

Visualize with Rational Software Modeler: UML 2.0 modeling

A tour of Rational Software Modeler's Visual UML 2.0 modeling tools

Skill Level: Introductory

[Eric Long \(elong@us.ibm.com\)](mailto:elong@us.ibm.com)
Software Engineer
EMC

23 May 2006

For the architect, system analyst, or designer immersed in the development process, Rational Software Modeler offers a completely customizable, UML 2.0-based visual modeling and design tool that makes it simple to clearly document and communicate processes, flows, and designs. Teams find it easier to collaborate, since Rational Software Modeler integrates with other tools such as WebSphere Business Integration Modeler. Using easy-to-follow, step-by-step instructions, this tutorial lets you sample some of the visual UML 2.0 modeling capabilities. By the end, you'll know how to create a variety of different UML 2.0 diagrams, import existing modeling projects, and edit models.

Section 1. Before you start

About this tutorial

Learn how to create and import UML 2.0 modeling projects, models, and diagrams using Rational Software Modeler. This tutorial demonstrates several Rational Software Modeler UML 2.0 diagrams:

- Class diagrams
- Sequence diagrams
- Activity diagrams

If you are a business analyst, architect, developer, or someone who is interested in learning about Rational Software Modeler's UML 2.0 visual tooling capabilities, this tutorial is for you.

Objectives

After completing this tutorial, you will know how to use Rational Software Modeler's visual tools to create class, sequence, and activity diagrams.

Prerequisites

This tutorial assumes that you have some understanding of the Unified Modeling Language (2.0). Knowledge of UML 2.0 diagrams is helpful, but not required.

System requirements

To run the examples as demonstrated in this tutorial, you need to have Rational Software Modeler installed on your machine.

If you don't already have a copy, you can download a [free trial version of Rational Software Modeler](#). You also need to download the [sample payroll application](#). During the tutorial, you will import this file into Rational Software Architect.

Section 2. Models and designs supported by Rational Software Modeler

Rational Software Modeler lets you create, import, and edit many types of UML 2.0 models and the diagrams associated with those models. Here is a list of the "out-of-the-box" models and diagrams you can create:

- Use-Case Model

- Activity diagrams
 - State machine diagrams
 - Use-case diagrams
 - Analysis Model
 - Class diagrams
 - Sequence diagrams
 - Design Model
 - Class diagrams
 - Communication diagrams
 - Component diagrams
 - Composite structure diagrams
 - Deployment diagrams
 - Sequence diagrams
 - Enterprise IT Design Model
 - Corba Template Model
 - XSD Model
-

Section 3. Import a UML Modeling Project

To get started with this tutorial, you need to do some initial set up. First, import a project into Rational Software Modeler using the Project Interchange option.

Import the Payroll Application Project

Importing a project using Rational Software Modeler is very easy:

1. Open Rational Software Modeler.
2. From the Window menu, select **Open Perspective > Modeling** to open the Modeling perspective.


