Getting Started with Borland® Together® Edition for *SAP NetWeaver*™ *Developer Studio* (Together)

VERSION 6.2

Borland®
Together®

Borland Software Corporation
100 Enterprise Way, Scotts Valley, CA 95066-3249
www.borland.com

TogetherSoft Corporation, a wholly owned subsidiary of Borland Software Corporation, may have patents and/or pending patent applications covering subject matter in this document. Please refer to the product CD or the About dialog box for the list of applicable patents.

Copyright © 2002-2003 TogetherSoft Corporation, a wholly owned subsidiary of Borland Software Corporation. All rights reserved. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. SAP NetWeaver is a trademark of SAP AG. All other marks are the property of their respective owners.

Updated: August 8, 2003

Contents

About this guide 5
Navigating around Together. 5
Creating a project 7
Creating a use case diagram 8
Using a class diagram 12
Adding methods and attributes 14
Creating relationships and links 16
Creating hyperlinks 17
Creating a sequence diagram 18
Creating the initial sequence 19
Associating an object with a class 20
Resolving sequence diagram problems 23
Using the Properties view. 25
Using the Overview 26
Using and creating patterns 27
Managing diagram views 31
Specifying a detail level. 31
Hiding/Showing Information 32
Running quality assurance. 33
Running audits 34
Running metrics 36
Saving and loading metric results 39
Setting rules for quality assurance. 39
Comparing metrics results 41
Generating documentation 43
Exporting/Importing XMI projects 47
Exporting XMI projects 47
Importing XMI projects 49
Setting UML profiles 50
Creating a collaboration diagram 51

About this guide

The goal of this guide is to provide you with a basis for understanding and using Borland® Together® Edition for *SAP NetWeaver™ Developer Studio* (Together). It shows how to create a sample project and uses the project to demonstrate the product's primary features.

As such, it is recommended that you follow the examples in this guide in sequential order. For more task-oriented instructions, refer to the User Guide in the product's online help.

This guide provides examples for:

- Navigating around Together
- Creating a project
- Creating a use case diagram
- Using a class diagram
- Creating a sequence diagram
- Using the Properties view
- Using the Overview
- Using and creating patterns
- Managing diagram views
- Running quality assurance
- Generating documentation
- Setting UML profiles

Navigating around Together

Together provides a Modeling perspective and corresponding Diagram view. The Modeling perspective is central to designing projects in UML and creating source code for your project. Note that a key feature of Together is its ability to simultaneously synchronize diagrams and source code.

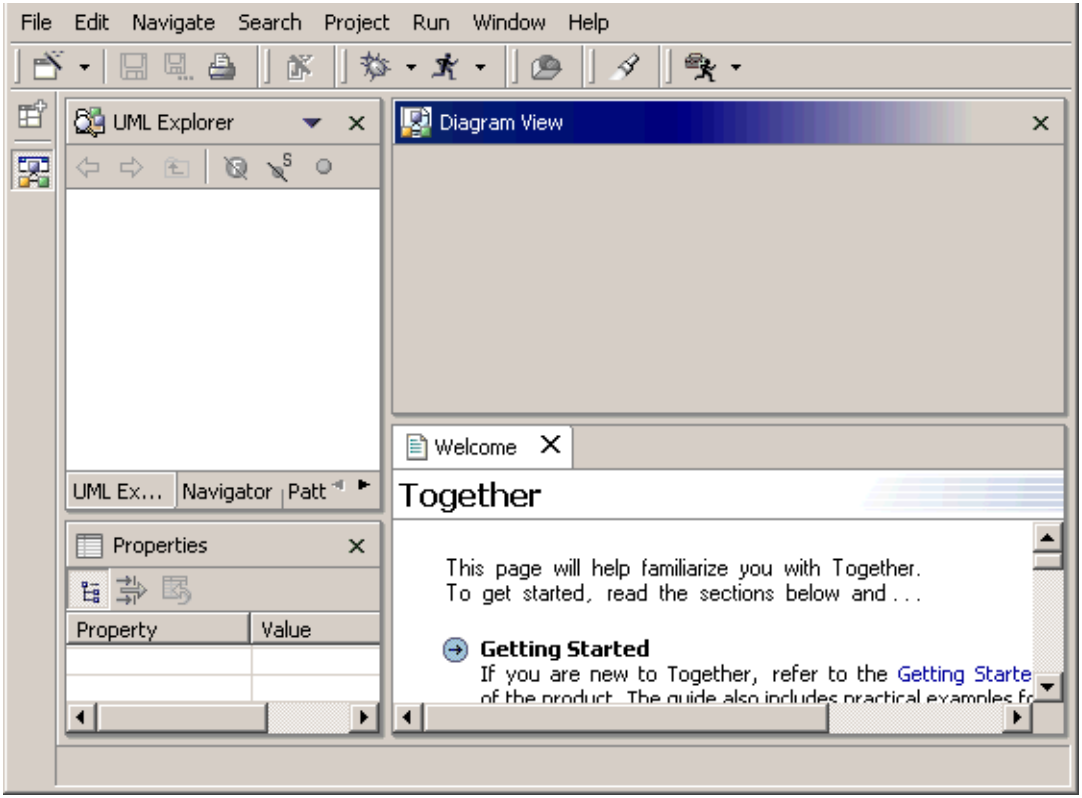
When you start Together for the first time, the Modeling perspective opens by default. You can also open the Modeling perspective manually and show the Diagram view.

To open the Modeling perspective and show the Diagram view:

1 Choose **Window > Open Perspective > Other > Together Modeling**. If the Modeling perspective is open but not active, click the Together Modeling button on the vertical toolbar at the left of the window.

2 Choose **Window > Show View > Diagram View**. The Modeling perspective is displayed similar to the one shown in Figure 1.

Figure 1 Modeling Perspective



The Modeling perspective has four main areas:

- **Navigation view** – By default, this area contains three views:
  - **UML Explorer** – UML tree of projects that you can use to manage elements, open, create, delete etc.
  - **Navigator** – Standard view.
  - **Patterns Recognizer**– Lists the recognized patterns contained in the projects.
- **Diagram view** – Displays the open diagrams. This view provides a tab for each open diagram.
- **Properties view** – In the Modeling perspective, this view shows the properties for a selected element. Properties for each element are usually broken into four categories:

- Description – Textual description of element
- Hyperlinks – Links to other elements or external files/documents
- Common properties – UML properties
- Requirements – Requirement information

- **Java Editor** – Standard Java™ editor

Creating a project

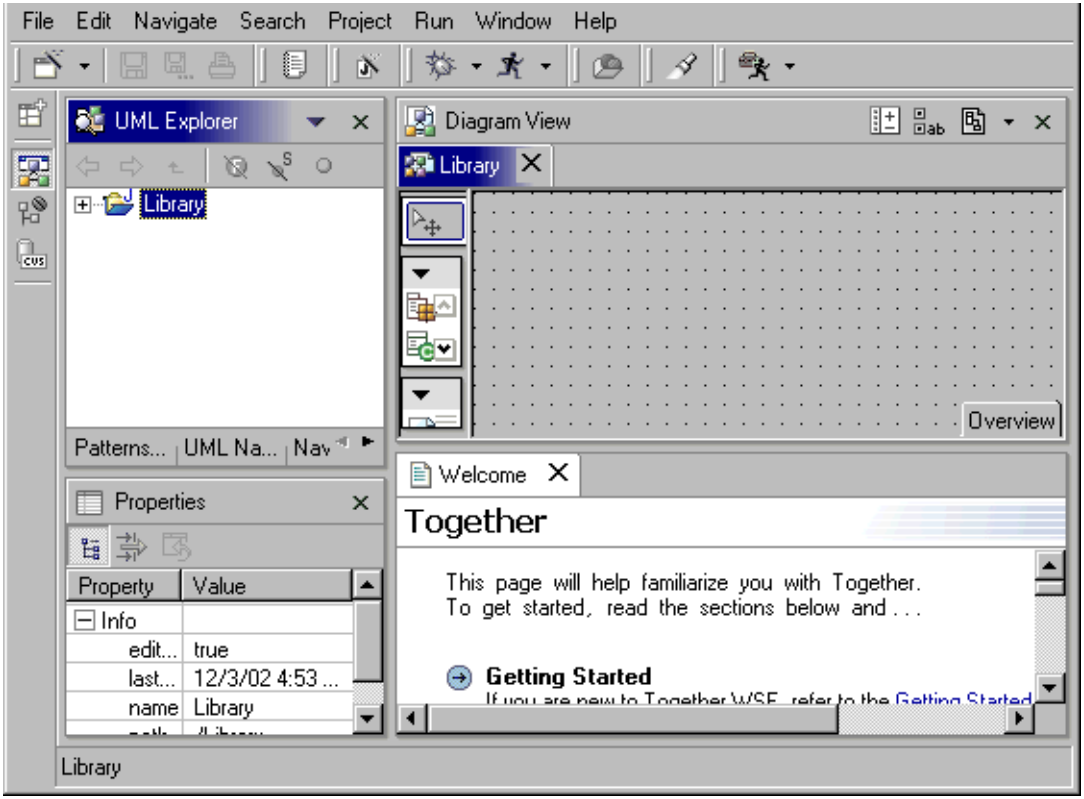
This guide models a simple library to explore the various features of Together.

To create a new project:

1. From the main menu, choose **File > New > Project > Together > Java Modeling Project**.
2. Click **Next,** and enter "Library" as the project name.
3. Click **Finish**.

The default diagram for the project opens in the Diagram View, as shown in Figure 2.

Figure 2 Library project as viewed from the Modeling perspective



Creating a use case diagram

These instructions assume that you have created the Library project as explained in the previous section.

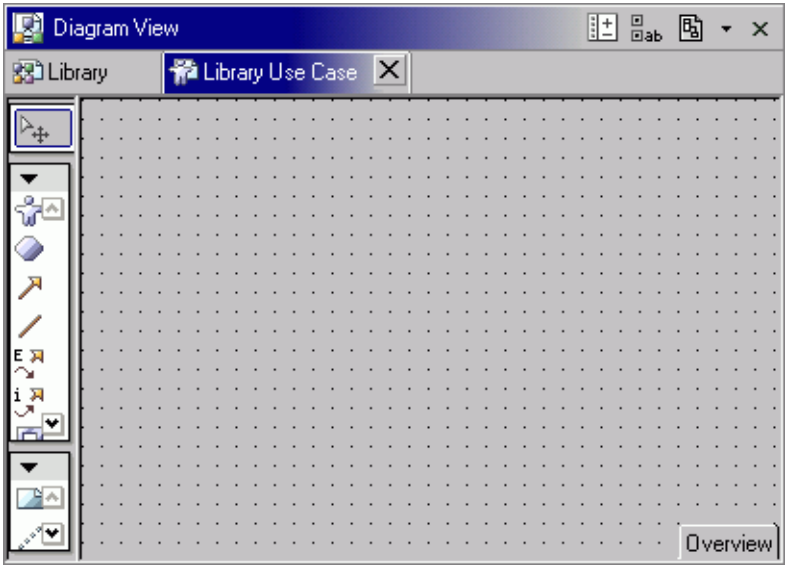
To create a use case diagram:

1. In the UML Explorer, right click on the **Library** project and choose **New > UML Diagram**.
2. In the resulting dialog box, choose **Use Case Diagram** as the type. Enter "Library Use Case" as the name. Click **Finish**.

Note You can use the UML Diagram menu, as described in this step, to create any of the diagrams supported by Together.

The Library Use Case tab appears as shown in Figure 3.

Figure 3 Library Use Case tab



3 From the Diagram View toolbar, click on the Actor button , and then click on the diagram background. Enter "Librarian" as the actor name.

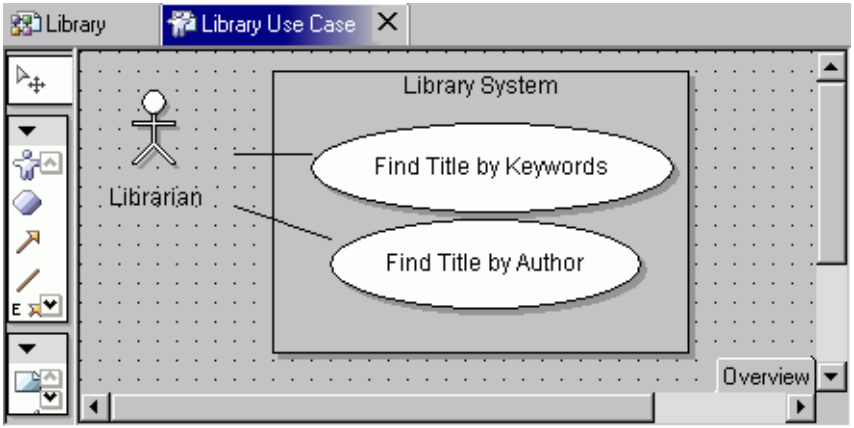
4 Click the System Boundary button , and then click the diagram background to the right of the Librarian. Rename the system boundary "Library System."

5 Click the Use Case button , and create two use cases inside the system boundary. Rename the use cases "Find Title by Keywords" and "Find Title by Author."

6 Click the Communicates link button and drag-and-drop from the actor to the use case. Repeat this step to create a link for each use case.

7 Your use case diagram should be similar to the diagram in Figure 4.

Figure 4 Sample use case diagram



8 Select the **Librarian** actor to view the properties for Librarian in the Properties view as shown in Figure 5.

Figure 5 Properties view for Librarian

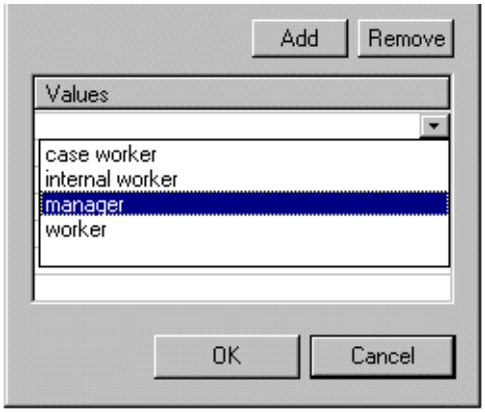
9 In the Properties view, click on the stereotype field. An information button ... appears as shown in Figure 5. You may need to scroll to the right to see this button.

