Hello World: Rational Software Architect

Design a simple phone book application

Skill Level: Introductory

Tinny Ng (tng@ca.ibm.com)
Advisory Software Developer
IBM Toronto

05 May 2006

Welcome to the first tutorial in the "Hello, World! Series", which will provide high-level overviews of various IBM software products. This tutorial introduces you to IBM Rational Software Architect, and highlights some basic features of Rational Software Architect with a hands-on exercise. Learn how to design an application using UML diagrams, publish the model information into a Web page, and transform the design to Java code using Rational Software Architect.

Section 1. Before you start

About this series

This series is for novices who want high-level overviews of various IBM software products. The modules are designed to introduce the products, and draw your interest for further exploration. The exercises only cover the basic concepts, but are enough to get you started.

About this tutorial

This tutorial provides a high level overview of Rational® Software Architect, and shows some basic features of Rational Software Architect with a hands-on exercise. The exercise has step-by-step instructions for designing an application using UML diagrams, publishing the model information into a Web page, and transforming the design to Java™ using Rational Software Architect.

Prerequisites

This tutorial is for application designers at a beginning level, but you should have a general familiarity with using the Eclipse development environment.

To run the examples in this tutorial, you need to install IBM Rational Software Architect v6.0.

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

Animated demos

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

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

Section 2. Introduction

IBM Rational Software Architect is an integrated design and development tool that unifies architecture, design, and development within one tool. It includes the full functions of:

- Rational Application Developer - a development tool that lets you do J2EE development, XML, Web Service development and more
- Rational Software Modeler - a modeling tool that lets you visually model systems and applications using Unified Modeling Language (UML) notation

Rational Software Architect unifies them and is built on top of the open and extensible Eclipse platform, which leverages several industry standards.

Rational Software Architect is mainly used by software architects and senior developers within a development team for specifying and maintaining all aspects of an application's software architecture. Its UML 2.0 support lets users capture and communicate all aspects of an application architecture using a standard notation. Users can use patterns and transformations to define and implement their applications.

Rational Software Architect supports an extensive list of features which uniquely differentiate itself from its competitors. The following table highlights some of the key differentiators. The first three are illustrated in the following exercise. The exercise steps you through designing an application using UML diagrams, publishing the model information into a Web page, and transforming the design to Java using Rational Software Architect. To further advance your skills, refer to the Resources for more information.

Table 1. Some key features of Rational Software Architect

Feature	Benefit
UML 2.0 modeling support for analysis and design using Use Case, Class, Sequence, Activity, Composite Structure, State Machine, Communication, Component, and Deployment diagrams.	UML 2.0 allows you to capture and communicate all aspects of an application architecture using a standard notation that is recognized by many different stakeholders.
Generate HTML, PDF, and XML reports from UML designs.	Create reports and documentation that can be reviewed by team members or other stakeholders.
Uses transformations to generate Java code, C++, or Enterprise JavaBeans code	Automate the repeatable task of generating code from design models. Transformations can be customized to tailor code generation patterns to an organization's needs.
UML Class diagram editing for Java code, EJB code, and Database objects.	Uses UML notation to provide abstract views of Java code, EJB code, and database objects to simplify the development and understanding of new and existing applications.
Java method body visualization using UML 2.0 Sequence diagrams.	Use UML 2.0 sequence diagram constructs to understand the flow of a Java method.
WS-I compliant Web services and service oriented architectures.	Integrates your business applications.
Apply and author patterns and transforms.	Allows organizations to capture and promote "recipes" that can be used to increase the predictability

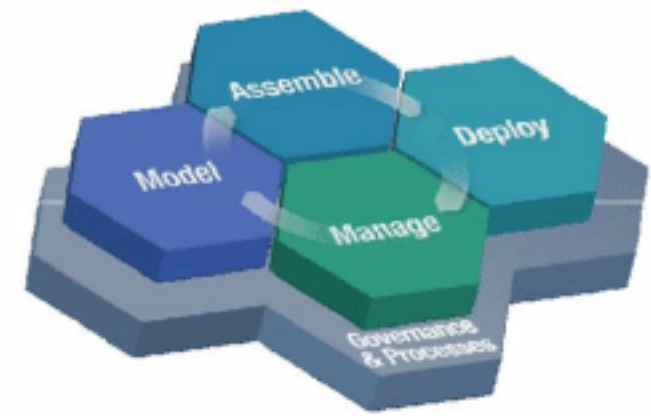
	and repeatability of software development. The authoring and apply capabilities support teams in "developing for reuse" and "developing with reuse".
Asset Browser for accessing reusable assets.	Supports OMG Reusable Asset Specification and supports users in browsing repositories containing reusable assets. Repositories can be structured so that assets can be found easily.
Open API to support customizing and extending the modeling environment. UML profile creation and editing to customize the properties stored in UML models.	Organizations can develop plug-ins customize the analysis and design tools for their environment and process. Supports the creation of an ecosystem allowing vendors to develop integrations.
RUP configuration for Software Architects with context-sensitive and dynamic process guidance.	Process guidance and user assistance is provided dynamically as the user works with the tool.

Section 3. How does Rational Software Architect fit into SOA?

Service Oriented Architecture (SOA) is an architectural style for building distributed systems that deliver application functionality as services to be used by end-user applications or for building other services. It enables customers to create sophisticated applications and solutions swiftly and easily by assembling from new and existing services. Each business function in a company can be implemented as a service which can then be integrated with other services to fulfill the company's business requirements. Companies in every industry are seeking ways to respond more quickly and effectively to changing market conditions. To achieve this level of business flexibility, many companies are implementing SOA by developing service-oriented applications

