IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler, Part 2: Patterns advanced usage and accelerators palette configuration

Thomas Gschwind (thg@zurich.ibm.com)
Research Staff
IBM

Jana Koehler (koe@zurich.ibm.com)
Research Staff Manager
IBM

Janette Wong (janette@ca.ibm.com)
Senior Technical Staff Member
IBM

Cedric Favre (ced@zurich.ibm.com)
PhD Student
IBM

Wolfgang Kleinoeder (wbk@zurich.ibm.com)
Research Emeritus
IBM

Alexander Maystrenko (oma@zurich.ibm.com)
PhD Student
IBM

Krenar Muhidini
Former intern
IBM

09 December 2009
(First published 20 July 2009)

This series walks you through the IBM Pattern-based Process Model Accelerators V2.0 for WebSphere® Business Modeler, a set of plug-ins for IBM WebSphere Business Modeler that add patterns, transformations, and refactorings to your business process modeling environment. In Part 2 we show you how to apply patterns with business items and business item states to create pattern-based process models with data flow. We also explain how to configure the Accelerators palette to suit your needs.

View more content in this series

Introduction

This article is Part 2 of a series of 4, introducing IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler (hereafter referred to as the accelerators). These accelerators provide you with a set of plug-ins for IBM WebSphere Business Modeler that add patterns, transformations and refactorings to your business process modeling environment. The plug-ins also include a feature to automatically detect and correct control-flow errors.

By using the accelerators you move away from a traditional business process modeling approach where process models are drawn by dragging and dropping elements on the drawing canvas that are then manually connected. The accelerators enable you to create business process models of higher quality by composing them from larger building blocks or by applying semantically correct change operations to your entire model with a single click. Your models will contain significantly less modeling errors, modeling becomes a much more fun exercise and you will experience productivity gains of about 70 % compared to the traditional approach.

This article provides:

- A systematic description of the 8 patterns available in this release of the accelerators.
- Instructions for how to configure the Accelerators palette according to your modeling needs.

A *pattern* encapsulates a simple and elegant solution to a specific, but frequently re-occurring problem. Patterns are created by observing (or “mining”) a variety of solutions that different people have created over time when working on the same problem. The pattern encapsulates the “essence” that is common to all these different solutions and that ensures that the solution solves the problem. Examples of famous collections of patterns are the object-oriented design patterns by Gamma et al, the software architecture patterns by Buschmann et al, and the workflow patterns by van der Aalst et al (see Resources for detailed references).

Download the accelerators.

The business process patterns that we provide in this release of the accelerators encapsulate typical process model fragments that occur again and again in business process models. The encapsulated fragments represent common, easy-to-reuse control-flow structures that you can also associate with business items and business item states. When composing a process from these patterns, it is guaranteed that the process correctly executes in a process simulation environment such as the one provided by WebSphere Business Modeler, and that it can be mapped to a combination of workflow patterns for a correct implementation in a process runtime engine.

We cover the following patterns in this article:

-  Insert Task
-  Insert Process

-  Sequence
-  Alternative Compound
-  Parallel Compound
-  Loop
-  Alternative Branch
-  Parallel Branch

The above list also introduces you to the icons of these patterns in the Accelerators palette.

Prerequisites

This article is Part 2 of a series, and assumes that you are familiar with Part 1, which means:

- You have IBM WebSphere Business Modeler V6.2.0.1 with Fixpack 1 installed.
- You installed the accelerators and worked through the example modeling project `HiringExample.mar`.

We also assume that you have basic knowledge of WebSphere Business Modeler, i.e., that you are familiar with the product and you have gained some modeling experience while creating business process models. You should also be familiar with the basic model elements such as gateways, tasks, subprocesses, start and terminate events from the Business Process Modeling Notation (BPMN) as available in WebSphere Business Modeler. The help available in WebSphere Business Modeler provides you with the necessary background.

Part 1 gave you detailed information on where to download the accelerators and how to install them as well as how to validate your installation. Part 1 also guided you through a step-by-step tutorial based on the example project during which you learned to apply 5 of the patterns, namely Insert Task, Sequence, Parallel Compound, Loop, and Alternative Branch. In addition, it introduced you to control-flow analysis, which allows you to easily locate modeling errors in your process model. Part 1 also explained how you apply transformations and refactorings to correct errors and to further improve your models.

In this article, we describe all 8 patterns in detail, in particular by looking at how to create models with data flow. This article will serve mostly as a reference for you where you can look up detailed information on best practices for using the patterns, and additional instructions for configuring the Accelerators palette.

General hints for using the Patterns

Applying a pattern in WebSphere Business Modeler consists of three simple steps:

1. Identify the location to apply a pattern by selecting one or two connections
2. Invoke the pattern from the Accelerators palette or the menu
3. Provide optional parameters to the pattern by filling in a pattern wizard

Step 1: Identify the location to apply a pattern

You must have a single connection or a pair of connections selected to apply a pattern. The selection is different for the three groups into which the patterns can be organized:

- The first group comprises the *Insert Task, Insert Process* and *Sequence* patterns, which add a single task or process, or a sequence of tasks to a process model and reconnect it to the existing flow.
  For these patterns you should select only a single connection. When you select a single connection, Modeler applies the pattern to refine this connection, i.e., it inserts the process fragment that results from instantiating the pattern into the process model by splitting the selected connection into two connections connecting the pattern to the existing process model. If the connection has associated data, it automatically reuses this data in the instantiation of the pattern.
- The second group comprises pattern compounds, i.e., a combination of tasks and gateways that exhibits a regular structure. The following compounds are available: *Alternative Compound, Parallel Compound*, and *Loop*.
  For these patterns you can select one connection or a pair of connections. If you select a single connection, Modeler refines this connection with the selected pattern. If you select a pair of connections, Modeler reuses the process fragment that is identified by this pair of connections as a part of the pattern. We explain the reuse of process fragments in a pattern when describing the patterns in more detail.
- In the third group, you find patterns that allow you to easily add additional branches to your model: *Alternative Branch* and *Parallel Branch*.
  For these patterns you must have a pair of connections selected. Modeler places these patterns between the two selected connections. We describe this behavior in detail in the description of these patterns.

Step 2: Invoke a pattern

Following your selection of connections, you have two choices to invoke a pattern:

- Click on the drawing canvas of your business process model diagram and then invoke the pattern at the bottom of the pull-down menu, or
- Click on the corresponding graphical symbol in the Accelerators palette.

We explained both choices in Part 1 of this series.

We recommend that you save your model before you apply a pattern, which makes it easier to undo unwanted changes by simply discarding a model and reopening it again in its last saved state. Alternatively, you can use the undo function provided by WebSphere Business Modeler. This will undo each of the individual editing operations out of which the pattern is composed in a step-by-step manner.

For patterns that you can invoke on a single connection, the following error message is shown when no connection is selected:

```
To apply the pattern, please select a connection.
```

For patterns that you can invoke on a single connection or a pair of connections, Modeler displays the error message from Figure 1 when you have nothing selected. If your connection pair does not delimit a process fragment to which the pattern is applicable, the following error message is displayed:

```
To apply the pattern, please select a single connection
 or a pair of connections that delimits a fragment.
```

It is safe for you to apply any of the 8 patterns; you cannot introduce a modeling error into your business process model by instantiating a pattern.

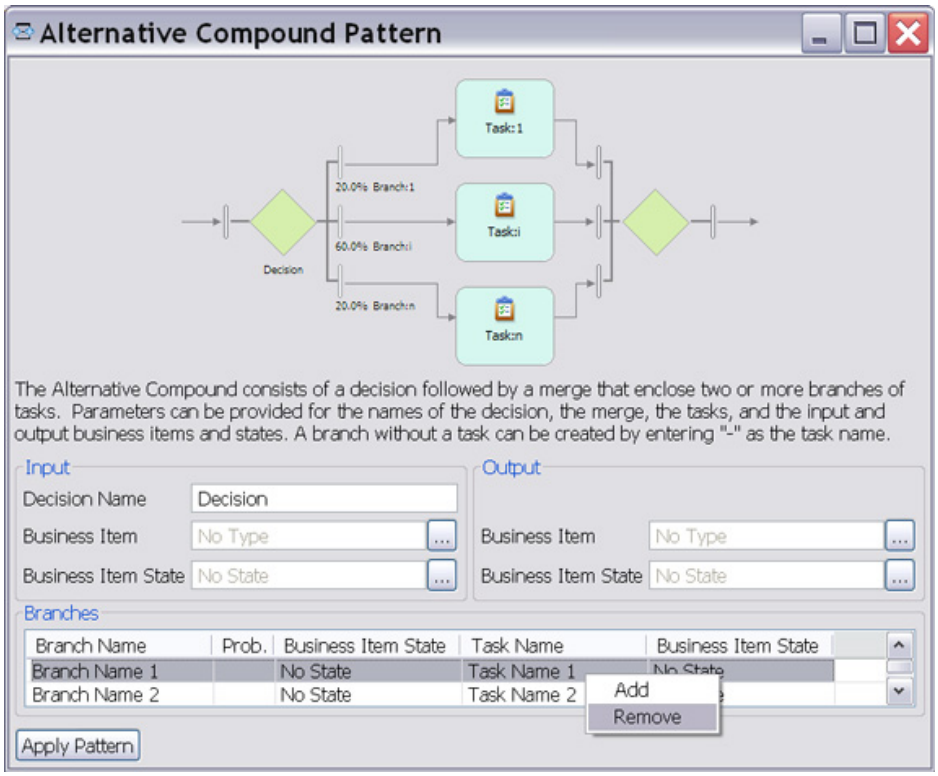
However, applying a pattern rarely leads to a perfect layout of the resulting process model. Invoke the **Auto-Layout Left to Right** to improve the layout of the model.

Step 3: Provide parameters in a pattern wizard

Several of the patterns come with a user interface, called the pattern wizard that allows users to instantiate the pattern with parameters. A pattern wizard is composed of three parts:

- In the upper part, it shows the name of the pattern and a schematic picture under which a textual explanation of the pattern can be found containing hints on how to use the pattern.
- In the middle part of the wizard, information about gateways and their possible input and output business items and their states can be specified.
- In the lower part, tabs can be found to add information about the tasks that are part of the pattern. The tabs contain lists that allow users to add more tasks or branches to a pattern. To add or remove a row from the list, click right and select **Add** or **Remove** from the context menu (see Figure 1).

Figure 1. Pattern wizard example



While a pattern wizard is open, you can continue to edit the model. If you accidentally delete the connection that you selected for applying the pattern, the wizard becomes inactive and the pattern cannot be applied anymore. In this case, close the wizard.

All information in a wizard is optional, which means that a pattern can always be applied without entering any information in the wizard.

At the bottom of each wizard, you find the **Apply Pattern** button. Click this button to apply the pattern to your process model. If you decide to not apply a pattern, simply close the wizard window. All parameters will be discarded. Each wizard is shown and described in detail in this article.

In this article, we adopt the common guidelines for the description of patterns by specifying the following information:

- **Pattern name:** A descriptive name that helps to identify and refer to the pattern.
- **Intent:** A description of the goal behind the pattern and the reason for using it.
- **Also known as:** Other names for the pattern.
- **Structure:** An explanation of the pattern wizard (if available) and the main parameters of the pattern.
- **Consequences:** A description of the results, side effects, and trade-offs caused by using the pattern.
- **Sample:** An example illustrating how to use the pattern.

