process.

Animated demos

If this is your first encounter with a developerWorks tutorial that includes demos, here are a few things you might want to know:

• Demos are an optional way to see the same steps described in the tutorial. To see an animated demo, click the Show me link. The demo opens in a new browser window.
• Each demo contains a navigation bar at the bottom of the screen. Use the navigation bar to to pause, exit, rewind, or fast forward portions of the demo.
• The demos are 800 x 600 pixels. If this is the maximum resolution of your screen or if your resolution is lower than this, you will have to scroll to see some areas of the demo.
• JavaScript must be enabled in your browser and Macromedia Flash Player 6 or higher must be installed.

Section 2. Process overview

The plan

In this tutorial, you are going to use IBM WebSphere Business Integration Modeler to model a business process and some business objects, then define how the steps of the business process work. After defining the costs and other attributes of the
tasks in the process, run simulations of the process to estimate the benefits the new process will provide.

There are six objects to model:

- A PurchaseOrder
- A LogMessage
- Three warehouses: WarehouseA, WarehouseB and WarehouseC
- A LoggingFacility

Start the business process model by defining these objects, then define the data flows, decision nodes, and other parts of the process.

1. The input to the process is a PurchaseOrder. A PurchaseOrder contains a unique ID and the quantity of items ordered.

2. The PurchaseOrder is first sent to WarehouseA. WarehouseA compares its inventory to the items listed in the PurchaseOrder. If WarehouseA can ship an item, it sets the quantity of that item to 0. WarehouseA returns an updated PurchaseOrder that contains the unique ID and any items that it didn't ship.

3. If the updated PurchaseOrder has any items left (it has at least one item with a quantity greater than zero), it is sent to WarehouseB.

4. WarehouseB processes the order just as WarehouseA did, then returns the updated order.

5. If the order has any items left, it is sent to WarehouseC.

6. If the updated order returned by WarehouseC has any items left, you call the LoggingFacility to record the items that didn't ship.

7. The output from the process is an updated purchase order that contains a list of items that did not ship.

An important part of the process design is that the input to and output from the process are PurchaseOrders. That means you can use the process in a modular way. If some other process needs to handle a purchase order, it can invoke the process and receive an updated PurchaseOrder. That metaprocess can then do something else with the PurchaseOrder if any items didn't ship.

Starting Business Integration Modeler

Start Business Integration Modeler. Make sure you're in the Business Modeling Perspective:
Figure 1. The Business Modeling Perspective

Right-click in the Project Tree view and select New > Business Modeling Project:

Figure 2. Creating a new Business Modeling Project

The Quickstart wizard opens. Enter the values as shown here and click Finish.

Figure 3. The Quickstart wizard

The wizard creates a new project named SOA2005. Within that project, it creates a process catalog cleverly named Processes, and it creates a new business process named ProcessAndShipOrder within the process catalog.
The process workspace

When Business Integration Modeler is finished creating your new project, you see a view similar to this:

Figure 4. The process workspace

(Yes, the microscopic text here is illegible; it's just shown to give you a sense of how the overall workspace should look. Business Integration Modeler defines other views, but this is the one you'll use here.)

Configuring the Modeler

Before you begin, make sure the tool is configured properly.

Would you like to see these steps demonstrated for you?

Show me

Using the toolbar icons, set the user profile to Intermediate and the mode to BPEL mode:
Section 3. Defining business items

Creating a data catalog

Before defining business items, create a data catalog to store them.

1. Right-click the project name and choose **New > Data Catalog**.

Would you like to see these steps demonstrated for you?

- Show me
2. Give the data catalog the intriguing name **DataCatalog** and click **Finish**.

*Figure 8. Naming the new data catalog*

---

Creating a new business item - The PurchaseOrder

Would you like to see these steps demonstrated for you?

*Show me*

Your first business item is the purchase order. Right-click the data catalog and select **New > Business Item**:

*Figure 9. Creating a new business item*
Create a new business item called **PurchaseOrder**:

**Figure 10. Naming the new business item**

(The comment at the bottom of the panel is optional.)

**Defining the attributes of a business item**

Would you like to see these steps demonstrated for you?

