IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler, Part 1: Quality and change management using process patterns

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

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

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

Cedric Favre (ced@zurich.ibm.com)
PhD Student, Zurich Research Laboratory
IBM

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

Alexander Maystrenko (oma@zurich.ibm.com)
PhD Student, Zurich Research Laboratory
IBM

Krenar Muhidini
Former intern
IBM

09 December 2009
(First published 03 June 2009)

The IBM® Pattern-based Process Model Accelerators are a set of plug-ins that add patterns, transformations, and refactorings to your business process modeling environment. In Part 1 of this series, you will learn 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. You also learn how to automatically detect control-flow errors.

View more content in this series

Introduction

This article is Part 1 of a series introducing IBM Pattern-based Process Model Accelerators V2.0 (hereafter referred to as the accelerators) for WebSphere® Business Modeler (hereafter referred to as Business Modeler). The accelerators are a set of plug-ins for 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.

View the video for this article.
Download the accelerators.

With the traditional business process modeling approach, process models are drawn by dragging and dropping elements on the drawing canvas and then manually connecting the elements. With the accelerators, you 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 can contain significantly less modeling errors, and you can see productivity gains of about 70% compared to the traditional approach.

This article uses a modeling scenario to show you step-by-step how to use five patterns, five transformations, and three refactorings. The V2.0 release of the accelerators contains a total of 8 patterns, 10 transformations and 7 refactorings. This series covers all the accelerators in detail:

- Part 2 describes all eight accelerator patterns in depth and shows you how to configure the Accelerators palette to suit your modeling needs.
- Part 3 discusses each available transformation in detail, and provides hints on using transformations.
- Part 4 describes how to improve a model's structure with refactoring.

Installing the accelerators

This article assumes that you have basic knowledge of Business Modeler, and that you have some experience creating business process models. You should also be familiar with the basic model elements such as gateways, tasks, subprocesses, and start and terminate events from the Business Process Modeling Notation (BPMN) as available in Business Modeler. The online help in Business Modeler provides you with the necessary background.

To repeat the steps in this article, you need:

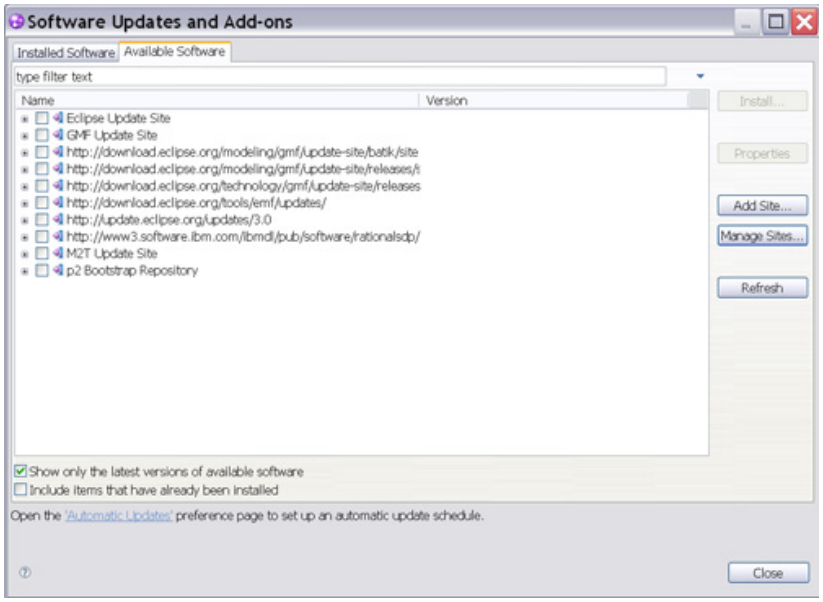
- IBM WebSphere Business Modeler V6.2.0.1 (you must apply Fixpack 1 to Business Modeler V6.2.0).
- The accelerators (download the accelerators).
- The example modeling project HiringExample.mar (this file is included with the accelerators).

Download the accelerator.zip file to your computer. Unzip the file to a directory, for example, C:\temp\acceleratorUpdatesite. The file contains a local Eclipse update site. By using an update site, you can properly manage the configuration of your Business Modeler installation.

To install the accelerators in Business Modeler V6.2, perform the following steps:

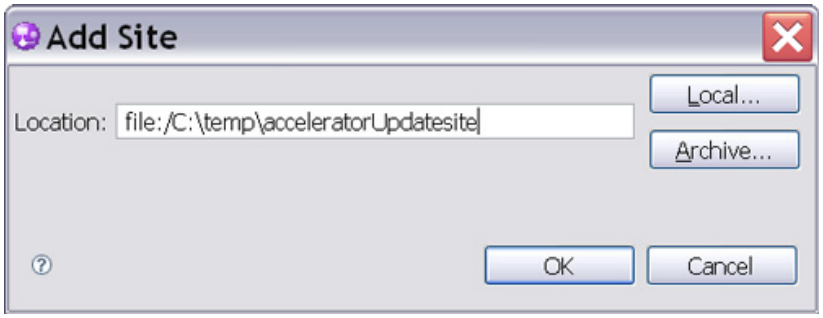
1. Start WebSphere Business Modeler Basic or Advanced.
2. Select **Help > Software Updates …**
3. Click on the Available Software tab (Figure 1).

Figure 1. Installing the accelerators in Business Modeler



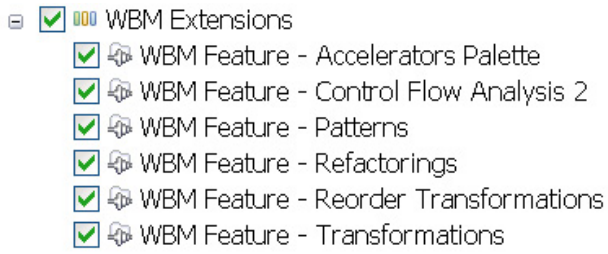
4. Click **Add Site…**. In the window, click **Local…** as shown in Figure 2. A browser window opens that allows you to navigate to the unzipped update site in your file system. Browse to C:\temp\acceleratorUpdatesite. The file location is added in the **Location** field. Click **OK** to close this window.

Figure 2. Adding a local update site



5. The local update site is added to the Available Software tab. You should see the following Business Modeler Extensions listed (Figure 3).
6. Mark the check box in front of the extensions as shown. Then click **Install …** in the upper right corner of the window.

Figure 3. Choosing Business Modeler extensions



7. Eclipse calculates requirements and dependencies. You should see an Install window opening. If you encounter problems, verify that you have Business Modeler V6.2 with Fixpack 1 installed.
8. Make sure that all features are marked and click **Next.** Accept the terms in the license agreements, click **Next** again, then click **Finish**. When prompted for verification, click on **Install All**. The accelerators will be added to your installation of Business Modeler.
9. You will be asked to restart Business Modeler. Answer **Yes**.

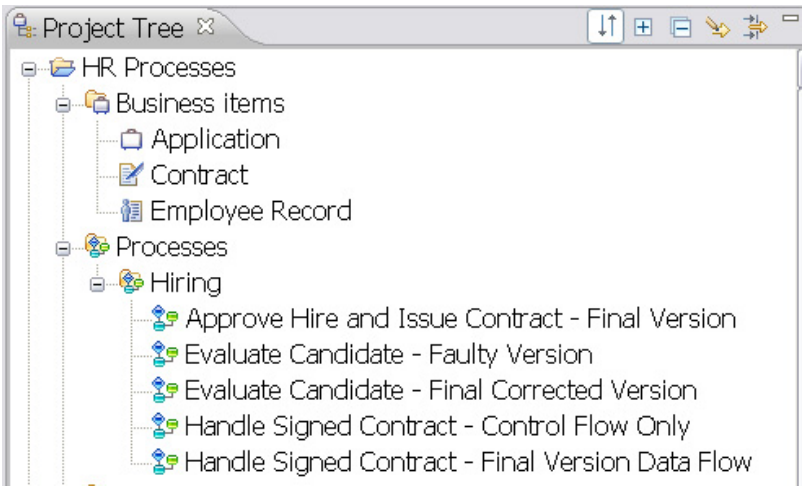
To uninstall the accelerators properly, select **Help > Software Updates …** and scroll down the Installed Software Tab. Search for and select any features that you want to uninstall and click **Uninstall …** Review and confirm that you want to uninstall the checked items that are listed in the next view, then click **Finish**.

Invoking the accelerators in Business Modeler

After successful installation of the accelerators, restart Business Modeler and verify that the accelerators are available in your modeling environment. Apply the 4-pane Layout such that you can see the project tree view in the upper left, the drawing palette and canvas in the upper right, the Outline view in the lower left, and the Attributes view in the lower right.