Note Several properties contain the information button or the drop down arrow. These provide you with additional choices for selection or entry, either as drop down lists or dialogs.

10 Click the information button ... to open the Stereotype dialog shown in Figure 6.

11 Click **Add** and then click the drop down arrow . Select **manager** from the drop down list and click **OK**.

Figure 6 Adding a stereotype to an actor



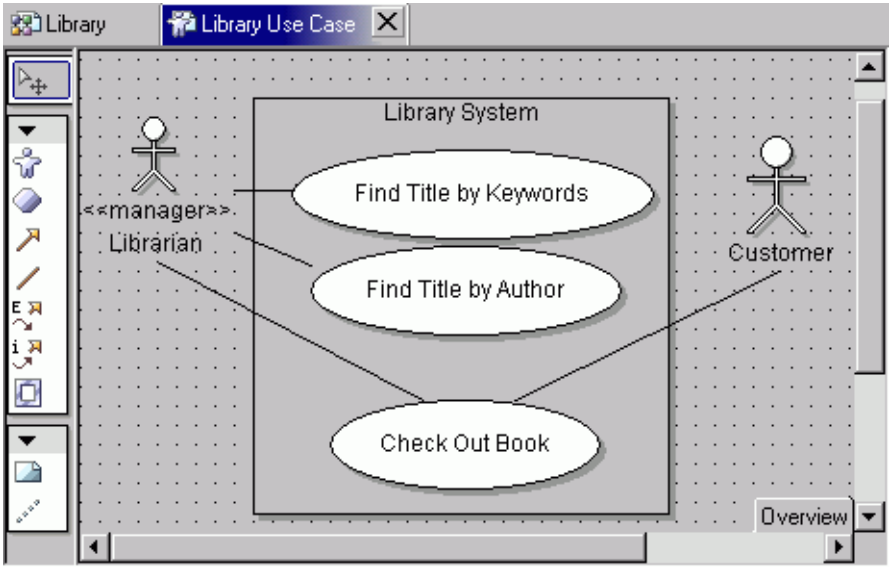
12 Add another use case inside the Library System boundary. Rename it "Check Out Book."

13 To the right of the Library System boundary, add another actor. Rename it "Customer."

14 Link the use case to both Customer and Librarian.

15 Your diagram should be similar to the one in Figure 7.

Figure 7 Use case diagram for Library project



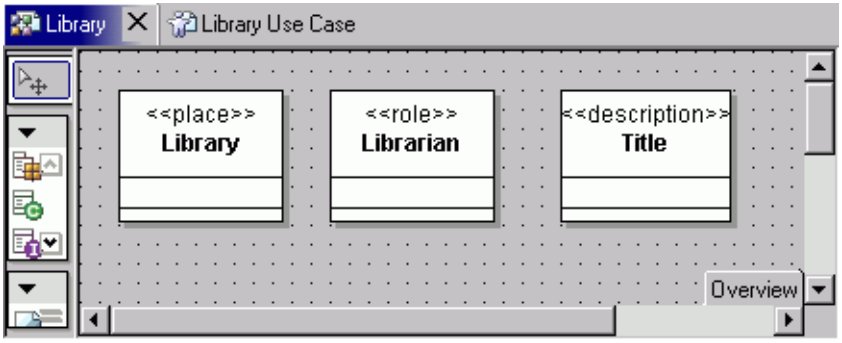
Using a class diagram

Together creates a class diagram for each package in a project. The Library project has a "default package" diagram that represents the package at the root level of the project. This default is a class diagram.

To open and use the default package diagram:

1. In the UML Explorer, expand **Library > (default package) > (default diagram)**.
2. Double click **(default diagram)**. The Library class diagram opens in the Diagram view.
3. To create a class, click the Class button and then click on the diagram background. Rename the class "Library."
4. Select the **Library** class to view its details in the Properties view.
5. Click on the stereotype value field to view the information button ... . You may need to scroll to the right to see the information button.
6. Click the information button to open the Stereotype dialog. Click **Add** and then click the drop down arrow . Select *place* and click **OK**.
7. Repeat steps 3-6 to create classes for `Librarian` (stereotype = role) and `Title` (stereotype = description). Your diagram should be similar to the diagram in Figure 8.

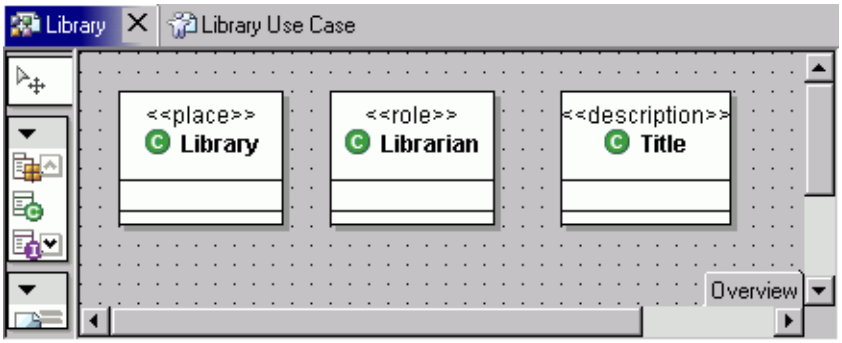
Figure 8 Library diagram with Library, Librarian, and Title classes



The screen shots used in this guide show a standard UML view for diagrams, as shown in Figure 8. By default, Together opens class diagrams in this view. However, Together can also display icons in class and package diagrams as shown in Figure 9.

To set view management properties, choose **Window > Preferences**. In the resulting dialog box, expand **Modeling** and then select **View Management**. In the **Details** tab, check the option *Show Icons on Class and Package Diagrams*. Click **OK**.

Figure 9 Class diagrams showing icons



As alternatives to the steps previously described, Together provides two other methods for creating a class:

- Right click on the diagram background and choose **New > Class**.
- In the UML Explorer, right click on the package and select **New > Class**.

Adding methods and attributes

As part of the requirements identified for the Library system, Table 1 lists the methods and attributes that you need to add to the classes you have created.

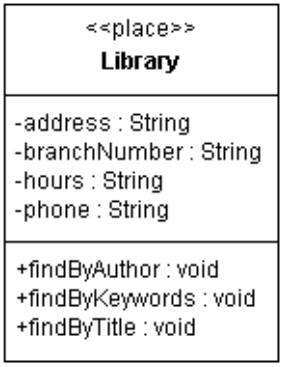
Table 1 Requirements for the Library sample project

Class	Attributes	Methods
Library	address branch number hours phone	findByAuthor findByKeywords findByTitle
Librarian	librarian ID name access level	
Title	title author publisher genre	getTitle setTitle

To create members for a class:

1. From the Library diagram, right click on the `Library` class and choose **New > Attribute**.
2. Rename the attribute `address:String`.
3. In the Properties view, click in the field for **visibility** and choose **private** from the drop down list.
4. By default, Together creates *private int* attributes and *public void* return methods. When you rename a class member on the diagram, you can also enter the type.
5. From the Library diagram, right click on the `Library` class and choose **New > Operation**.
6. Rename the operation `findByAuthor`.
7. Repeat the previous steps for the remaining attributes and methods for the `Library` class as listed in Table 1. Your Library class node should be similar to Figure 10.

Figure 10 Library class with attributes and methods



8 Continue adding the attributes and methods listed in Table 1 to the `Librarian` and `Title` classes. For `getTitle`, specify **String** for type.

9 In order to specify a return value of `String` for `getTitle`, you need to configure properties for Java as follows. From the main menu, choose **Window > Preferences**. In the resulting dialog box, expand **Modeling** and then select **Java**. Under *Java Bean Properties Support*, uncheck *Recognize Java Bean Properties* and *Hide Java Bean Properties Participants*.

   Select the `branchNumber` attribute of `Library`. In the Properties view, enter "Branch Number" as the alias value as shown in Figure 11.

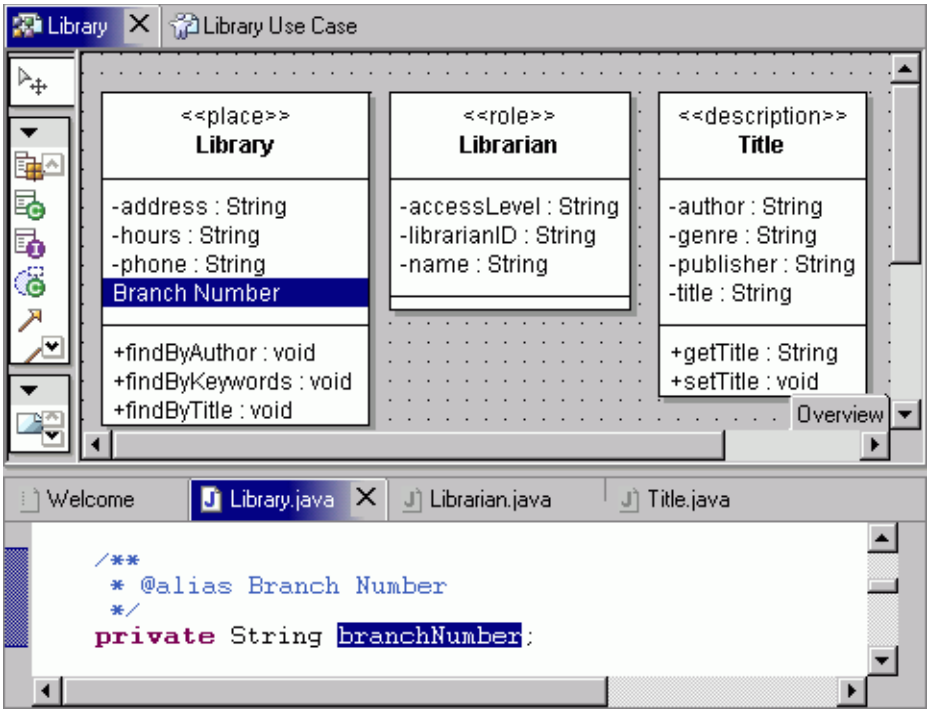
Note Some of the attributes in the table have names with spaces, which cannot serve as actual attribute names. For example, branch number becomes `branchNumber`. The *alias* property is useful for specifying a different name from that in the actual code.

Figure 11 Creating an alias for an attribute

Property	Value
⊟ Common properties	
alias	Branch Number
associates	
final	false
initial value	
name	branchNumber
static	false
stereotype	
transient	false
type	String
visibility	private
volatile	false
⊟ Hyperlinks	

The alias changes only the visual label for the attribute in the diagram, as shown in Figure 12. The attribute remains as `branchNumber` within the source code.

Figure 12 Alias for branchNumber and corresponding code



Creating relationships and links

The sample Library project includes two relationships:

- A `Librarian` is associated with a `Library`
- A `Library` consists of many `Titles`

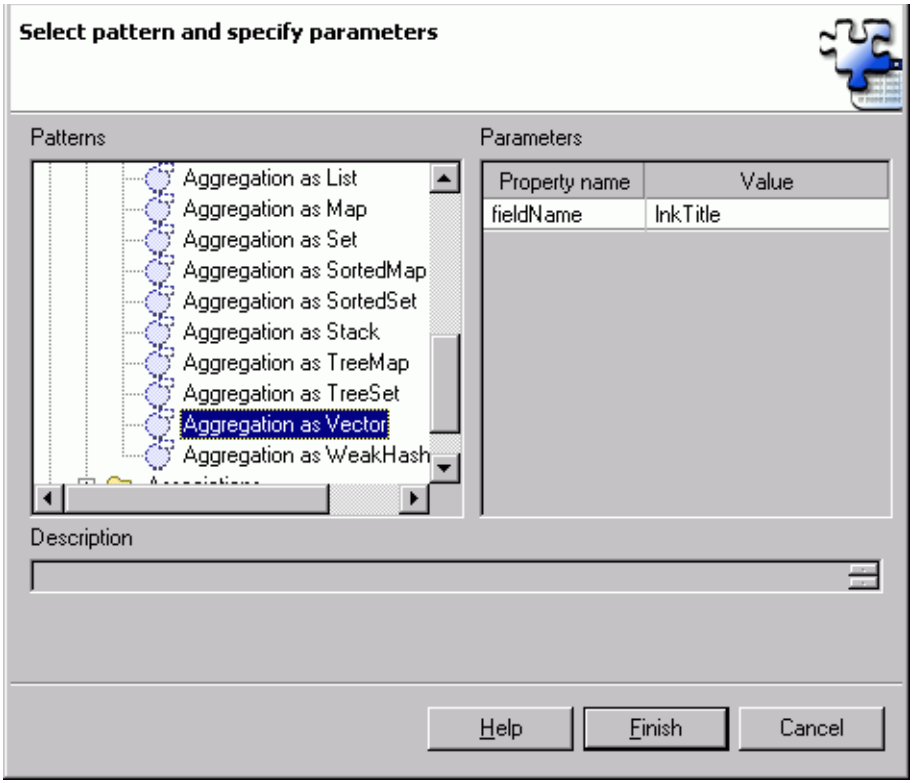
You can create an association link to establish the relationship between `Librarian` and `Library` as "client" and "supplier." For the second relationship, you can create an aggregation since `Library` has several titles. By using the Link By Pattern button, you can easily implement the aggregation with a `Vector`.