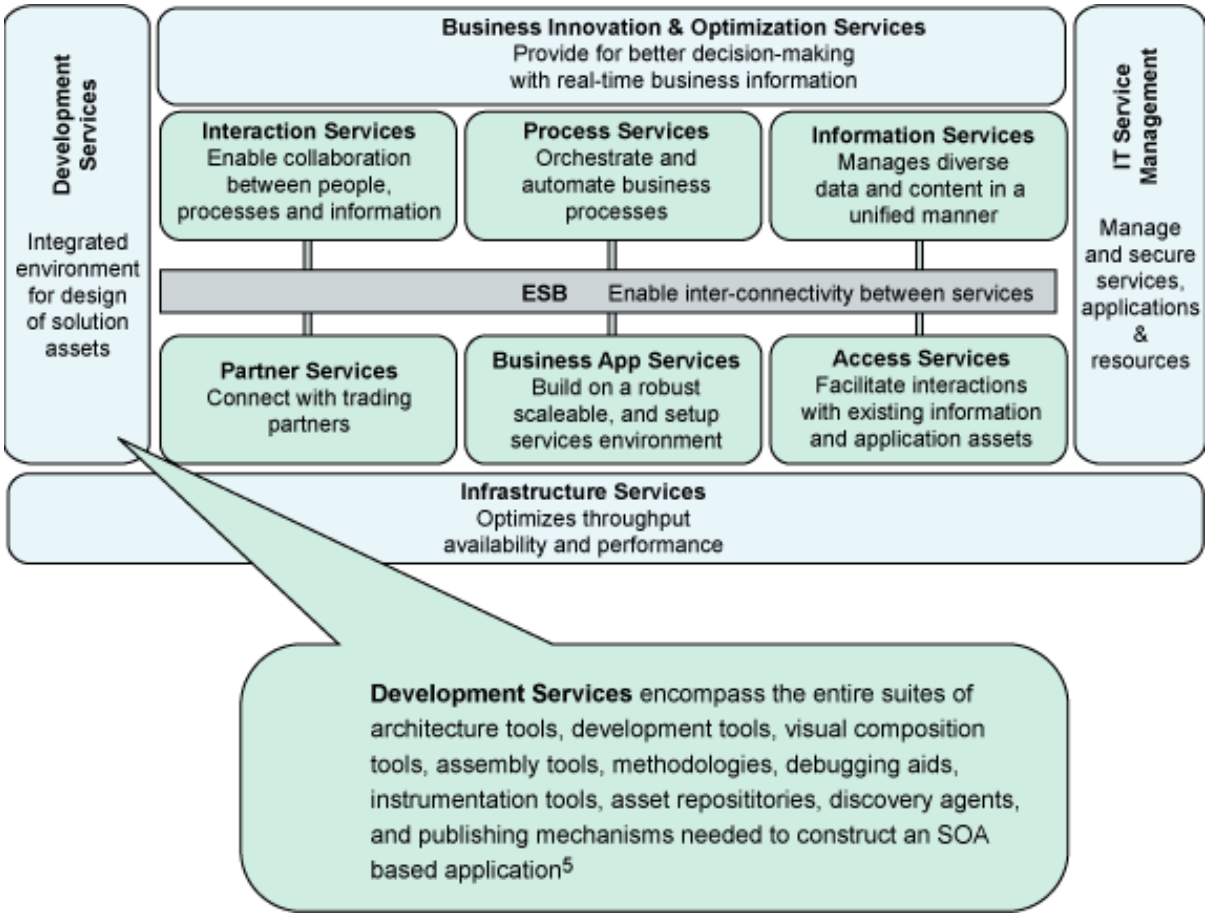
Getting started with SOA is easy with the IBM SOA Foundation - an integrated, open-standards-based set of software, best practices and patterns for Service Oriented Architecture. The software that comprises IBM SOA Foundation supports each stage of the SOA life cycle which includes four stages: model, assemble, deploy and manage. Underpinning all of these life-cycle stages are governance and processes that provide guidance and oversight for the SOA project.

Figure 1. SOA Life Cycle



The IBM Rational Software Architect, as a robust and sophisticated design and development tool, is part of the IBM SOA Foundation and supports the model phase of the SOA life cycle. It is part of the development services of the SOA Reference Architecture, and gives companies the tools they need for modeling service-oriented applications.

Figure 2. SOA Reference Architecture



Rational Software Architect allows software architects to visually model and design flexible services architecture using the open standard Unified Modeling Language (UML), and automatically apply design patterns for SOA from analysis and design to implementation. Besides, there are a number of new SOA design resources available as Rational Software Architect plug-ins to further assist users with solution

design in a service-oriented world. Please refer to the Resources for more information.

Section 4. What is UML?

UML was first released by the Object Management Group (OMG) in 1997, and is currently being upgraded to version 2.0. UML is designed to bring together the development community with a stable and common design language that could be used to develop and build applications. It is a modeling language, and is programming-language independent.

UML provides various diagrams that let users capture and communicate all aspects of an application architecture using a standard notation that is recognized by many different stakeholders. There are 13 official UML 2.0 diagrams, each of which is a different view that shows different aspects of the system:

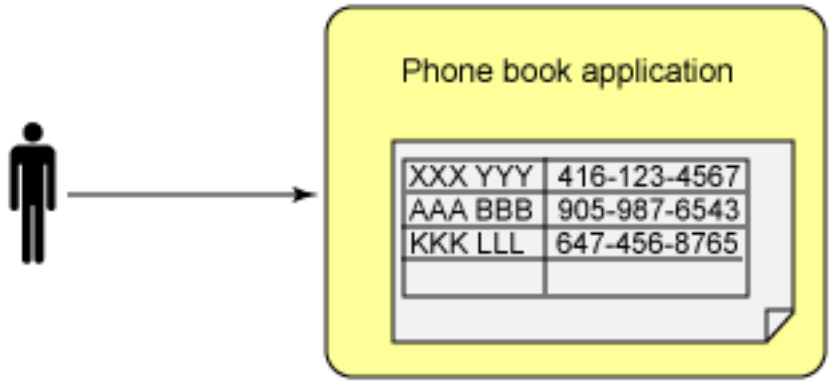
- Activity diagram
- Class diagram
- Communication diagram
- Component diagram
- Composite structure diagram
- Deployment diagram
- Interaction overview diagram
- Object diagram
- Package diagram
- Sequence diagram
- State machine diagram
- Timing diagram
- Use case diagram

In the following exercise learn how to create a use case diagram, a class diagram, and a sequence diagram, and then publish and transform the design using Rational Software Architect.

Section 5. Designing a phone book application

In this exercise you'll design a very simple phone book application, as shown in Figure 3, that stores user entered phone numbers to be retrieved later.

Figure 3. Phone book application



To begin:

1. Start Rational Software Architect if it's not already started: From Windows select **Start** > **Programs** > **IBM Rational** > **IBM Rational Software Architect v6.0** > **Rational Software Architect**.
2. A window appears asking for the workspace directory. Select **OK** to accept the default.

Would you like to see these steps demonstrated for you?