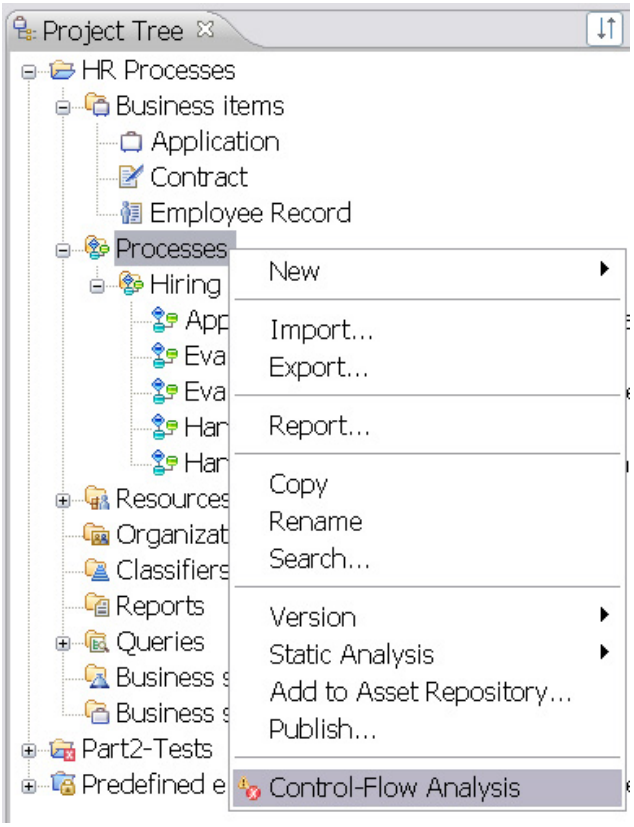
Import the HiringExample.mar file into Business Modeler, into a project that you name `HR Processes`. For example, if you have unzipped the downloaded `accelerator.zip` into your `C:\temp\acceleratorUpdatesite` directory, you can find the `HiringExample.mar` file in the `example` subdirectory in the `acceleratorUpdatesite` directory. In the project tree view, you should see the content shown in Figure 4.

Figure 4. The project tree



Select a process catalog or one of the processes in the project tree. Click right the process to invoke the pull-down menu. You should see a new entry **Control-Flow Analysis** at the bottom of this menu.

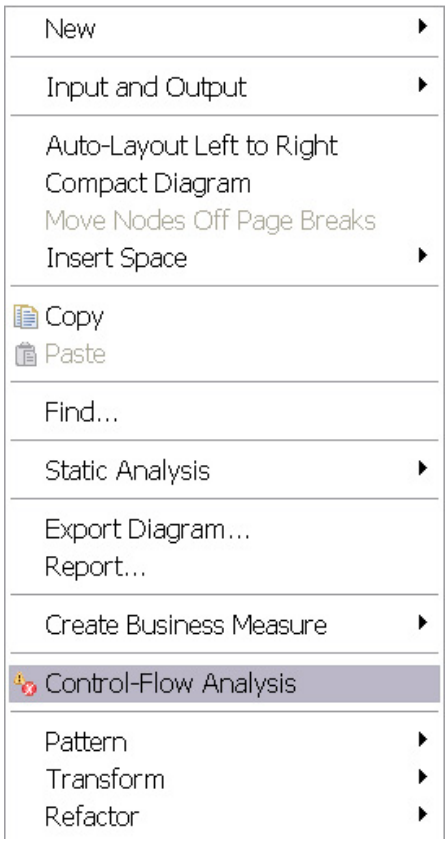
Figure 5. Starting control-flow analysis



Open one of the business process models. The process will open on your drawing canvas. Make sure that you are viewing the process in Free-Form Layout.

Click right anywhere in the white space of your drawing canvas and invoke the pull-down menu. At the bottom of this menu, you should see **Control-Flow Analysis**, followed by the **Pattern**, **Transform**, and **Refactor** submenus.

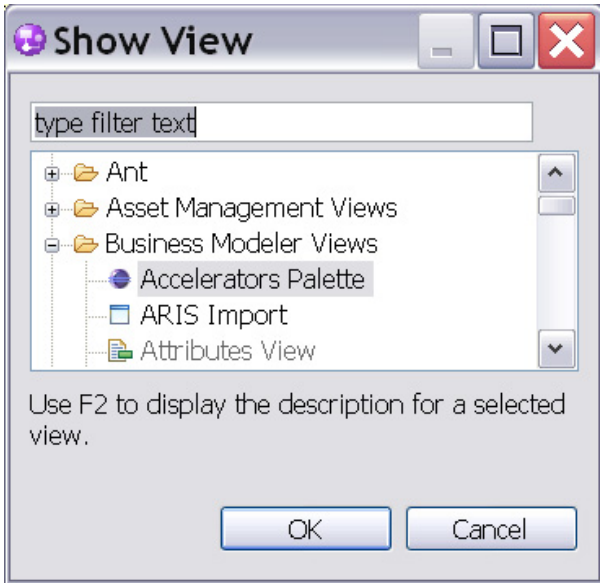
Figure 6. Menu items for accelerators



The Control-Flow Analysis is available to you in Free-Form Layout and in Swimlane Layout; the patterns, transformations, and refactorings are only available in Free-Form Layout.

Select **Window > Show View > Other …**. In the **Show View** window that opens, expand the Business Modeler Views folder. Select the **Accelerators Palette** and click **OK**.

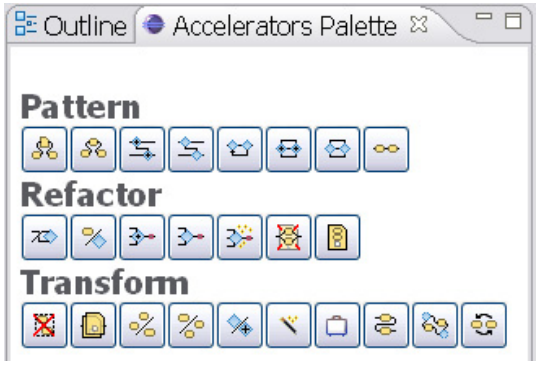
Figure 7. Activating the Accelerators Palette view



A palette with icons for all accelerators is added to your attributes view at the bottom of Business Modeler. You can see a description of each icon when you hover the mouse over the icon. By clicking on an icon, you can invoke an accelerator.

You can also detach the palette from the view or attach it elsewhere in Business Modeler. Furthermore, the palette names and entries can be configured. Part 2 of this series describes how to detach and configure the palette.

Figure 8. The Accelerators Palette icons



The example modeling scenario

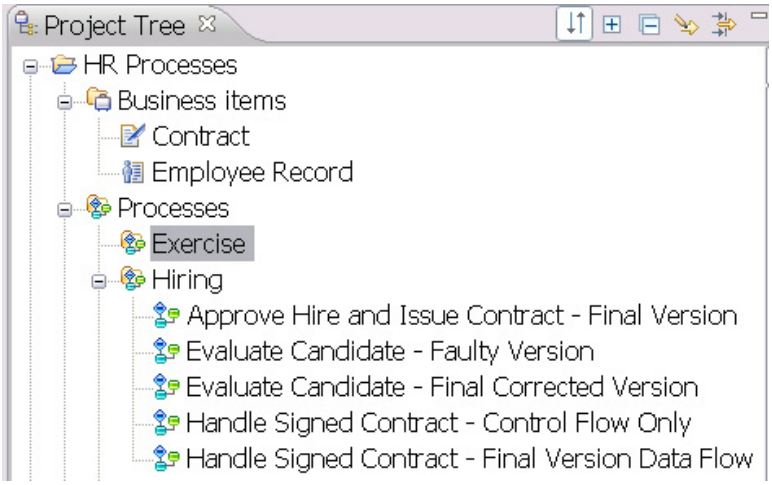
When importing the HiringExample.mar file, you created a modeling project named HR Processes that was added to your project tree. It contains three business items Application, Contract, and Employee Record and the following process models in a process catalog named Hiring.

- Approve Hire and Issue Contract – Final Version
- Evaluate Candidate – Faulty Version
- Evaluate Candidate – Final Corrected Version

- Handle Signed Contract – Control Flow Only
- Handle Signed Contract – Final Version Data Flow

The processes in the Hiring catalog show the final versions of the process models that you will create or change in this article. We suggest that you do not work directly on these process models to keep them unchanged for your reference. In this article, you will create new versions of all processes in a process catalog named Exercise, which you create in the default Processes catalog.