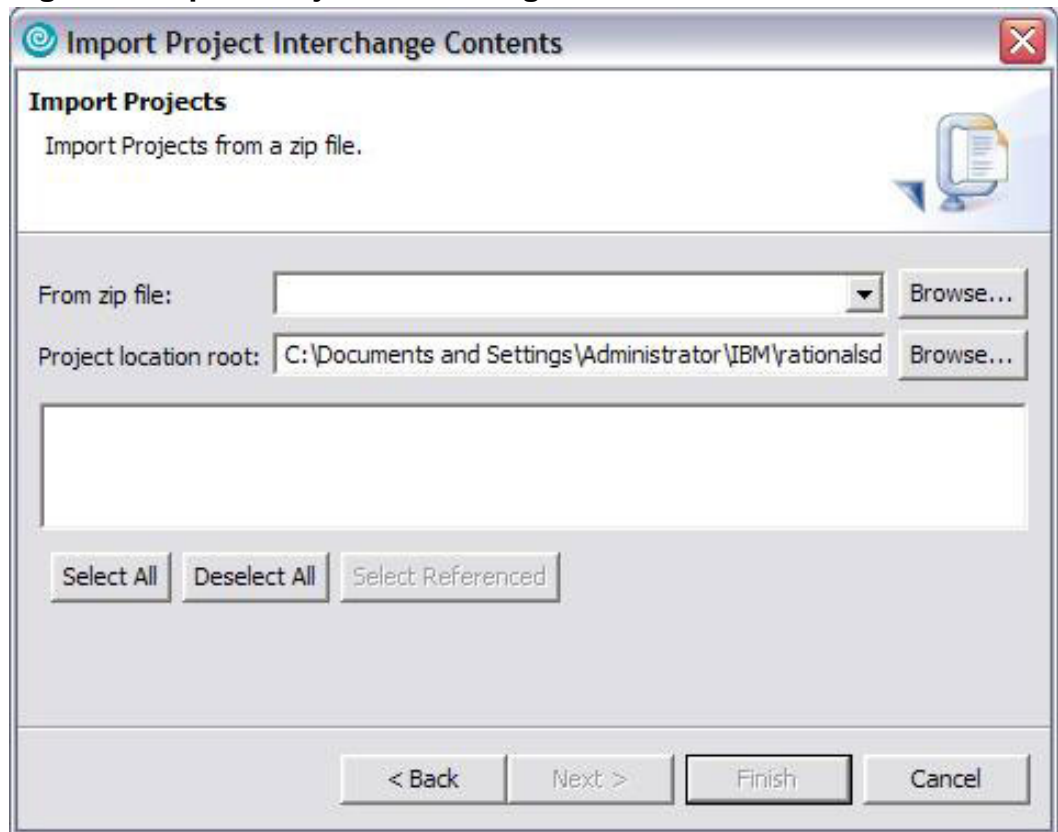
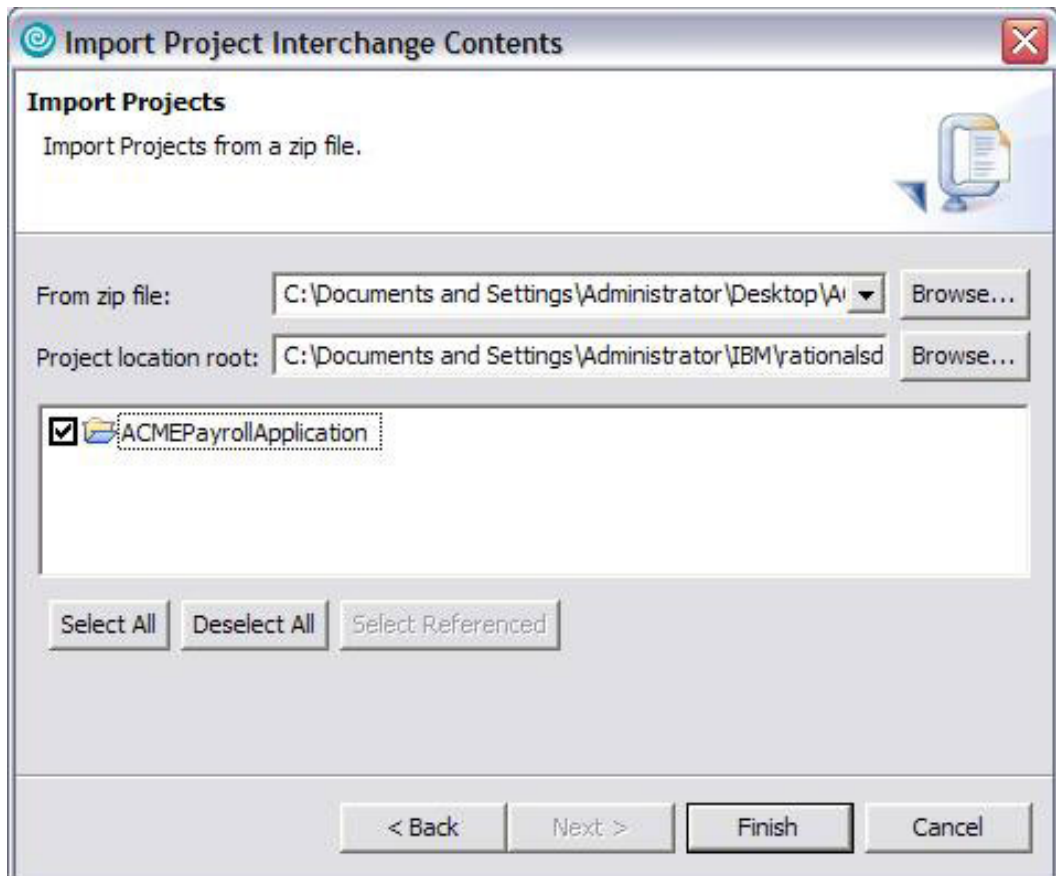
3. Make sure all of your open projects are closed.
4. Select **File > Import....**
5. Select **Project Interchange**  **Project Interchange** (or zip file).
6. Click **Next**.
7. Next to the From zip file: field, click **Browse...:**