To create relationships and links:

1 Click the **Association Link** button . Click on **Library** and then drag-and-drop the link to **Librarian**. You have established the relationship from "client" to "supplier." Notice that as you move over a class, a black border highlights the location that Together recognizes as a valid destination for dropping the end of the link.

2 Click the **Link By Pattern** button . Click on **Library** and then drag-and-drop the link to **Title**. The Apply Pattern dialog opens as shown in Figure 13.

Figure 13 Apply Pattern dialog



3 Choose **java.util Collections > Aggregations > Aggregation as Vector**. Rename the field name to "titleList" to indicate a collection. Click **Finish**.

Together automatically adds the link information to the code by placing tags above the attribute. In this example, the link is an aggregation linked to the `Title` class. By double clicking on a class you can open its source in the Java Editor. For example, the source for the `titleList` attribute in `Library` is as follows:

```
/**@link aggregation <{Title}>*/
java.util.Vector titleList=null;
```

Note For instructions on how to use patterns to create or modify existing links and classes, see "Using and creating patterns" on page 27.

Creating hyperlinks

The Library has a `findByKeywords` method for searching for titles. The requirements for this method are defined as the use case you created earlier, *Find Title by Keywords*. By using hyperlinks, you can link diagrams and elements to express these types of relationships and record them in the model for others to use.

To create a hyperlink between the method and use case:

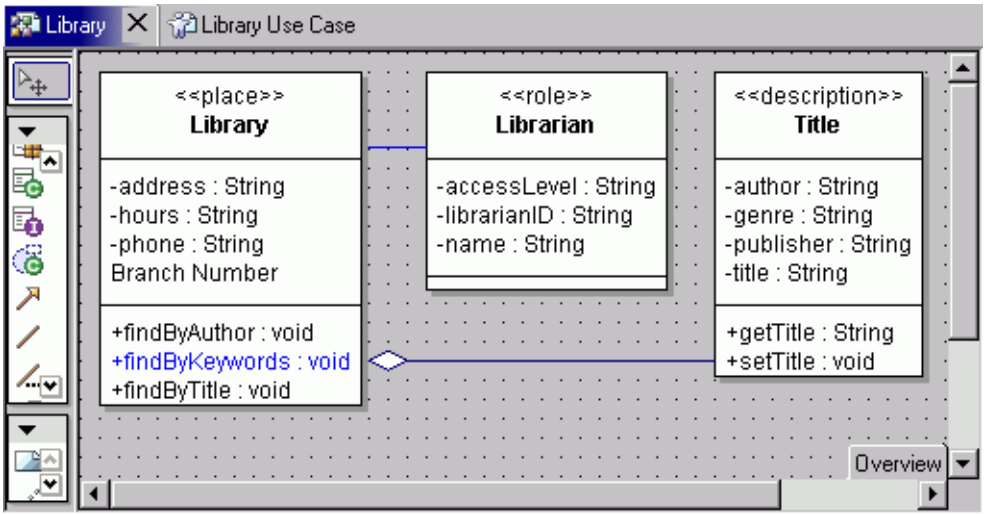
1 Right click on the **findByKeywords** method and choose **Hyperlinks > Edit**. The Hyperlinks dialog opens.

2 From the **Model Elements** tab on the left, select **Library > Library Use Case > Library System > Find Title by Keywords**.

3 Click **Add** to add the element to the Selected column on the right. Then click **OK**.

The method with the newly created hyperlink is highlighted in blue as shown in Figure 14. The corresponding use case is also highlighted in blue.

Figure 14 Hyperlinked element



4 To test the hyperlink, right click on the method and choose **Hyperlinks > Find Title by Keywords**. The diagram containing the corresponding use case opens with the use case selected.

Tip The Hyperlinks Dialog has tabs for choosing the hyperlink target. You can use the External Documents tab to connect to an existing document. This is helpful for linking an element such as *Find Title by Keywords* to requirements specified in another location. When you select the hyperlink, Together launches the default application associated with the file.

Creating a sequence diagram

Sequence diagrams are used to design the dynamic aspects of an object model. Together provides two modes for creating sequence diagrams:

- A simple “sketch pad”

- Two-way with source code and classes

Creating the initial sequence

You can create a sequence diagram in the Library project to model some steps necessary to check out a book.

To create a sequence diagram:

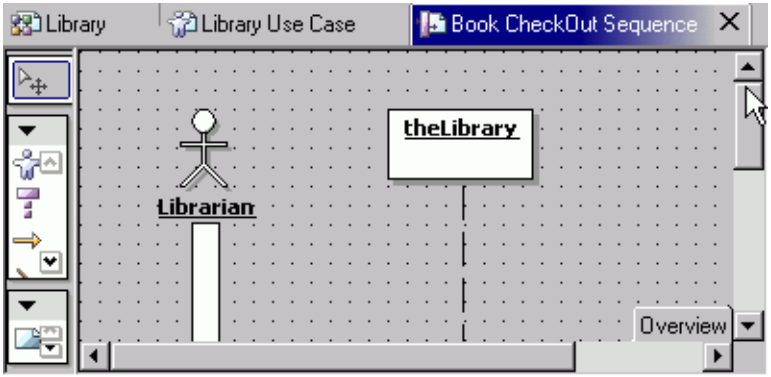
1 In the Library diagram, right click the background of the project and choose **New Diagram > Sequence**.

2 Click the new diagram to select it. Then go to the Properties view and change the name of the diagram to "Book Checkout Sequence."

3 From the Diagram view toolbar, click the Actor button , and then click the diagram background. Enter "Librarian" as the actor name.

4 Click the Object button and click to the right of the actor line. Name the object "theLibrary" as shown in Figure 15.

Figure 15 Librarian actor and the Library object

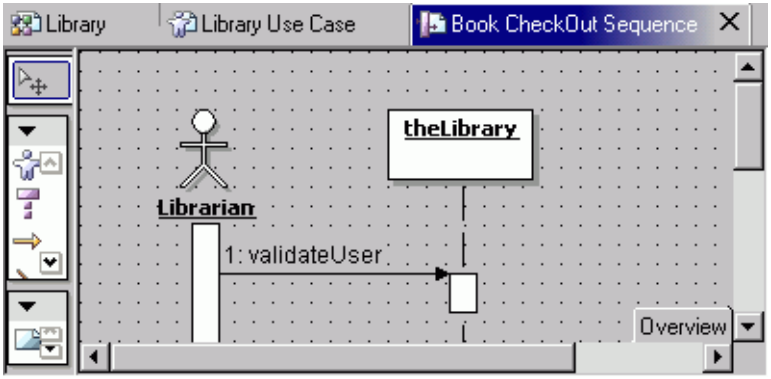


The steps that follow model the requirement of validating the user. The Librarian (having already captured the scanned Customer ID from their library card) will send a message to the Library asking for a validation of the Customer (at the same time it checks for any overdue fines).

5 To add the first message, click the Message button and then drag and drop from the Actor line to theLibrary lifeline. Figure 16 shows the resulting message line.

6 The in-place editor activates. Rename the message "validateUser," as shown in Figure 16.

Figure 16 Validate User message



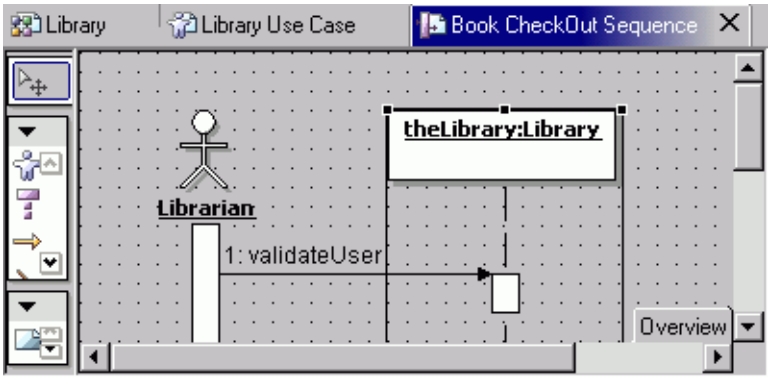
Associating an object with a class

To associate an object with a class:

1 Right-click on **theLibrary** object and choose **Select Class** > **Library**. The resulting sequence diagram is shown in Figure 17. Notice that the object name is followed by the name of the associated class.

Note After you associate a class to an object, you can right click the object and select **Unlink Class** to remove the association.

Figure 17 An object associated with a class



2 Select the validateUser message and edit the Properties view with the following changes as shown in Figure 18:

- Add the argument `ID`.
- Add the return type `float` (in case any money is due).

Figure 18 Properties for validateUser message

Property	Value
⊟ Properties	
arguments	ID
condition	
creation	false
destruction	false
iteration	
non-atomic delivery	false
return	float
return message	false
sequence number	1
synchronization	call

The diagram immediately updates to show the changes.

Steps 3-5 assure that the `Library` class delegates the process of validating a user.

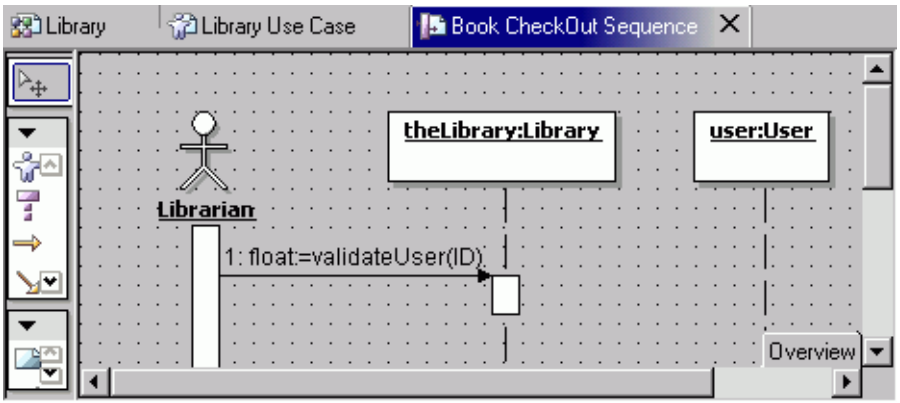
3 Click the Object button and then click to the right of the Library object. Rename this object "user."

4 Right-click on the **user** object and choose **New > Class**. The **New Object's Class dialog** opens.

5 For Name, enter "User" and then click **Finish**.

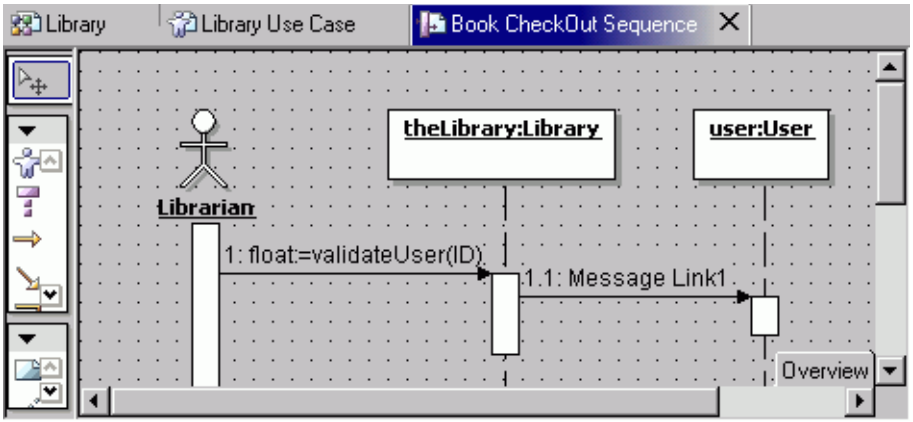
6 Your sequence diagram should be similar to the one shown in Figure 19.

Figure 19 Sequence diagram with User class



7 Add a new message (more specifically, a static finder or constructor). Instead of using "sketch" mode, create a real operation in the new `User` class. Click the Message button, and then drag and drop from theLibrary activation bar to the user lifeline. A second message line appears as shown in Figure 20.

Figure 20 Sequence diagram with two message lines



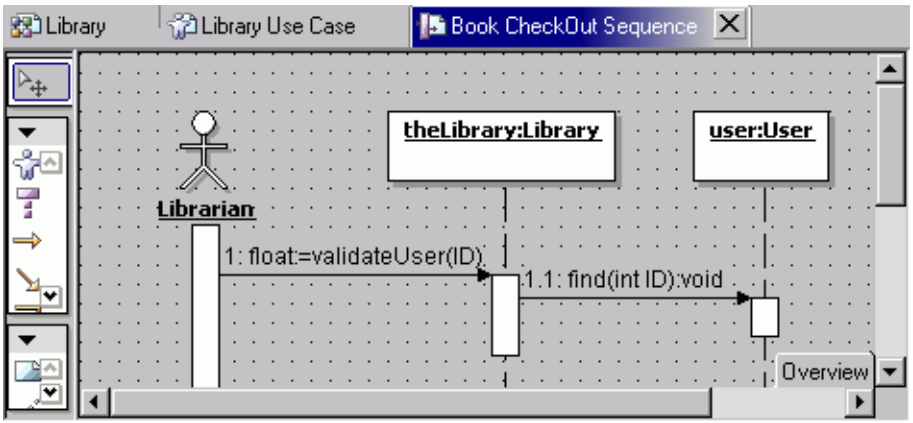
8 Right click on the new message line and choose **New > Operation**. The New dialog opens.