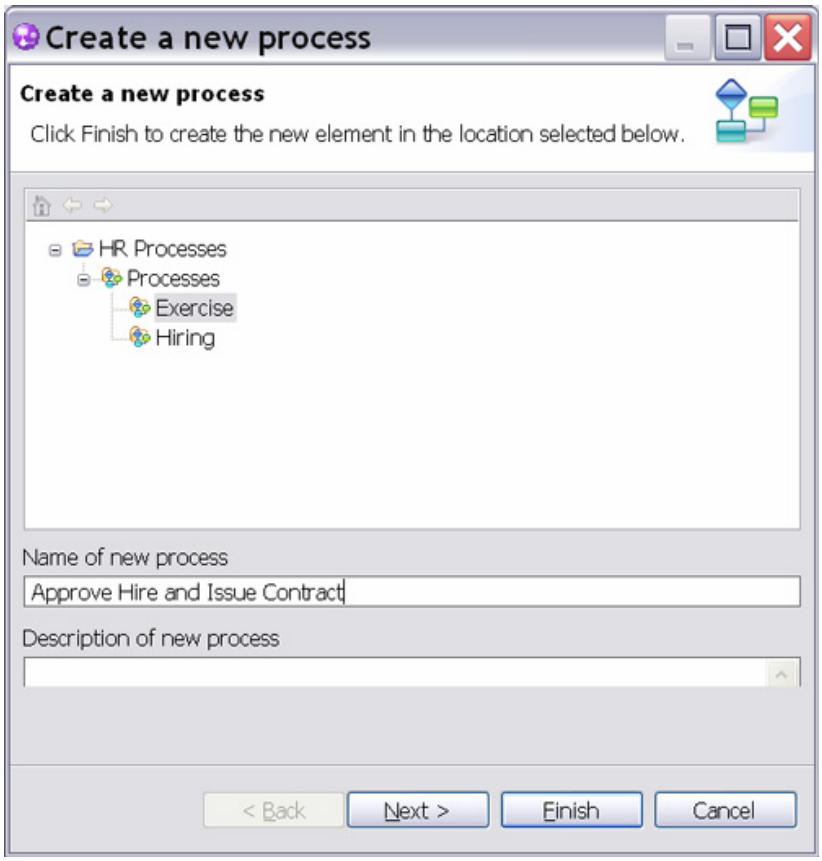
Figure 9. Creating the Exercise process catalog



Composing the processes from patterns

Click right on the Processes catalog in the project tree and select **New > Process Catalog**. Name the new process catalog `Exercise` and click **Finish**. Create a new process model in the Exercise process catalog, name it `Approve Hire and Issue Contract`, and click **Finish**.

Figure 10. Creating a new process model



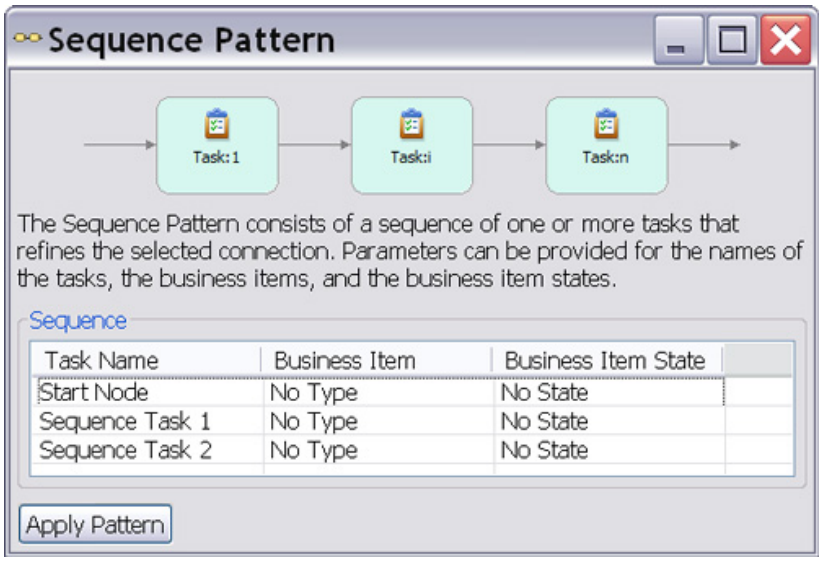
Add a start and terminate event to the process diagram. Connect the two events and select the connection as shown in Figure 11.

Figure 11. Connecting the start and terminate events

Invoke the Sequence pattern either from the Accelerators palette by clicking on the  icon, or right-click on the drawing canvas and select **Pattern > Sequence …** A wizard opens (Figure 12).

You must have a connection selected in order to successfully apply a pattern! To ensure that your selected connection remains selected when you invoke the pattern, do not move your mouse after you click on the connection to select it.

Figure 12. Wizard for the Sequence Pattern



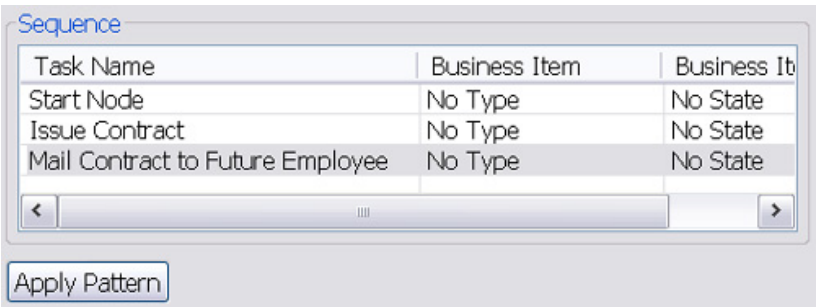
The upper part of the wizard in Figure 12 shows a picture of the pattern illustrating the process fragment structure that you can create by using this pattern. Below the picture is a description of the pattern and a table where you can enter the pattern parameters.

A pattern consists of a number of connected tasks and optional gateways. To apply the pattern, you define the names of the tasks. If you want to connect them by data flow, you additionally specify any business items and optionally the business item states. In this article, you will only instantiate patterns with task names and therefore focus on control flow only. Part 2 of this series shows how to correctly specify business items and states to create patterns with data flow.

In the pattern wizard, the start event is shown in the first row of the table in the Task Name column. In this table cell, the pattern wizard always shows the name of the model element from which your selected connection originates.

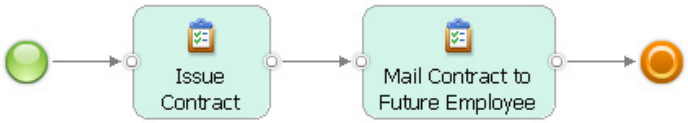
Instantiate the Sequence pattern as shown in Figure 13. Enter the task names `Issue Contract` and `Mail Contract to Future Employee` in the second and third rows of the first column. Then click **Apply Pattern**.

Figure 13. Specifying tasks of the Sequence Pattern



A sequence of these two tasks is added to your process, refining the selected connection.

Figure 14. Task sequence in the process flow

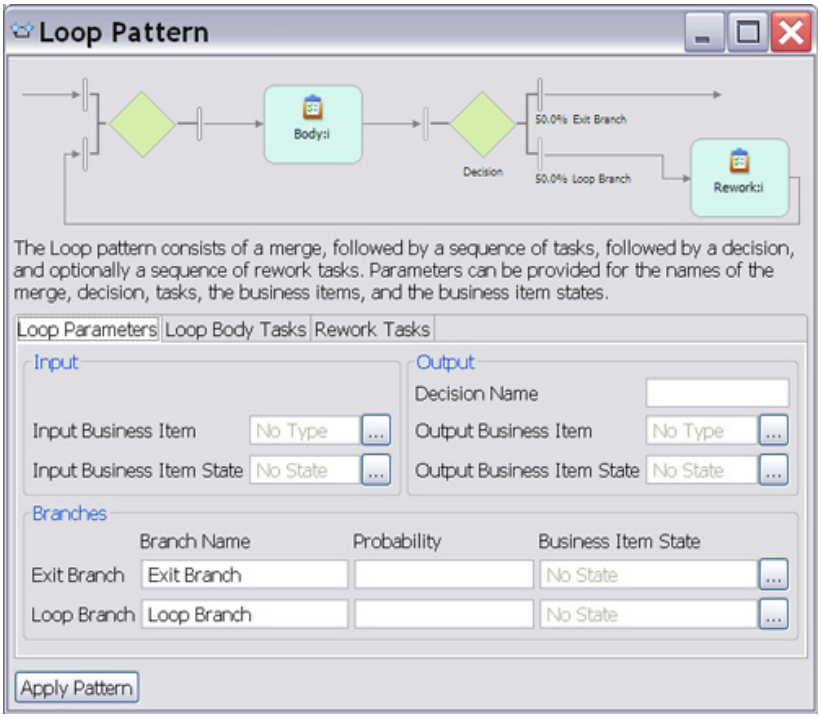


This creates an initial simple process that you will further refine by applying more patterns. The process begins with a loop where the employee data is reviewed and, if necessary, completed. This is followed by two approval steps where the contract is approved by management and human resources.