Figure 1. Import Project Interchange Contents window

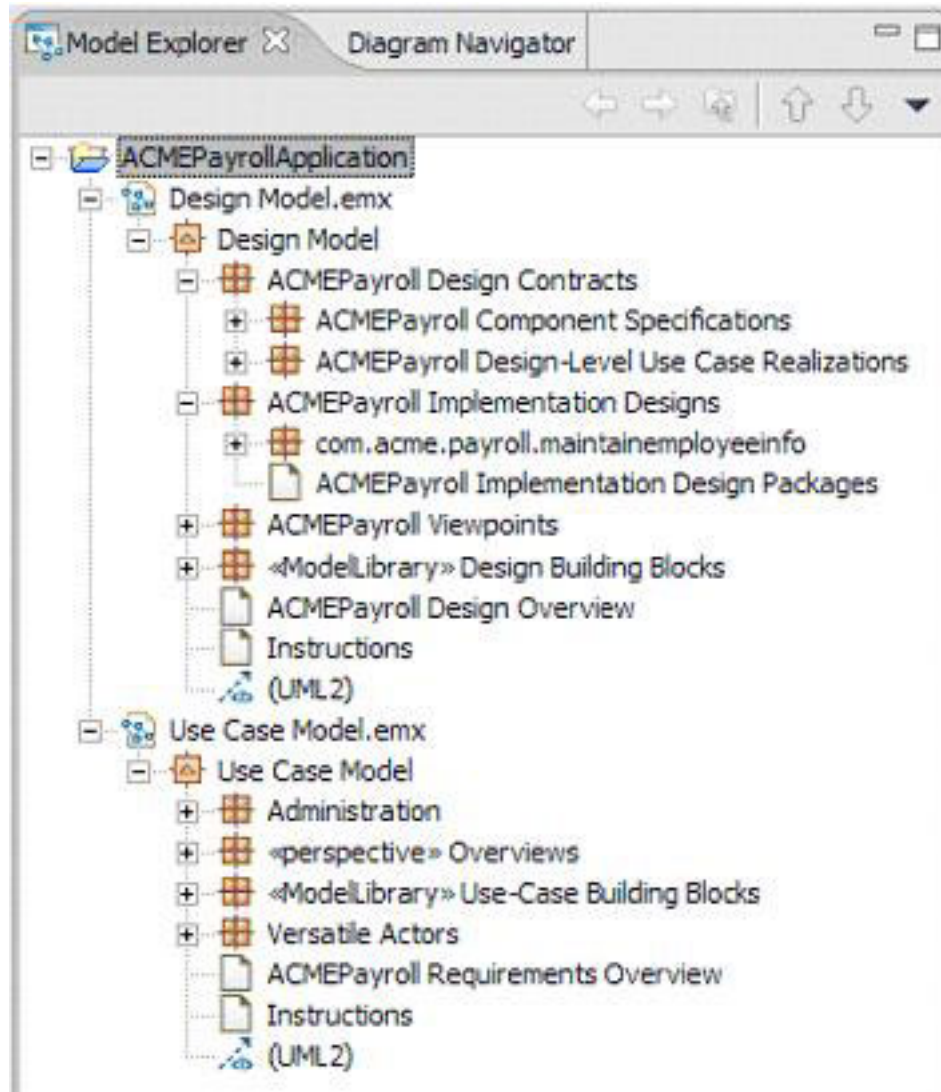


8. Find ACMEPayrollModel.zip (where you downloaded the file at the beginning of this tutorial).
9. Click **Open**.
10. Select **ACMEPayrollApplication** in the Import Projects list:

Figure 2. Import Projects window



11. Click **Finish**.
 12. You should see the ACMEPayrollApplication project in the Model Explorer view. Double-click on both of the `.emx` files and explore their contents.
- Figure 3. Model Explorer view**



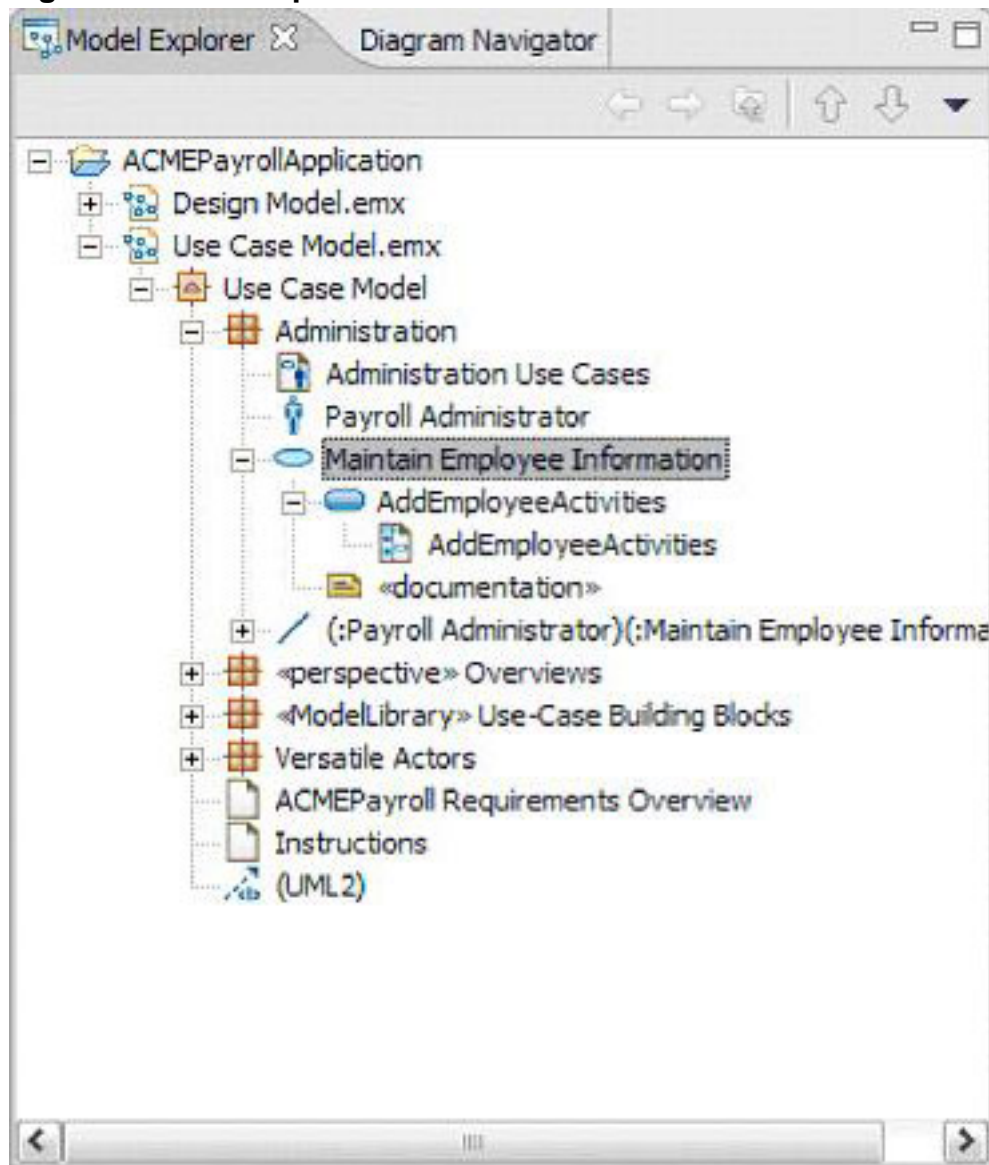
Now on to the more exciting part of the tutorial. Let's create UML class diagrams.