Show me

Creating a UML project

First create a UML project named MyPhoneBookUMLProject:

1. From the workbench, select **File** > **New** > **Project > Other**.
2. Select **UML Project** and then select **Next**.
3. Enter `MyPhoneBookUMLProject` as the project name, and select **Next**.
4. Enter `Phone Book UML Model` as the file name of the UML Model, uncheck the box **Create a default diagram in the new model**, then select **Finish**.

Section 6. Creating a use case diagram

A use case diagram models the behavior of a system and helps to capture the requirements. It identifies the interactions between the system and its actors, and defines the scope of the system.

Actor

Represents a role of a user that interacts with the system. The user can be a human user, an organization, a machine, or another external system.

Use case

Describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system.

The use cases and actors shown in a use case diagram describe what the system does and how the actors use it, but not how the system operates internally. To relate an actor and a use case, you can create an *association relationship* to indicate the connection between the two model elements.

For our simple phone book application, assume there is only one actor, **Any User**, who can carry out the following two use cases against the system:

Add an entry

Enter a unique person name and a phone number using the provided application user interface. The system processes the entered data and stores it.

Search for a phone number

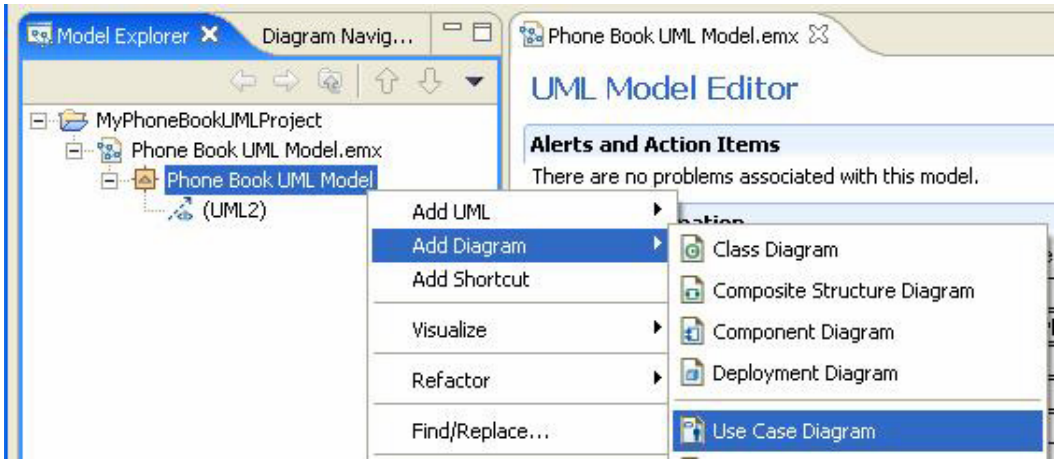
Retrieve a phone number by entering a unique person name using the provided application user interface. The system locates the phone number and returns it to the actor.

Would you like to see these steps demonstrated for you?

Show me

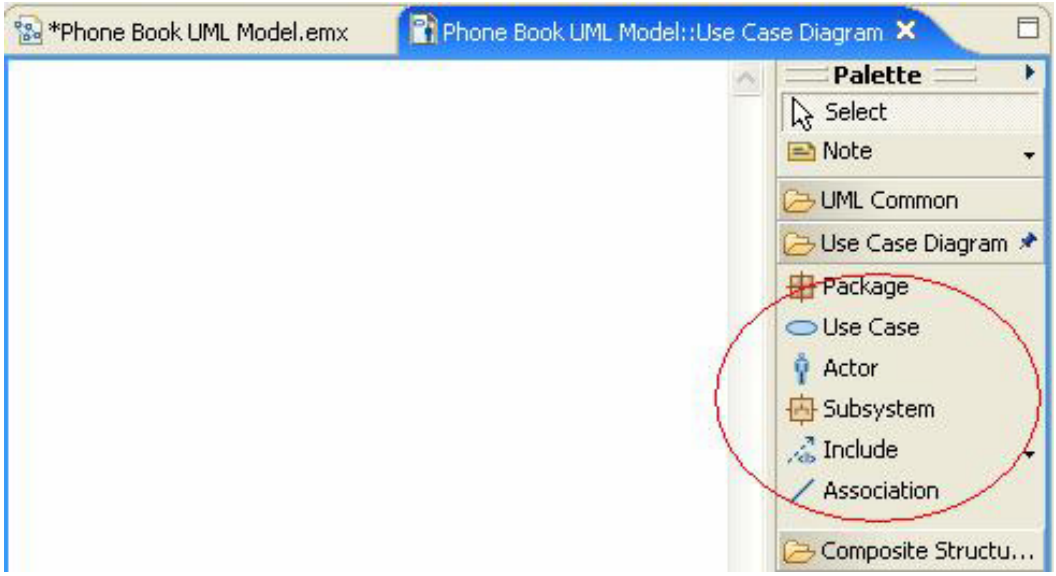
To create a use case diagram that lists the two use cases:

1. In the Model Explorer view, right-click **Phone Book UML Model** and select **Add Diagram** > **Use Case Diagram**.
   Figure 4. Adding a use case diagram



2. Enter `User Case Diagram` as the name of the generated diagram to replace the default name Diagram1. Now you can draw the use case diagram by adding various model elements from the Palette to the diagram.