- **Related patterns:** A short discussion of other patterns that have a relationship with the pattern or that are especially useful in a combined application with the pattern. We will also point to transformations and refactorings that are useful in the pattern context.

In case of strong similarity of the information, we provide one description that covers several patterns. When describing the wizards, we focus on those fields where information can be entered by the user.

Insert Task, Insert Process, and Sequence patterns

Pattern name: Insert Task, Insert Process, Sequence

Intent: The Insert Task, Insert Process, and Sequence patterns allow you to refine a selected connection with a single task or subprocess, or a sequence of tasks.

Also known as: Sequential, serial or linear routing (flow) of tasks and subprocesses.

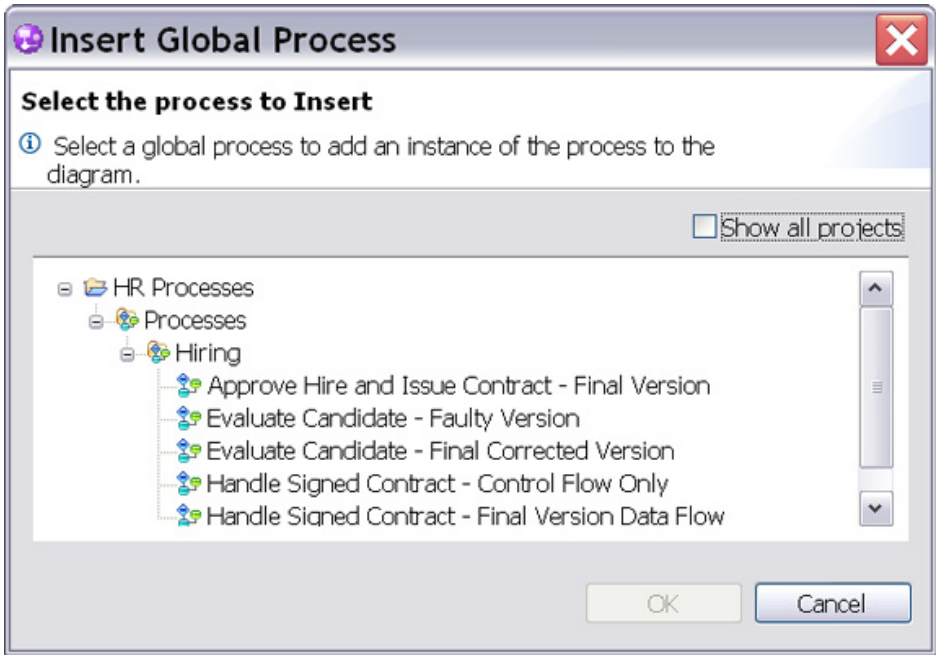
Structure: The Insert Task and Insert Process patterns are variants of the Sequence pattern that allow you to refine a selected connection with a single task or global process. , These two specific instances of the pattern have been explicitly added to the accelerators because this editing step occurs so frequently. In contrast, the Sequence pattern allows you to specify a sequence of tasks with optional business item outputs and business item states.

The **Insert Task** pattern does not have a wizard. It simply inserts a task with a new default name on the selected connection, saving you from the burden of manually dropping a task on the canvas and reconnecting this task. After applying the pattern, you can edit the name of this task.

The Insert Process pattern opens a dialogue window that allows you to select a global process from the existing business process models in the modeling project, as shown below.

The **Insert Process** pattern selects the list of global processes from your current modeling project or any project in its reference group. Select a process from the list, then click **OK**. The process is added to the model and automatically connected using the selected connection (see Figure 2).

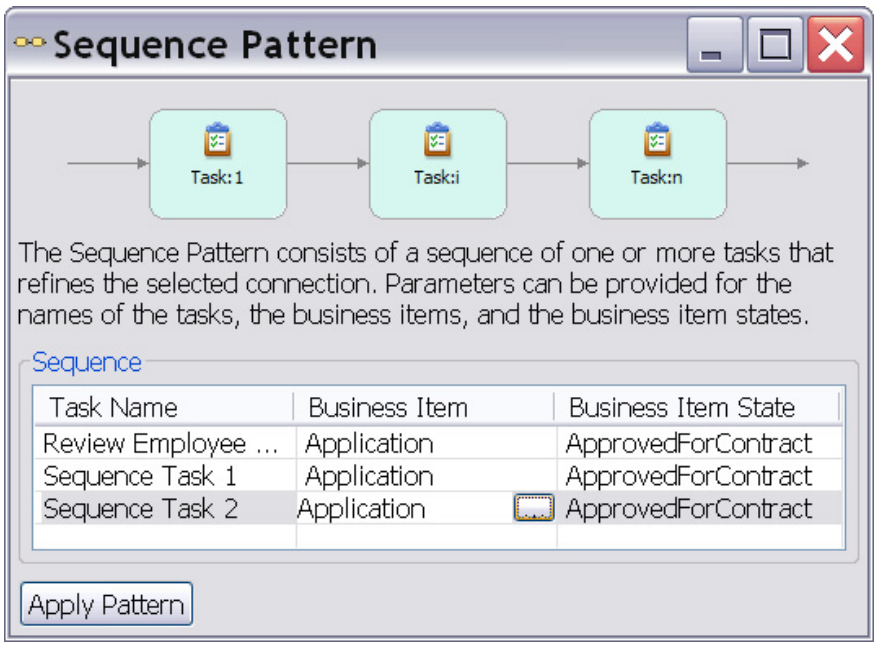
Figure 2. Inserting a global process



To add another modeling project from your workspace to the reference group of your current project, right-click on your current project, select **Edit Reference Group**… and follow the instructions.

The **Sequence pattern** allows you to specify a list of task names in a wizard. The wizard shows a list of task names with optional business item and business item state columns as shown in Figure 3.

Figure 3. Sequence Pattern wizard



The name shown in the first row is the name of the task or other model element from which the selected connection starts. For example, when you select one of the output branches starting from a decision, the name of the selected output branch is shown in the **Task Name** field of the first row.

If a selected connection has an associated business item and business item state, this business item and business item state appear in the first row as the output of the model element. They are also used to instantiate the business item and business item state fields in all other rows.

You can overwrite all fields with new values. You can also overwrite the values in the first row of the wizard. However, only the change of the output business item and business item state is applied to the selected connection, while a change of the task name is ignored.

To overwrite a field, click in a field of a business item or business item state column. A button appears as shown in Figure 5. When you click this button, the next windows lets you select your choice directly from the business items and states that are defined in your modeling project. If no states are defined for a business item, the value `No State` is shown in the respective field and cannot be changed.

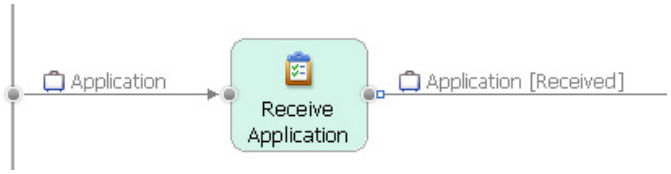
To add or remove additional tasks to/from the list, right-click and select **Add** or **Remove** from the context menu.

Consequences: The patterns add a local task, global process or sequence of local tasks to the selected connection. There are no specific patterns to add global tasks or local processes, but you can convert a local task to a global task or a local process by using the **Convert To** feature from WebSphere Business Modeler. If you enter the name of a local task in the wizard of the Sequence pattern that is already used in your process model, a new task with this name is added and a `Duplicate Name` error is shown in the Errors view.

Sample: This example shows the Sequence pattern illustrating the usage of business items and states. We reuse business items as defined in the sample modeling project `HiringExample.mar` that is available as a download with this series.

Our initial example model contains a single task `Receive Application` that has an `Application` business item as input and output. When the `Application` business item leaves this task, it is in `Received` state (Figure 3).

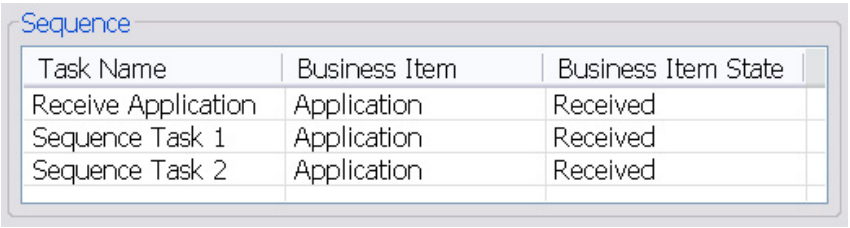
Figure 4. Initial example model containing a single task



We apply the Sequence pattern to the outgoing connection of this task to add further tasks to the process that handle the application, i.e., we select the outgoing connection of the `Receive Application` task before we invoke the Sequence pattern.

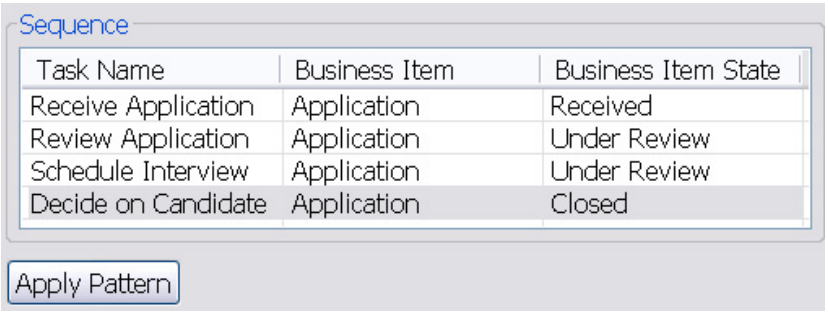
The wizard of the Sequence pattern is pre-instantiated as Figure 6 shows. Therefore the `Receive Application` task appears in the first row with the `Application` business item as its output and the `Received` state. The next two rows are instantiated with default names for two new tasks, while the business item and business item state are inherited from the values in the first row.

Figure 5. Sequence pattern wizard pre-instantiation



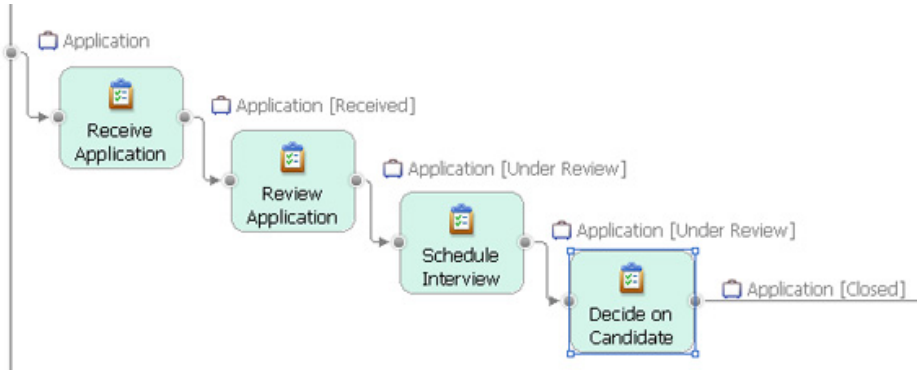
Click on the **Task Name** and **Business Item State** fields in the second and third row and change their values as shown in Figure 6 (change `Sequence Task 1` to `Review Application` and the **Business Item State** to `Under Review`. Add a fourth row to the list and set its **Task Name** to `Decide on Candidate`, and its **Business Item State** to `Closed`.