Select again the connection leaving the start event and invoke **Pattern > Loop …** or click on the  icon in the Accelerators palette. The Loop pattern wizard is shown in Figure 15. The picture shows you the overall structure of a loop fragment. It begins with a merge, followed by a number of body tasks that connect to a decision. The decision has an exit branch that terminates the loop and a loop branch that reconnects back to the merge. On the loop branch, a number of rework tasks can be specified. Three tabs are provided by the wizard:

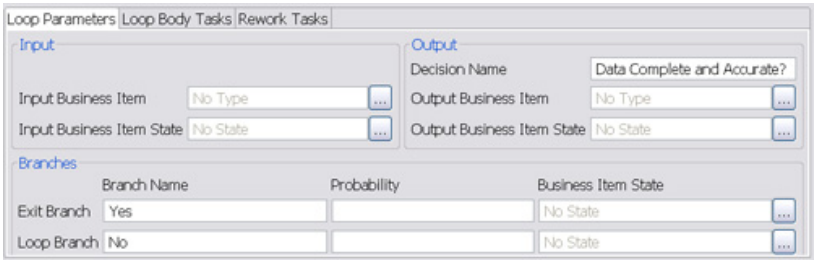
1. Loop Parameters
2. Loop Body Tasks
3. Rework Tasks

Figure 15. Wizard for the Loop Pattern



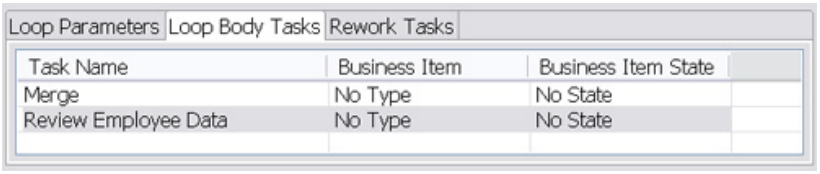
In this article, you will only specify the name of the decision and the names of the exit and loop branches. Enter `Data Complete and Accurate?` in the **Decision Name** field, `Yes` in the **Exit Branch Name** field and `No` in the **Loop Branch** field (Figure 16).

Figure 16. Specifying loop parameters



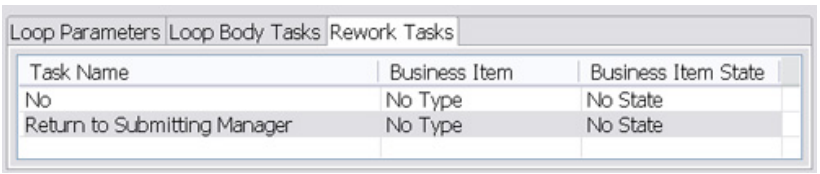
Then open the Loop Body Tasks tab. In the Task Name column, you see Merge as the task name in the first row because the loop body always starts in the merge gateway. In the second row, enter `Review Employee Data` as the task name (Figure 17).

Figure 17. Specifying the tasks in the loop body



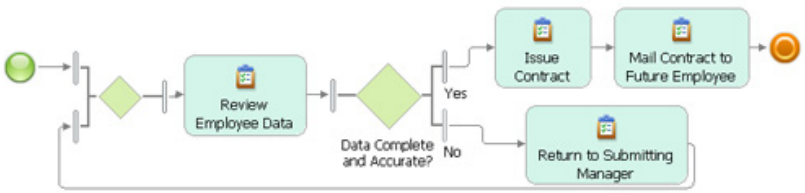
Open the Rework Tasks tab. In this tab, you see the name of the Loop Branch shown as the task name in the first row. Enter `Return to Submitting Manager` as the name of the rework task in the second row as shown in Figure 18. Then click **Apply Pattern**.

Figure 18. Specifying rework tasks



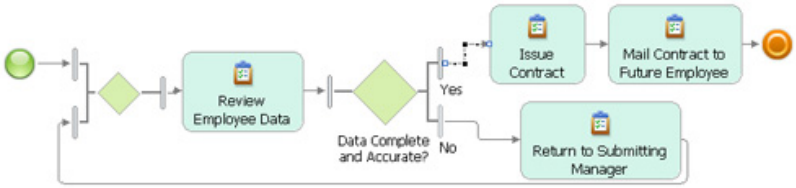
Click right on the drawing canvas and select **Auto-Layout from Left to Right** to improve the layout of the generated process. Your process should now look similar to Figure 19.

Figure 19. Result of applying Sequence and Loop patterns



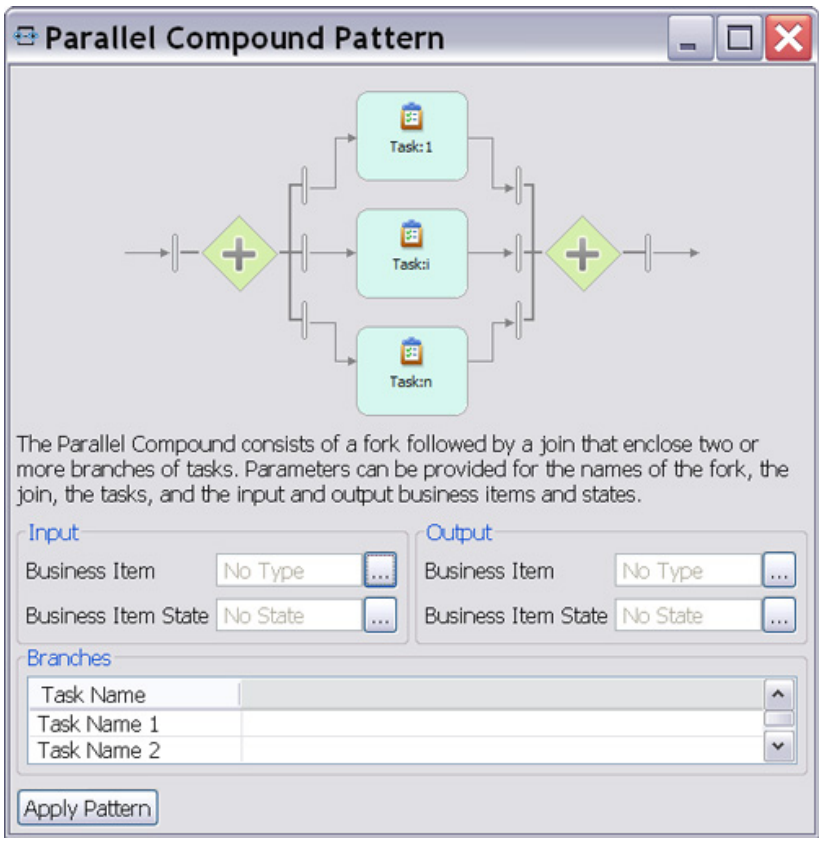
Now add the required management approval tasks. Select the Yes branch of the Data Accurate and Complete? decision (Figure 20).

Figure 20. Selecting the Yes branch



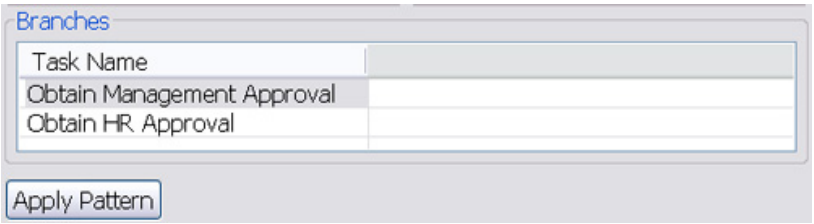
Invoke **Pattern > Parallel Compound …** or click on the  icon in the Accelerators palette. This pattern allows you to add tasks to your process model that should be performed in parallel. Figure 20 shows the wizard for the Parallel Compound pattern.

Figure 20. Wizard for the Parallel Compound pattern



A Parallel Compound is a number of tasks, each occurring on a separate branch, which start in a fork gateway and end in a join gateway. In the table in the wizard, you specify the names of these tasks. Enter `Obtain Management Approval` as the task name in the first row and `Obtain HR Approval` as the task name in the second row as shown in Figure 21. Then click **Apply Pattern**.