**Show me**

To define the attributes of a purchase order:
1. In the Business Item Attributes panel, add 12 data fields (orderID, quantity605001 through quantity605010 and numItems):

   **Figure 11. Defining the attributes of the purchase order**

   ![Table of attributes](image)

   *Note: In a real-world application, you would use an array to store all of the items in the purchase order instead of creating a separate field for each item in the catalog. The method demonstrated here simplifies the business items and the XML schemas. Or maybe it just serves as an exercise for the user.*

2. By default, every attribute of a business item is a string. You want all of the quantities of a purchase order to be integers. To change the data type of an item, double-click in the Type column, click the button that appears, then select the data type from the window:

   **Figure 12. Selecting the data type of an attribute**
3. Save and close the definition of the purchase order.

Defining the LogMessage business item

The other business item to create is a LogMessage. This is the data you'll send to the logging facility to record an event. To create a new business item named LogMessage:

Figure 13. Defining the LogMessage

There are four properties to define for the LogMessage: orderID comes from the purchase order, service is the service that generated the event (WarehouseA, for example), eventID is one of a set of predefined event codes (IN STOCK and END ORDER are two examples), and description is some text that describes the event.

The business item should look like this:

Figure 14. Defining the attributes of the log message
Save and close the definition of LogMessage.

Section 4. Defining resources

Defining a resource type

Now that the business items have been created, create the resources you need. To make your model more robust, define two types of resource (warehouses and logging facilities), then define the instances of the actual objects you'll use.

1. Right-click the Resources folder and select **New > Resource Definition**:  
**Figure 15. Defining the Warehouse resource type**

2. Create a new resource definition named **Warehouse**:  
**Figure 16. Naming the resource type**
3. Make sure this is an individual resource. (A bulk resource is something like power or water.) Save and close the resource definition.

4. Create a new resource definition for a LoggingFacility:
   **Figure 17. Defining the LoggingFacility resource type**
5. Save and close this resource definition as well.

Defining a warehouse resource

Now that you've defined the Warehouse resource type, it's time to create an actual Warehouse.

Would you like to see these steps demonstrated for you?

Show me

Right-click the Resources folder again and select **New > Resource**:

**Figure 18. Defining a new resource**

Each warehouse is an individual resource and is based on the Warehouse resource definition you just created.

**Figure 19. Naming the new resource**
Defining the costs of a warehouse

1. Double-click the definition of WarehouseA. You see a section for **Costs**: Figure 20. Defining costs for a warehouse

2. Click the Add button. The cost for a warehouse is measured per time unit:
3. Define the cost of WarehouseA to be $800.00 per hour:

4. Create the resources WarehouseB and WarehouseC the same way. Define the cost of WarehouseB to be $1,500.00 per hour and the cost of WarehouseC to be $450.00 per hour.

Defining a logging facility resource

Finally, create a new LoggingFacility resource named Logger:

Define a cost for the Logger as $50.00 per hour.
At this point you've created all the objects you need to define our process. In the next section you'll start defining the process by declaring the inputs and outputs from the process itself.

---

Section 5. Defining the process - Part one

At this point you've created all the resources you need to define the process. Start defining the process by declaring the inputs and outputs from the process itself. Then draw the steps in the process, defining the inputs and outputs for each step along with the resources (warehouses, for example) needed for each step. Finally, draw connectors and branches between the steps to complete the process.

Configuring your workspace

To get started, move on to the main drawing area. Click the 2-pane layout button:

**Figure 24. Switching to the 2-pane layout**

This makes the drawing area occupy the top 2/3 of the screen:

**Figure 25. The 2-pane layout**
Defining the process inputs

The first step in defining the process is to define its inputs and outputs. In this example, the input and output are both PurchaseOrders.

Would you like to see these steps demonstrated for you?

Show me

1. At the bottom of the drawing area, click the Attributes View tab:
   **Figure 26. The Attributes View tab**

2. The Attributes View lets you define the attributes of the process itself. Select the Inputs tab:
   **Figure 27. Attributes View - Inputs tab**
3. The input to your business process is a PurchaseOrder. To create that input, click the Add button in the Input settings section of the panel: 

**Figure 28. Adding an input to the process**

![Input settings](image)

4. Double-click the default name (Input) and change it to InputPO.

5. Double-click the data type, then click the button that appears: 

**Figure 29. Editing an input**

![Input settings](image)

6. In the Select type window, select **Complex type**, then expand your project to find the PurchaseOrder data type: 

**Figure 30. Defining the data type of an input**

![Select type](image)

7. Click **OK** and you'll see the input to the process:

**Figure 31. The input to the process, fully defined**

![Input settings](image)

There's an Input Logic tab, but you won't use it here. The Input Logic tab is used to
define criteria for the input. For example, you can define logic that says a PurchaseOrder has to have at least five items for this business process to be invoked.