Figure 6. Updated Sequence pattern wizard



Click **Apply Pattern** and layout your process model. It contains the tasks shown in Figure 7, connected by data flow.

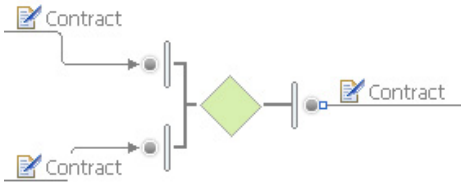
Figure 7. HiringExample process model after application of the Sequence pattern



Avoid replacing a business item on a branch of a gateway, because this can lead to getting incompatibilities in your process reported in the Errors view when you save the model. As an

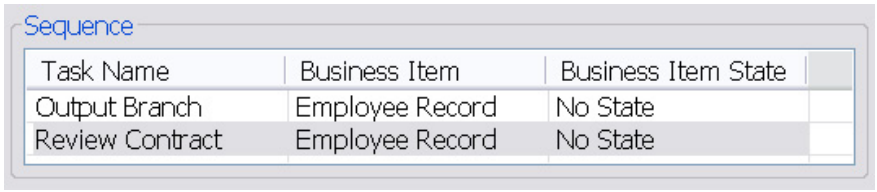
example, select the outgoing connection of the merge that routes a Contract business item as Figure 8 shows.

Figure 8. Outgoing merge connection



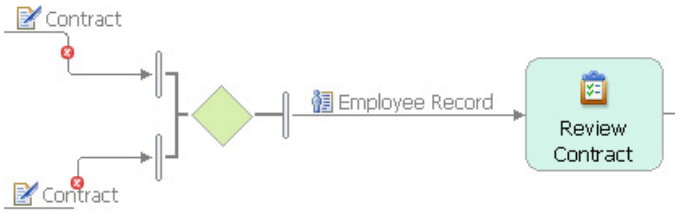
Apply the Sequence pattern to the output branch of this merge and replace the `Contract` business item of the output branch by the `Employee Record` as Figure 9 shows.

Figure 9. Applying the Sequence pattern with a changed business item



As a result, the merge receives two `Contracts` on its incoming branches that it magically changes to an Employee Record on its output branch as Figure 10 shows.

Figure 10. Contract changed to Employee record



Related patterns: These patterns can be applied on any connection in your process model. Use them in particular after an application of the Alternative Branch and Parallel Branch patterns to add tasks and subprocesses to the branch created by these patterns. Similarly, apply them after the Alternative and Parallel Compound or Loop patterns to add more tasks and subprocesses to the specified branches.

Alternative Compound

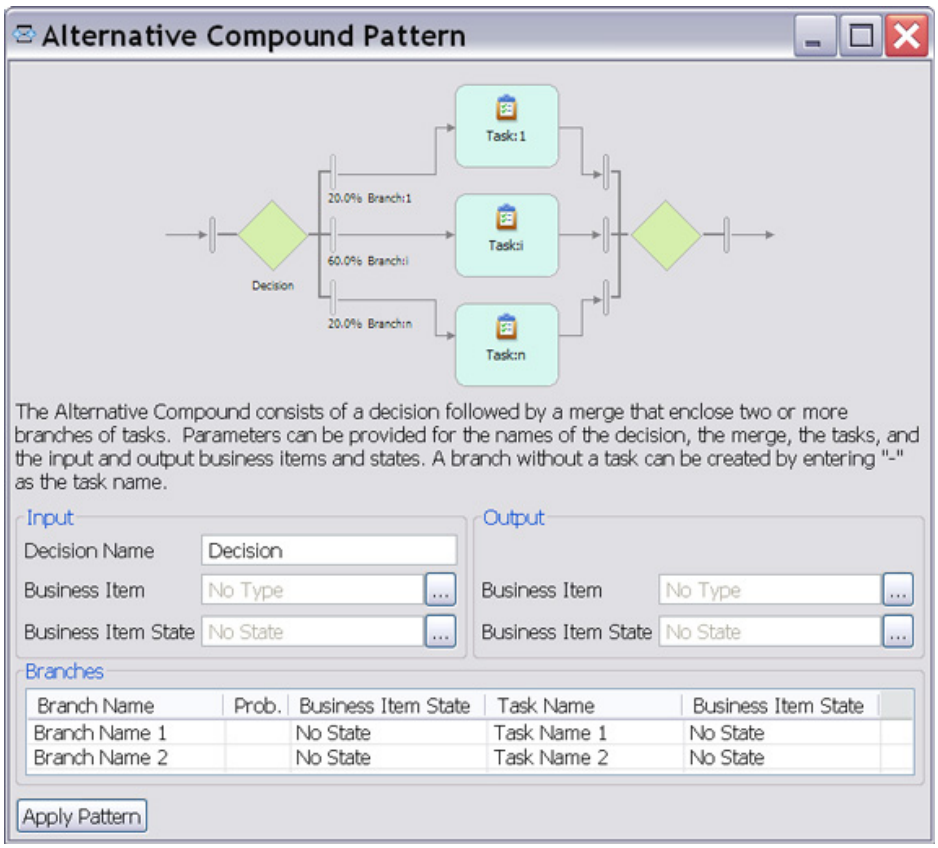
Pattern name: Alternative Compound

Intent: Use this pattern to add a flow of several alternative branches to your process model that begins with a decision and ends with a merge.

Also known as: The Alternative Compound pattern is a combination of the Exclusive Choice and the Simple Merge workflow patterns (see Russell et al. in Resources below) that enclose several alternative branches. The pattern implements the conditional routing and asynchronous joining of several alternative flows. The behavior is comparable to a case or switch statement in programming languages.

Structure: You can invoke the alternative compound pattern by selecting one connection or a pair of connections. A wizard opens as Figure 11 shows:

Figure 11. Alternative Compound pattern wizard



The wizard allows you to enter information about the input of the decision that starts the pattern, and about the output of the merge that ends the pattern. For the decision, you can enter its name, incoming business item and business item state. For the merge, you can enter the the outgoing business item and business item state. Recall that a merge can have a user-defined name in WebSphere Business Modeler, but this name is not visible in the diagram and is therefore not editable in the wizard.

You can also enter the following information for two or more output branches of the decision:

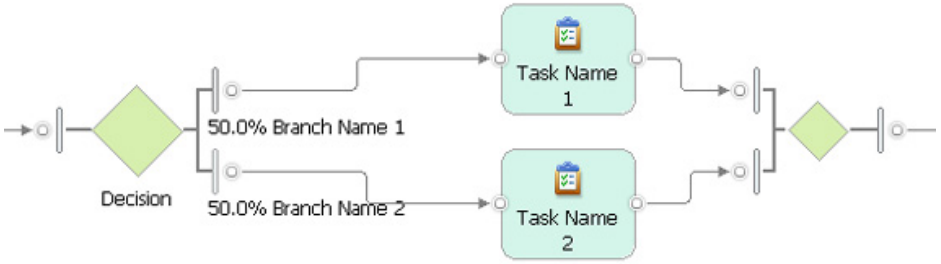
- The name of each branch.
- The branch's probability
- The state of the business item when it leaves the decision and enters the task on this branch
- The name of the task on this branch

- The state of the business item when it leaves the task.

To add or remove additional branches to/from the list, right-click and select **Add** or **Remove** from the context menu.

Consequences: When you apply the pattern without entering any information in the wizard, the selected connection is refined with the process fragment shown in Figure 12:

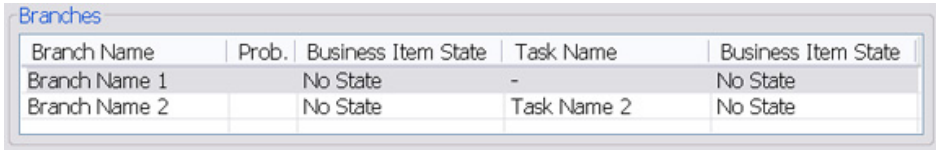
Figure 12. Default output of Alternative Compound process wizard



The first two branches will by default obtain a 50% probability. Any further branch will obtain a 0% probability by default. When you specify percentages for each branch that do not add up to 100%, WebSphere Business Modeler reports this error when you save the model. The pattern will not attempt to correct branch percentages, because it cannot know what the correct percentages are.

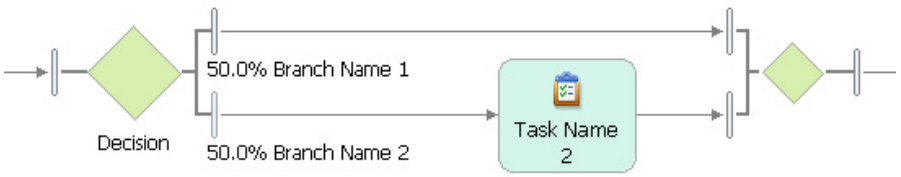
You can create an alternative compound with an empty branch not containing a task by specifying a dash “-” as the name of the task, as Figure 13 shows:

Figure 13. Alternative Compound pattern wizard



The wizard inputs from Figure 13 create a process fragment as shown in Figure 14:

Figure 14. Process fragment created for compound with empty branch



You can also select a pair of connections and then apply the pattern. Based on the connection pair, the pattern wizard automatically identifies the process fragment that is delimited by these two connections. If this fragment has your selected connections as its only incoming and outgoing connections, it uses it as an argument to the pattern wizard and adds it on one of the branches of the pattern by specifying a start “*” as the name of the task on this branch.

Note that the textual explanation of the pattern in the wizard adjusts by displaying additional text when you select two connections, and points you to this option.

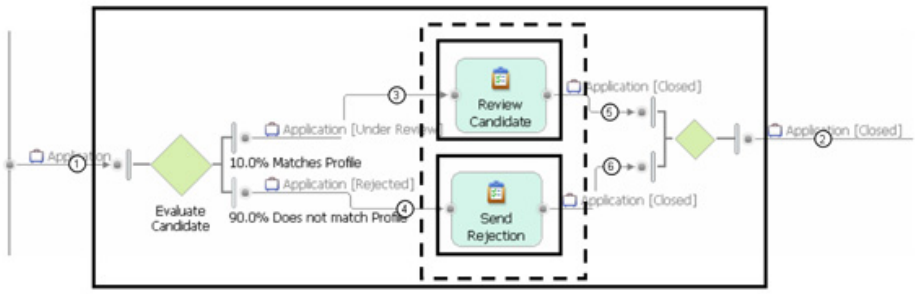
```
The Alternative Compound consists of a decision followed by a merge that enclose two or
more branches of tasks.  Parameters can be provided for the names of the decision, the
merge, the tasks, and the input and output business items and states. A branch without a
task can be created by entering "-" as the task name. The currently selected fragment
can be used instead of a task by using "*" as the task's name.
```

If you forget to put the *, the following error message reminds you:

```
If a pair of connections has been selected, use "*" as the name of
exactly one task. If only a single connection has been selected,
"*" is not allowed as a task name.
```

Let's take a closer look at the notion of a process fragment. Figure 15 shows 4 fragments in a process model. Only the three fragments surrounded by a solid box each have a single incoming connection 1, 3, or 4 and outgoing connection 2,5, or 6 and can be reused in the pattern. The part of the process model surrounded by a dashed box has two incoming connections 3 and 4 (the two output branches of the decision) and two outgoing connections 5 and 6 (the two input branches of the merge). If you select two of these connections, for example the lower output branch of the decision and the upper input branch of the merge, you have not selected a proper fragment that can be reused in the pattern.