Figure 21. Tasks for the Parallel Compound pattern



Your process should now look similar to Figure 22. Recall that you modeled it by instantiating three patterns: Sequence, Loop, and Parallel Compound.

Figure 22. Final version of the Approve Hire and Issue Contract process

Applying transformations and refactorings

When creating the next example process Handle Signed Contract, you will use two more patterns and some of the transformations and refactorings provided by the accelerators.

In the Exercise process catalog, create a new process and name it `Handle Signed Contract`. Create the following tasks:

- `Receive Signed Contract`
- `Verify Necessary Signatures`
- `Contact Employee and Complete Data`
- `Obtain Social Security Information`
- `Check Work Permit Status`

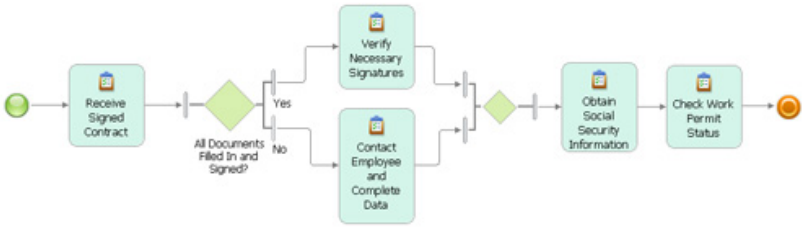
Place the tasks and the start and terminate events on the drawing canvas as shown in Figure 23. Note that we placed three of the tasks in an approximate row, and the other two tasks in an approximate column with sufficient extra white space between the column and the tasks in the row. Imagine that a gateway could be placed there.

Figure 23. Tasks in the Handle Signed Contract process

Invoke **Transform > Autolink Elements** or click on the  icon in the palette. Control-flow connections are automatically added to your model. Apply the Auto-Layout, name the decision

`All Documents Filled In and Signed?` and name its output branches `Yes` and `No`. Then save your model. It should look similar to Figure 24.

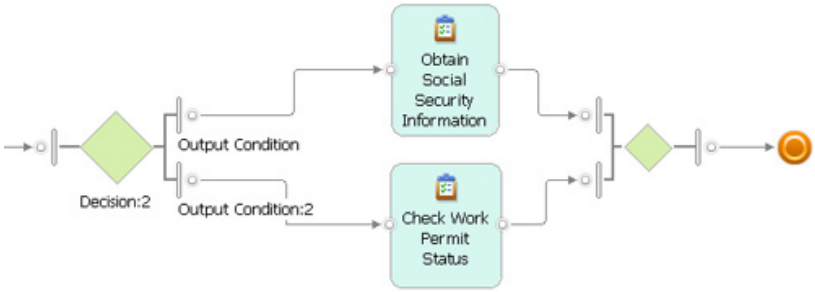
Figure 24. Applying the AutoLink Elements transformation



The Autolink Elements transformation automatically connects tasks based on their geometric position on the drawing canvas. It also inserts decision and merge gateways when you indicate alternative branches by placing tasks in a column and leaving sufficient white space before and after a column of tasks. You can also apply this transformation to only a selected set of model elements and thereby use it to create complex models. This means that you can autolink a few elements, place additional elements on the canvas, autolink them again with the already linked process fragment, and so on. Refer to Part 3 of this series for further details.

The process model is not as accurate as we would like it to be. For example, we only need to check the work permit status of foreigners, but for non-foreigners we obtain their social security information. To change the process accordingly, press SHIFT and select the Obtain Social Security Information and Check Work Permit Status tasks. Invoke **Transform > Convert to Alternative Compound** or click on the  icon with both tasks selected. Apply the Auto-Layout. Your process fragment will change (Figure 25).

Figure 25. Applying the Convert to Alternative Compound transformation



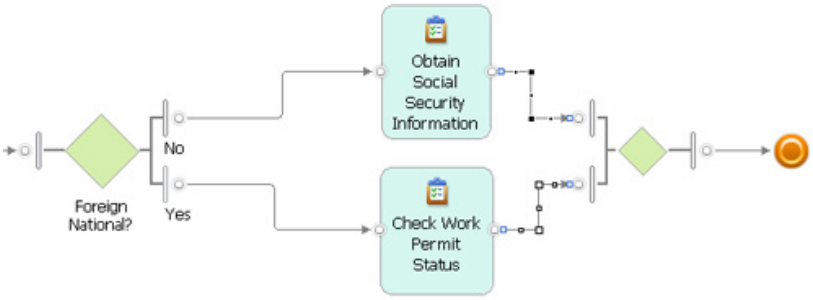
The two selected tasks are placed on two different branches opened by a decision and closed by a merge. Change the name of the decision to `Foreign National?`. Enter `No` as the name of its upper output branch and `Yes` as the name of its lower output branch.

The Convert to Alternative Compound transformation allows you to quickly reorder sequences of tasks such that they are placed on different alternative branches of your process model. Refer to Part 3 of this series for further details.

The next editing step introduces an additional branch to obtain work permits in case the Check Work Permit Status task reveals that a foreign national needs a work permit. Press SHIFT and

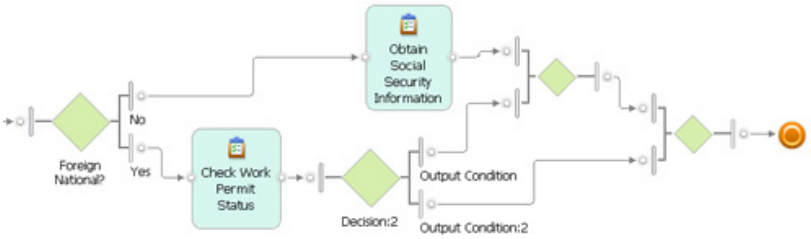
select the outgoing connection of the Check Work Permit Status task first, then select the outgoing connection of the Obtain Social Security Information task as shown in Figure 26.

Figure 26. Adding a branch to the process using the Alternative Branch pattern



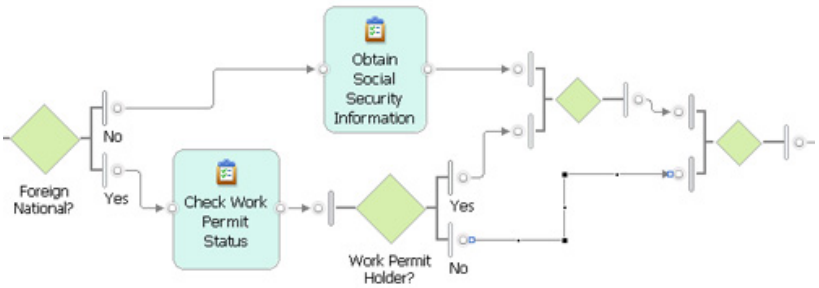
Invoke **Pattern > Alternative Branch** or click on the  icon in the palette. Apply the Auto-Layout. A new decision-merge branch is added to your model (Figure 27).

Figure 27. The new decision-merge branch



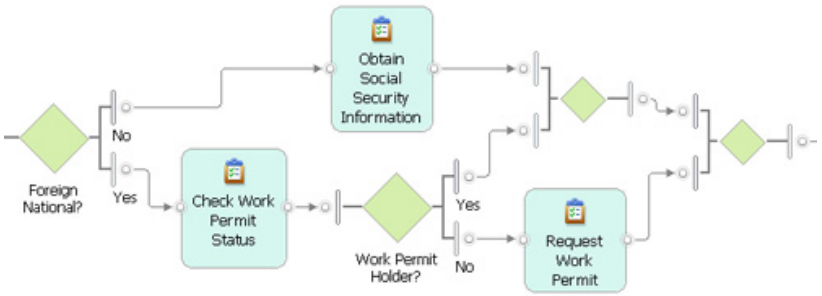
Name the new decision `Work Permit Holder?`. Name its upper branch `Yes`, and its lower branch `No`. Then select the lower branch as shown in Figure 28.

Figure 28. Selecting the lower branch



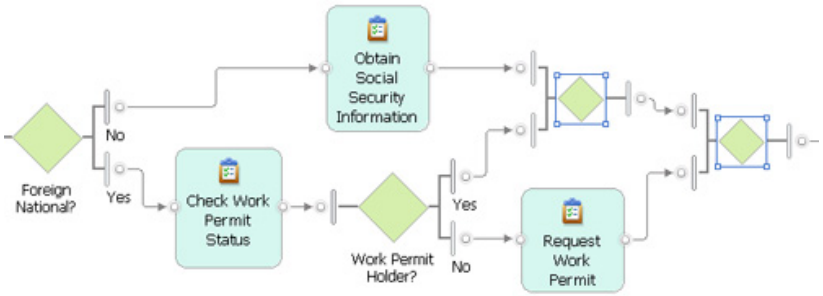
Invoke **Pattern > Insert Task** or click on the  icon in the palette. A new task is added to the `No` branch of the `Work Permit Holder?` decision. Name this task `Request Work Permit`.