Figure 5. Adding model elements



3. Select **Actor** in the Palette, then click anywhere in the diagram to create an Actor. Name it `Any User`.

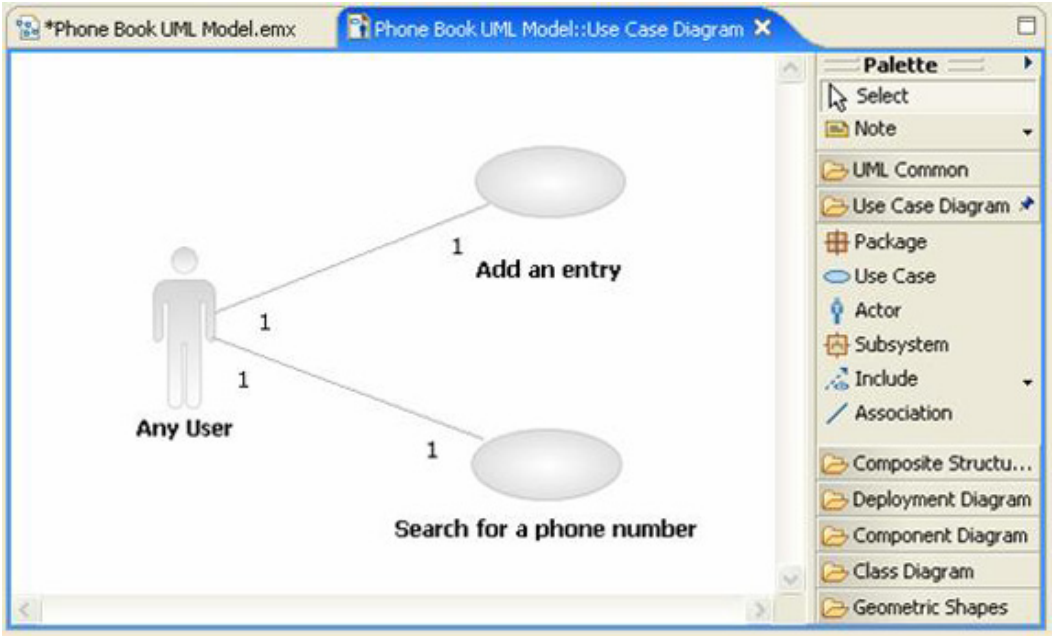
4. Select **Use Case** in the Palette, then click anywhere in the diagram to create a Use Case. Name it `Add an entry`.

5. Similarly, create another use case called **Search for a phone number**.

6. Select **Association** in the Palette. Draw the association relationship line from the actor **Any User** to the use case **Add an entry** to initiate a relationship between the two model elements.

7. Similarly, create another association relationship between the actor **Any User** and the use case **Search for a phone number**.

8. The complete use case diagram should look similar to Figure 6. Select **Ctrl-S** to save the diagram.
Figure 6. Complete use case diagram



Section 7. Creating a class diagram

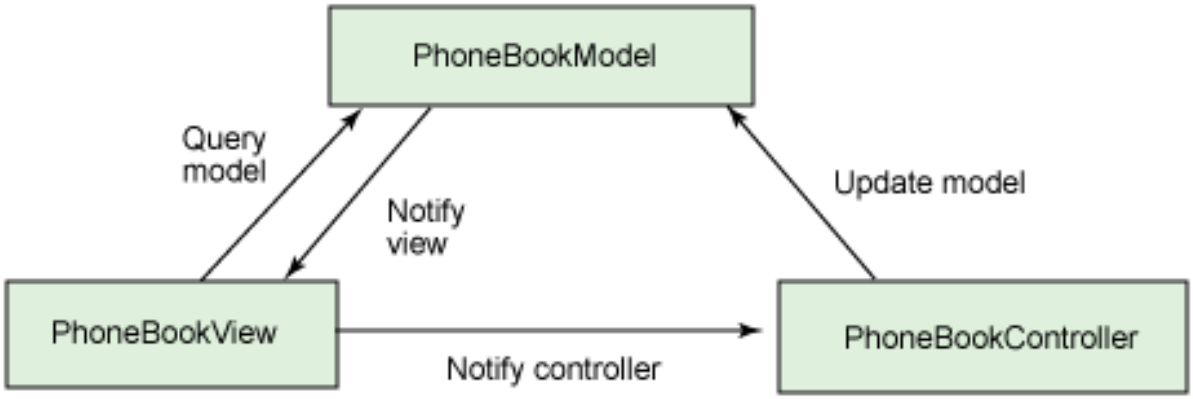
Class diagrams are the blueprints of your system. Use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services they provide.

Design the simple phone book application using the Model-View-Controller (MVC) architecture pattern, as shown in Figure 7. (See Resources for information on MVC.) The following three classes will be created:

- `PhoneBookModel` - A class that manages the phone book entries and captures the state of the application. Whenever the state is changed, it notifies `PhoneBookView`, which should then refresh the user interface based on the state of the application.
- `PhoneBookView` - A class that manages the graphical or textual interface to the user based on the state of the application, and notifies `PhoneBookController` when an input is received.
- `PhoneBookController` - A class that controls the operation of the entire application. It changes the model state of the application and updates the data model based on user input.

The purpose of this exercise is to show the use of Rational Software Architect when designing an application. The design itself is not the focus, and you can proceed with a different design.

Figure 7. MVC design



Would you like to see these steps demonstrated for you?

Show me

Let's create a class diagram that reflects the design in Figure 7.

1. In the Model Explorer view, right-click **Phone Book UML Model** and select **Add Diagram** > **Class Diagram**.
2. Enter `Class Diagram` as the name of the generated diagram to replace the default name Diagram1.
3. Select **Class** in the Palette, then click anywhere in the diagram to create a class. Name it `PhoneBookModel`.
4. Right click the created class **PhoneBookModel** and select **Add UML** > **Operation** to create an operation for this class. Name it `setState`.
5. Similarly, create the rest of the elements as shown in Table 2. The operation names are chosen without any programming language assumptions.
   Table 2. Class information

Class	Operation(s)
PhoneBookModel	addAnEntry searchPhoneNumber getSearchResult getState
PhoneBookView	stateHasChanged changeView getUserInput
PhoneBookController	userHasInput

start

6. Now create some associations to relate these three classes together. As shown in Figure 8, click the arrow that appears next to Association] from the Palette and select **Directed Association**.
Figure 8. Directed association relationship