Figure 15. Fragments in a process model



If two connections do not identify a valid selection of a fragment that can be reused in the pattern, an error message tells you to revise your selection of connections.

```
To apply the pattern, please select a single connection
or a pair of connections that delimits a fragment.
```

Sample: We show two examples. The first example illustrates the use of business items and business item states in an alternative compound. In this example, we create the process model shown in Figure 16. The second example illustrates the reuse of a process fragment as an argument in the pattern wizard with two connections selected.

1. Create a new process model and connect the start and terminate events. Then select this connection as Figure 16 shows:

Figure 16. Start and terminate events

2. Instantiate the pattern as is shown in Figure 17. You can use the Application business item as defined in the sample modeling project `HiringExample.mar` that is available with this series. Select Closed as the value for the Business item state fields of the Output and the Branches. Note that the fields to specify business items and business item states are synchronized with each other such that their values remain consistent with each other. For example, you can specify different business item states on each of the branches only when you select No State as the value of the business item input state of the decision. If you want a specific output state of the business item in the merge, then the task on each branch must provide this output state.

Figure 17. Instantiating the Alternative Compound pattern

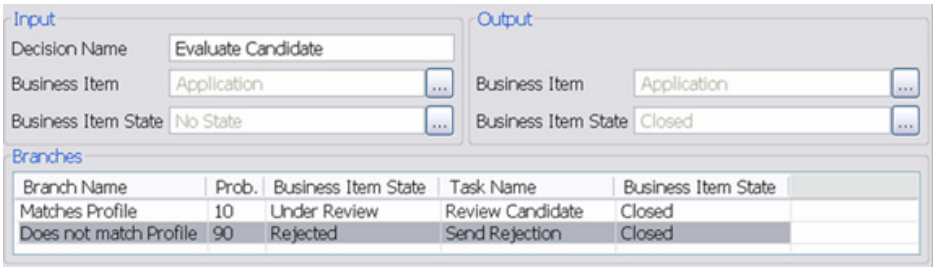
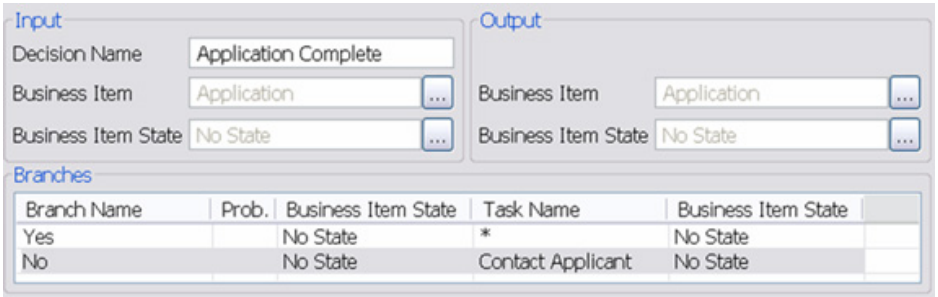


Figure 18 shows the resulting process model.

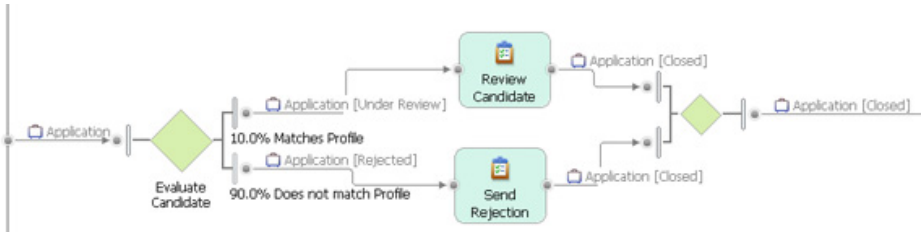
Figure 18. Resulting process model after applying the Alternative Compound pattern



With our second example, we show the reuse of a process fragment:

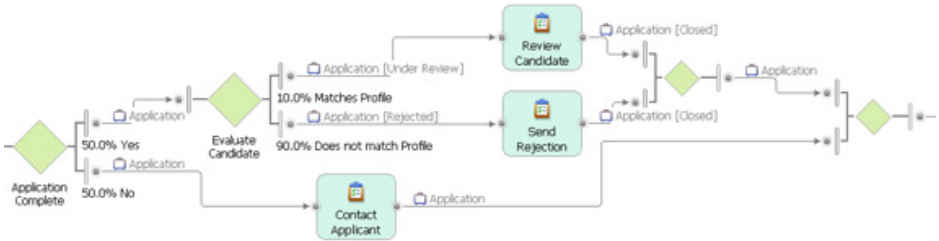
1. Select the incoming connection 1 of the decision and the outgoing connection 2 of the merge as marked in Figure 15 from the alternative compounded that we created in the first example. The connection pair identifies the largest fragment shown in Figure 15.
2. Invoke the pattern and insert * as the name of the task in the first branch. Fill in the other fields of the wizard as Figure 19 shows. Click **Apply Pattern**.

Figure 19. Alternative Compound pattern wizard when reusing a process fragment



The process fragment between the two selected connections is reused on the upper branch of the pattern, as Figure 20 shows:

Figure 20. Reused process fragment



Related patterns: To add more tasks or subprocesses to a branch of the pattern, apply the Insert Task, Insert Process or Sequence patterns to a connection on this branch. To model a process where control can flow from one branch to another before the two branches are merged, apply the Alternative Branch pattern. To merge several gateways into a single gateway, use the Merge Elements refactoring. For refactorings, see Part 4 of this series.

Parallel Compound

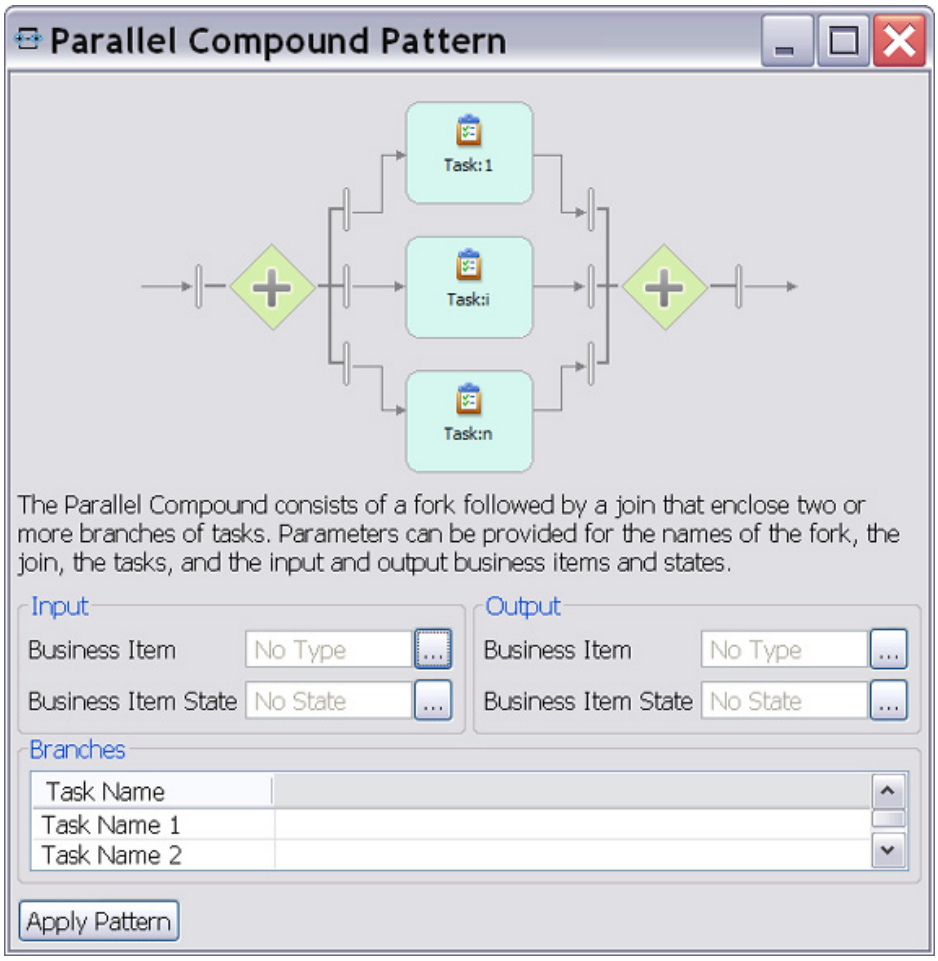
Pattern name: Parallel Compound.

Intent: Use this pattern to add a flow of several parallel branches to your process model that begins with a fork and ends with a join.

Also known as: The Parallel Compound pattern is a combination of the Parallel Split and the Synchronization workflow patterns (see Russell et al. in Resources below) that enclose several parallel branches. This pattern implements the parallel routing and rendezvous of flows.

Structure: You can invoke the parallel compound pattern by selecting one connection or a pair of connections. A wizard opens as Figure 21 shows:

Figure 21. Parallel Compound pattern wizard

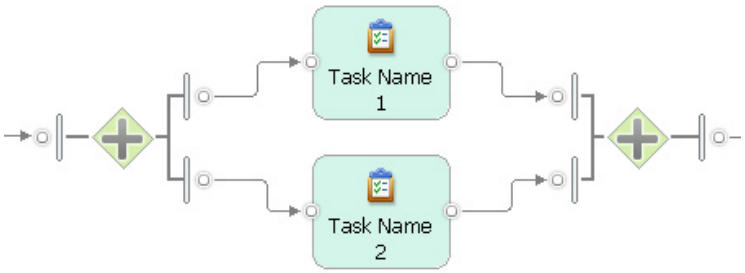


In this wizard you can enter information about the input business item and business item state of the fork that starts the pattern, and about the output business item and business item state of the join that ends the pattern. Recall that a fork or join can have a user-defined name in WebSphere Business Modeler, but these names are not visible in the diagram and are therefore not editable in the wizard.

You can also specify the name of one task for each parallel branch. The wizard inherits the input and output business items and business item states from the information provided for the fork and join, which are therefore not editable for the task. To add or remove additional branches to or from the list, right-click and select **Add** or **Remove** from the context menu.

Consequences: When you apply the pattern without entering any information in the wizard, Modeler refines the selected connection with the process fragment shown in Figure 22:

Figure 22. Default process fragment for Parallel Compound pattern

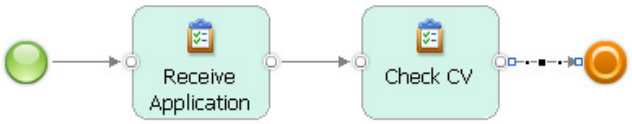


No empty branch can be created in this pattern because all parallel branches are always executed, and executing an empty branch is not meaningful. Therefore when you provide a dash "-" as the name of the task in a branch, pattern interprets the dash as the task's name, in contrast to the Alternative Compound pattern where the dash indicates an empty branch.

Sample: We show an example where we recreate a part of the Evaluate Candidate process from Part 1 of this series, but this time we also model business items and business item states.