9 For Name, enter "find" and then click **Finish**.

10 Return to the Library class diagram view and select the `User` method named `find`. In the Properties view, change the value of *parameters* to "int id." This step illustrates how the sequence diagram automatically reflects changes to the Library class diagram

11 Return to the sequence diagram to see the updated parameter as shown in Figure 21.

Figure 21 Updated sequence diagram



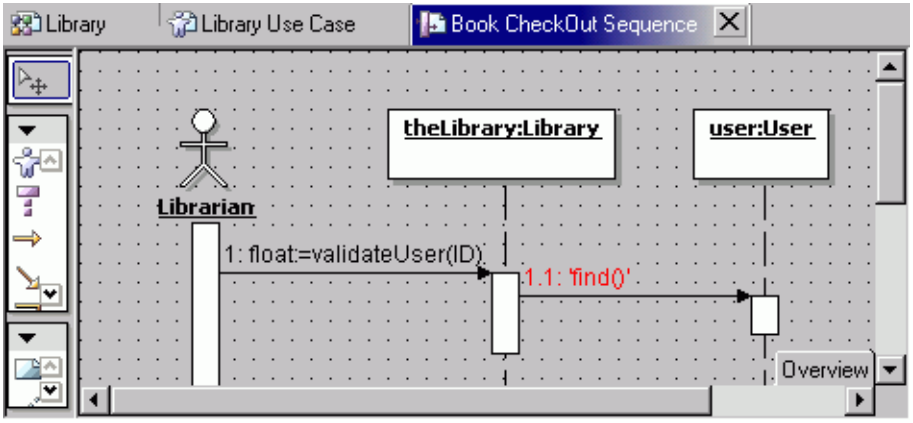
Resolving sequence diagram problems

The previous section demonstrated that Together automatically synchronizes changes between the sequence diagram and class diagram. However, the example that follows illustrates instances when you need to resolve differences between the sequence and class diagrams.

1 Open the Library class diagram and delete the `find` operation.

2 Return to the sequence diagram. The *find* operation is now highlighted in red and appears in single quotes, as shown in Figure 22. This signifies that the operation in the `User` class is no longer available.

Figure 22 Sequence diagram that requires resolution



3 To resolve the sequence diagram, right click on the diagram and choose **AutoFix** to display the Solve Problems dialog. The Solve Problems dialog indicates the problem, suggests a default solution, and lists other available solutions as shown in Figure 23.

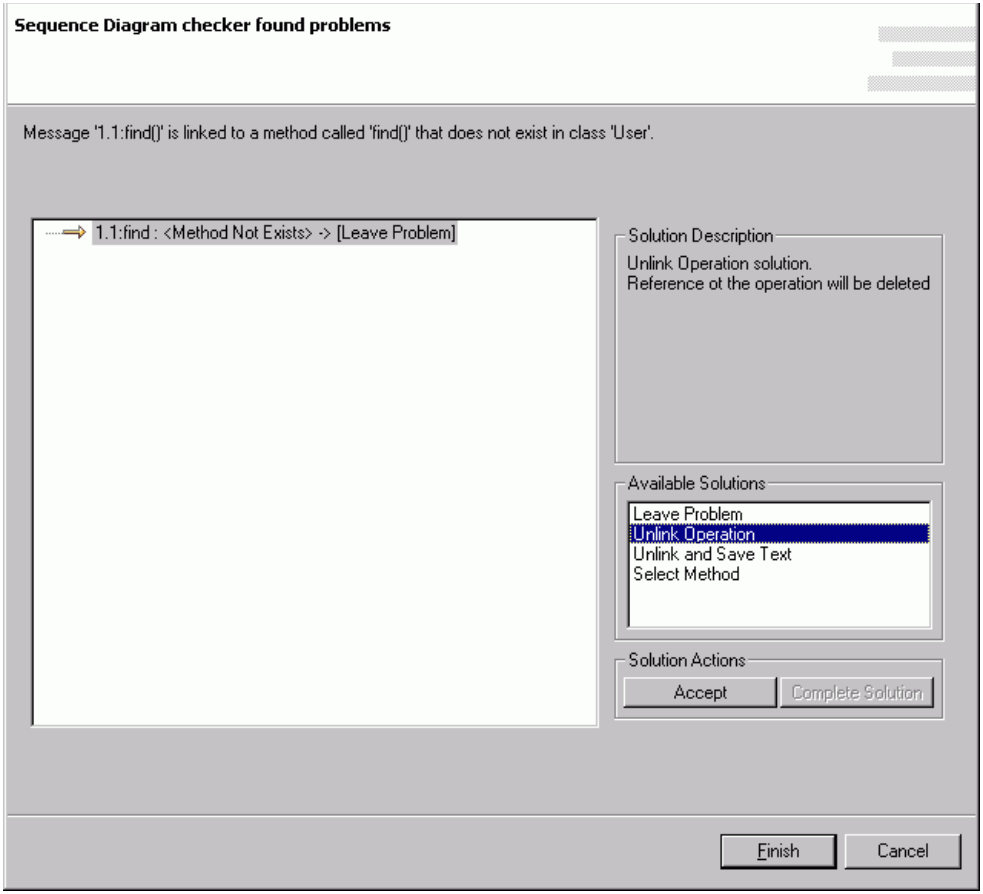
4 Choose the **find** method.

5 From Available Solutions choose **Unlink Operation**.

6 For Solution Actions, click **Accept**.

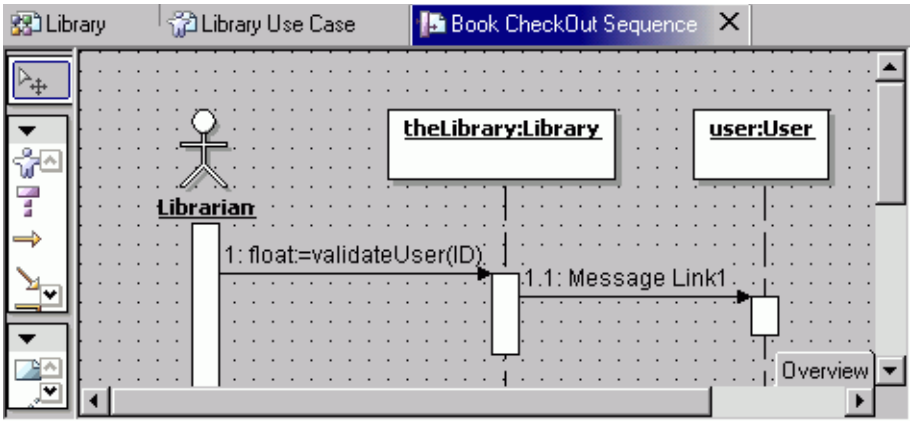
7 Click **Finish**.

Figure 23 Solve Problems dialog



The message link remains displayed but the operation is now unlinked as shown in Figure 24.

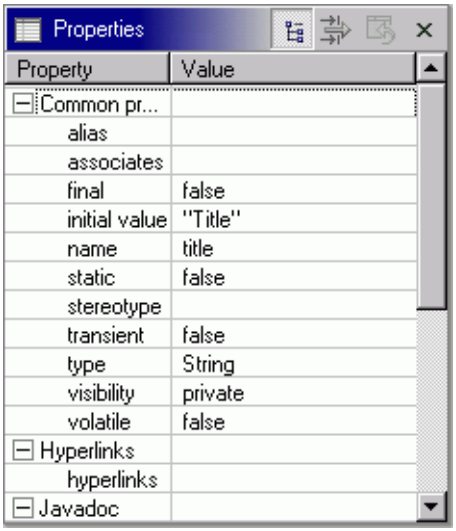
Figure 24 Sequence diagram with an unlinked operation



Using the Properties view

You can use the Properties view to view and edit values such as alias, name, stereotype, and others as shown in Figure 25.

Figure 25 Properties view



To add details to the Title class:

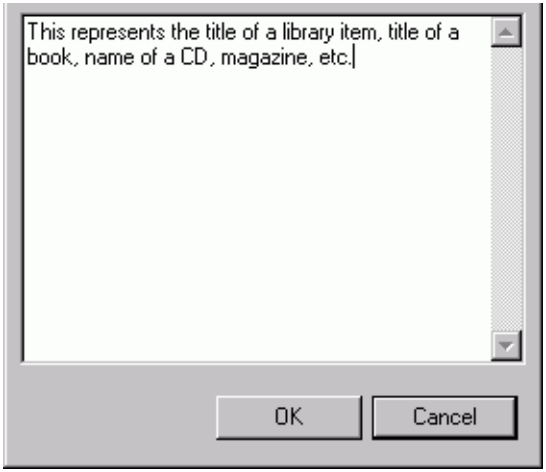
1 In the Properties view of the Library diagram, select the title attribute.

2 For *initial value*, enter “Title” (include the quotes), as shown in Figure 25.

3 Also in Properties, expand *Javadoc*, click in the field for *comment*, and then click the information button. The Enter text dialog opens.

4 Enter a comment for the title attribute as shown in Figure 26, and then click **OK**.

Figure 26 Enter text dialog



Using the Overview

The overview feature of the Diagram view provides a thumbnail view of the current diagram. The **Overview** tab is located in the bottom right hand corner of every diagram, as shown in Figure 27.

Figure 27 Overview tab

To use the overview feature:

1 On the Library diagram, click the **Overview** tab. The pane expands to show a thumbnail image of the current diagram as shown in Figure 28.

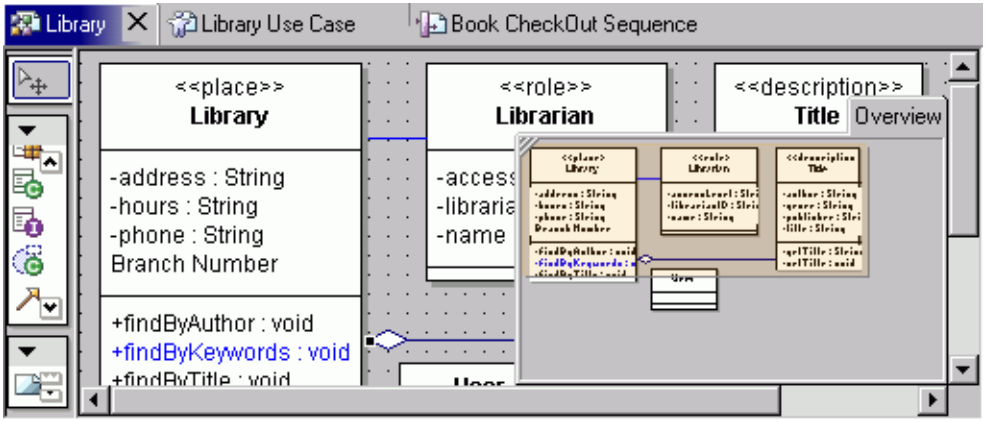
2 Use the mouse to click on the shaded area and drag it. This is a convenient way to scroll around the diagram.

Tip If you click on an area not covered by the shaded area, you will jump to that area in the diagram.

3 Alter the size of the Overview pane by clicking on the upper left hand corner of the pane and dragging to resize it.

The Overview pane automatically closes when you select an element on the diagram.

Figure 28 Thumbnail of Library diagram



Using and creating patterns

Together provides support of frequently used patterns such as the GoF patterns. You can use patterns to create or modify existing links and classes.

For the following example, assume that the Title class requires a dynamic system so that after a Title has been returned, the counter is updated and other systems are notified. The Observer pattern is useful for designing such a system.

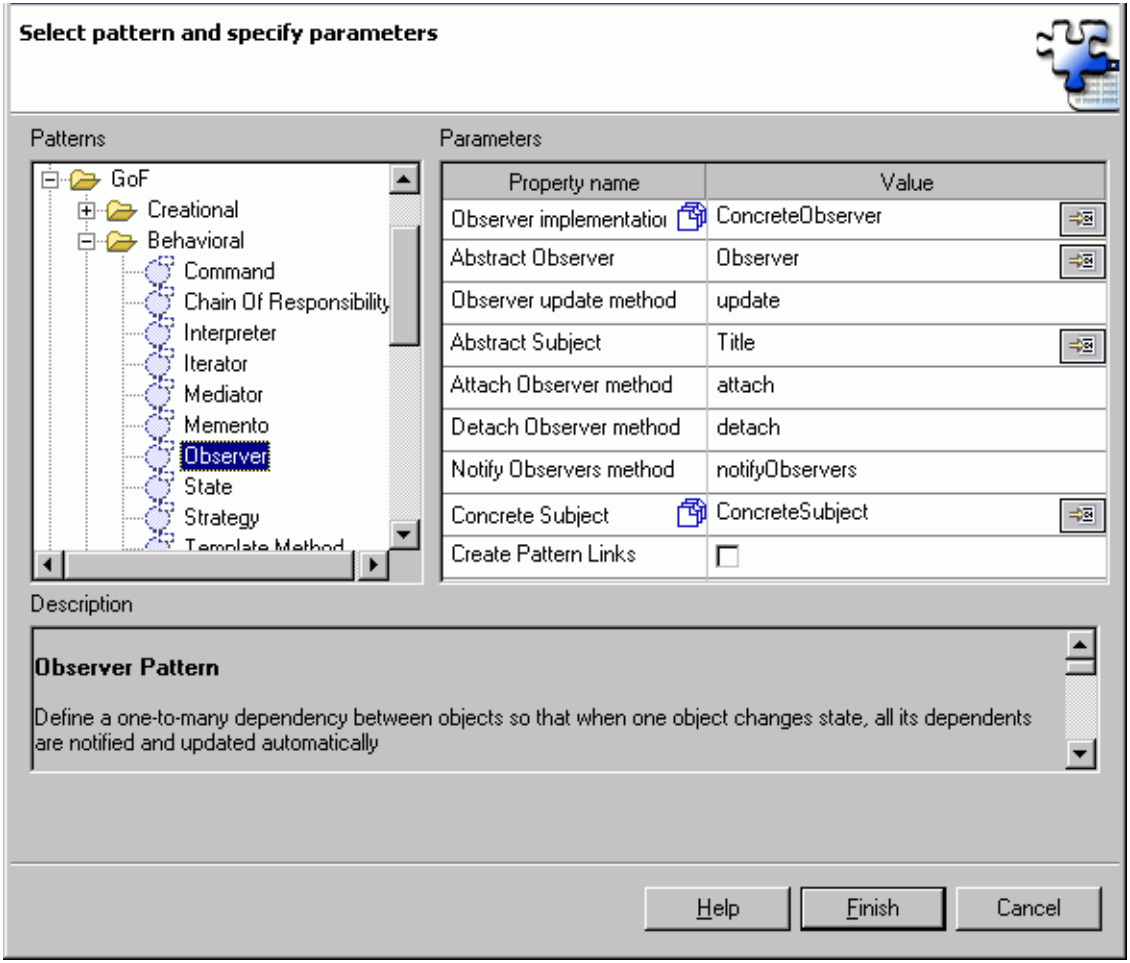
Note For instructions on how to create a link based on an existing pattern using Create Link by Pattern, see "Creating relationships and links" on page 16.