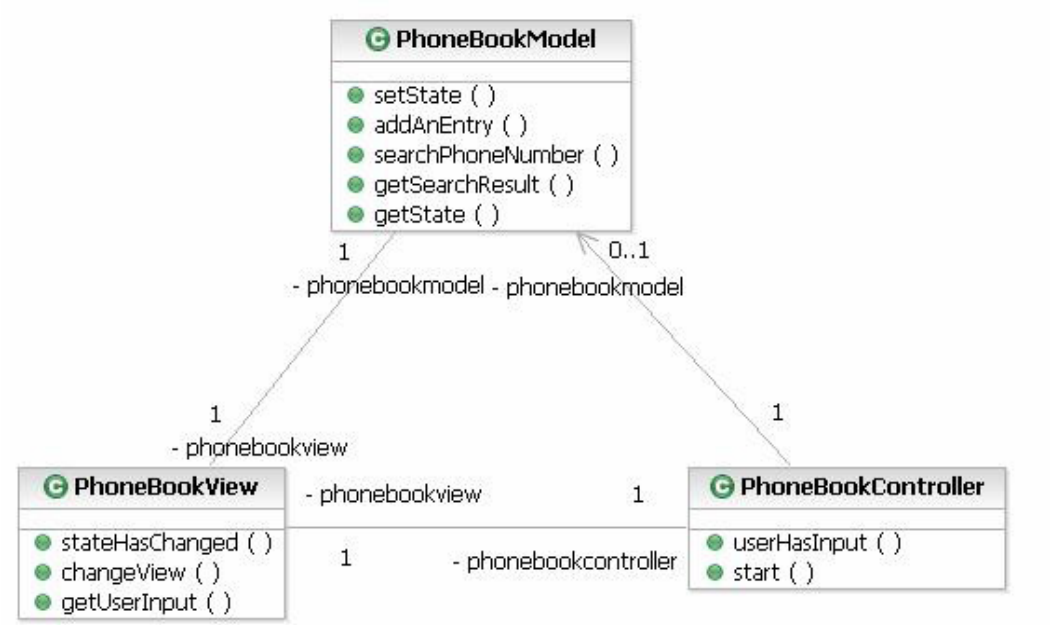
7. Draw the directed association relationship line from the class **PhoneBookController** to **PhoneBookModel** (the order is important) to initiate a **Directed Association** relationship between these two classes. A Directed Association relationship means the first one is aware of the latter, but not the other way round.

8. Similarly, create the following relationships:

   - Create an Association relationship between the class **PhoneBookModel** and **PhoneBookView**.
   - Create an Association relationship between the class **PhoneBookView** and **PhoneBookController**.

   An Association relationship without any direction means the two connected classes are aware of each other.

9. The complete class diagram should look similar to Figure 9. Select **Ctrl-S** to save the diagram.
Figure 9. Complete class diagram



Section 8. Creating a sequence diagram

A sequence diagram in UML shows the chronological sequence of messages between instances in an interaction. It consists of an interaction that is represented by lifelines and the messages that they exchange over time during the interaction.

In this section you're going to realize the use case "Search for a phone number" and show the associated interaction using a sequence diagram. The use case starts with the actor **Any User**, who makes use of the interface provided by `PhoneBookView` to request a search. `PhoneBookView` notifies `PhoneBookController` about the user's request. `PhoneBookController` then updates the data model that is stored in `PhoneBookModel`. Because there is a change of the model, `PhoneBookModel` notifies `PhoneBookView`, which should then refresh the user interface to reflect the latest state of the application.

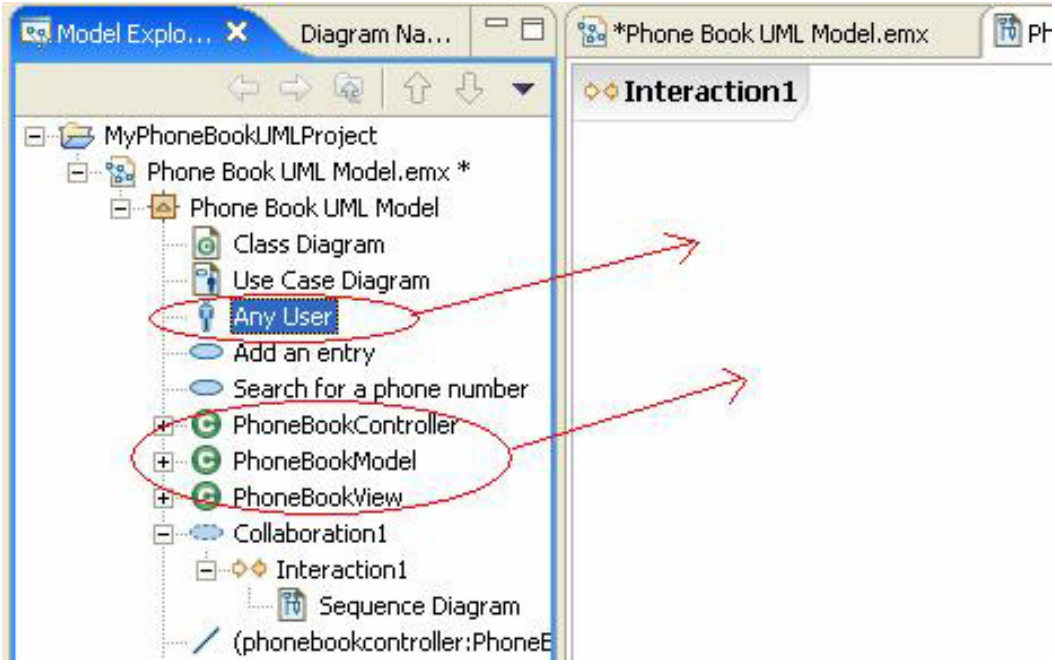
Would you like to see these steps demonstrated for you?

Show me

1. In the Model Explorer view, right-click **Phone Book UML Model** and select **Add Diagram** > **Sequence Diagram**.
2. Enter `Sequence Diagram` as the name of the generated diagram to replace the default name Diagram1.

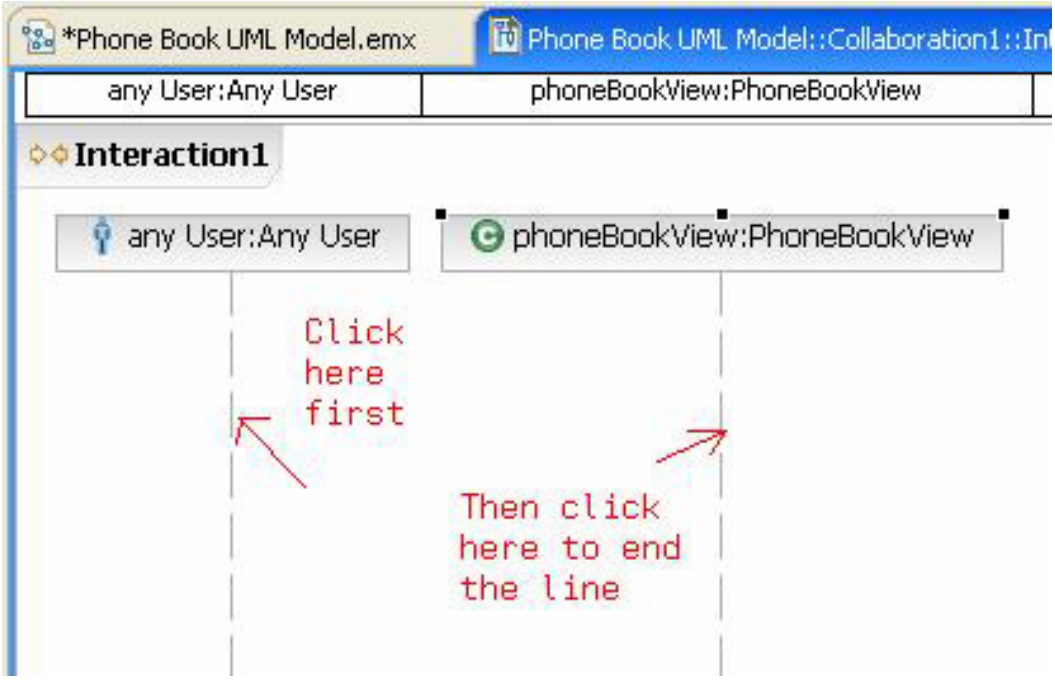
3. Drag the actor Any User from the Model Explorer view to the diagram to create an instance of the actor, as shown in Figure 10. Similarly, create instances of PhoneBookView, PhoneBookController and PhoneBookModel by dragging them to the diagram.