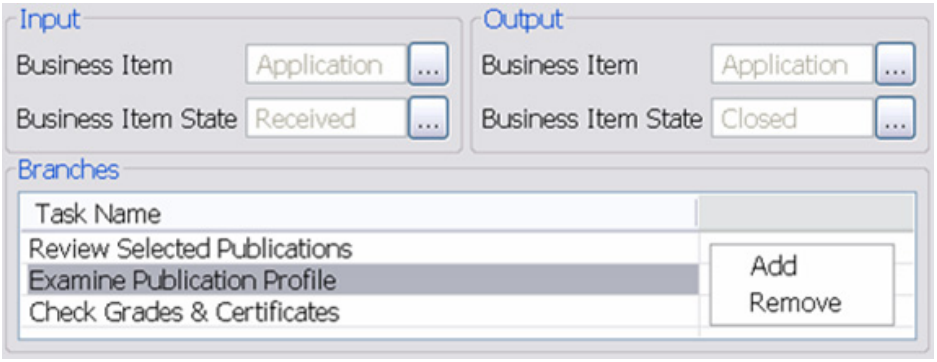
1. Create an initial part of this process and select the connection between the Check CV task and the terminate event as Figure 23 shows:

   Figure 23. Initial Evaluate Candidate process

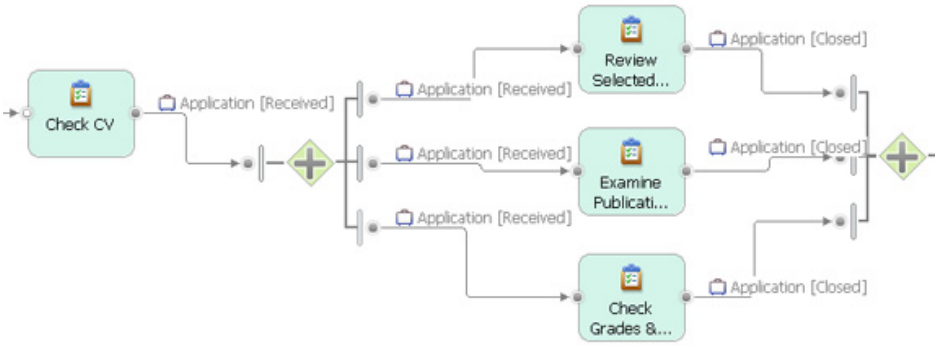
2. Invoke the Parallel Compound pattern.
3. Instantiate it as Figure 24 shows by creating three branches containing the tasks `Review Selected Publications`, `Examine Publication Profile`, and `Check Grades & Certificates`. Recall that you need to add a new branch first before you can enter the third task name.
4. Select the **Application** business item as input and output of the compound.
5. Select **Received** as its input state and **Closed** as its output state. You can use the Application business item as defined in the sample modeling project `HiringExample.mar` that is available with this series.
6. Click **Apply Pattern**.

Figure 24. Parallel Compound pattern with three instantiated branches



The pattern instantiated with data flow is added to your process model. The pattern adds the Application business item as output of the Check CV task to properly connect the required input for the fork, as Figure 25 shows:

Figure 25. Process model after pattern application



As in the case of the Alternative Compound pattern, you can select two connections to reuse a process fragment on one branch of the Parallel Compound pattern. See the description of the Alternative Compound pattern for further information.

Related patterns: To add more tasks or subprocesses to a branch of the pattern, apply the Insert Task, Insert Process or Sequence patterns. To add additional parallel branches to the pattern or to add additional parallel connections between existing branches, select two connections and apply the Parallel Branch pattern (PROD NOTE: Add link to Parallel Branch section). Merge multiple forks and joins using the Merge Elements refactoring (see Part 4 of this series).

Loop

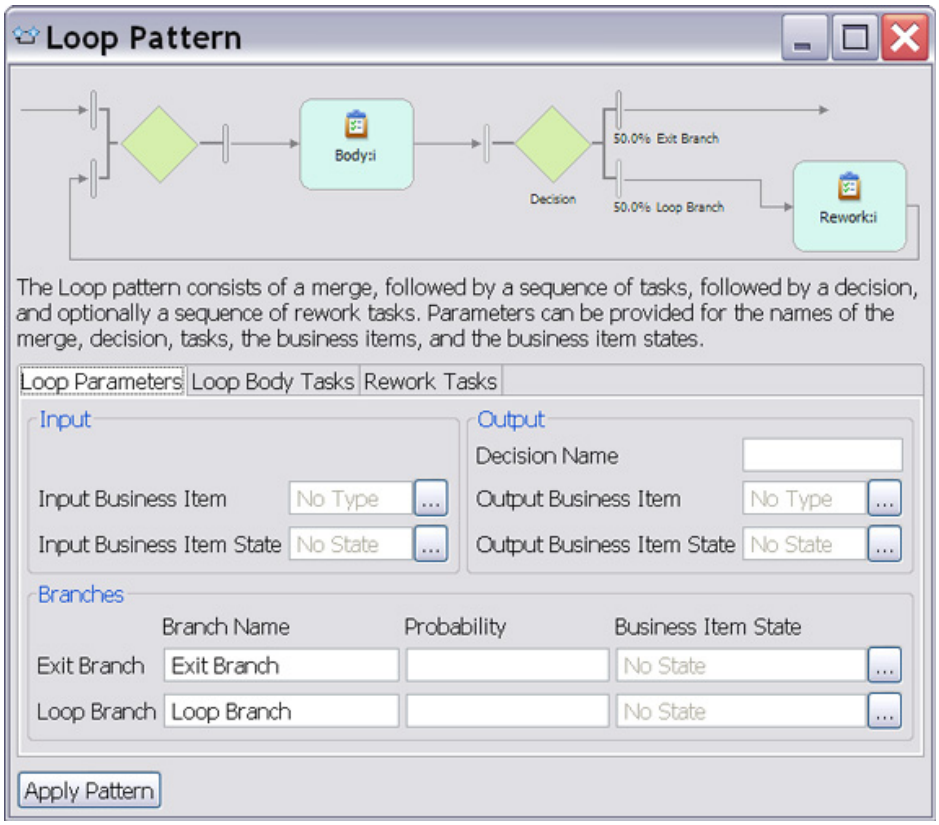
Pattern name: Loop

Intent: The Loop pattern allows you to add a cycle to your process model that begins with a merge, comprises a sequence of loop tasks, and ends with a decision. On the backward connection from the decision to the merge, you can add so-called *rework tasks* that are executed before the loop body is entered again.

Also known as: The Loop pattern allows users to add structured loops (iterations, cycles) in the form of a while … do (pre-test) or a repeat … until/do … while (post-test) loop to the model. It corresponds to the Structured Loop workflow pattern (see Russell et al. in Resources below). Syntactically, it is a combination of the Simple Merge followed by the Exclusive Choice workflow patterns.

Structure: You can invoke the Loop pattern by selecting one connection or a pair of connections. A wizard opens as Figure 26 shows:

Figure 26. Loop pattern wizard

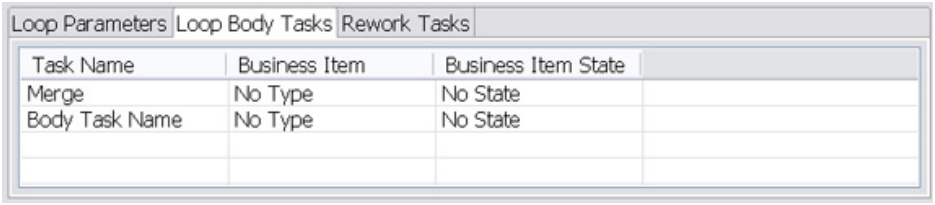


The wizard contains three tabs named **Loop Parameters, Loop Body Tasks**, and **Rework Tasks**.

In the **Loop Parameters** tab you enter information about the input business item and business item state for the merge, the output business item and business item state for the decision, and the decision name. Furthermore, you can specify the names and probabilities of the exit branch that leaves the decision and the loop branch connecting the decision back to the merge. Note that the business item and business item state fields of the branches are synchronized with the corresponding fields for the merge and decision gateways. For example, the output business item state of the decision is the same as the business item state of the exit branch. The business item state entered for the loop branch is the same as the business item state in the first row of the Rework Tasks tab. The business item and business item state of the last rework task are the same as the input business item and business item state of the merge.

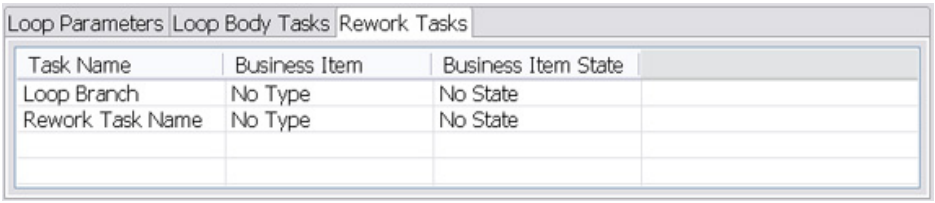
In the **Loop Body Tasks** tab shown in Figure 27, you enter information about the tasks between the merge and the decision in the loop body. The first row shows the merge from which the loop body starts. You can edit the output business item and business item state for the merge. In the subsequent rows, you can add more tasks with their output business items and business item states. To add or remove additional tasks to/from the list, right-click and select **Add** or **Remove** from the context menu.

Figure 27. Loop Body Tasks tab



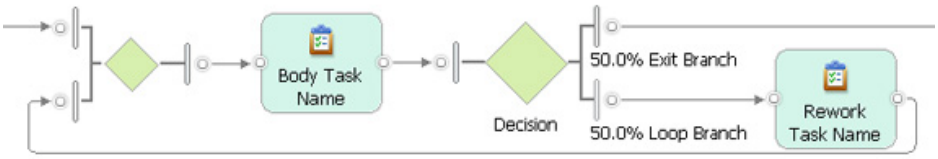
In the **Rework Tasks** tab shown in Figure 28, you can add tasks for the backward connection, i.e., the loop branch that leads back from the decision to the merge. The first row shows the name of the loop output branch of the decision. In the subsequent rows, you can add more tasks with their output business items and business item states. To add or remove additional tasks to/from the list, click right and select **Add** or **Remove** from the context menu.

Figure 28. Rework Task tab



Consequences: When you apply this pattern without entering any information in the wizard, the selected connection is refined with the process fragment shown in Figure 29:

Figure 29. Result of applying the Loop pattern without providing an instantiation



Modeler adds the merge and decision to the model. Between them, it adds a default task named Body Task Name. The two branches of the decision are named Exit Branch and Loop Branch. The exit branch connects to the originally selected connection, while the Loop Branch connects back to the merge. On the loop branch, Modeler places a default rework task called Rework Task Name.

The outcome of the decision gateway determines whether the loop is repeated. This means, the loop body tasks will be executed at least once. The Rework tasks will only be executed when the decision outcome activates the loop branch and the loop repeats. By leaving the loop body empty without any tasks and placing only rework tasks on the loop branch, a while-do (pre-loop test) loop

is defined, because all rework tasks are only executed when the decision outcome activates the loop branch. If you leave the loop branch empty without any tasks and only place tasks on the loop body between the merge and the decision, a do-while (post-loop test) loop is defined.

Loops created with the loop pattern are always properly nested within each other. When using this pattern, you cannot create unstructured cycles with overlapping connections.

Sample: We show an example illustrating the use of business items and business item states. In this example, we create the initial part of the Approve Hire and Issue Contract process that you worked on in Part 1 of this series, but refine it with data flow.