Figure 29. Using the Insert Task pattern



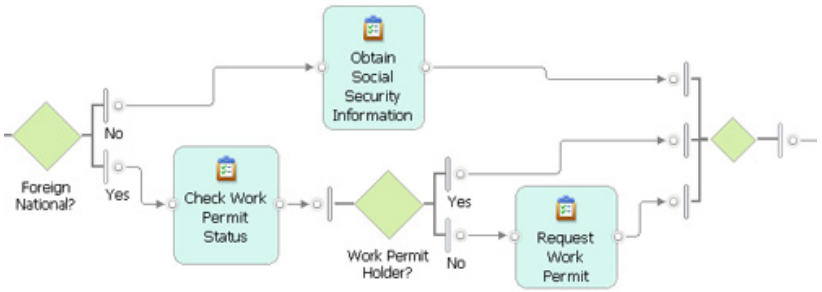
To make your process model tidier, select the two merge gateways and invoke **Transform > Merge Elements** or click on the  icon.

Figure 30. Applying the Merge Elements transformation



The two selected gateways are merged into a single gateway and the control flow is properly reconnected. The process model is now changed as shown in Figure 31. If your model shows incoming connections into the merge that cross each other, try **Refactor > Automatically Order Branches** or click on the  icon in the palette. The Automatically Order Branches refactoring can help to improve the layout of your model if a slight repositioning of the merge gateway on the canvas does not have the desired effect.

Figure 31. Applying the Automatically Order Branches refactoring



Finally, select the incoming connection of the terminate event and apply the Sequence pattern to add two more tasks `Create Employee Record` and `Prepare Welcome Package` at the end of the process.

Figure 32. Adding two new tasks

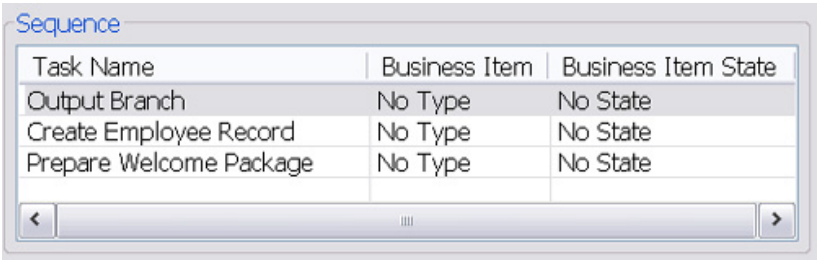
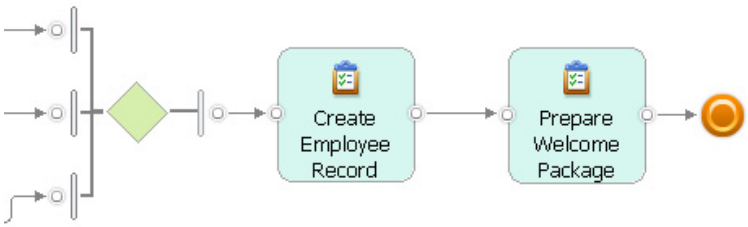
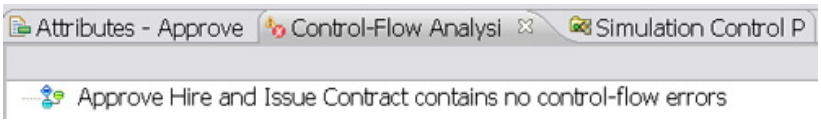


Figure 33. The process flow for the new tasks



Your process is now completely modeled. Recall that we applied the Autolink Elements transformation, followed by the Convert to Alternative Compound transformation, followed by the Alternative Branch and Insert Task patterns. Then we applied the Merge Elements transformation, followed optionally by the Automatically Order Branches refactoring. Finally, we applied the Sequence pattern. With these 7 accelerators, we modeled a quite complex process in a fast and accurate manner. To verify that your process has no control flow errors, run the Control-Flow Analysis. It will confirm that your process is correct.

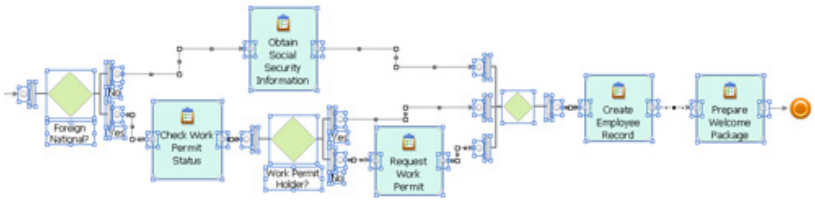
Figure 34. Result of the control-flow analysis



It is a recommended modeling practice to use subprocesses to obtain a better structure for large process models. A subprocess encapsulates a connected fragment of the process containing tasks and gateways and is properly connected to the surrounding parent process. The accelerators help you to quickly add subprocesses to your model while properly restoring the control flow between the new subprocess and the parent process.

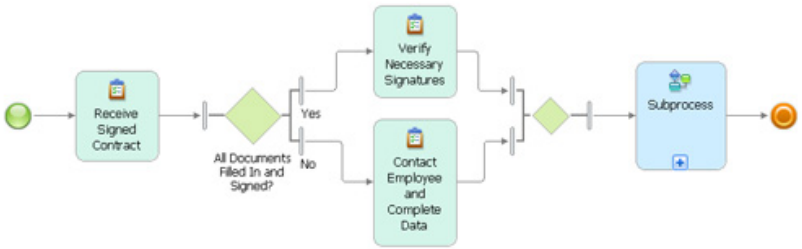
Select the process fragment that starts in the Foreign National? decision and ends before the terminate event as shown in Figure 35. To keep this fragment selected when you invoke the next refactoring, purposefully position the cursor on one of the selected elements in order to preserve your selection.

Figure 35. Selecting a process fragment



Invoke **Refactor > Extract Subprocess** or click on the  icon. A new subprocess is created and opened on the drawing canvas. The selected process fragment is moved into this subprocess. The Handle Signed Contract process now contains a subprocess as shown in Figure 36.

Figure 36. The extracted subprocess

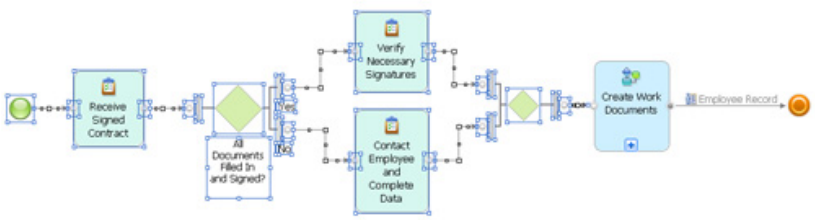


Name this subprocess `Create Work Documents`.

Now, add data flow to the Handle Signed Contract process model. When doing traditional modeling, you have to select each connection and invoke **Associate Data …** Select the last connection in the process model connecting the Create Work Documents subprocess to the terminate event. Use the traditional **Associate Data…** to associate the connection with the Employee Record business item. Imagine that you had to do this for each connection in your process model. This would be quite cumbersome.

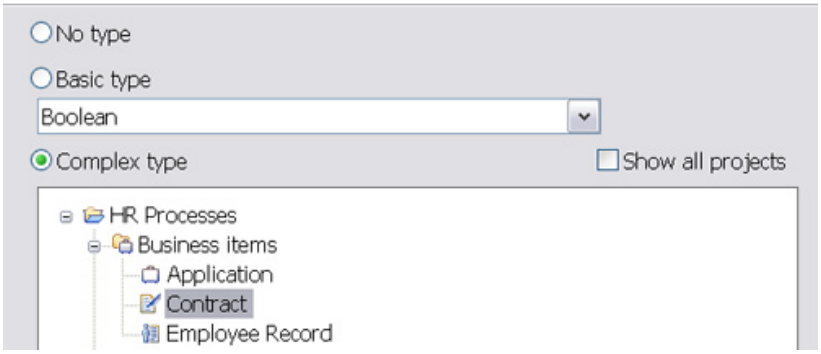
With the new Associate Data transformation, you can now easily associate data with multiple connections through one single operation. Select all the other connections of your process model by pressing and holding the left button of your mouse and pulling it over the fragment of the process model that begins with the start event and ends with the connection leading to the Create Work Documents subprocess. Note that the subprocess itself is not selected (Figure 37).