Figure 10. Drag the model elements to the sequence diagram

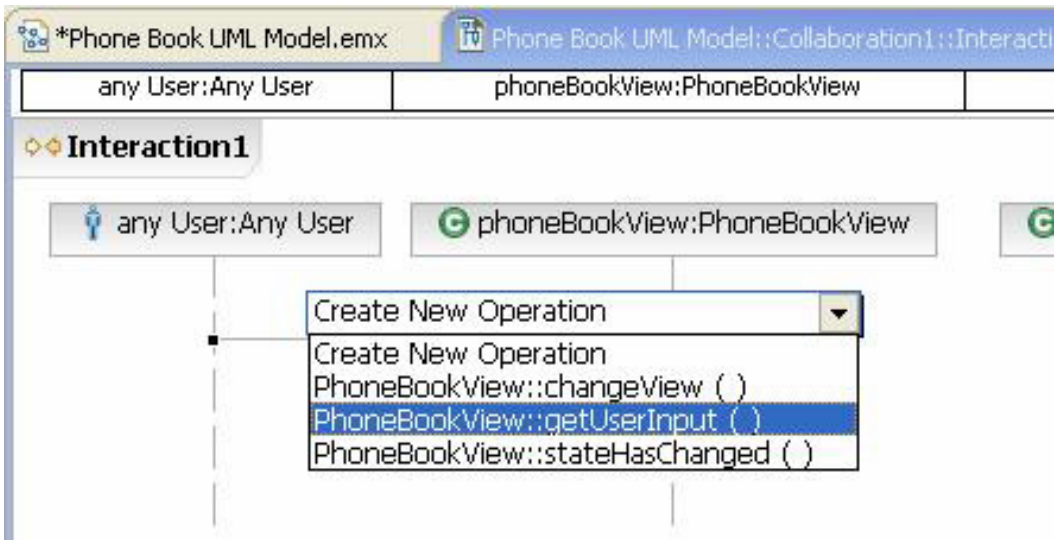


4. Select **Asynchronous Message** in the Palette. As shown in Figure 11, click the line under **any User: Any User** and then the line under **phoneBookView:PhoneBookView**.

Figure 11. Create message line



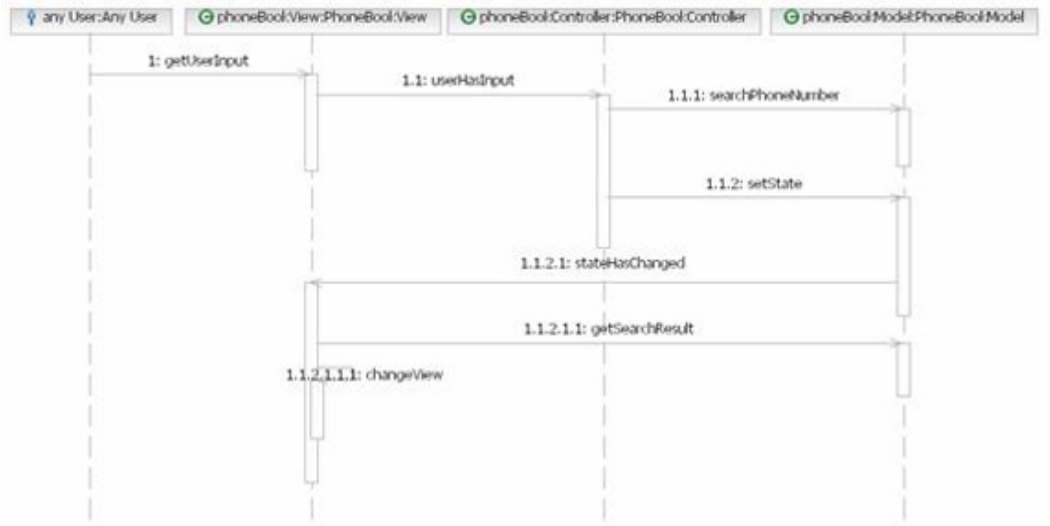
5. Select the operation **PhoneBookView::getUserInput()** from the drop down list.
Figure 12. Select operation for a message line



6. Similarly, create the following Asynchronous Message lines shown in Table 3. To create an Asynchronous Message to call itself, simply click the instance bar directly without any dragging.
Table 3. Message lines for the sequence diagram

From instance	To instance	Operation
phoneBookView	phoneBookController	PhoneBookController::userHasInput()
phoneBookController	phoneBookModel	PhoneBookModel::searchPhoneNum
phoneBookController	phoneBookModel	PhoneBookModel::setState()
phoneBookModel	phoneBookView	PhoneBookView::stateHasChanged()
phoneBookView	phoneBookModel	PhoneBookModel::getSearchResult()
phoneBookView	phoneBookView	PhoneBookView::changeView()

7. The complete sequence diagram should look like Figure 13. Select **File** > **Save All** to save everything.
Figure 13. Complete sequence diagram



Section 9. Publishing the design

By publishing the model information, you can share it with people who do not have the modeling tool. Rational Software Architect supports two publishing capabilities:

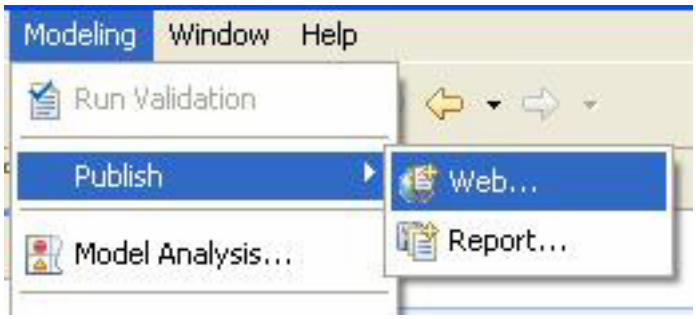
- Publishing models to a Web page
- Publishing a model information report

To publish your design to a Web page:

Would you like to see these steps demonstrated for you?