Defining the process outputs

To define the outputs of the process:

1. select the Outputs tab:
   Figure 32. Attributes View - Outputs tab

2. Create an output named OutputPO of type PurchaseOrder:
   Figure 33. Adding an output to the process

As with the Inputs section, ignore the Output Logic panel.

Creating a process task

Now start drawing tasks and defining the process. The first step is to create a process task.

Would you like to see these steps demonstrated for you?

Show me

1. Switch to the Diagram tab at the bottom of the drawing area:
   Figure 34. The Diagram tab

2. In the tools palette, click Create a local task in the upper-left corner:
   Figure 35. Create a local task
3. Draw a box on the drawing surface and label the first task **Send order to Whse A:**

*Figure 36. Labeling a task*

---

**Defining task attributes**

1. With the task selected, go to the Attributes view in the bottom panel:

   *Figure 37. The Attributes view*

<table>
<thead>
<tr>
<th>General</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Input logic</th>
<th>Output logic</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Send order to Whse A - Attributes View</strong>&lt;br&gt;<strong>General information</strong>&lt;br&gt;This section provides general information about this task.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Click the Inputs tab, then click **Add** to define a new input named **AInputPO** of type **PurchaseOrder**:

   *Figure 38. Defining a new task input*

<table>
<thead>
<tr>
<th>Name</th>
<th>Associated data</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>AInputPO</td>
<td>PurchaseOrder</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
3. Go to the Outputs tab and define an output named AOutputPO of type PurchaseOrder. The Outputs tab should look like this:

Figure 39. Defining a new task output

Creating a decision node

Next, add a decision node to the process. The output from the Send order to Warehouse A task is an updated PurchaseOrder. If that PurchaseOrder has no items, you’re done with the process. If there is at least one item left on the PurchaseOrder, you have more work to do.

Would you like to see these steps demonstrated for you?

Show me

To add a decision node:

1. Click Create a simple decision:

Figure 40. Creating a simple decision node

2. Draw a decision node on the drawing area and label it Did A ship everything?:

Figure 41. Labeling a decision node
3. With the decision node selected, go to the Attributes view at the bottom of the window and define the decision node’s input, a PurchaseOrder named \textit{DecisionAInput}:

\textbf{Figure 42. Defining the input to a decision}

When you define the input for a simple decision, WebSphere Business Integration Modeler automatically defines outputs of the same data type as the input. To clarify the process diagram, rename them \textit{DecisionAYesOutput} and \textit{DecisionANoOutput}:

\textbf{Figure 43. The outputs from a decision}

\textbf{Defining a data flow}

Use the Connections tool to define how data flows between the process steps you've defined so far:
1. Click the Connections button in the tool palette:
   **Figure 44. The Connections button**

2. Click in the input area for the process, then click the input (the left side) for the *Send order to Warehouse A* task. The drawing area should look like this:
   **Figure 45. Connecting the process input to a task**

3. The process diagram indicates that the input to the process is a PurchaseOrder and that the PurchaseOrder flows from the entry point of the process to the Send order to Whse A task. Connect the output of Send order to Whse A to the input of Did A ship everything?:
   **Figure 46. Connecting a task’s output to a decision node**
Defining other process tasks

Before you move on, create two more tasks and two more decision nodes. Name the two tasks *Send order to Whse B* and *Send order to Whse C*, and name the two decision nodes *Did B ship everything?* and *Did C ship everything?* The diagram should look something like this:

**Figure 47. The drawing canvas with three tasks and three decision nodes**

Be sure to define all the inputs and outputs as you did for the first step and the first decision node.