Section 4. Create and edit a UML activity diagram

In this section of the tutorial, you will create an activity diagram in the ACMEPayrollApplication project to illustrate the flow in a use-case.

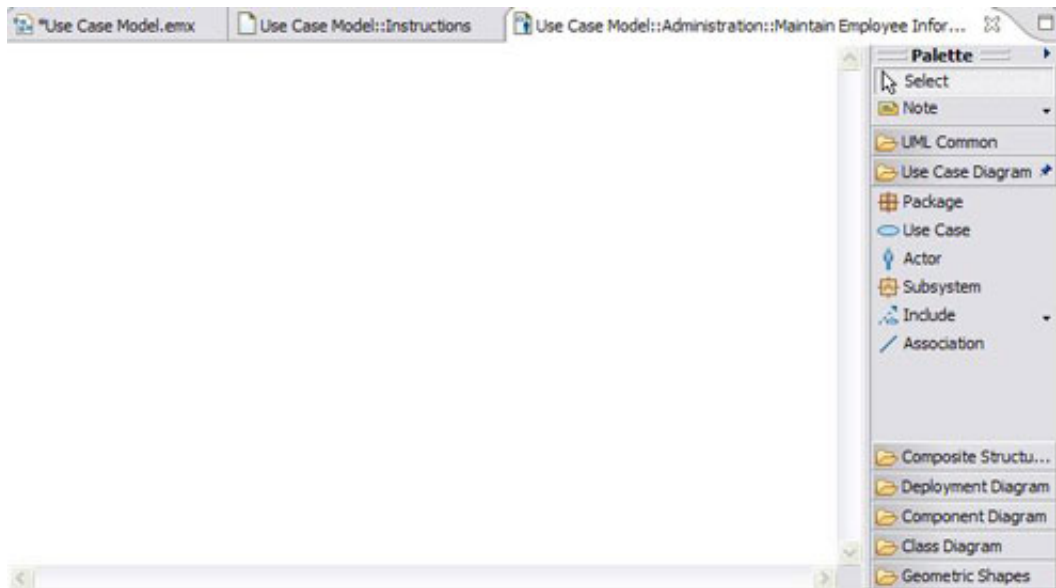
1. In the Model Explorer view, expand the Administration package in the Use Case Model.


2. Right-click the Maintain Employee Information use-case.
3. Select **Add Diagram > Activity Diagram**.
4. Name the activity and the diagram `AddEmployeeActivities`:

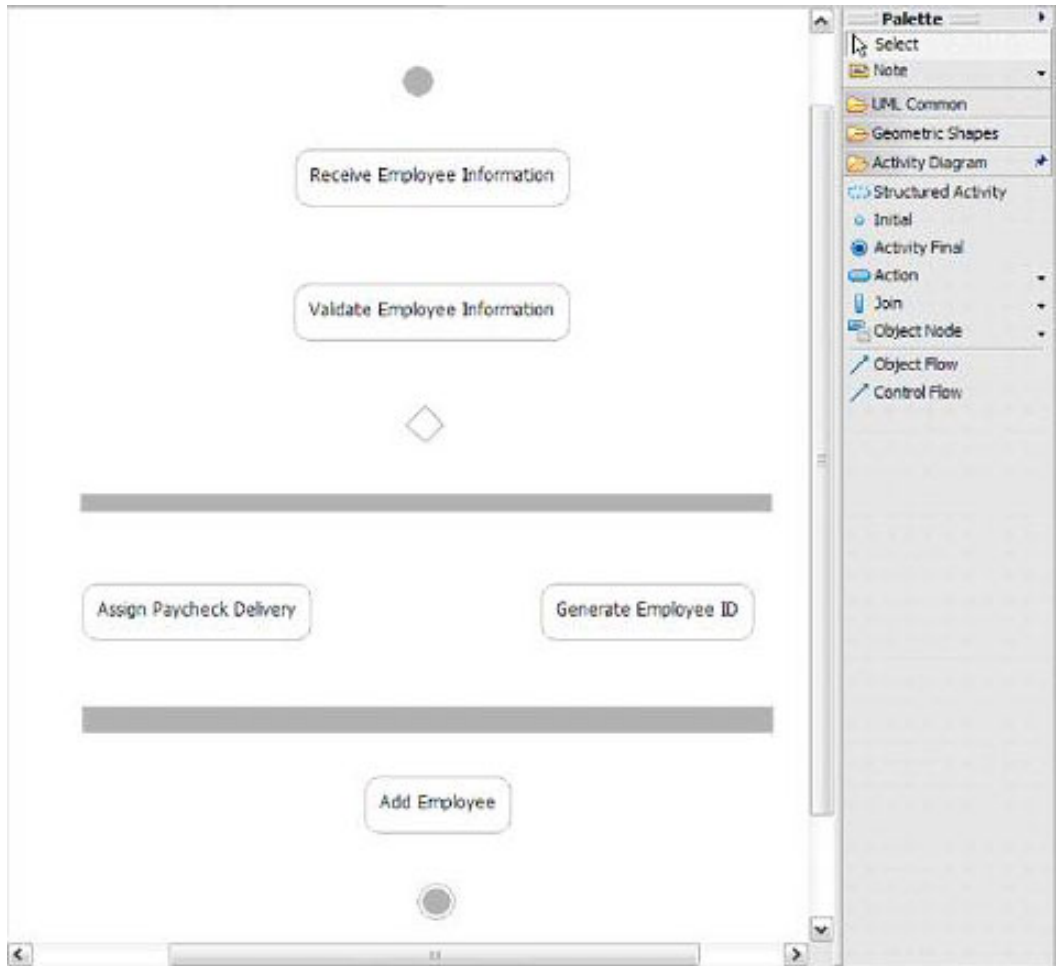


5. The activity diagram opens in the Diagram editor.

Figure 5. Activity diagram 1

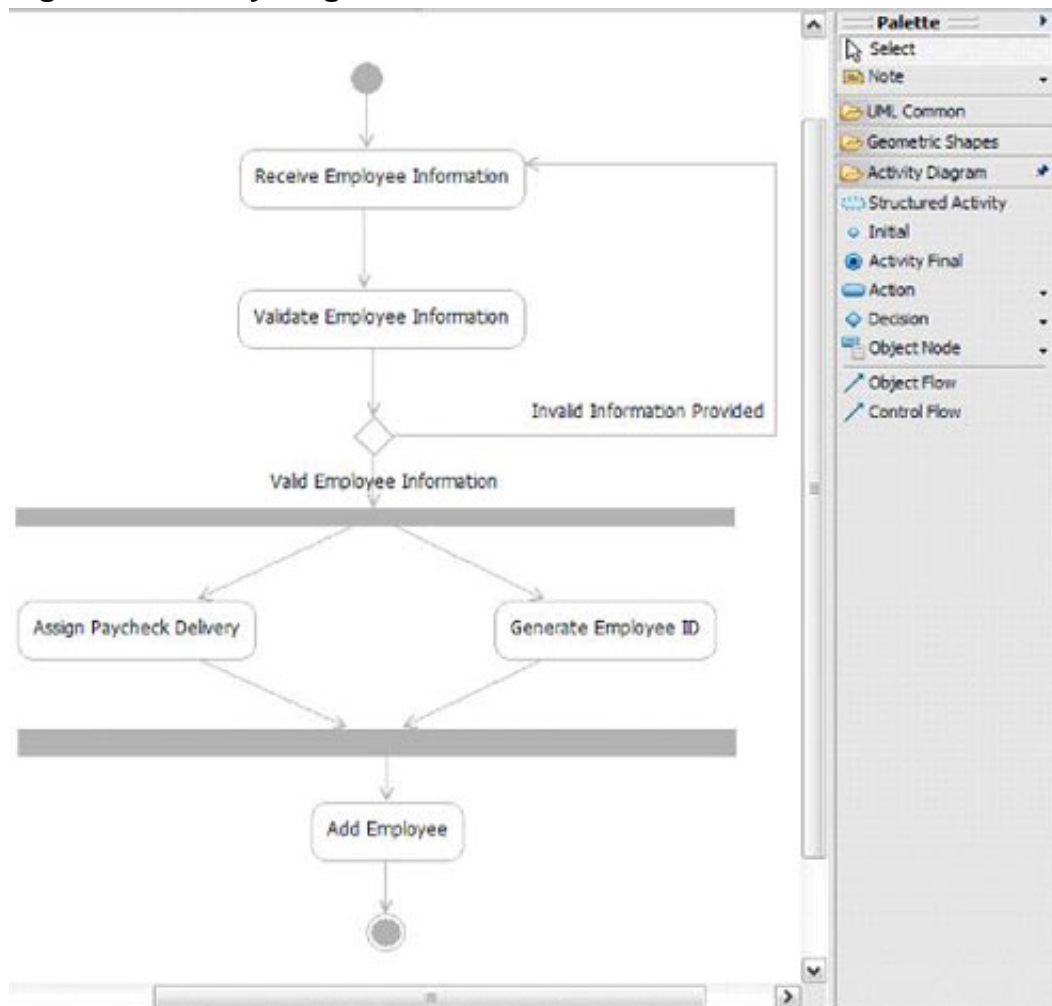


6. From the Palette  (right of the editor), add the following nodes to the diagram:
 1. An Initial node with no name.
 2. An Action node named `Receive Employee Information`.
 3. An Action node named `Validate Employee Information`.
 4. A Decision node (which is a type of Control node) with no name.
 5. A Fork node (size appropriately) with no name.
 6. An Action node named `Assign Paycheck Delivery`.
 7. An Action node named `Generate Employee ID`.
 8. A Join node (size appropriately) with no name.
 9. An Action node named `Add Employee`.
 10. An Activity Final node with no name.
7. Arrange the nodes to look similar to this:
Figure 6. Activity diagram 2



8. From the Palette, add the following Control Flow relationships (only provide names where indicated):
 1. From the Initial node to the Receive Employee Information action.
 2. From the Receive Employee Information action to the Validate Employee Information action.
 3. From the Validate Employee Information action to the Decision node.
 4. From the Decision node to the Receive Employee Information action named `Invalid Information Provided` (you might need to adjust the line to route around other actions).
 5. From the Decision node to the Fork node named `Valid Employee Information`.

6. From the Fork node to the Assign Paycheck Delivery action.
 7. From the Fork node to the Generate Employee ID action.
 8. From the Assign Paycheck Delivery action to the Join node.
 9. From the Generate Employee ID action to the Join node.
 10. From the Join node to the Add Employee action.
 11. From the Add Employee action to the Activity Final node.
9. Your diagram should look similar to this:
Figure 7. Activity diagram 3



Great! Now, on to creating a sequence diagram.

