

Getting Started with

Borland® Together® Edition for Eclipse

(version 7.0)

Borland® Together®

Integrated and Agile Design Solutions

Borland®
Excellence Endures™

Borland Software Corporation
100 Enterprise Way
Scotts Valley, California 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. The furnishing of this document does not give you any license to these patents.

Copyright © 1998-2004 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. IBM and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both. All other marks are the property of their respective owners.

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

Updated: November 12, 2004

Contents

| | | | |
|---|----|--|----|
| About this guide | 5 | Specifying options for sequence diagram generation. | 33 |
| Navigating around TogetherEC | 5 | Converting between sequence and collaboration diagrams | 35 |
| Viewing other TogetherEC perspectives. | 7 | Using the Properties view | 35 |
| Views associated with each Together Perspective. | 8 | Using the Overview | 36 |
| Activating Together Capabilities | 9 | Using and creating patterns | 37 |
| Creating a project | 10 | Managing diagram views | 40 |
| Working with the Diagram View. | 11 | Specifying a detail level. | 41 |
| Working with the Diagram View Toolbar | 12 | Hiding/Showing Information | 41 |
| Using the Tools Palette. | 13 | Running quality assurance. | 43 |
| Hiding and showing the tools palette | 13 | Running audits | 43 |
| Hiding and showing groups of diagram elements | 14 | Running metrics | 45 |
| Scrolling in the tools palette | 16 | Saving and loading metric results | 48 |
| Creating a use case diagram | 16 | Setting rules for quality assurance. | 49 |
| Using a class diagram | 20 | Comparing metrics results | 50 |
| Adding methods and attributes | 21 | Generating documentation | 51 |
| Creating relationships and links | 24 | Exporting/Importing XMI projects | 55 |
| Creating hyperlinks | 26 | Exporting XMI projects | 55 |
| Creating a sequence diagram. | 27 | Importing XMI projects. | 56 |
| Creating the initial sequence | 27 | Creating a Project from an MDL model | 56 |
| Associating an object with a class | 28 | About Path Aliases | 58 |
| Resolving sequence diagram problems | 31 | MDL Import Notes | 59 |
| Generating a sequence diagram from existing source code | 33 | Setting UML profiles | 59 |

About this guide

The goal of this guide is to provide you with a basis for understanding and using Together® Edition for Eclipse (TogetherEC or Together). It explains how to create a sample project and then uses the project to show how to use 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 *TogetherEC User Guide* in the product's online help.

This guide provides examples for:

- Navigating around TogetherEC
- Activating Together Capabilities
- Creating a project
- Working with the Diagram View
- 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
- Creating a Project from an MDL model

Navigating around TogetherEC

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

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

To open the Together Architect perspective and show the Diagram view:


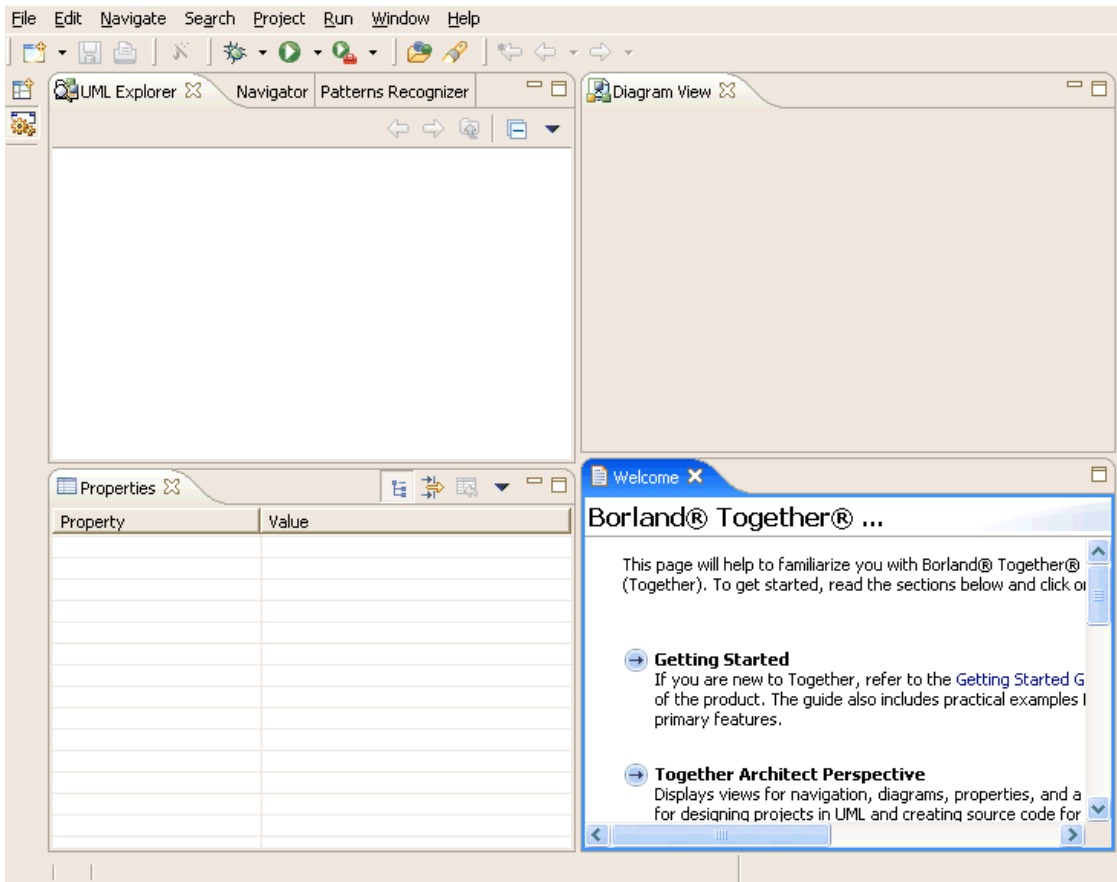
- 1 Choose **Window > Open Perspective > Other > Together Architect**. If the Architect perspective is open but not active, click the Together Architect button  on the vertical toolbar at the left of the window.
- 2 Choose **Window > Show View > Diagram View**. The Architect perspective is displayed as shown in Figure 1.