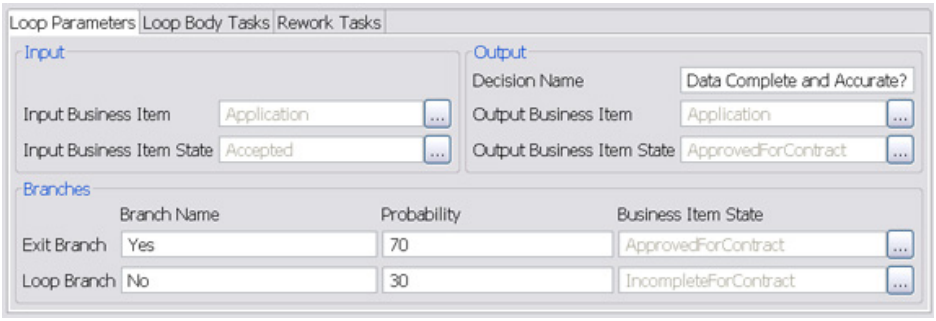
1. Create a small process model consisting of a start event, followed by the Issue Contract task, and ending in a terminate event, as Figure 30 shows:

   Figure 30. Initial process model

2. Select the connection between the start event and the Issue Contract task.
3. Invoke the loop pattern. Fill in the Loop Parameters tab as Figure 31 shows, with `Data Complete and Accurate?` as the name of the decision.
4. Select the Application business item as input of the merge and output of the decision. The business item input state is `Accepted`, and its output state is `ApprovedForContract`.
5. Name the exit branch `Yes` and the loop branch `No`. Enter a `70` % probability that the exit branch is taken and a `30` % probability that the loop branch must be taken.
6. Select **IncompleteForContract** as the business item state for the loop branch. The state for the exit branch is already set from the output business item state of the decision.

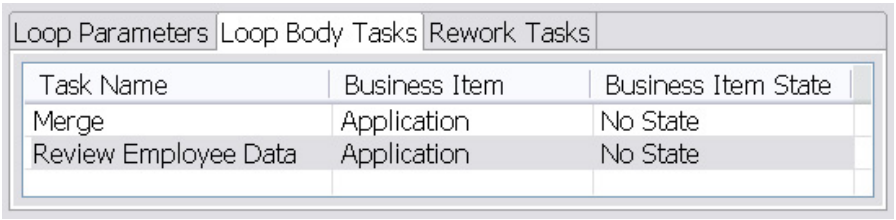
   Figure 31. Loop Parameters tab values

7. Next fill in the Loop Body Tasks tab as in Figure 32. Enter the task `name Review Employee Data` in the second row. Select the **Application** Business item, but do not select a specific business item state because the Application item can be in two different states after the review.
8. The Business Item State field for the merge can show the value `Accepted` at this stage, but when you fill in the Rework Tasks, it will be synchronized to show the value `No State`. Note

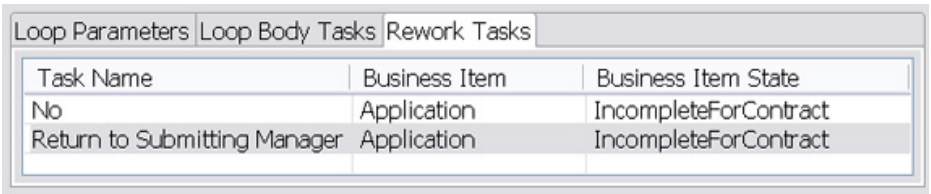
that `No State` in WebSphere Business Modeler indicates 'any state' – a business item is always in some state, but it can, for example, be in one of several possible output states after a task has been executed. This is the case with the Review Employee Data task that sets the state of the Application to either IncompleteForContract or ApprovedForContract, as we can see from the business item states of the loop and exit branches.

Figure 32. Loop Body Tasks tab values



9. Now fill in the Rework Tasks tab as in Figure 33. Enter the task name `Return to Submitting Manager` in the second row, select the `Application` Business item, and specify `IncompleteForContract` as its state. The first row should already show the information that you entered in the Loop Parameters tab for the loop branch.

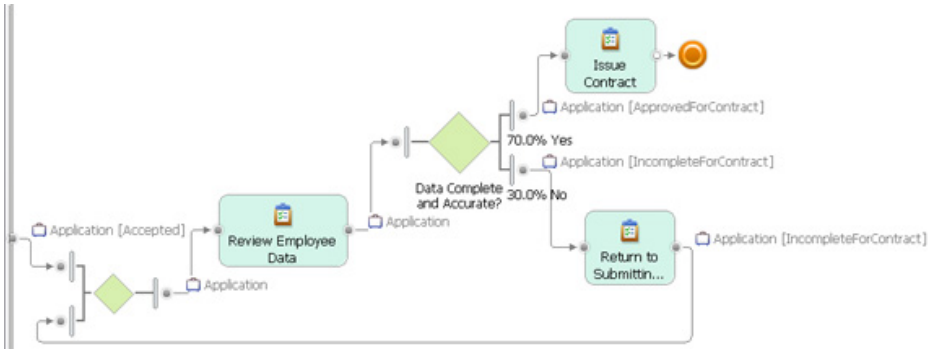
Figure 33. Rework Tasks tab values



10. Click **Apply Pattern**.

As you can see in Figure 34, The start event is replaced by data flow showing the Application business item entering the process in the Accepted state. The decision branches show the two possible outcome states of the Review Employee Data task: ApprovedForContract and IncompleteForContract. For the ApprovedForContract state, the exit branch named Yes is taken and it connects to the Issue Contract task. For the IncompleteForContract state, the loop branch named No is taken and it leads back to the merge. On the loop branch, we can see the Return to Submitting Manager task as the only rework task in this example.

Figure 34. Updated Issue Contract process model



Related patterns: Use the Sequence or Insert Task/Insert Process patterns to add more loop body or rework tasks and subprocesses to a loop. You can add unstructured cycles to your model by applying the Alternative Branch pattern. See the description of this pattern for more details.

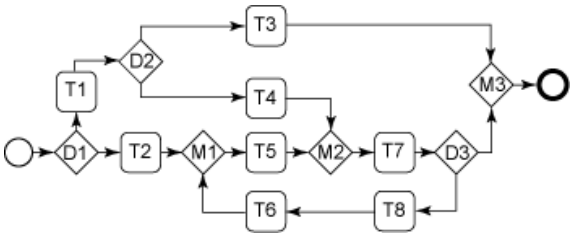
Alternative Branch and Parallel Branch

Pattern name: Alternative Branch Parallel Branch

Intent: The alternative and parallel branch patterns allow you to create a new connection that begins in the connection that you select first and ends in the connection that you select second. For the Alternative Branch pattern, Modeler adds a decision and a merge gateway at the beginning and end of the new connection. For the Parallel Branch pattern, Modeler adds a fork and a join gateway at the beginning and end of the new connection..

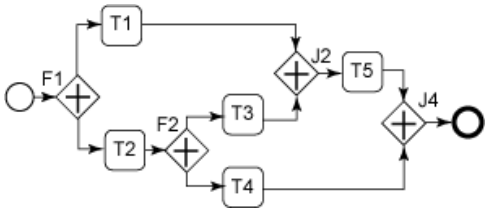
These patterns make it possible to create processes with very flexible branching flows, which go beyond the block-like structures you can create with the Sequence, Compound, and Loop patterns. In addition, the Alternative Branch pattern lets you add unstructured cycles to your model. Figures 35 and 36 illustrate process models you can create using the Branch patterns.

Figure 35. Process model with unstructured alternative branches including a cycle



The process model in Figure 35 contains three merges and three decisions, but these merges and decisions are not placed in well-structured blocks such that one decision would always be exactly matched by a merge. We call such a flow *unstructured*. The process model also contains a cycle involving merges M1 and M2 and decision D3.The two merges M1 and M2 specify two possible entries into the cycle.

Figure 36. Cycle-free process model with unstructured parallel branches



The process model in Figure 36 shows an unstructured model with parallel flows starting and ending in different forks and joins. For example, join J2 synchronizes two branches that are opened by fork F1 and fork F2. Note that a process fragment with forks and joins must never contain a cycle, i.e., a backward connection leading back to a join. A cycle or loop that begins in a join will always create a situation called a deadlock. This deadlock occurs because the join waits

for input from all branches before it can execute, but the loop branch can only provide input after the fork has executed. The control-flow analysis contained in this release of the accelerators lets you automatically detect deadlocks and other types of control-flow errors. See Part 1 of this series for details.

Also known as: The Alternative Branch pattern is a combination of the Exclusive Choice followed by the Simple Merge workflow patterns (see Russell et al. in Resources below).When used to create backward connections, it lets you add the Arbitrary Cycles pattern to your process models. The Parallel Branch pattern is a combination of the Parallel Split followed by the Synchronization workflow patterns (see Russell et al. in Resources below).

Structure: The patterns do not have a wizard. To create unstructured models, select two connections by holding down the SHIFT key. First click the connection where the branch should begin, and second click the connection where the branch should end.

Consequences: To create a correct model, you must only use one type of branch in a fragment of your model. If you mix both types within a fragment, your model will always contain control-flow errors. We have guarded the application of both branch patterns in the accelerators to prevent you from applying a branch incorrectly. You will see the following error messages when trying to apply the Alternative Branch pattern to a fragment that contains a fork or join gateway:

```
To apply this pattern, the enclosing fragment must not contain a fork or a join.
```

A parallel branch may only be added to process fragments that do not contain a decision or merge gateway:

```
To apply this pattern, the enclosing fragment must not contain a decision or a merge.
```

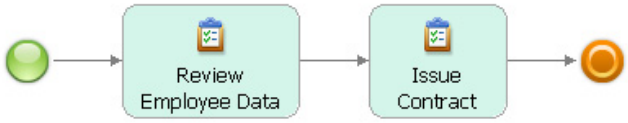
Furthermore, a process fragment with forks and joins must never contain a cycle, i.e., a backward connection leading back to a join. The pattern prevents you from adding a cycle to your model when applying the Parallel Branch pattern. It displays the following error message:

```
The application of the Parallel Branch pattern must not introduce a cycle.
```

Sample: In our first example, we look at the most common use case for the Alternative Branch pattern, namely the creation of a cyclic process by adding a backward connection to the model.

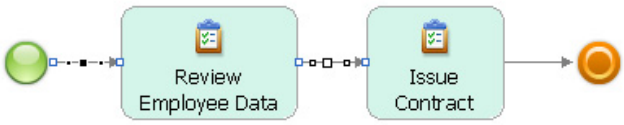
1. Create a process model that contains the Review Employee Data and Issue Contract tasks in a sequence as Figure 37 shows:

   Figure 37. Initial process model



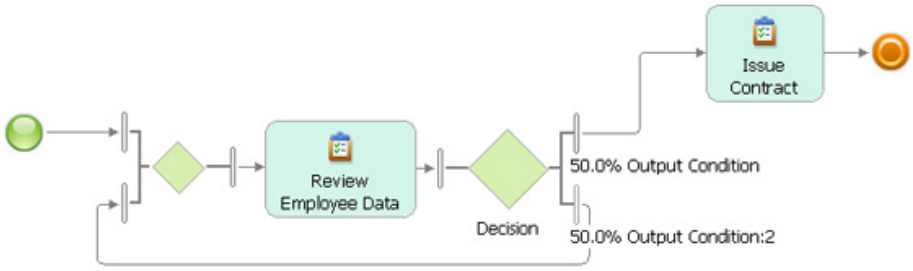
2. Now hold down the SHIFT key and select the connection between the two tasks, followed by the connection between the start event and the Review Employee Data task. Note the two different markings of the selected connections Figure 38 distinguish the order of selection.