Show me

1. Select **Phone Book UML Model** in the Model Explorer view. Select **Modeling** > **Publish** > **Web**.
   Figure 14. Publish design to a Web page

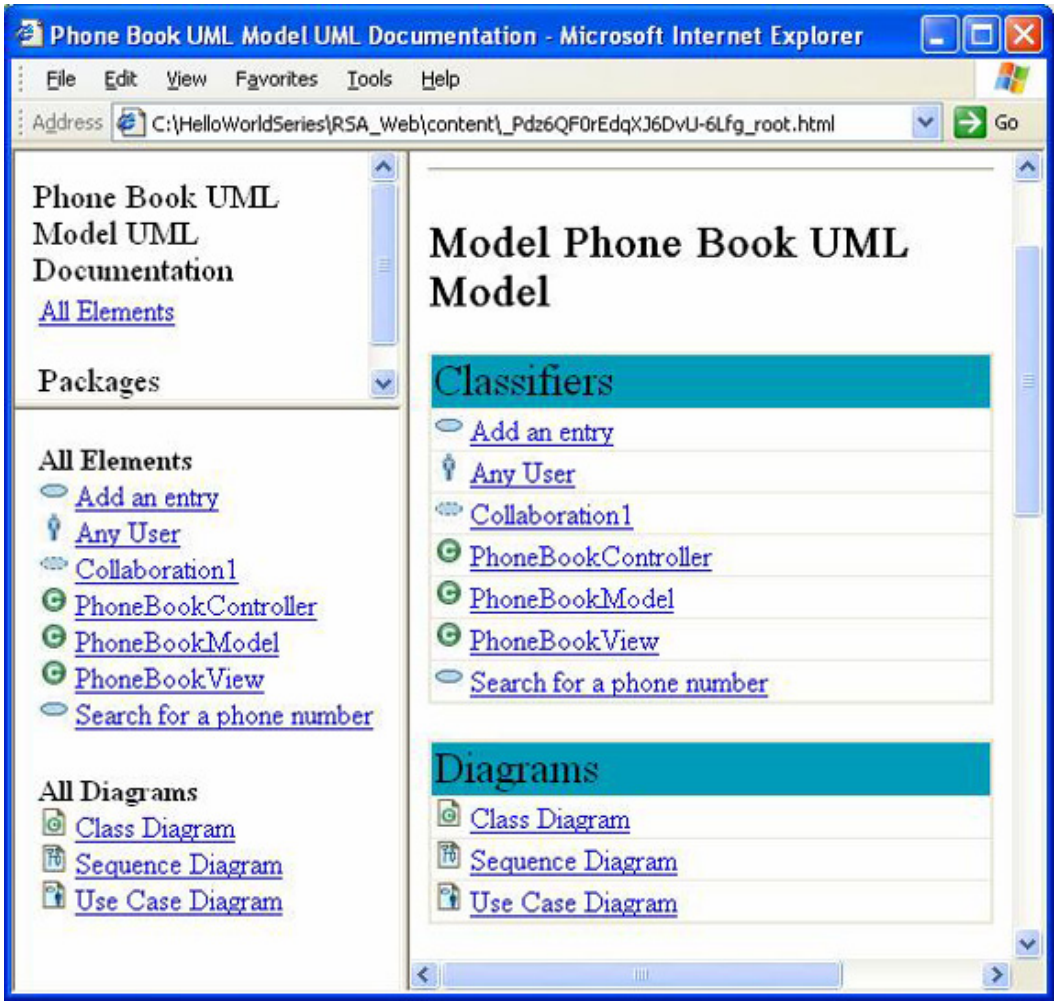


2. Specify the target location of the generated HTML files, for example C:\HelloWorldSeries\RSA_Web from Download, and select **OK**. The model is then published into HTML files that are stored in the specified location.

3. Open the file C:\HelloWorldSeries\RSA_Web\index.html using any Web browser.
Figure 15. Published Web page

4. Select the link **Phone Book UML Model**.

5. Navigate the published model by clicking the elements links and the diagrams.
Figure 16. Published Phone Book UML Model



Section 10. Transforming UML to Java

The transformation function is a key feature of Rational Software Architect. You can transform the design from UML to C++, CORBA, EJB code, JACL, Java code, and so on, and begin your implementation phase with a head start. In this section, you'll transform the UML design into Java code where you can fill in the programming details and run the application.

Would you like to see these steps demonstrated for you?

Show me

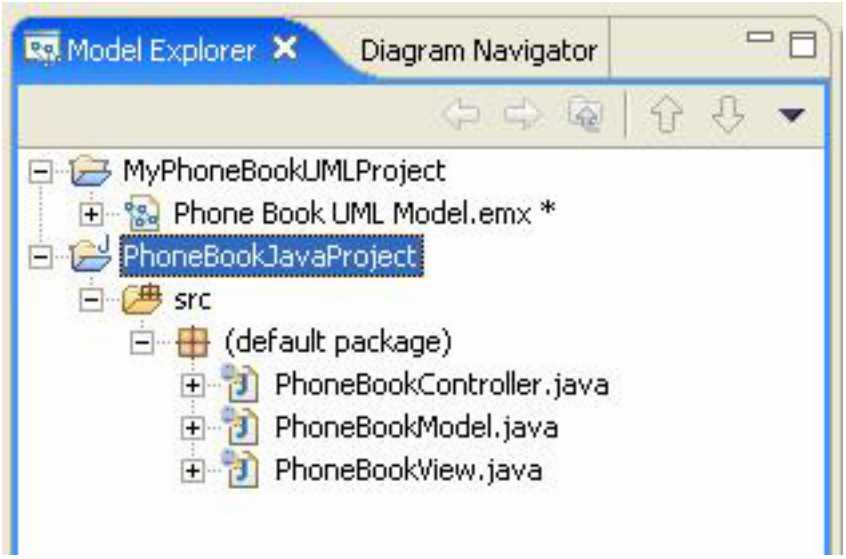
1. In the Model Explorer view, right-click **Phone Book UML Model** and select **Transforms** > **UML to Java**.