To apply a pattern:

1 On the Library class diagram, right click on the **Title** class and choose **Apply Pattern**. The Apply pattern dialog opens.

2 Choose the **GoF > Behavioral > Observer** pattern so that Title appears in the Abstract Subject field. (This is because you are creating a pattern with an existing class.) Figure 29 shows the corresponding Apply pattern dialog.

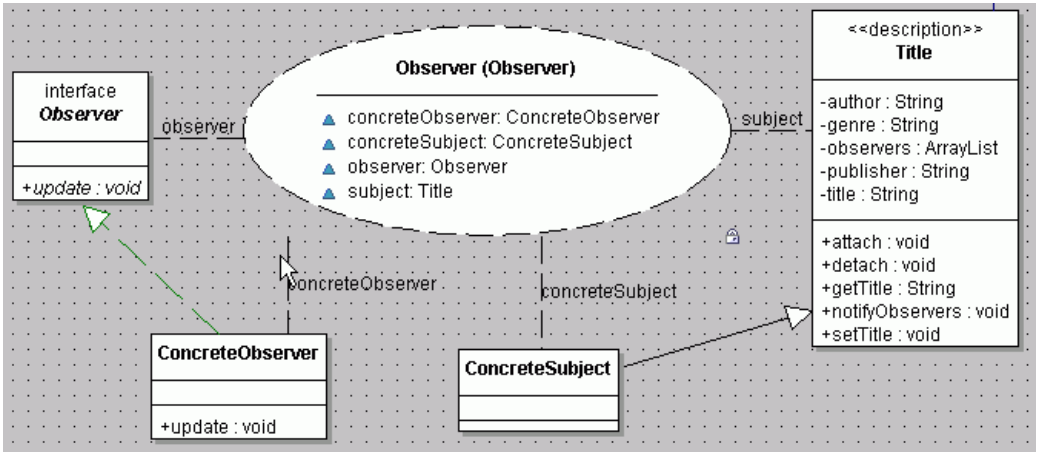
Figure 29 Apply pattern



3 Click **Finish**.

The diagram is updated as shown in Figure 30. `Title` is updated with the notification and observer methods (`attach` and `detach`). The other classes and interfaces have been created ready for use. Together recognizes the pattern and visualizes the pattern element on the diagram. In addition, it lists the participants of the pattern and the pattern links.

Figure 30 Applying the Observer pattern to a diagram

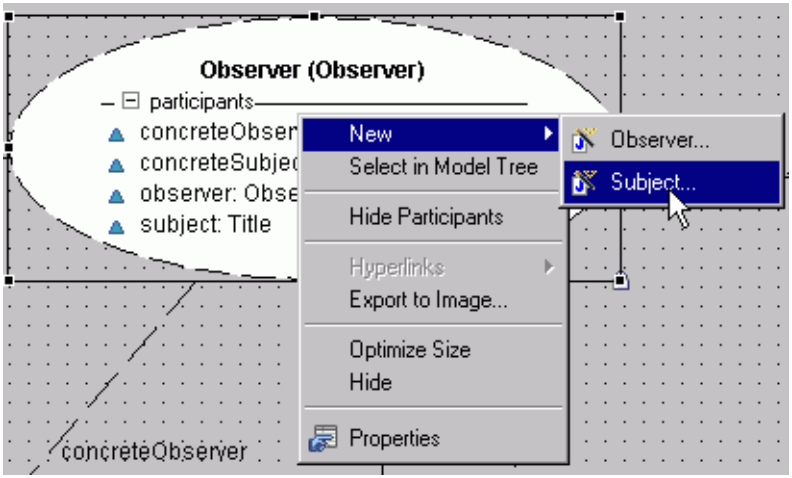


The pattern elements on the diagram have specific pattern actions that you can choose relevant to the pattern as shown in Figure 31.

If the Observer pattern "bubble" doesn't appear in your diagram, you may need to enable patterns.

1. In the menubar, select **Window > Preferences**.
2. Expand **Modeling > Patterns**.
3. Check the Recognize... checkbox on the General tab.
4. In the Patterns Recognizer view, right click on the project and select **Rebuild Patterns**.

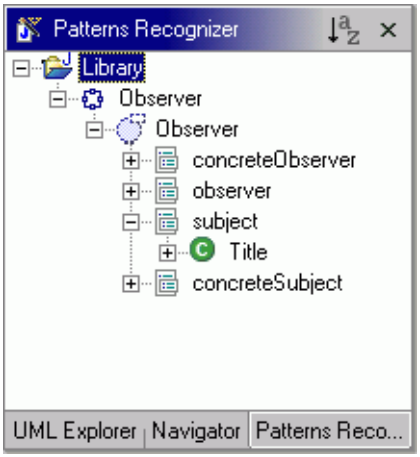
Figure 31 Applying pattern-specific options



The Patterns Recognizer view shows a list of the recognized patterns in the open projects. The Library project has only one pattern, as shown in Figure 32.

To set preferences for pattern recognition, choose **Window > Preferences > Modeling > Patterns**. This allows you to enable or disable recognition as well as specify which patterns should be recognized.

Figure 32 Patterns Recognizer view



Managing diagram views

View management controls which diagrams and elements are displayed in Together. This section explains two features for specifying diagram views: detail level and filters.

Specifying a detail level

You can use detail level to view the different levels of detail shown in class diagrams. Detail levels include Analysis, Design, and Implementation.

To specify a detail level:

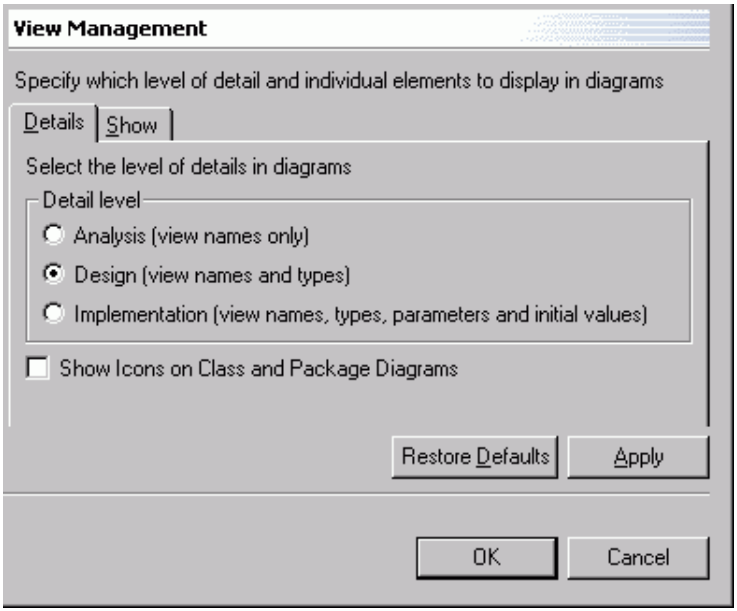
1 Choose **Windows > Preferences** to view the Preferences dialog shown in Figure 33.

2 On the left, expand **Modeling > View Management,** and select the **Details** tab.

3 Choose the detail level and click **OK**.

Note If you choose **Analysis** level, all information regarding types, return values, visibility and parameters will be hidden, leaving the classes and their members with text labels.

Figure 33 Preferences dialog



Hiding/Showing Information

When dealing with large projects, the amount of information shown in a diagram can become overwhelming. In Together, you can selectively show or hide information.

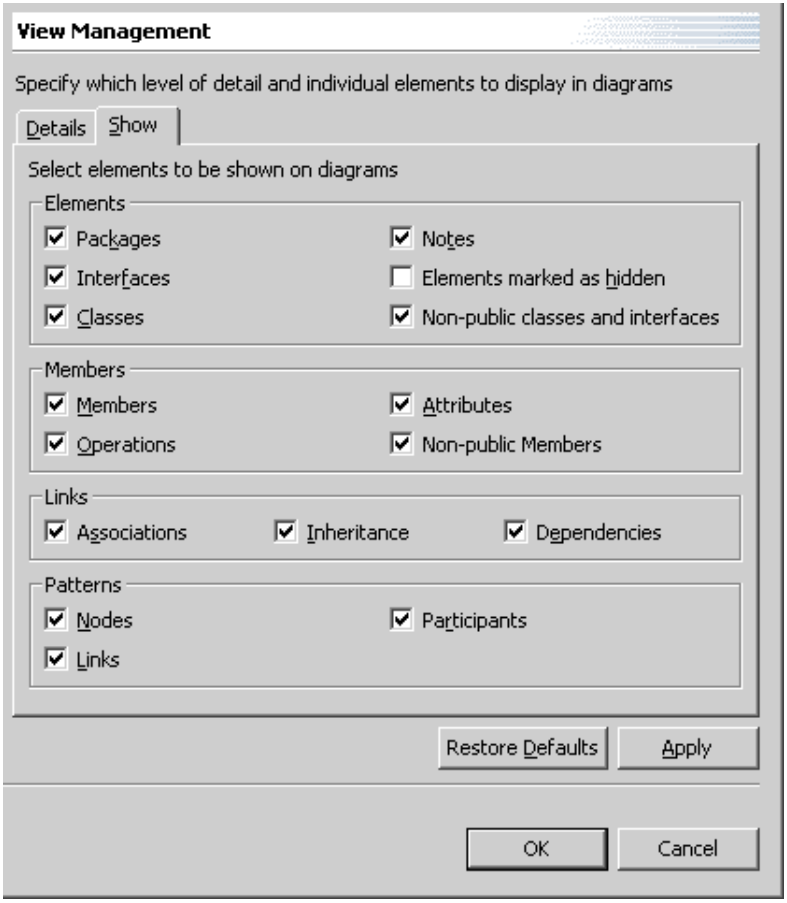
The UML Explorer has filters to show or hide information in the tree. You can also hide and show elements on diagrams. The following example demonstrates how to hide and show information on the Library class diagram created earlier in this guide.

To hide/show elements:

1. Right click on the `Library` class node in the diagram and choose **Hide/Show > Attributes**. The `Library` attributes disappear from the diagram.
2. To show the hidden attributes, right click the `Library` node and choose **Hide/Show > Attributes**.
3. To hide the entire `Library` class, right click the `Library` node and choose **Hide**.
4. To show the `Library` class, right click on the diagram background and choose **Hide/Show**. The Show Hidden dialog opens.
5. Click **Library** on the Hidden Elements column, then click **Remove**. Library moves to the Diagram Elements column.
6. Click **OK**. The `Library` class appears on the diagram.

For global control over the views, you can use the filters in **Window > Preferences > Modeling > View Management > Show** as shown in Figure 34.

Figure 34 Preferences dialog for view management



The filters shown in Figure 34 are global filters. To specifically filter classes, you can uncheck **Members**. This results in disabling the Attributes, Operations and Non-public Members filters.

Running quality assurance

You can run quality assurance (QA) on your project to check the quality of your code against a set of predefined measurements.

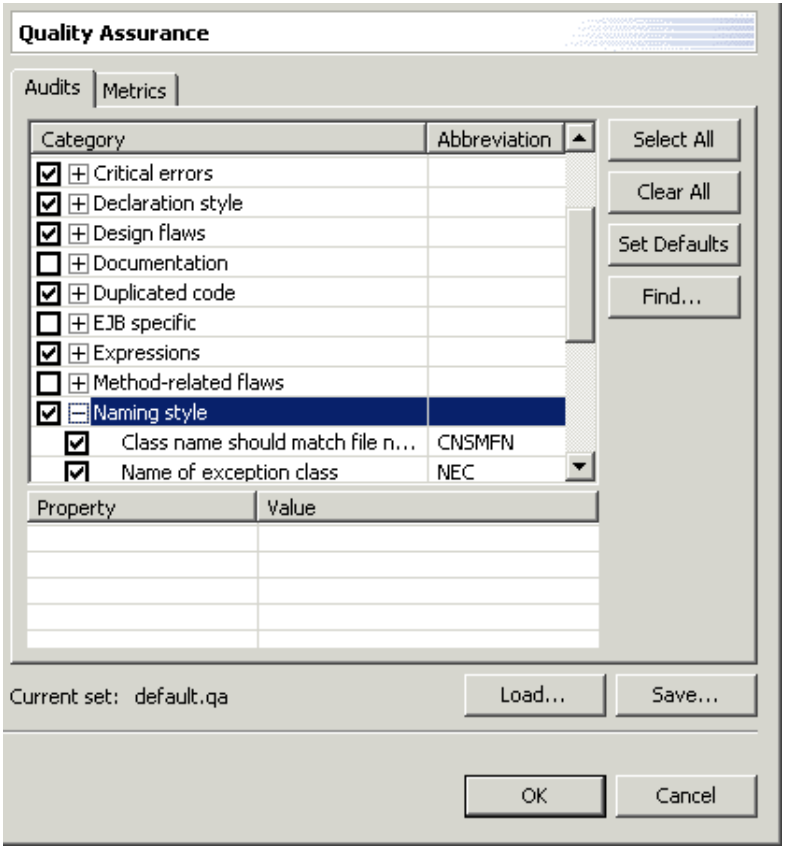
Running audits

To run audits on the Library project:

1 In the UML Explorer, right click on the Library class diagram and choose **Quality Assurance > Audits**.

2 The Run QA, Audits Execution dialog opens. Click Preferences. The Audit preferences dialog opens as shown in Figure 35. You can use this dialog to choose the specific audits you want to run.