Figure 38. Initial process model with connections selected



3. Apply the Alternative Branch pattern, which add a cycle around the Review Employee Data task, i.e., this task becomes part of the loop body that you just created as Figure 39 shows:

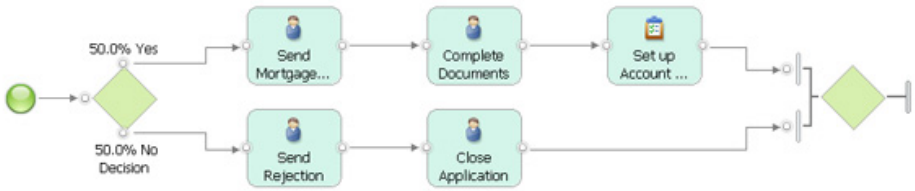
Figure 39. Process model with Alternative Branch pattern applied



4. To add rework tasks to this loop, select the backward connection (i.e., the loop branch leaving the decision) and apply the Sequence pattern as explained earlier in this article.

In our second example, we discuss a use case where you need to add an additional connection between two alternative branches. Take a look at the model of a mortgage approval process with two alternative branches shown in Figure 40:

Figure 40. Initial Mortage Approval process model



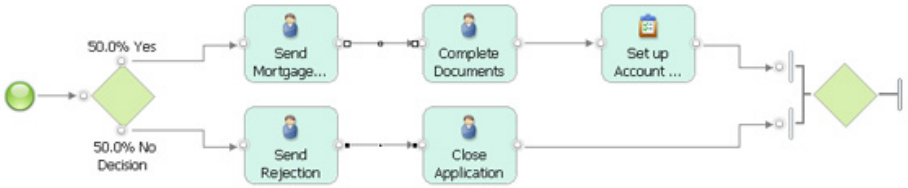
If the customer is not creditworthy, the bank sends a rejection and closes the customer application. If the customer is creditworthy, the bank sends a mortgage offer, completes the documents, and sets up an account for paying out the mortgage.

We notice that the process model assumes that the customer accepts the offered mortgage. This may not always be the case. If the offer is rejected by the customer, the bank employee should contact the customer to find out why, and then close the application.

To make this change in the process model:

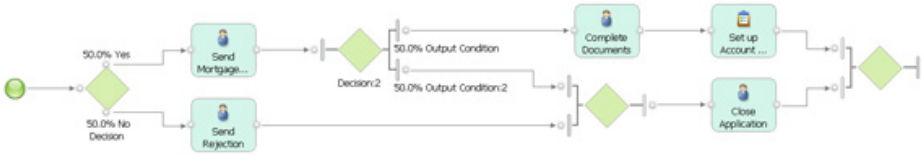
1. Hold down the SHIFT key and first select the connection between the Send Mortgage… and Complete Documents tasks in the Yes branch of the decision, and then select the connection between the Send Rejection and Close Application tasks in the No branch of the decision, as Figure 41 shows:

   Figure 41. Mortgage Approval process with connections selected



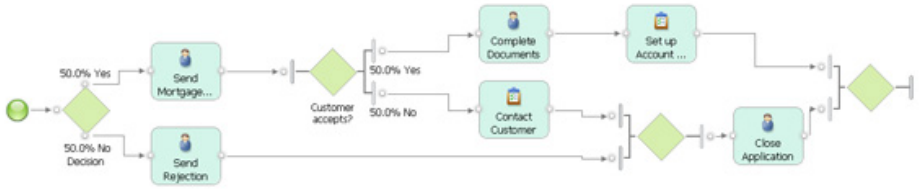
2. Invoke the Alternative Branch pattern. A new connection connects the two branches that is guarded by new decision and merge gateways, as Figure 42 shows:

   Figure 42. Mortgage approval process with the Alternative Branch pattern applied



3. Place the Contact Customer task on this connection by invoking the Insert Task pattern. Change the description of the decision to `Customer Accepts?`. The result is a process model where the customer is contacted when he does not accept the offer, as Figure 43 shows:

   Figure 43. Mortgage approval process with Insert Task pattern applied

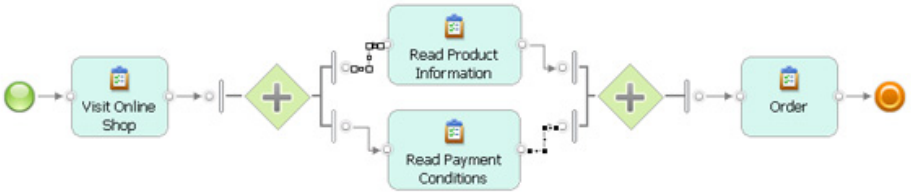


We could apply the Alternative Branch pattern again to allow the bank to revise its offer after it has contacted the customer by adding a backward connection from the output of the Contact Customer task to the input of the Send Mortgage task.

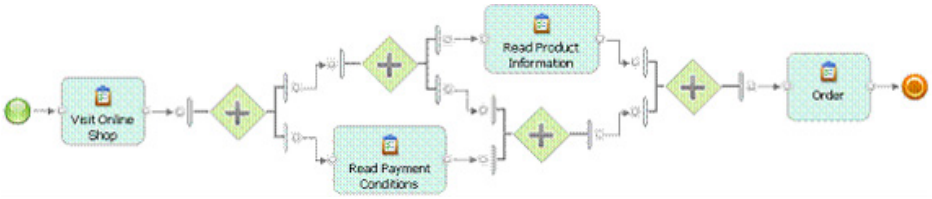
In our third example, we look at a use case for the Parallel Branch pattern. Our example describes the activities of a user when ordering products in an online shop. We would like to modify this process such that the user must additionally read warranty conditions and accept the payment and warranty conditions once he has read both. In parallel, he can read the product information.

To make this change:

1. Select the incoming connection of the Read Product Information task, followed by the outgoing connection of the Read Payment Conditions task as Figure 44 shows:
   Figure 44. Initial Online Shop process model



2. Modeler adds a new parallel branch to the model as Figure 45 shows:
   Figure 45. Online Shop process with a new parallel branch



3. Add the Read Warranty Conditions task to this branch, and add the Accept Conditions task to the connection leaving the join added by the pattern using the Insert Task pattern twice. Figure 46 shows the complete process model.
   Figure 46. Online Shop process with new tasks added

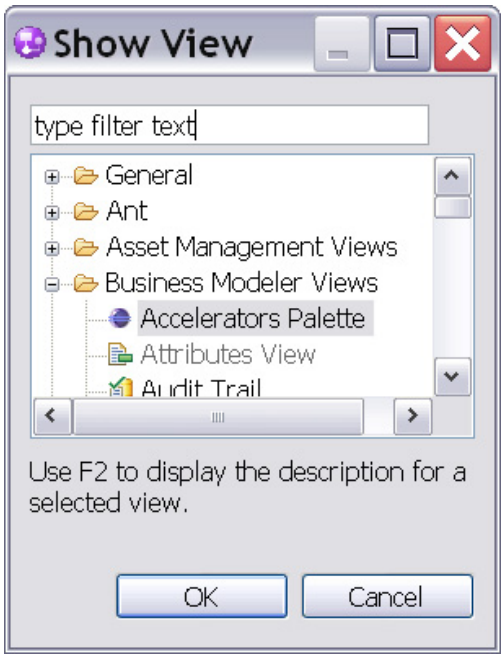
Related patterns: To add additional tasks and subprocesses to the created branches, use the Insert Task, Insert Process, and Sequence patterns. To improve the layout of the resulting process model, you might need to reorder the branches entering or leaving a gateway. The Automatically Order Branches refactoring can automate this for you. To combine gateways that connect directly to each other into a singe gateway, apply the Merge Elements transformation. See Parts 3 and 4 of this series for more details on transformations and refactorings.

Configuration of the Accelerators palette

The Accelerators palette is a configurable Eclipse view that gives you easy access to the available patterns, transformations, and refactorings. You can place it anywhere on your screen and customize it to your needs.

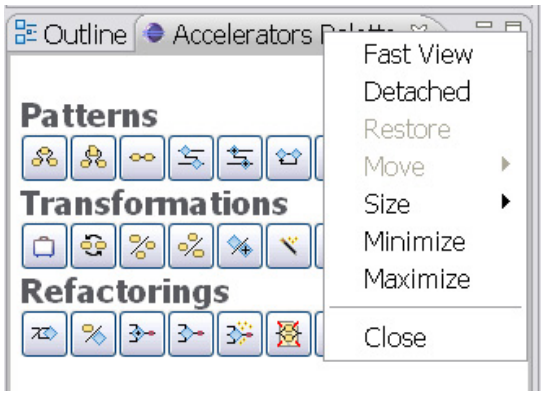
1. To open the Accelerators palette, select **Window > Show View > Other...** in WebSphere Business Modeler. Then scroll down to Business Modeler Views, select **Accelerators Palette** and click **OK**, as Figure 47 shows:

Figure 47. Opening the Accelerators palette



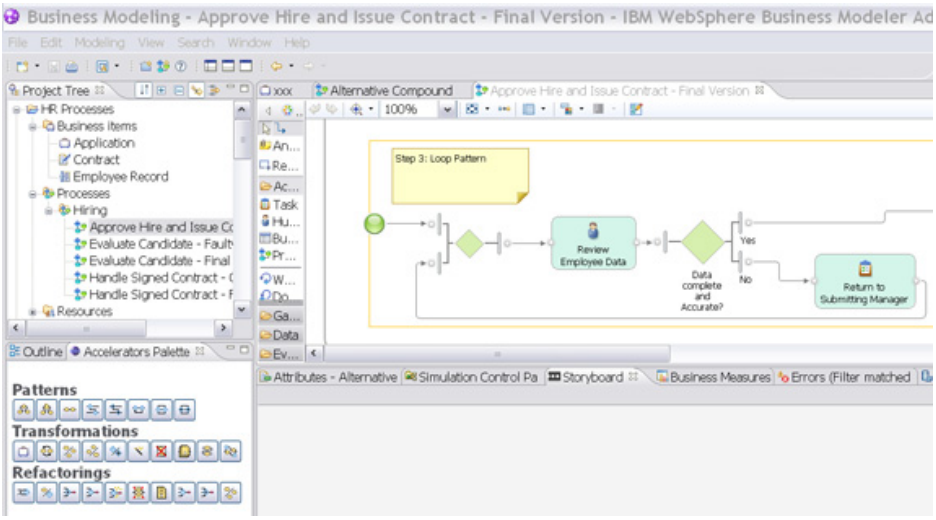
2. The palette will install as a new view at the bottom of WebSphere Business Modeler. You should only need to install the palette once, which makes it available for you the next time you open Modeler on the same workspace. However, when you switch between the 4-pane, 2-pane and 1-pane layouts in Modeler, the palette closes and you must reopen it.
3. Click on an icon in the palette to invoke a pattern, transformation, or refactoring. Hover with your mouse over an icon to see a short description of it.
4. Right-click on the grey tab of the Accelerators palette view and select **Detached** as Figure 48 shows:

Figure 48. Figure 48. Detaching the palette



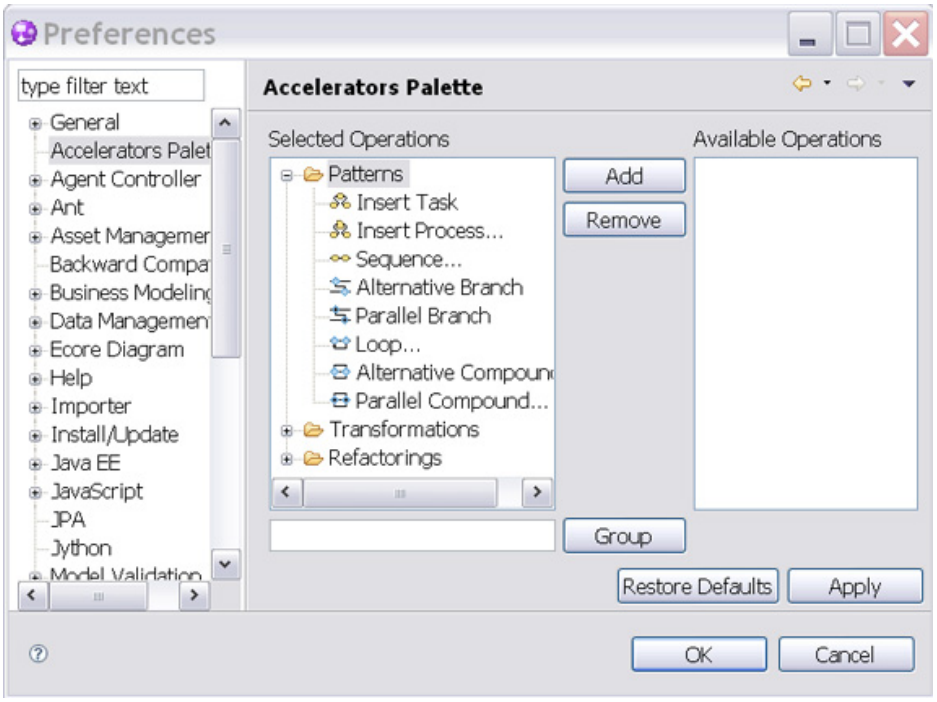
5. You can now move the palette to any position on your screen. You can also resize the palette as any other window. We found it convenient to add the palette to the lower left pane in Modeler as Figure 49 shows:

Figure 49. Moving the palette to lower left of Modeler



6. The Accelerators palette by default shows all icons of all patterns, transformations and refactorings. However, you can also select your subset of accelerators and arrange them in groups with names of your choice.
   For this purpose, select **Window > Preferences.** In the Preferences view that opens**,** select the Accelerators Palette as Figure 50 shows:

Figure 50. Preferences view



The Selected Operations view lists all operations, i.e., patterns, transformations, and refactorings that are currently shown in the Accelerators palette. The Available Operations

view shows you any remaining operations that are not yet shown in the palette. By default all available operations are selected for the palette, and the Available Operations view is empty.

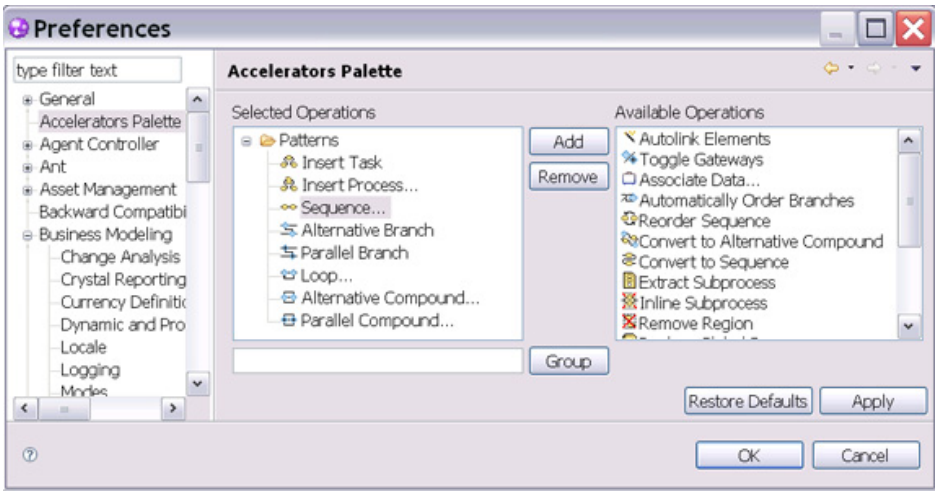
You can perform the following configuration operations:

- Define and add your own groups by entering a group name in the text box below the Selected Operations view and pressing the **Group** button. The new group is added to the Selected Operations view.
- Rename a group by double-clicking and editing a group name in the Selected Operations view.
- Remove a group by selecting the group in the Selected Operations view and then click on the **Remove** button. The group will disappear and its operations will be listed in the Available Operations view.
- Add an operation to a group by first selecting the group in the Selected Operations view, then selecting the operation in the Available Operations view and clicking the **Add** button. The operation will disappear from the Available Operations view and be added to the selected group. You do not need to expand the + sign in front of the group to add an operation.
- Remove an operation from a group by expanding the + sign in front of the group, then select the operation and click the **Remove** button. The operation will disappear from the group and be added to the Available Operations view.
- Move an operation from one group to another by dragging and dropping the operation within the Selected Operations view.
- Change the order of operations within a group by dragging and dropping the operation within the group expanded in the Selected Operations view.

You can only add an operation to one group at a time, but not to multiple groups simultaneously. To return to the initial default setup of the palette, click on **Restore Defaults**. When adding or removing operations to or from a group using the Add or Remove button, you can select multiple operations in the views by holding down the SHIFT or CTRL keys

In Figure 51, we configured the Accelerators palette so that it only shows the eight available patterns.

Figure 51. Accelerators palette configured to show the available patterns



For this configuration, delete all groups except the Patterns group. Reorder any patterns within the Patterns group by dragging and dropping them within the group as you like. Then click **Apply** and **OK**. Your changes should be immediately visible in the palette.

If a configuration change is not visible in the palette, close and re-open the palette for the view to refresh, i.e., select **Window > Show View > Other...**. Then scroll down to Business Modeler Views, select **Accelerators Palette** and click **OK**. A palette configuration is valid with respect to a workspace. This means, for each workspace you can have a different configuration, but it also requires that you configure your palette each time you switch to a new workspace if you want to deviate from its default configuration.

Summary

In this article we introduced you to the patterns available in release 2.0 of the IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler. The patterns let you add common and widely used control- and data-flow structures, such as parallel and alternative branches or loops, to your model in a simple, fast and correct manner. Sample scenarios illustrate the use of the patterns and explain in detail how patterns are instantiated with business items and business item states.

This article also described how to configure the Accelerators palette.

Resources

Learn

- IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler, Part 1: Quality and change management using process patterns (developerWorks, June 2009): This article describes how to compose your business process model by instantiating predefined patterns, and how to apply complex changes to your model with a single click by invoking a transformation or refactoring.
- N. Russell, A.H.M. ter Hofstede, W.M.P van der Aalst, and N. Mulyar: Workflow Control-Flow Pattern Library: A revised View. BPM Center Report BPM-6-22, BPMcenter.org, 2006.
- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: Pattern-Oriented Software Architecture – A System of Patterns, Wiley 1996.
- E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.
- Tutorials and Samples for WebSphere Business Modeler Version 6.2: Learn about business process modeling with WebSphere Business Modeler V6.2 and download additional example models.
- WebSphere Business Process Management V6.2 Information Center: Access more information about Business Process Management with WebSphere V6.2 http://www-01.ibm.com/support/docview.wss?uid=swg21366595
- T. Gschwind, J. Koehler, J. Wong: Applying Patterns during Business Process Modeling. Proceedings of the 6th Intern. Conference on Business Process Management, Springer 2008.This article describes the scientific foundations underneath the capabilities described in this article.
- J. Koehler, J. Vanhatalo: Process anti-patterns: How to avoid the common traps of business process modeling, Part 1 (developerWorks, Feb 2007): This article describes typical modeling errors in process models and how to recognize them.
- J. Koehler, J. Vanhatalo: Process anti-patterns: How to avoid the common traps of business process modeling, Part 2 (developerWorks, April 2007): Part 2 of this series further describes data-flow modeling errors commonly found in process models.

Get products and technologies

- Trial: IBM WebSphere Business Modeler Advanced: Download a free trial version of IBM® WebSphere® Business Modeler Advanced V6.2, IBM's premier business process modeling and analysis tool for business users that offers process modeling, simulation, and analysis capabilities to help business users visualize, understand, and document business processes for continuous improvement. .
- Downloadl: Process Model Accelerators for WebSphere Business Modeler: Download ta set of plug-ins for IBM WebSphere Business Modeler V6.2 that add patterns, transformations and refactorings to the business process modeling environment.

Discuss

- Forum: WebSphere Business Modeler and WebSphere Business Modeler Publishing Server Forum : This forum is for WebSphere Business Modeler and WebSphere Business Modeler Publishing Server customers to post questions and share experiences and solutions .

About the authors

Thomas Gschwind

Thomas Gschwind is a Research Staff Member at the IBM Zurich Research Laboratory and a lecturer at the university of Zurich. His research interests are business process modeling and software engineering. Thomas has developed the software technology foundations underlying the patterns, and led the conceptual design and implementation of the refactoring and transformation operations. You can reach Thomas at thg@zurich.ibm.com.

Jana Koehler

Jana Koehler is a Research Staff Member and manager at the IBM Zurich Research Laboratory. She works on technologies for business process management and distributed systems, and leads the Business Integration Technologies team at the lab. Jana has contributed to the conceptual design of the patterns, refactoring, and transformation operations, and has authored the technical documentation. You can reach Jana at koe@zurich.ibm.com.

Janette Wong

Janette Wong is a Senior Technical Staff Member at IBM. She leads patterns work in the BPM area. Janette led the initial inputs from the field, contributed to refining the pattern ideas and documentation, and the subdequent promotion of this patterns work. You can contact Janette at janette@ca.ibm.com.

Cedric Favre

Cedric Favre is a graduate of the Ecole Polytechnique Fédérale de Lausanne, Switzerland, and a PhD student at the IBM Zurich Research Lab. He designed and implemented the Control-Flow Analysis as part of his Master's thesis during an internship at the IBM Zurich Research Lab. You can reach Cedric at ced@zurich.ibm.com.

Wolfgang Kleinoeder

Wolfgang Kleinoeder is an IBM Research emeritus at the IBM Zurich Research Laboratory working on implementing and testing the Pattern-based Process Model Accelerators. You can reach Wolfgang at wbk@zurich.ibm.com.

Alexander Maystrenko

Alexander Maystrenko is a graduate of the Kiev Polytechnic Institute, Ukraine, and PhD student working at the IBM Zurich Research Laboratory. He helped implement the refactoring and transformation operations during an internship at the IBM Zurich Research Laboratory. You can reach Alexander at oma@zurich.ibm.com.

Krenar Muhidini

Krenar Muhidini is a student at Jacobs University Bremen and was a summer intern in the IBM Zurich Research Laboratory helping to implement the Pattern-based Process Model Accelerators.