2. Select **Create new Target Container...**.

3. In the New Java Project panel, enter `PhoneBookJavaProject` as the Java project name, and select **Create separate source and output folder**. Select **Finish**.

4. Select **Run** to start the transformation.

5. In the Model Explorer view, a Java project PhoneBookJavaProject is created automatically with three Java files underneath.
Figure 17. Generated Java project



Let's take a look at the **PhoneBookController.java**, shown below. The generated class has two attributes, **phonebookmodel** and **phonebookview**, which are generated as a result of the directed association and association relationships, respectively. The two operations (**userHasInput** and **start**) that you added earlier when drawing the class diagram are also generated.

```
public class PhoneBookController {
    /**
     * Comment for <code>phonebookmodel</code>
     * @generated "UML to Java (com.ibm.xtools.transform.uml2
     */
    private PhoneBookModel phonebookmodel;

    /**
     * Comment for <code>phonebookview</code>
     * @generated "UML to Java (com.ibm.xtools.transform.uml2
     */
    private PhoneBookView phonebookview;

    /**
     * @generated "UML to Java (com.ibm.xtools.transform.uml2
     */
    public void userHasInput() {
        // TODO Auto-generated method stub
    }

    /**
     * @generated "UML to Java (com.ibm.xtools.transform.uml2
     */
    public void start() {
        // TODO Auto-generated method stub
    }
}
```

You can modify the files to implement the application based on the design infrastructure. A sample implementation is in the Downloads section. It implements a line command interface and stores the phone entries into a local file.

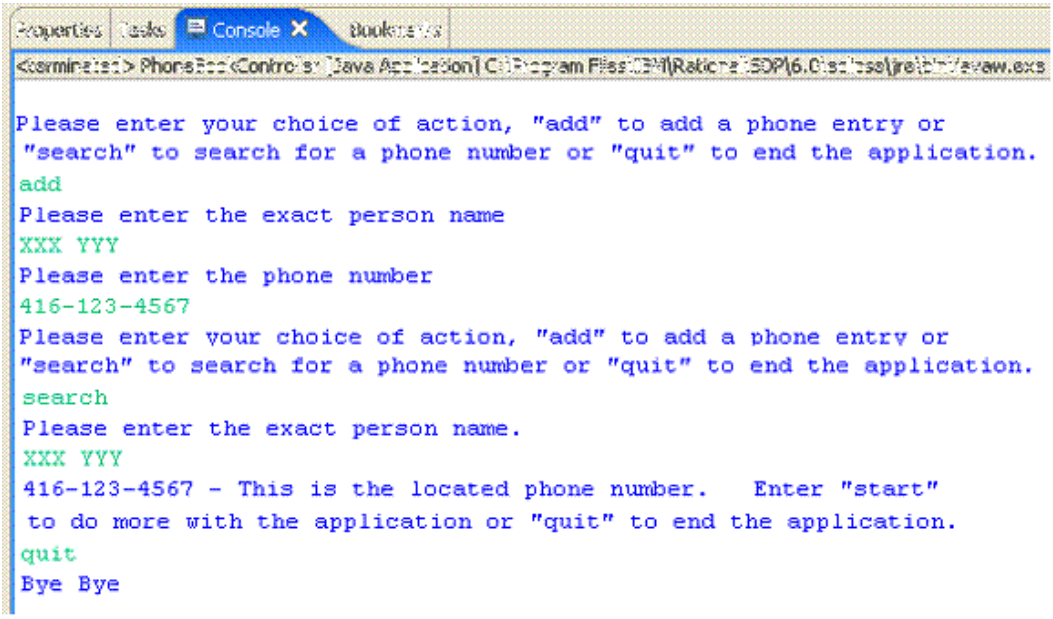
The design just lays out the infrastructure, which can lead to many different implementations. For example, you can create a GUI interface instead of a line command interface, store the data with EJB components instead of to a file, or use the Observer pattern to implement the notification mechanism. The provided sample implementation is just one of many ways to implement the design.

Now copy and paste the sample implementation.

1. Select **File** > **Save All** to save everything.
2. Select **PhoneBookController**, then select **Run** > **Run As ...** > **Java Application** to run the phone book implementation as a Java application.

3. Go to the Console view, as shown in Figure 18, and interact with the application. Validate that you can perform the two use cases, **Add an entry** and **Search for a phone number**, that were discussed earlier. Remember, the purpose of use cases is to define the behavior of a system and capture the requirements. It is important that the implementation fulfills the requirements, and is working as expected.

Figure 18. Running the sample phone book application



Section 11. Summary

This tutorial provided an introduction to Rational Software Architect, and highlighted some basic features. The exercise showed you how to design an application using UML diagrams, how to publish the model information into a Web page, and how to transform the design to Java using Rational Software Architect.

Stay tuned for the next part of this series, which will introduce you to Rational Application Developer.

Downloads

Description	Name	Size	Download method
Sample phone book application	phone.zip	4KB	HTTP

Information about download methods

Resources

Learn

- IBM Rational Software Architect: Find technical documentation, how-to articles, education, downloads, and product information about Rational Software Architect.
- IBM Service Oriented Architecture (SOA): Learn more about SOA from IBM.
- Donald Bell's article, UML basics: An introduction to the Unified Modeling Language, provides an overview of UML.
- More on Model-View-Controller.

Get products and technologies

- Download a free trial version of Rational Software Architect V6.0.
- Build your next development project with IBM trial software, available for download directly from developerWorks.
- Discover new SOA design resources that are available for download.

Discuss

- Participate in the discussion forum for this content.

About the author

Tinny Ng
Tinny Ng is a scenario architect at the IBM SWG Scenario Analysis Lab, whose mission is to improve the cross-brand integration capability of IBM SWG products. She has more than 10 years of experience in software development, from design to implementation, including application building, packaging, testing, and support.