Defining other data flows

To connect all the steps you've defined so far, use the Connections tool to draw the following links:

- From the No branch of Did A ship everything? to the input of Send order to Whse B
- From the output of Send order to Whse B to the input of Did B ship everything?
- From the No branch of Did B ship everything? to the input branch of Send order to Whse C
- From the output of Send order to Whse C to the input of Did C ship everything?
After defining these links, the drawing canvas should look something like this:

**Figure 48. The data flows between the warehouses**

![Diagram showing data flows between warehouses](image)

Rearranging the process diagram

Right-click the drawing area and select **Auto-Layout Left to Right**:

**Figure 49. The Auto Layout menu**

![Auto Layout menu](image)

If you reduce the size of the display to 50%, you'll see something like this:

**Figure 50. The rearranged diagram**

![Rearranged diagram](image)
Section 6. Defining the process - Part two

Defining branching logic

With the decision nodes defined, you need to define the logic used to determine which branch should be followed. To define the logic for the two output branches from the decision:

1. Go to the Output branches tab and select **Yes**.
   **Figure 51. Selecting a decision branch**

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>DecisionAYesOutput</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>DecisionANoOutput</td>
<td>No</td>
</tr>
</tbody>
</table>

   Would you like to see these steps demonstrated for you?
   
   ![Show me button]

2. Define the logic for the Yes branch. The decision you're defining is whether or not Warehouse A shipped everything in the order. If that's true, the input purchase order (Decision A input) will have a `numItems` field equal to 0.

3. Scroll down to Decision Branch Condition at the bottom of the panel and click **Edit Expression**:  
   **Figure 52. The Edit Expression button**
The Expression Builder
When you click the Edit Expression button, the Expression Builder appears:

Figure 53. The Expression Builder

1. In the First term drop down list, select **Modeling artifact**:
   Figure 54. Defining the first term of the expression
2. Expand Processes > ProcessAndShipOrder > Did Warehouse A ship everything? > Input:
Figure 55. Defining the details of the first term of the expression

3. Scroll to the bottom of the list and select numItems, then select Operator > is equal to:
Figure 56. Defining the operator of the expression

4. For the second term, choose Number from the list menu, then make sure the number value is set to 0:
5. Click **Apply**. The Expression text field displays the text of the expression: **Figure 58. The text of the expression**

<table>
<thead>
<tr>
<th>Expression text</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Processes.ProcessAndShipOrder.Did A ship everything?.Input.numItems</code> is equal to 0.0</td>
</tr>
</tbody>
</table>

6. Click **OK**.

7. Scroll back to the top of the panel and select the **No** branch. If you scroll to the bottom, you'll notice that Business Integration Modeler has entered the negation of the Yes branch: **Figure 59. The automatically generated negation of the first branch of the decision**

<table>
<thead>
<tr>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Processes.ProcessAndShipOrder.Did A ship everything?.Input.numItems</code> is not equal to 0.0</td>
</tr>
</tbody>
</table>

This branch is true if `numItems` is not equal to 0. You can edit this expression if you want, but the default expression does what you want. This decision has only two outputs, so the second branch is automatically generated as the negation of the first.

**Defining the logging task**

Now that you've defined the major tasks, the connections between them and the logic of the decision nodes, there are a couple of things left before our process diagram is finished. If there are any items that weren't shipped by any of the warehouses, you need a task to handle that. You also need a merge node to make sure the updated PurchaseOrder is returned correctly.

The first task is to log any items that weren't shipped by any of the warehouses. To start, scroll to an empty part of the drawing area and create a new task named `Log items not shipped`:

**Figure 60. The logging task**
Define an input named \textit{LoggerInput} and an output named \textit{LoggerOutput}. Both are PurchaseOrders, as you would expect. Create a connection from the output from the No branch of the \textit{Did Warehouse C ship everything?} decision to the input of the \textit{Log items not shipped} task.

Defining a merge node

The final step in building the diagram is to merge all the different threads of the process. No matter what has happened with the order, you need to return a PurchaseOrder from the process.

Would you like to see these steps demonstrated for you?

1. Hold down the mouse button as you click the Create a join node button:
   \textbf{Figure 61. Selecting the merge node function}

2. Click \textbf{Merge} to switch this button to Merge mode and draw a merge node on the diagram:
   \textbf{Figure 62. Drawing a merge node}
3. You need four branches in all, so right-click the merge node and select **Add branch** twice:

**Figure 63. Adding a branch to a merge node**

4. From top to bottom, name the four inputs **A shipped everything**, **B shipped everything**, **C shipped everything**, and **Incomplete order**. The data type of all four inputs is **PurchaseOrder**, as you'd expect. Your input settings should look like this:

**Figure 64. Defining the inputs for a merge node**

The merge node with four inputs should look like this after the inputs are defined:

**Figure 65. The fully-defined merge node**
5. Define the following connections:
   - From the Yes branch of Did A ship everything? to the first (topmost) input of the merge node
   - From the Yes branch of Did B ship everything? to the second input of the merge node
   - From the Yes branch of Did C ship everything? to the third input of the merge node
   - From the output of Log unshipped items to the fourth (bottom) input of the merge node
   - Finally, from the output branch of the merge node to the output of the process itself

When all the connections are defined, the merge node should look like this:

**Figure 66. The merge node with all connections defined**
Deleting the start and end nodes

Because you're using the process input and output as the start and end nodes, you don't need the start and end nodes that are added to the diagram by default.

Would you like to see these steps demonstrated for you?

Show me

To delete them:

1. Scroll back to the process input node:
   Figure 67. The start and end nodes

2. Click the background of the canvas and draw a rectangle around the two nodes to select them, then click delete:
   Figure 68. Deleting the start and end nodes
The final diagram should look something like this:

Figure 69. The complete process diagram

Be sure to save the diagram before you go on to the simulation portion of this tutorial.

Section 7. Simulating the process

Creating a simulation profile

One of the strongest features of Business Integration Modeler is its ability to simulate a business process. You can create simulation profiles that allow you to save a snapshot of the process, then compare snapshots after running different simulations.

For this demo, you'll define some conditions for a scenario and run through it several times. What happens if Warehouse A is much more expensive than Warehouse B? What happens if Warehouse C processes orders twice as fast as the other two warehouses? These are the sorts of questions Business Integration Modeler helps to answer.

To get started, right-click on the process definition and choose Simulate:
This creates a simulation profile, which is a group of settings that define how the process will be evaluated. See a slightly modified version of the diagram in the upper-right portion of the window:

The notation (Simulate) is in the tab above the diagram and there are zeros and bars at various connections. You'll see what those are used for in a few minutes.
See the details of the simulation profile in the lower-right portion of the window:

**Figure 72. The simulation details**

![Simulation Details](image)

**Defining resource requirements**

You could spend days examining all of the simulation settings, but keep it simple here.

Would you like to see these steps demonstrated for you?

[Show me](#)

To define the resources needed for the process:

1. Select the *Send order to Warehouse A* task. The Attributes View looks something like this:

   **Figure 73. The simulation attributes view**

   ![Attributes View](image)

2. Click the Resources tab to define the resources needed for this task.

3. Scroll down to the Individual resource requirements section and click **Add**
to create a new resource requirement.

4. Double-click the requirement name and change it to Warehouse.

5. Double-click in the Individual resource column, click the button that appears, and select WarehouseA:

Figure 74. Defining a resource requirement

6. Double-click in the Time required column and click the button that appears. Specify 45 minutes for WarehouseA to complete this task. The Resources tab should look like this:

Figure 75. A defined resource requirement

7. Define other resource requirements similarly. For the task Send order to Warehouse B, use WarehouseB for 90 minutes. For Send order to Warehouse C, you need WarehouseC for 15 minutes. For the task Log items not shipped, you need Logger for 1 minute.

Running the simulation

Now that you've defined the properties of the simulation, it's time to run it.
Would you like to see these steps demonstrated for you?

Show me

Click the Control Panel tab at the bottom of the window:

Figure 76. The Control Panel tab

On the control panel, click the run button in the upper-right corner to run the simulation:

Figure 77. The run button

An animation shows you how the process is going:

Figure 78. Simulation animation

In this screen capture, a PurchaseOrder is moving from the No branch of the Did A ship everything? decision to the Send order to Warehouse B task. You can use random numbers or mathematical expressions to define how branching is done within your simulation; as you complete this tutorial, keep things simple.

Analyzing simulation results
If you run the simulation several times, the process should go through different paths. Compare the time and costs of each iteration to see how efficiently the process worked. To fully analyze your business process, run many different simulations, changing the resource requirements and other factors each time.