Section 5. Create and edit a sequence diagram

Create a sequence diagram for the RemoveDeletedEmployees flow you created in the Maintain Employee Information use-case realization in part two of this tutorial.


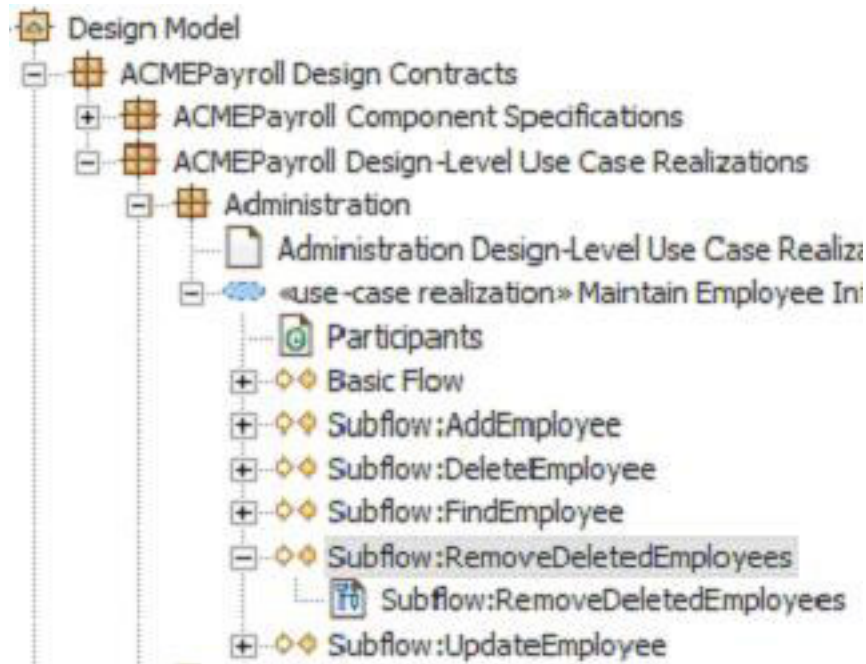
1. In the Model Explorer  view, expand **Design Model** > **ACMEPayroll Design Contracts** > **ACMEPayroll Design-Level Use Case Realizations** > **Administration**.
2. Right-click **[[use-case realization]] Maintain Employee Information**.
3. Select **Add Diagram** > **Sequence Diagram**.
4. Name both the interaction and the diagram
Subflow:RemoveDeletedEmployees:

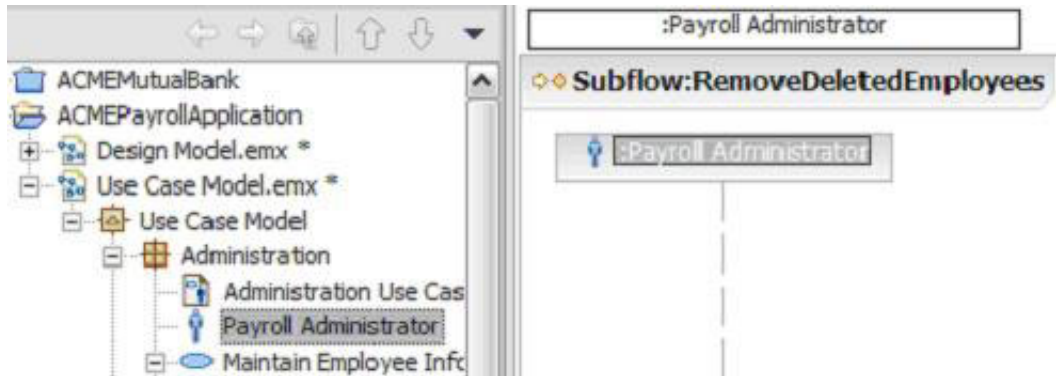
Figure 8. Model Explorer view




5. In the Model Explorer view, expand **Use Case Model.emx** > **Administration** to find the Payroll Administrator actor.
6. Drag-and-drop the Payroll Administrator actor on to the sequence

diagram and press **F2**, then **Delete**, then **Enter** (this removes the auto-assigned name of *payrolladministrator*).

Figure 9. Add PayrollAdministrator



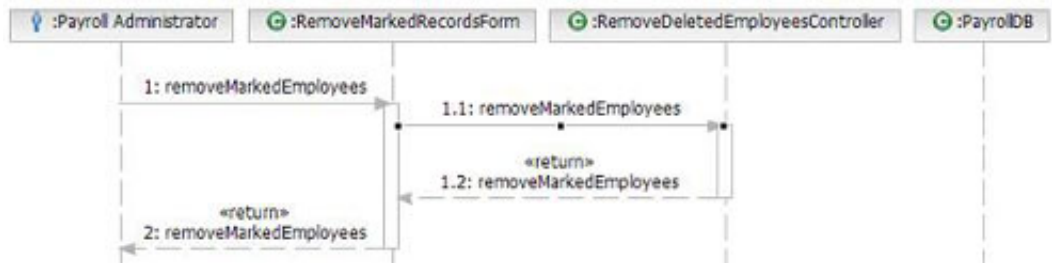
7. From the Palette, select the **Sequence Diagram** drawer . 
8. Select the **Lifeline** tool and click on the diagram.
9. Select **Create New Class**.
10. Name the class `RemoveMarkedRecordsForm` and click **OK**.
11. If prompted for a name for the element, press **Delete** and then **Enter**.
12. Repeat the same steps to add two more unnamed classes to the diagram:
 - `RemoveDeletedEmployeesController`
 - `PayrollIDB`
13. You should now see:



14. From the Palette, select the Sequence Diagram drawer.
15. Select the Synchronous Message tool.
16. Click-and-hold on the **Payroll Administrator** lifeline and drag it to the `RemoveMarkedRecordsForm` lifeline, then release.
17. Name the operation `removeMarkedEmployees` and click **OK**.