Figure 35 Audit preferences



3 Click **OK** to accept the preferences. Click OK again to run the audits. Together generates the audits in the Audits view, as shown in Figure 36.

Figure 36 Audits view (your results may differ)

Audit

Description	!	Resource	In Folder	Location	Target
Field 'accessLevel' is not used	⚠	Librarian.java	/Library	line 9	Line
Field 'librarianID' is not used	⚠	Librarian.java	/Library	line 13	Line
Field 'name' is not used	⚠	Librarian.java	/Library	line 11	Line
Importing a type from the java.lang package	⚠	Librarian.java	/Library	line 1	Line
Field 'address' is not used	⚠	Library.java	/Library	line 20	Line
Field 'branchNumber' is not used	⚠	Library.java	/Library	line 18	Line
Field 'hours' is not used	⚠	Library.java	/Library	line 16	Line
Field 'lnkLibrarian' is not used	⚠	Library.java	/Library	line 12	Line
Field 'phone' is not used	⚠	Library.java	/Library	line 14	Line
Field 'titleList' is not used	⚠	Library.java	/Library	line 10	Line
Importing a type from the java.lang package	⚠	Library.java	/Library	line 1	Line

4 In the Audit view, double click the second entry for *Field 'librarianID' is not used*. Together opens the source in the Java Editor, and highlights the line as shown in Figure 37.

Figure 37 Java Editor showing problematic source code

Welcome | Library.java | Librarian.java

```
    private String librarianID;

}
```

5 From the Audit view, right click *Field 'librarianID' is not used* and choose **Description.** This opens a description of the Field is Not Used audit.

6 Click the Synchronize with help view button.

7 Select an audit failure with a different description and bring the Help window to the foreground. It contains the description for this audit.

Note As you run different audits, a history of your audits is stored in a list accessed by the down arrow next to the history button .

8 Go to the Library diagram, right click on the **Title** class and choose **Quality Assurance > Audits**. This restricts the audit to the `Title` class as shown in Figure 38.

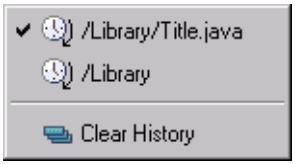
Figure 38 Audit view showing Title class

Audit

Description	!	Resource	In Folder	Location	Target
Field 'author' is not used	⚠	Title.java	/Library	line 16	Line
Field 'genre' is not used	⚠	Title.java	/Library	line 12	Line
Field 'publisher' is not used	⚠	Title.java	/Library	line 14	Line
Field 'title' is not used	⚠	Title.java	/Library	line 21	Line

9 Click on the down arrow on the history button to see the list of audits you can run as shown in Figure 39.

Figure 39 List of audits



A checkmark next to an audit indicates what is currently displayed in the audit table. This audit history exists during the lifetime of your current Together session.

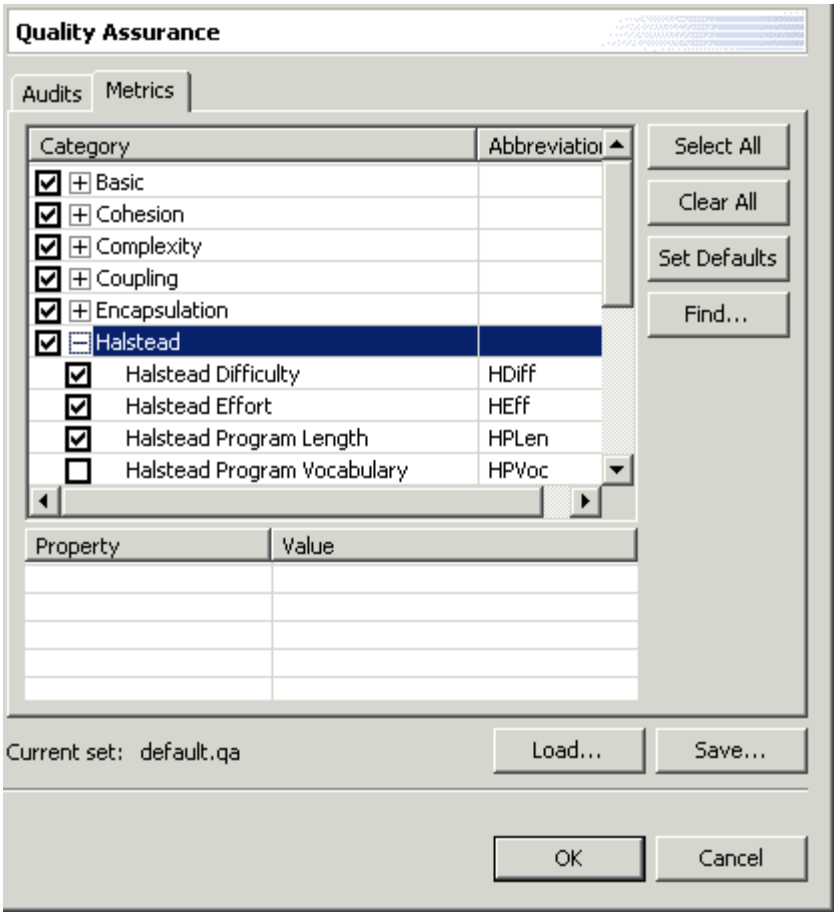
Running metrics

To run metrics on the Library project:

1 In the UML Explorer, right click on the Library project and choose **Quality Assurance > Metrics**.

2 The Run QA, Metrics Measurement dialog opens. Click **Preferences** to open the Metrics Preferences dialog as shown in Figure 40. Here, you can specify the metrics you want to generate.

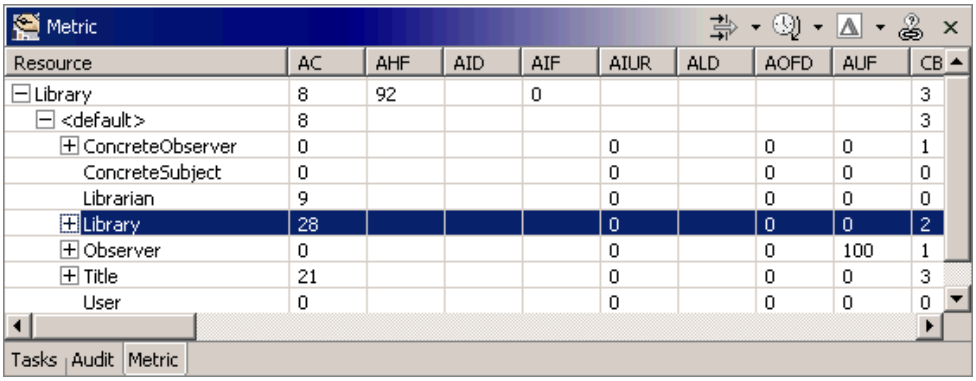
Figure 40 Metric preferences



3 Click **OK** to accept the preferences. Click **OK** again when you are ready to generate the metrics. This opens the Metric view showing the metrics generated.

4 Expand the Library tree to reveal the list of classes as shown in Figure 41.

Figure 41 List of classes



Resource	AC	AHF	AID	AIF	AIUR	ALD	AOFD	AUF	CB
Library	8	92		0					3
<default>	8								3
ConcreteObserver	0				0		0	0	1
ConcreteSubject	0				0		0	0	0
Librarian	9				0		0	0	0
Library	28				0		0	0	2
Observer	0				0		0	100	1
Title	21				0		0	0	3
User	0				0		0	0	0

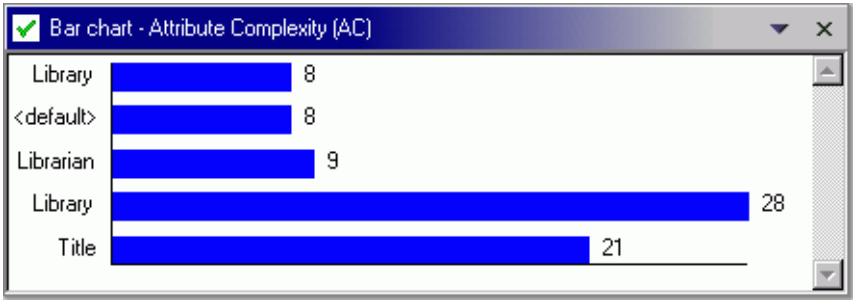
Tasks Audit Metric

5 Right click a metric column and choose **Description** to see more details.

Note The Metric table has a History and Synchronization button like the Audit table as described in "Running audits" on page 34. The Metric table also includes a Filters button so that you can display selected results only.

6 Right click in a column and choose **Bar Graph** to produce a bar chart of that column. Figure 42 shows the bar graph for the *AC* column.

Figure 42 Bar graph



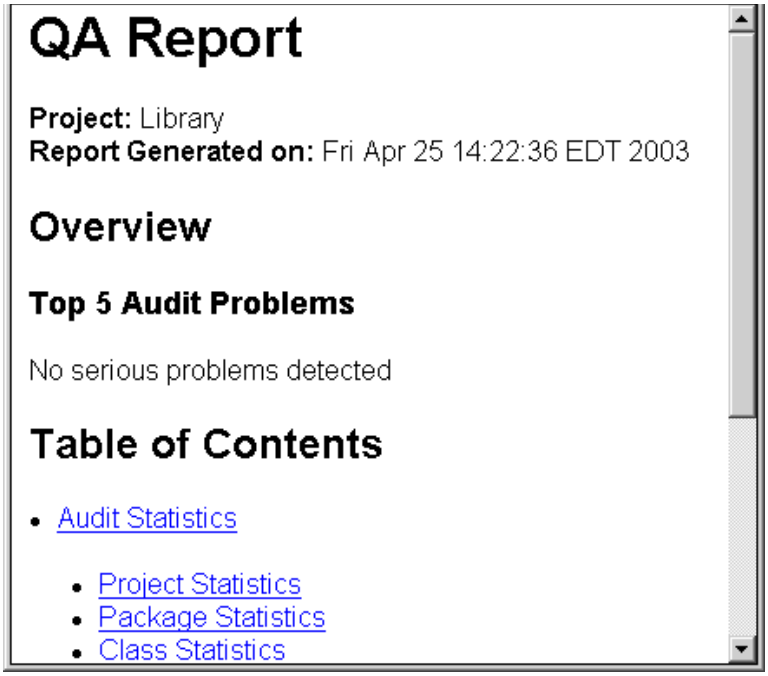
You can also choose **Kiviat Graph** to produce a graph based on the selected row.

7 To generate a report, right click on the table in the Metric view and choose **Generate Report**. This opens the Generate Report dialog.

8 Enter a location for saving the HTML report.

9 Check the option *Open in browser* and then click **OK**. Together generates the report, saves it, and opens it in a browser, as shown in Figure 43.

Figure 43 Sample QA report



Saving and loading metric results

To save metric results:

1 Right click on the table in the Metric view.

2 Choose **Save Metric Results**.

3 In the resulting dialog box, enter the name for the saved metrics and click **Save**.

To load metric results:

1 Right click on the table in the Metric view.

2 Choose **Load Metric Results**.

3 Choose the metrics to load from the resulting dialog box and click **Open**.

The results are added to the History list and Compare list.

Setting rules for quality assurance

When you run audits and metrics, Together runs a default set of rules through the source code.

To set rules for QA:

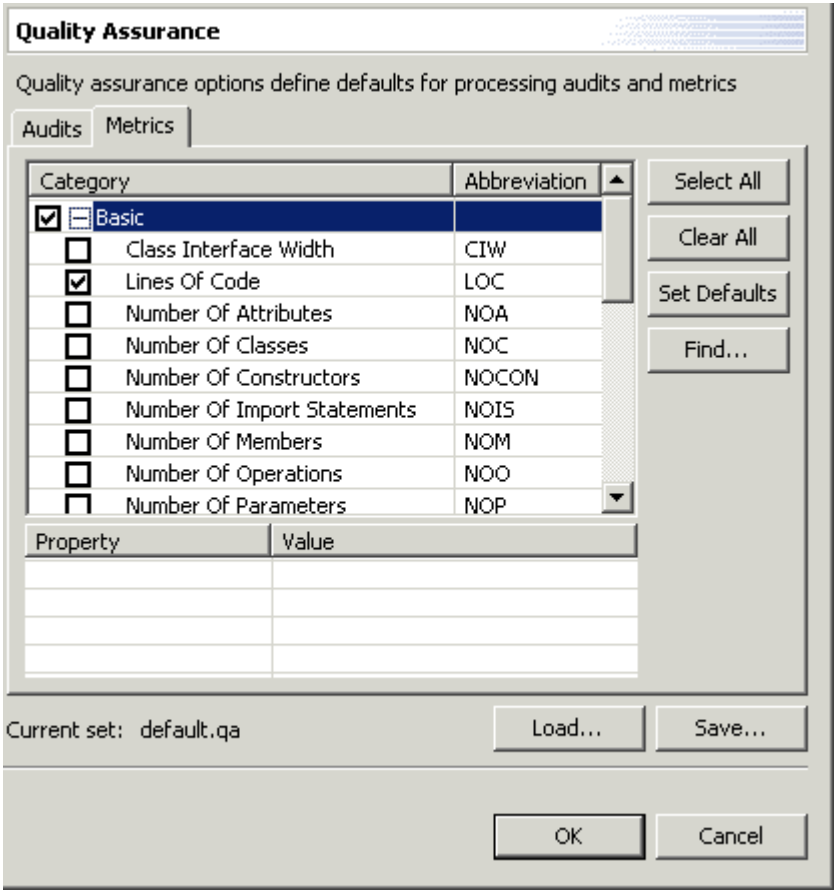
1 From the main menu, choose **Window > Preferences** to open the Preferences dialog.