As an example, here is one set of simulation results:

**Figure 79. One set of simulation results**

```
<table>
<thead>
<tr>
<th>Activity</th>
<th>Average Revenue</th>
<th>Average Execution Cost</th>
<th>Average Idle Cost</th>
<th>Average Alloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did A ship everything?</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Merge</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>ProcessAndShipOrder</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Send order to Whse A</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$400.00</td>
</tr>
</tbody>
</table>
```

Here is another set:

**Figure 80. Another set of simulation results**

```
<table>
<thead>
<tr>
<th>Activity</th>
<th>Average Revenue</th>
<th>Average Execution Cost</th>
<th>Average Idle Cost</th>
<th>Average Alloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did A ship everything?</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Did B ship everything?</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Merge</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>ProcessAndShipOrder</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Send order to Whse A</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$400.00</td>
</tr>
<tr>
<td>Send order to Whse B</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$6,000.00</td>
</tr>
</tbody>
</table>
```

As you can see, the cost of processing the second order was significantly higher because WarehouseB was involved. In a much more detailed model, of course, there would be costs and revenues associated with each step. As you add more detail to the model, the information you get from the simulation is much more useful. For example, a more detailed model might tell you that WarehouseC can ship the entire order 75% of the time. If orders were sent to WarehouseC, the first warehouse used, that might save a significant amount of money.

---

**Section 8. Summary**
Some thoughts about what you've done

In this tutorial, you've taken a look at WebSphere Business Integration Modeler, a powerful tool for modeling business processes. Although the sample you went through was relatively complex, there are many functions of Business Integration Modeler the tutorial didn't demonstrate. For example, you can define the statistical distribution of events. Experts in queuing theory can put their expertise to work:

- Business Integration Modeler is designed for business experts. These are people who know how goods, services, cash, and other resources flow through an organization. The assumption behind Business Integration Modeler is that the business experts are not traditional programmers.

- One of the oldest and most common problems in the development world is the frequent inability of programmers and non-programmers to communicate. Business Integration Modeler helps business experts define a business process in technical detail without writing code.

- The business process design here is ideal for developing solutions in a Service Oriented Architecture (SOA). In a perfect world, developers can start with the business process you've defined here and then connect each step in the process to a service already defined in an SOA.

- In the real world, it's unlikely that all of the services required by a business process will be available. The development shop might need to modify legacy applications or create new ones altogether.

- The level of detail provided by the process definition makes it much easier for developers to estimate the cost and time necessary to implement the changes.

- The level of detail provided by the simulation results makes it much easier for business experts to estimate the benefits of implementing the new process.

Using Business Integration Modeler, executives can use the detailed costs and benefits to make rational business decisions:

- A new business process might be significantly cheaper, but it might take twice as much time as the current process. If the organization's main priority is for the process to run as quickly as possible, it won't be implemented. On the other hand, if cost savings are more important than process throughput, the new process is likely to move ahead.

- A new business process might be cheaper and faster, but it might require substantial changes to a crucial and brittle legacy application. If that crucial application was originally written in COBOL 30 years ago by someone who died 10 years ago, the risks of modifying that legacy code might outweigh any benefits that might come from the new process.
• A new business process might not be faster or cheaper, but it might be the first step in implementing a complete SOA. Management might decide to implement an SOA-based business process as a pilot project. This new process might be seen as more of a learning experience than a critical business improvement.

With this powerful tool, you've designed and simulated a new business process. In a future tutorial you'll look at what's involved in implementing the new process in an SOA. For now, you should have a good understanding of how Business Integration Modeler can help you move to a process-oriented architecture.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
<th>Download method</th>
</tr>
</thead>
<tbody>
<tr>
<td>The workspace used in this tutorial.</td>
<td>SOA2005-ProcessModeler.zip</td>
<td>150 KB</td>
<td>FTP</td>
</tr>
</tbody>
</table>

**Information about download methods**

Business process modeling
Page 42 of 43
Resources

- Participate in the discussion forum for this content.
- The IBM Redbook *BPEL4WS Business Processes with WebSphere Business Integration: Understanding, Modeling, Migrating* takes an in-depth look at business process modeling. You can download both the PDF file and the source of the sample applications from the book's home page.
- Download WebSphere Business Integration Modeler V5.1 for Windows from the developerWorks Downloads page.

About the author

Doug Tidwell
Senior Programmer Doug Tidwell is IBM's evangelist for Web Services. He was a speaker at the first XML conference in 1997, and has been working with markup languages for more than a decade. He holds a Bachelors Degree in English from the University of Georgia and a Masters Degree in Computer Science from Vanderbilt University. He can be reached at dtidwell@us.ibm.com. You can also see his Web page at ibm.com/developerWorks/speakers/dtidwell/.