Figure 37. The connections to be selected



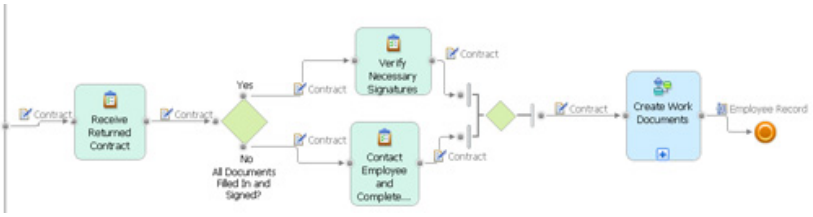
Now invoke **Transform > Associate Data …** or click on the  icon making sure to preserve your selection. A window opens to allow browsing to all available business items. Click the **Complex type** radio button and select the Contract business item as shown in Figure 38. Then click **OK**.

Figure 38. Associating the Contract business item



Data flow is assigned to the Handle Signed Contract process. With this last transformation, you have completed the model of this process.

Figure 39. The completed process model for Handle Signed Contract



To add data flow to the Create Work Documents subprocess, open this subprocess in a new page and apply the **Associate Data …** transformation as desired. To match the data flow in the Handle Signed Contract process, the Create Work Documents subprocess should take a Contract as input and provide an Employee Record as output. You can achieve this by applying the **Associate Data …**transformation twice.

Detecting and repairing modeling errors

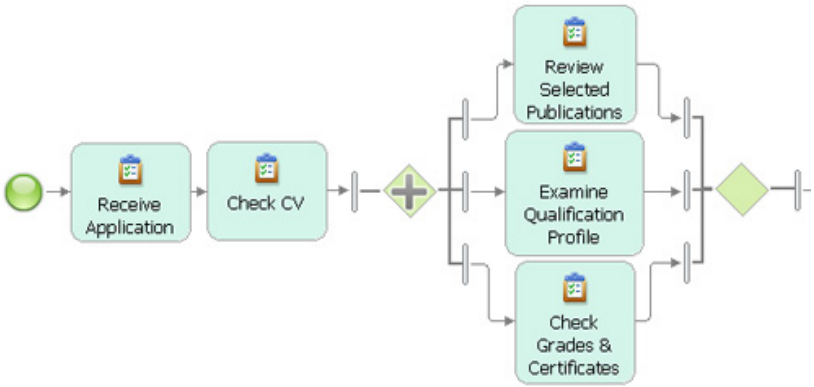
Many process models created using traditional methods contain modeling errors. When dragging and dropping single graphical model elements on a canvas, users often lose sight of the control flow that they design when connecting the single model elements and adding gateways to their process model. When creating a process model using the accelerators, notably the patterns, you cannot introduce modeling errors into your model. However, on traditionally created models, the accelerators help you to easily detect, locate, and correct errors in those models.

Copy the Evaluate Candidate-Faulty Version process from the Hiring process catalog to the Exercise process catalog and open it in the editor. The Evaluate Candidate process describes the evaluation and interview process for an applicant applying for a student internship or researcher position. This process is used by the research team co-authoring this article. It was modeled in

the traditional way without using any patterns, refactorings, or transformations. Unfortunately, it contains two control-flow errors.

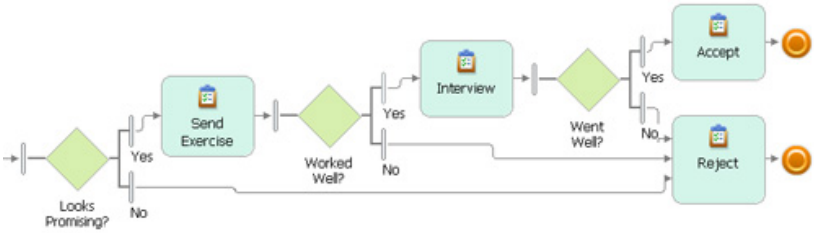
The initial part of this process is shown in Figure 40. The first task in the process is Receive Application, which captures the arrival of an application folder. The task is followed by the Check CV task to examine the curriculum vitae of the applicant. Then, a fork gateway follows that opens three parallel branches. The upper branch contains the Review Selected Publications task where available publications of the candidate are examined and some are selected for further review. The middle branch contains the Examine Qualification Profile task to check whether the skill profile of the candidate matches the skills needed by the team. The lower branch contains the Check Grades & Certificates task that looks at the formal documents included in the application. The three branches are closed by a merge gateway.

Figure 40. Initial part of the Evaluate Candidate process



The final part of the Evaluate Candidate process contains three decision gateways. If the candidate looks promising after the initial reviews, an exercise is sent to the candidate to test technical and collaboration skills. If the candidate scores well on the exercise, an interview is scheduled. If the interview is successful, the candidate is usually accepted. In all other cases, the candidate is rejected — that is, the three No branches of the decision enter the Reject task at the end of the process model.

Figure 41. The final part of the Evaluate Candidate process

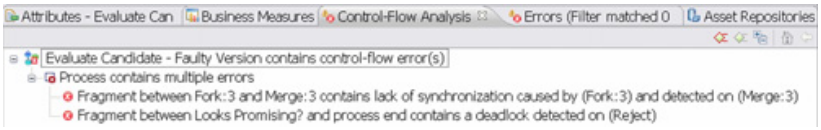


Each part of this process contains a very typical modeling error that usually happens when models are created in the traditional way. Click right on the Evaluate Candidate process in the project tree and invoke the **Control-Flow Analysis** from the menu. Alternatively, click right on the white space of the drawing canvas and invoke **Control-Flow Analysis** from the menu that is associated with the drawing canvas. A new view containing error messages is added at the bottom of Business Modeler (Figure 42).

Figure 42. Errors detected by the control-flow analysis

Click the plus sign (+) in front of the error message to view further details. More detailed error descriptions inform you about the type of error (lack of synchronization or deadlock) and the location of the error in some fragment of the process.

Figure 43. The detailed error description



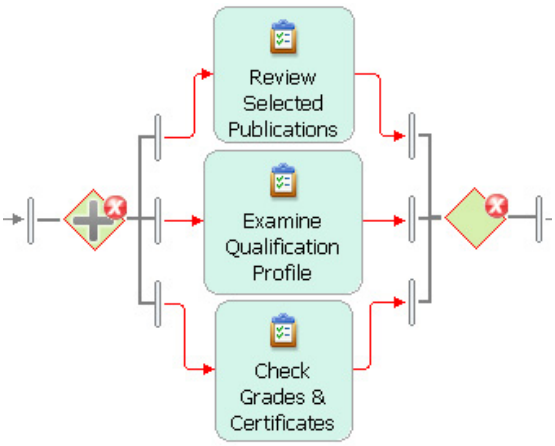
Our example process contains a lack of synchronization error in the initial part. The error is caused by the fork and detected on the merge. The final part of this process contains a deadlock that is detected on the Reject task. By double-clicking on an error message the fragment containing the error is highlighted in red (see Figures 44 and 45 in the following section).

A lack of synchronization modeling error

A lack of synchronization is an error where two or more parallel branches are not correctly synchronized in the process model. Because of this, all tasks following the insufficient synchronization are executed more often than intended.

In our example, the fork gateway opens three parallel branches that are synchronized by a merge gateway. The merge waits for one branch to finish and then continues to route the flow to the final part of the process. This means that, for each branch entering the merge, the final part of the process will be executed once (three times in total). Of course, this is not the intended process behavior.

Figure 44. A lack of synchronization error in the Evaluate Candidate process model

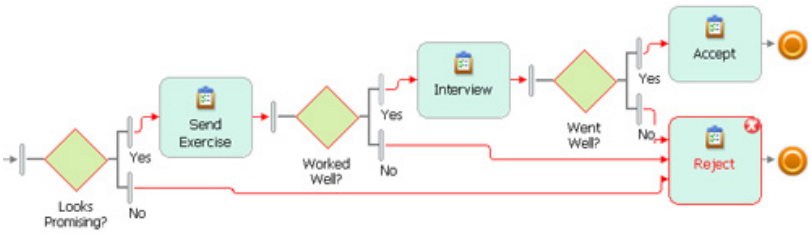


A deadlock modeling error

A deadlock is a modeling error that causes the control flow to stop at some task or gateway in the process model because the task or gateway waits for some input that can never arrive. In our example, the Reject task waits for three control inputs from each of its incoming connections. However, only one of the three required control flow inputs is activated in any execution of the process, because the three decision gateways always activate exactly one of their output branches. The control flow will therefore stop at the input of the Reject task, which can never execute.