2 Expand **> Quality Assurance** to view the QA preferences.

3 Choose the **Metrics** tab and click **Clear All**.

4 Expand the **Basic** metric and check **Lines of Code** as shown in Figure 44.

Figure 44 Metric preferences

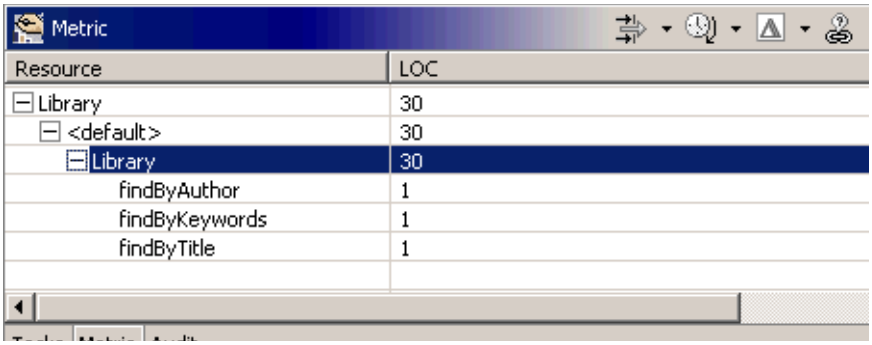


5 Click **Save**. You can create different sets of rules and distribute them to your team. If you have different sets, use **Load** to load the set you want to use.

6 Click **OK**.

7 Run the metrics on the `Library` class. In the diagram or in the UML Explorer, right click on the `Library` node and choose **Quality Assurance > Metrics**. The results are similar to those in Figure 45.

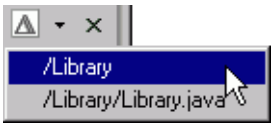
Figure 45 Results for saved set of rules (your numbers may differ)



Comparing metrics results

After you have run metrics on your project more than once, or loaded some existing metric results, the down arrow of the Compare With button contains a list of the results as shown in Figure 46.

Figure 46 Compare With menu on the Metric view



Compare With compares the current set of results with the one you choose on the menu.

In the previous sections, you modified the metrics to check the Lines of Code (LOC) and ran it on the `Library` class. For this example, you need to add another method to the `Library` class, a `getOverdueList` method.

To compare metric results:

1 Add the `getOverdueList` method to the `Library` class. Then run the metrics on the `Library` class. Figure 47 shows similar results.

Figure 47 Results of metrics for Library class with the new method (your numbers may differ)

Resource	LOC
Library	34
<default>	34
Library	34
findByAuthor	1
findByKeywords	1
findByTitle	1
getOverdueList	1

Tasks | Metric | Audit

2 Click the arrow next to the **Compare With** button and choose **/Library/Library.java** to compare these results to the previous metrics that you ran on the `Library` class.

The results compare the /Library/Library.java results similar to what is shown in Figure 48.

Figure 48 Comparing metric results (your numbers may differ)

Resource	LOC
Library (1)	30
Library (2)	34
<default> (1)	30
<default> (2)	34
Library (1)	30
Library (2)	34
findByAuthor	1
findByKeywords	1
findByTitle	1

In the example above, the LOC for `Library originally` had a value of 30. However, the latest metric results show the number of lines of code increasing to 34 (a combination of the `getOverdueList` code and blank line separators).

If you compare the latest metrics to the first metric result using all of the metric rules, only the Library row and LOC column are compared.

Generating documentation

Together features a UML documentation wizard that you can use to generate documentation for the Library project.

Before attempting to generate documentation, you should specify the location of Javadoc in the Together preferences.

To specify the location of Javadoc:

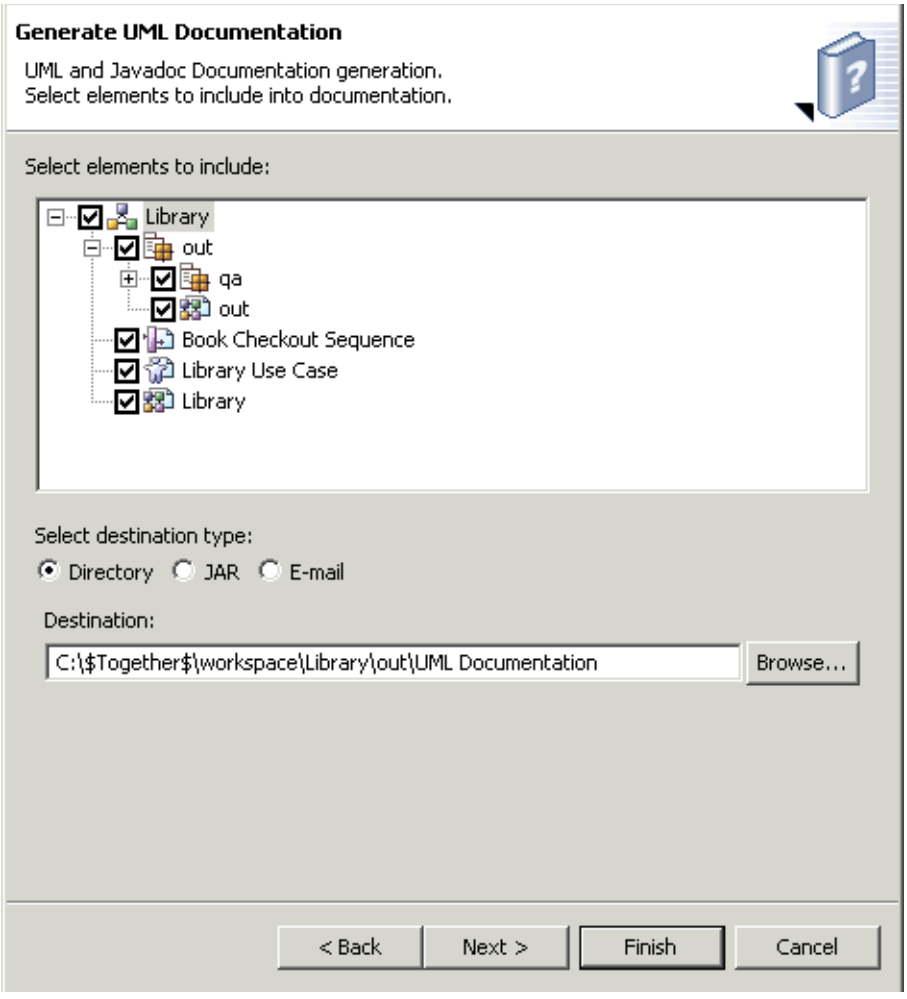
1. From the main menu, choose **Window > Preferences**.
2. In the tree on the left, choose **Java > Javadoc**.
3. Browse for the location of the Javadoc command.
4. Click **OK**.

When the Javadoc location is correct, you can generate project documentation.

To generate documentation for a project:

1. Choose **File > Export**. This opens the Export dialog.
2. Choose **UML Documentation** and click **Next**.
3. The first page of the UML Documentation wizard opens, as shown in Figure 49.

Figure 49 UML Documentation wizard showing project tree



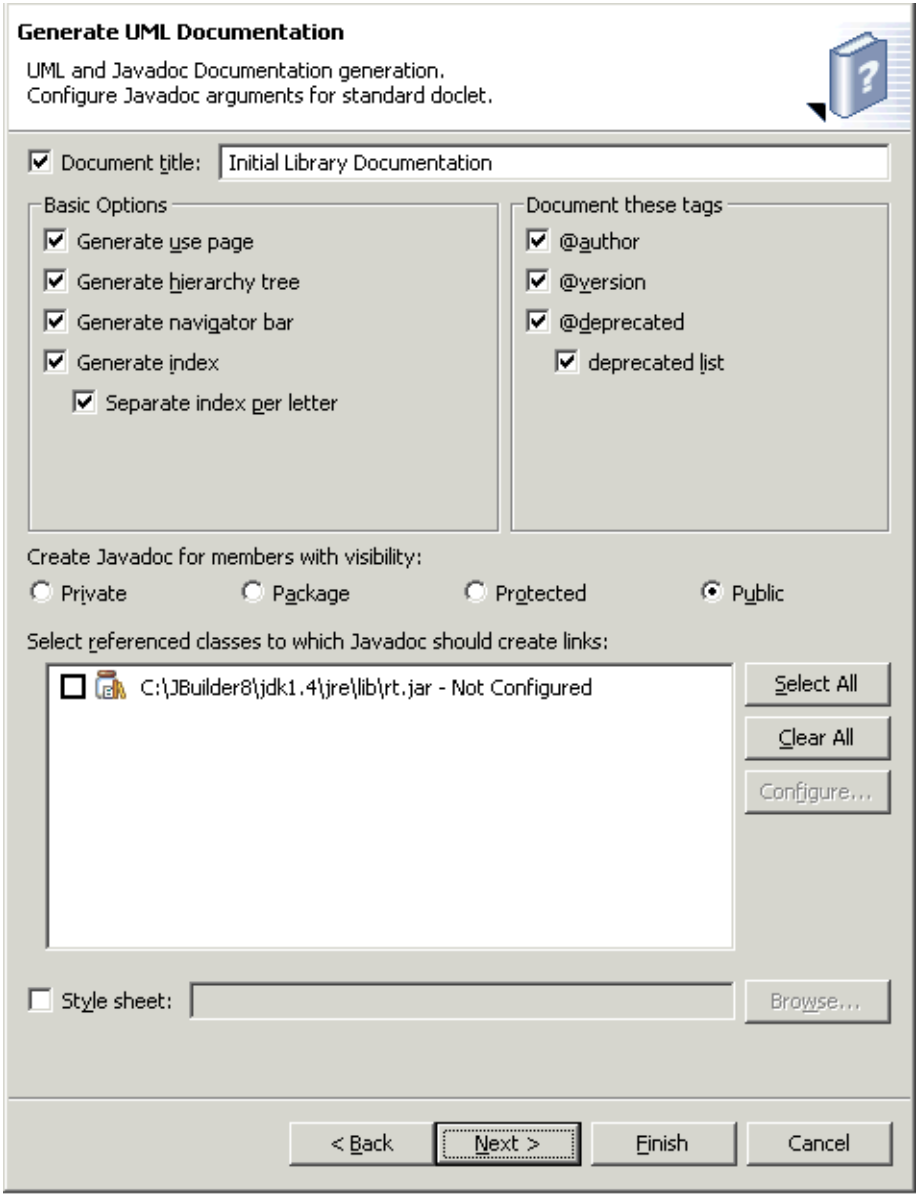
By default, the UML Documentation wizard generates documentation for your entire project. You can limit the scope of the documentation to a smaller set by expanding the tree at the top of this page and unchecking the parts you do not want.

4 On the first page of the wizard, you can accept the default name and path for the generated documentation or enter your own. Then click **Next**.

5 On this page, you can specify the content of the report, such as whether the report should include diagrams. Accept the defaults for a full report. Click **Next**.

6 Click the box next to the Document title field and enter “Initial Library Documentation” as shown in Figure 50. Leave the other options unchanged. (The location for “referenced classes” in the lower half of the dialog will differ from that shown in Figure 50.)

Figure 50 Documentation Wizard

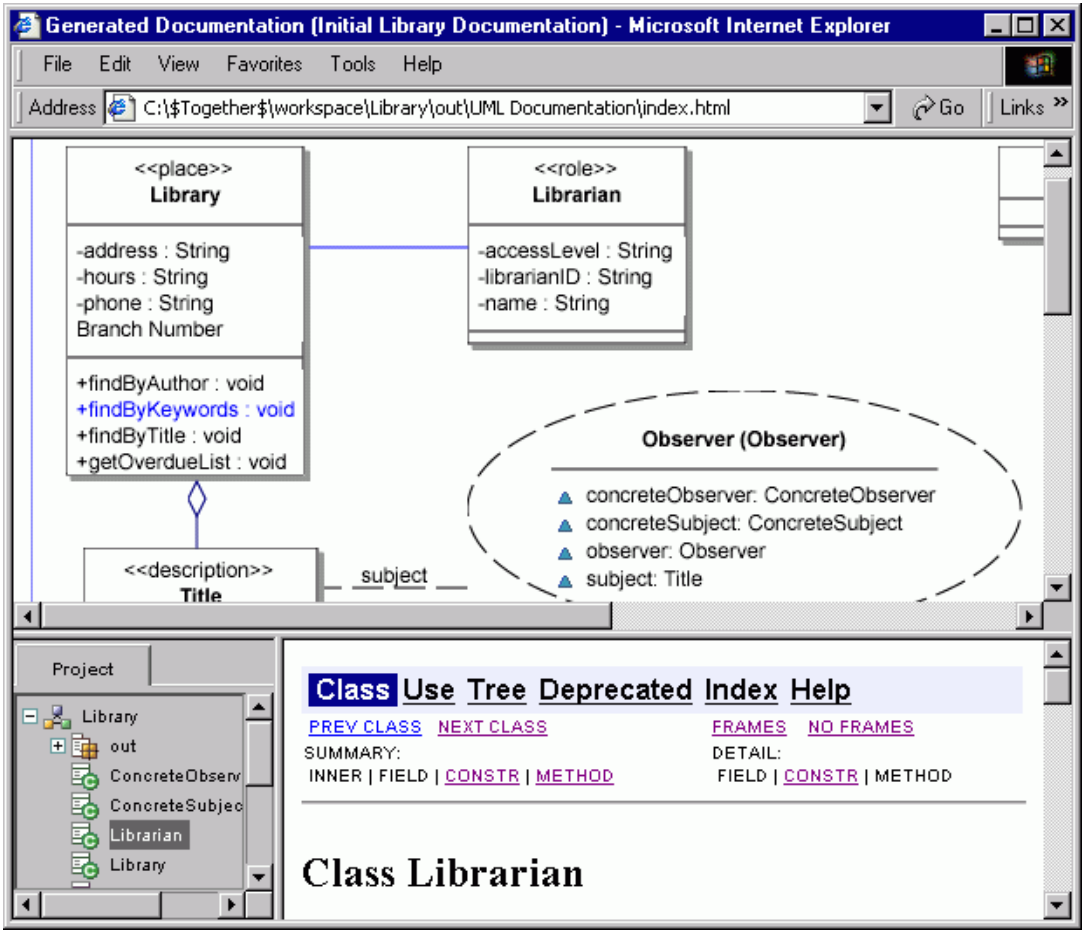


7 Click **Finish**.

8 When Together finishes generating the documentation, a dialog opens. Check *Open generated documentation in a browser,* and click **OK**.