18. Repeat these steps to create the `removeMarkedEmployeess` message from the `RemoveMarkedRecordsForm` lifeline to the `RemoveDeletedEmployeesController` lifeline.
19. Your sequence diagram should look like this:

Figure 11. Sequence diagram 2



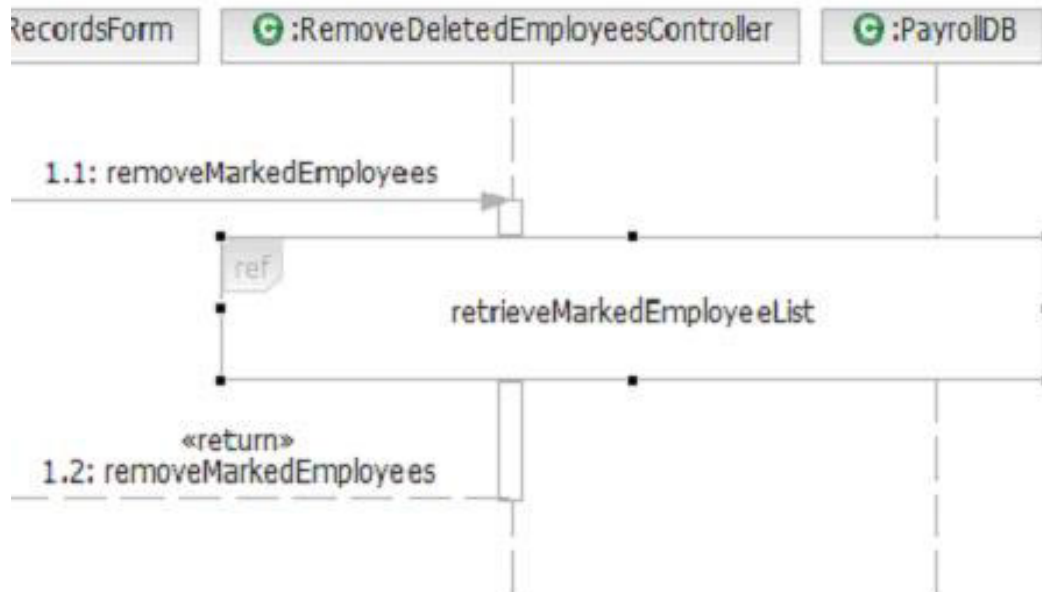
20. From the Palette, click **Interaction Occurrence**.
21. Click on the execution occurrence on the `RemoveDeletedEmployeesController` lifeline.
22. Select **Create New Interaction** and name it `retrieveMarkedEmployeeList`.
23. Resize the interaction occurrence box to span the `PayrollDB` lifeline.
24. When prompted, select **:PayrollDB** in the Add Covered Lifelines window, then click **OK**:

Figure 12. Add Covered Lifelines window



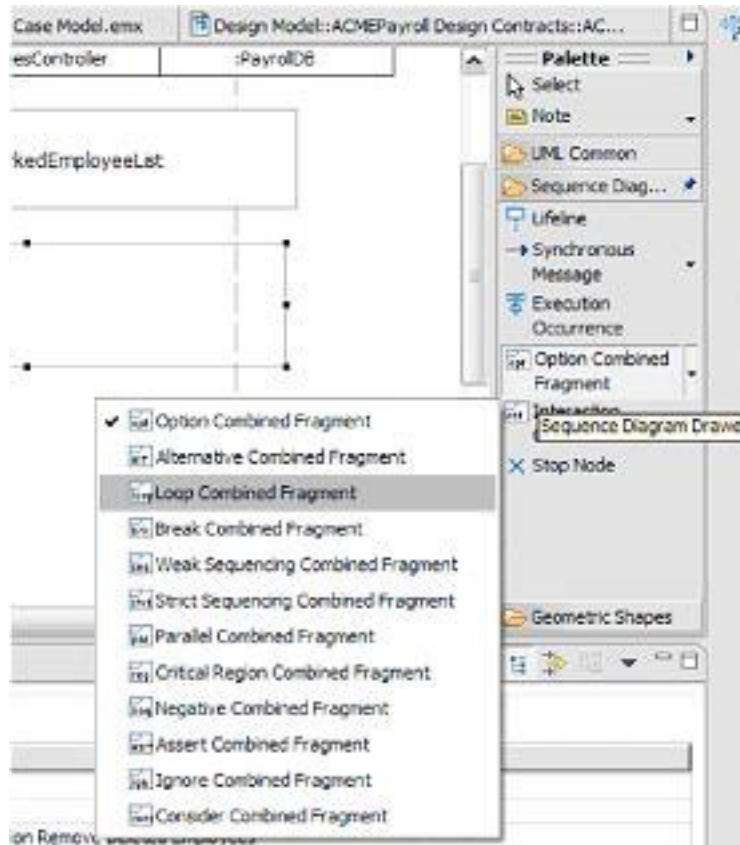
25. The retrieveMarkedEmployeeList interaction occurrence should look like this:

Figure 13. Sequence diagram 3



26. From the Palette, select **Option Combined Fragment**.
27. Click on the execution occurrence on the RemoveDeletedEmployeesController lifeline (just below the interaction occurrence you just created).
28. Type `hasMoreEmployees` as the guard condition.
29. Resize the interaction occurrence box to span the PayrollDB lifeline.
30. Again, when prompted, select **:PayrollDB** in the Add Covered Lifelines window, then click **OK**.
31. From the Palette, click the arrow beside the Option Combined Fragment tool and select **Loop Combined Fragment**:

Figure 14. Loop Combined Fragment



32. Click on the execution occurrence for RemoveDeletedEmployeeController within the area of the Option Combined Fragment box you previously placed.
33. Click **Enter** to accept the default parameters of **0,***.
34. Resize the interaction occurrence box to span the PayrollIDB lifeline.
35. Again, when prompted, select **:PayrollIDB** in the Add Covered Lifelines window, then click **OK**.
36. Within the Loop Combined Fragment, create the `deleteEmployee` synchronous message from the RemoveDeletedEmployeesController lifeline to the PayrollIDB lifeline.
37. Whew! Well, after all of that, your sequence diagram should look like this:
Figure 15. Sequence Diagram 5



On to UML class diagrams.

Section 6. Populate and edit a UML class diagram

So, you created a few different classes in the last section. This section is going to show how easy it is to take those same classes and view them in a class diagram.

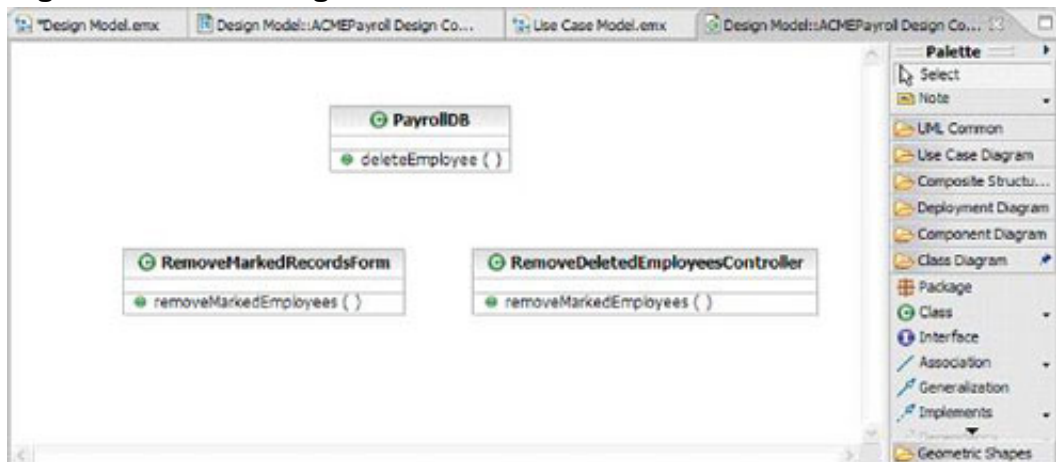
Populate a Class Diagram

1. In the Model Explorer view, expand **Design Model > ACMEPayroll Design Contracts > ACMEPayroll Design-Level Use Case Realizations > Administration > [[use-case realization]] Maintain Employee Information.**
Figure 16. Model Explorer view



2. Double-click the Participants diagram.
3. From the Model Explorer view, drag-and-drop the following classes onto the diagram:
 - PayrollDB
 - RemoveMarkedRecordsForm
 - RemoveDeletedEmployeesController
4. Your diagram should look like this:

Figure 17. Class Diagram



That's it! Feel free to add relationships, elements, and other components to the class diagram using the Palette.

Section 7. Conclusion

Congratulations! In this tutorial, you imported a UML Modeling Project and created a UML 2.0 activity diagram, sequence diagram, and class diagram. Even though the examples you created were fairly simple, you could see how easy it is to use Rational Software Modeler to create meaningful UML 2.0 models and diagrams.

Downloads

Description	Name	Size	Download method
Sample payroll application	ACMEPayrollModel.zip	20KB	HTTP

[Information about download methods](#)

Resources

Learn

- Visit the [developerWorks Rational zone](#) to expand your Rational skills.
- [Visualize with Rational Application Developer](#) (developerWorks, February 2006) details the visual tools available to visualize a number of different Java elements in Rational Application Developer.
- [Visualize with Rational Software Architect](#) (developerWorks, March 2006) details the visual tools available to create UML projects and models, apply design patterns to those models (new or existing), and transform UML models into source code or into a different type of model.
- [Discover IBM Rational visual tools for application development](#) (developerWorks, February 2006) details the visual tools available to visualize data elements in Rational Application Developer.
- Stay current with [developerWorks technical events and webcasts](#).

Get products and technologies

- Download a free trial version of [Rational Software Modeler](#).
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

About the author

Eric Long



Eric Long is a software engineer in the IBM Developer Skills Program. Eric graduated from the University of Texas with a degree in Computer Science, and currently works in Austin, Texas. He provides technical information to developers on emerging open source and industry trends and technologies through world-wide technical briefings, speaking engagements, workshops, Web content, and faculty consultations at IBM Academic Initiative member universities. His work also includes technical demos and content available at ibm.com/university and ibm.com/developerWorks.