Each decision in our example process opens two exclusive branches. Either the Yes or the No branch is taken. For example, if the first decision activates the No branch, the other two decisions are no longer taken as the control directly proceeds to the Reject task. It is therefore impossible for the Reject task to receive all three No inputs at the same time.

Figure 45. A deadlock error in the Evaluate Candidate process model



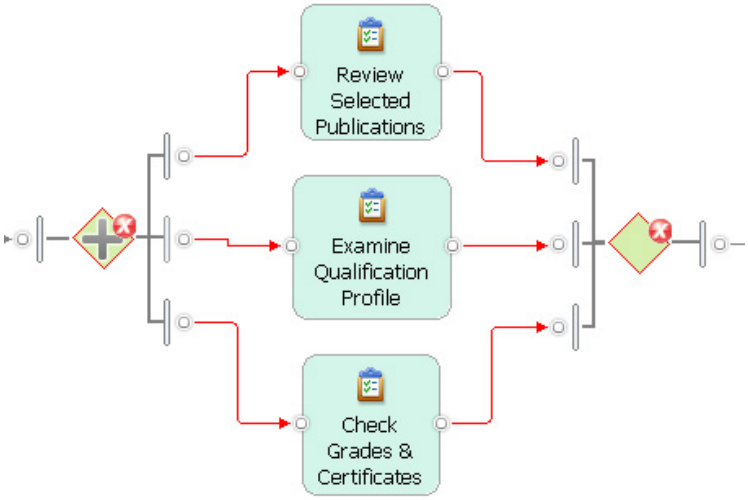
You can observe the problematic behavior caused by these two types of errors by simulating your process model with Business Modeler. To read more about modeling errors, see the articles in Resources.

Recall that the control-flow analysis detected two errors in the Evaluate Candidate process:

1. A lack of synchronization caused by pairing a fork gateway with a merge gateway.
2. A deadlock caused by a task that expects three inputs coming from alternative branches.

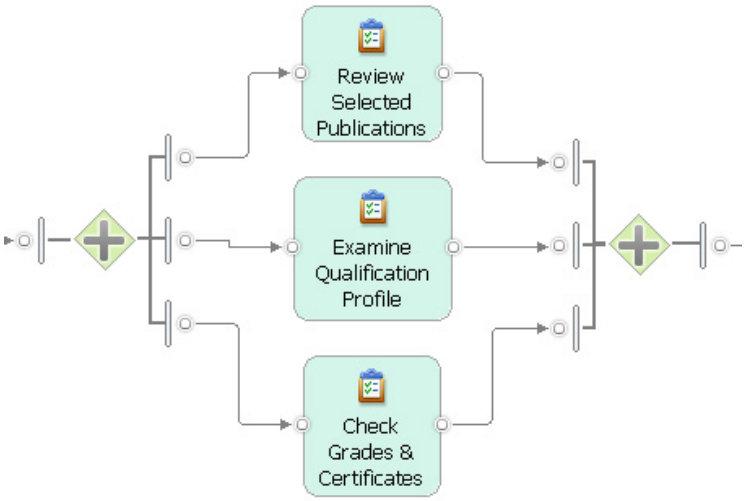
To identify the gateway causing the lack of synchronization, examine the affected process fragment. The fork is correct, because we want to execute all three tasks in parallel. Consequently, the merge must be replaced by a join that correctly matches the fork.

Figure 46. Examining the lack of synchronization error



To correct the lack of synchronization error, select the merge and invoke **Transform > Toggle Gateways** or click on the  icon in the palette. The merge will be replaced by a join. Save the process model and then run the Control-Flow Analysis again — the error is no longer reported (Figure 47).

Figure 47. Applying the Toggle Gateways transformation corrects the lack of synchronization error

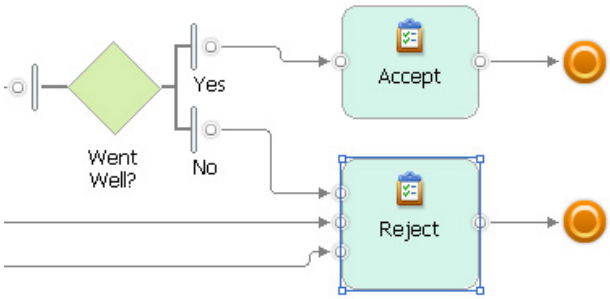


Select either the fork or the join and run the Toggle Gateways transformation again. It will now toggle both gateways simultaneously — that is, the fork is changed to a decision and the join is changed to a merge. The reason for this behavior: when you toggle a gateway that is part of a process fragment that contains an error, only the selected gateway is toggled. When you select a gateway that is paired with one or more correctly matching gateway(s), the set of matching gateways is always toggled together to ensure that you do not introduce a modeling error into your model when toggling gateways. In a nutshell, the Toggle Gateways transformation allows you to

correct errors caused by gateways, but it will always ensure that you do not transform a correct model into an incorrect one. See part 3 of this series for details.

To correct the deadlock error, first make the error more explicit. Select the Reject task and invoke **Refactor > Activity to Gateway Form** or click on the  icon in the palette (Figure 48).

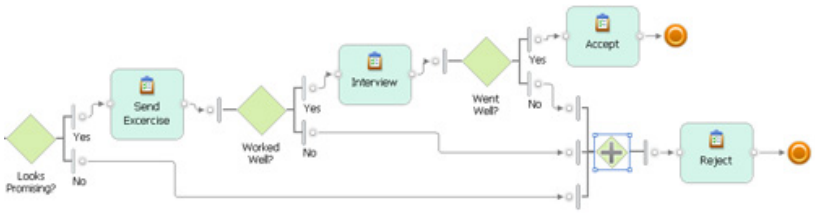
Figure 48. Applying the Activity to Gateway Form refactoring



The three connections that lead directly to the Reject task are converted to a join with three incoming connections and a single outgoing connection that leads to the task. The Activity to Gateway Form refactoring provides you with a correct translation of the input and output logic of a task into the corresponding gateways. See part 4 of this series for details.

You can now see that the outgoing flow of the three decisions is incorrectly matching the join gateway.

Figure 49. The join gateway that incorrectly merges the decision gateways



Select the join and invoke **Transform > Toggle Gateways** again to toggle the join into a merge. Save the corrected model and run the control-flow analysis again. No error is reported!

Summary

This article introduces you to the patterns, transformations, and refactorings available in IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler. Instead of modeling in a traditional way by placing tasks and gateways on the drawing canvas and connecting them one by one, you learn how to apply patterns to create entire process fragments and to apply transformations and refactorings to your model that encapsulate complex editing operations in a single click. In addition to saving time, you can automatically detect and correct modeling errors in the control flow of a business process model.

Parts 2, 3, and 4 of this series provide a complete overview of the available patterns, transformations, and refactorings that are available in this release of the accelerators.

Downloads

Description	Name	Size
Videos for this article	PatternAcceleratorsMay2009.zip	100MB

Resources

- View this article as a video. If you can't see the video portion, install the Camtasia codec file from Techsmith.
  - Video 1: Using patterns to compose a process model.
  - Video 2: Applying fast changes to a process model using transformations and refactoring.
  - Video 3: Detecting and correcting a control flow error with the accelerators.
- Workflow Control-Flow Pattern Library: A revised View. BPM Center Report BPM-6-22 (N. Russell, A.H.M. ter Hofstede, W.M.P van der Aalst, and N. Mulyar, 2006). See also Workflow patterns. This article provides a complete overview of workflow patterns. Some of these patterns can be recognized in the business process patterns provided by this release of the accelerators.
- Tutorials and Samples for WebSphere Business Modeler Version V6.2: 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.
- T.Gschwind, J.Koehler, J.Wong: Applying Patterns during Business Process Modeling. Proceedings of the 6th International Conference on Business Process Management, Springer 2008. This article describes the scientific foundations for the capabilities described in this series.
- Modeling business processes in WebSphere Business Modeler for BPEL transformation, (R. Kong): This article describes the requirements for modeling processes as input into Business Process Execution Language (BPEL) generations.
- Process anti-patterns: How to avoid the common traps of business process modeling, Part 1, J. Koehler, J. Vanhatalo (developerWorks, 2007): This article describes typical control-flow modeling errors in process models and how to recognize them.
- Process anti-patterns: How to avoid the common traps of business process modeling, Part 2, J. Koehler, J. Vanhatalo (developerWorks, 2007): Part 2 of this series further describes data-flow modeling errors commonly found in process models.

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.