9 The browser opens with a frameset to display the generated documentation. Expand the `Library` node in the tree in the lower left frame. Your browser displays a page similar to Figure 51. Notice that clicking a class name in the lower left frame opens the documentation in the lower right pane.

Figure 51 Resulting documentation



You can further explore the generated documentation with the following exercises

- Use the **Project** tab to navigate through the project.
- Click the `Library` class to display its corresponding documentation.
- Click `findByKeywords` to jump to that section in the documentation.

- Since the documentation is generated in the Javadoc style, use the hypertext to navigate through the documentation and access the index.

Note Because you opted to generate QA reports when you went through the UML Documentation wizard, the generated documentation contains the QA report among the project documentation. This documentation is stored within the Library project under the documentation directory you accepted or created in step 4. You can browse this documentation without Together.

- Close the documentation browser and reopen it by selecting **Navigate > Open UML Documentation** from the menu bar while the project is selected in the UML Explorer.

Exporting/Importing XMI projects

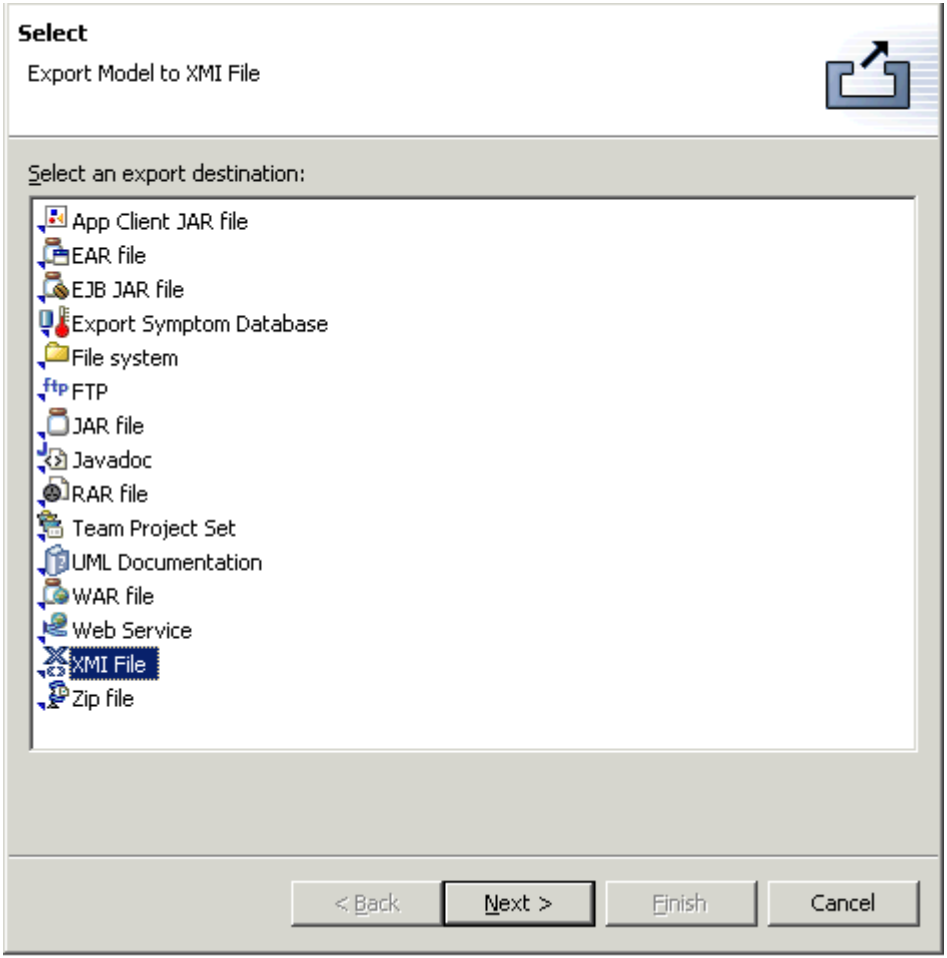
Together also features XMI import/export capability as a way of using external projects and of making them available outside Together.

Exporting XMI projects

To export a project as an XMI file:

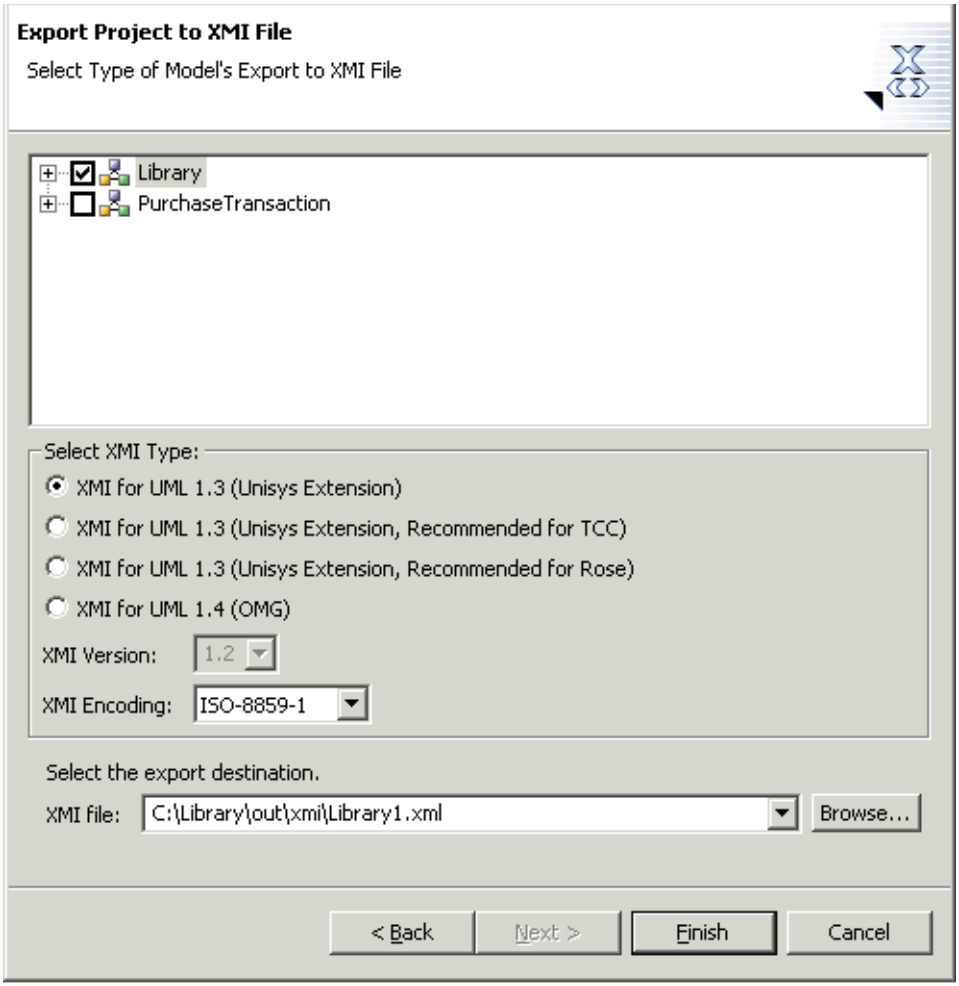
1. In the UML Explorer, right click on the Library project and choose **Export**.
2. In the Export wizard dialog, click **XMI File**. Click **Next**.

Figure 52 The Export wizard (Select dialog)



3 In the XMI Export dialog, click the box next to the Library project. You can also change the XMI Type information and destination of the XMI file. For this example, accept the defaults. Click **Finish**.

Figure 53 The XMI Export wizard



4 A confirmation dialog asks if you want to create the new directory. Click **OK.**

5 In the directory structure of your system, locate the new xml file: $Together$/workspace/Library/out/xmi./Library1.xml.

Importing XMI projects

Though this guide does not import an existing XMI file, the process is similar to exporting. Instead of **File > Export**, however, choose **File > Import** to begin the process. When the Export Wizard opens, choose **XMI file** from the list and follow the prompts from the wizard.

Setting UML profiles

Together uses profiles to support various extensions of UML. A single project can have multiple profiles. You can manage profiles through the project properties.

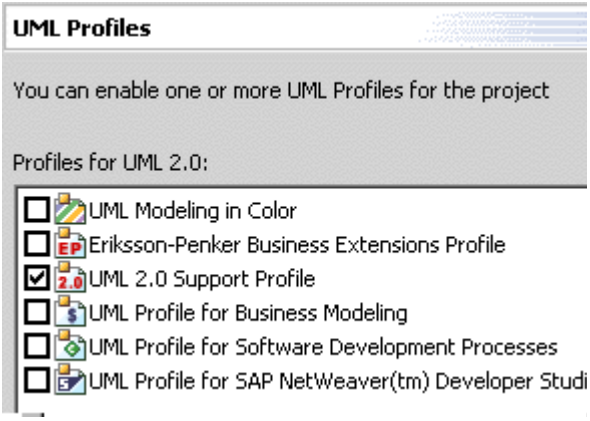
To enable the UML 2.0 profile:

1 From the UML Explorer view, right click on the project and choose **Properties**.

2 Click **UML Profiles**.

3 Check the **UML 2.0 Support Profile** option as shown in Figure 54, and click **OK**.

Figure 54 Properties for Library dialog (Your profile list may differ.)



The UML 2.0 Profile provides additional notation to the UML support by Together. The profile includes new notation for the following diagrams:

- activity
- component
- use case
- sequence
- state
- deployment

Note The UML 2.0 profile also provides a new collaboration diagram called the collaboration 2.0 diagram.

Creating a collaboration diagram

When you enable the UML 2.0 profile, all of the diagrams are available under the New Diagram menu.

To create a collaboration diagram:

1 In the UML Explorer, right click on the Library project and choose **New > UML Diagram**.

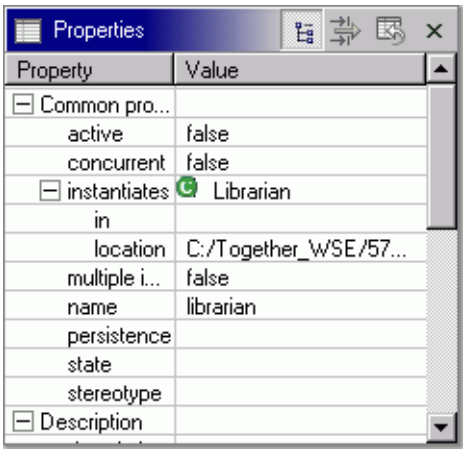
2 In the resulting dialog, choose **Collaboration Diagram** as the type. Name the diagram "Lending Books Interaction," and click **Finish**.

3 Click the Lifeline button and create a lifeline on the diagram. Name it "librarian."

4 In the Properties view, expand **instantiates** and click the Information button.

5 In the resulting Select Class dialog, add the `Librarian` class from the *Model Elements* column on the left to the *Selected* column on the right. Then click **OK**. The Properties view for this lifeline appears as shown in Figure 55.

Figure 55 Lifeline properties



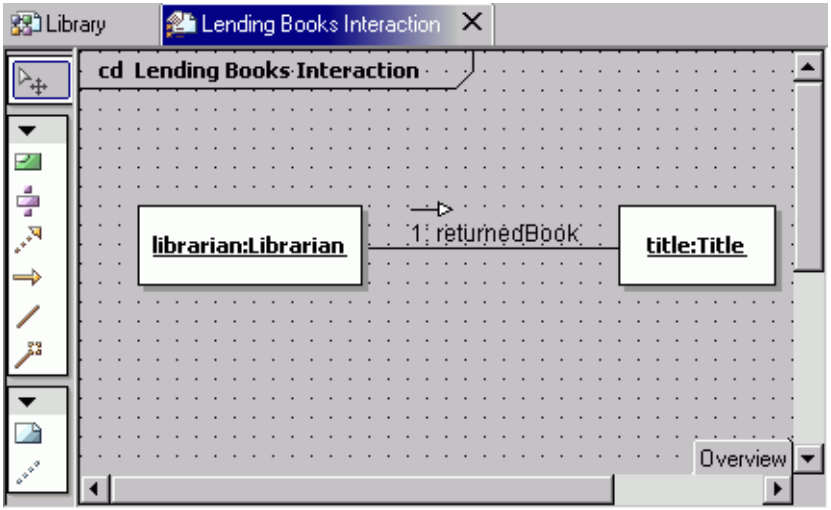
6 Create another lifeline element for `Title`.

7 Click the InterLifelineconnector button and connect the `Librarian` lifeline to the `Title` lifeline.

8 Double click on the message label on the connector to activate the in-place editor. Rename the message "returnedBook."

Your collaboration diagram should be similar to Figure 56.

Figure 56 Collaboration diagram



This product includes software developed by the Eclipse Project (http://www.eclipse.org).