Figure 1 Architect Perspective






The Together Architect 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 Eclipse 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** – 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

Viewing other TogetherEC perspectives

TogetherEC changes the user interface according to how you want to work with TogetherEC by providing four TogetherEC Perspectives to customize the TogetherEC-user experience. You can choose one of the following TogetherEC Perspectives:

-  Together Analyst
-  Together Architect
-  Together Developer

Together Architect is the default perspective. For more information, see “Navigating around TogetherEC” on page 5.

Note The UI elements shown may look different from those shown in this guide depending upon the actual set of enabled capabilities. For more information, see “Activating Together Capabilities” on page 9.

To choose a Together Perspective:

- 1 Choose Window > Open Perspective > Other. The Select Perspective dialog opens.
- 2 Select one of the Together Perspectives from the list, and click OK.

After you select a perspective, TogetherEC automatically customizes the UI to provide ready access to only the relevant elements of the UI, and to show only the information in the model that best supports the chosen perspective. UI elements and/or model information that are not generally relevant to the perspective are hidden. You can still access hidden information by changing the relevant configuration options and restoring hidden panes manually, but you may find it easier to just switch perspectives.

Important TogetherEC now supports activatable capabilities. A capability is a feature of TogetherEC, such as audits and metrics, generating documentation, XMI import/export, generating sequence diagrams, and so on. Enabling a specific Together Perspective does not automatically enable all of the associated menu items and features associated with that perspective, you must activate the associated capability for the feature. For more information, see “Activating Together Capabilities” on page 9.

Views associated with each Together Perspective

The views associated with each Together Perspective vary according to the perspective selected. The views that make up each Together Perspective are described below.

Together Architect Perspective

The Together Architect Perspective is the default perspective for TogetherEC. For more information, see “Navigating around TogetherEC” on page 5.

Together Analyst Perspective

The Together Analyst Perspective provides the following main areas:

- **Navigation view:** By default, this area contains the following tab:
 - **UML Navigator:** UML tree of projects that you can use to manage elements, open, create, delete, etc.
- **Diagram view:** Displays created and opened diagrams. When using multiple diagrams, this view provides a tab for each diagram.
- **Properties view:** This view displays the properties for a selected element. The properties for each element are usually divided into four categories:
 - **Description:** Textual description of element
 - **Hyperlinks:** Links to other elements or external files/documents
 - **Common properties:** UML properties
 - **Requirements:** Requirement information
- **Problems view:** Shows build problems that are either user-created or generated during the build process.
- **Tasks view:** Shows tasks (reminders) that are either user-created or generated during the build process.
- **Search view:** Displays the results of a search.

Together Developer Perspective

The Together Developer Perspective has the following main areas:

- **Navigation view:** By default, this area contains the following tabs:

- **UML Explorer:** UML tree of projects that you can use to manage elements, open, create, delete, etc.
- **Navigator:** Standard view
- **Diagram view:** Displays created and opened diagrams. When using multiple diagrams, this view provides a tab for each diagram.
- **Properties view:** This view displays the properties for a selected element. The properties for each element are usually divided 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
- **Problems view:** Shows build problems that are either user-created or generated during the build process.
- **Tasks view:** Shows tasks (reminders) that are either user-created or generated during the build process.
- **Search view:** Displays the results of a search.

Activating Together Capabilities

TogetherEC enables users to select specific user *Perspectives* or *Roles* (Analyst, Architect, Developer). When selecting a role, available views for the corresponding perspective are updated accordingly in the user interface. TogetherEC also allows users to activate or deactivate certain TogetherEC features or "capabilities." By default, most of the Together Capabilities are deactivated.

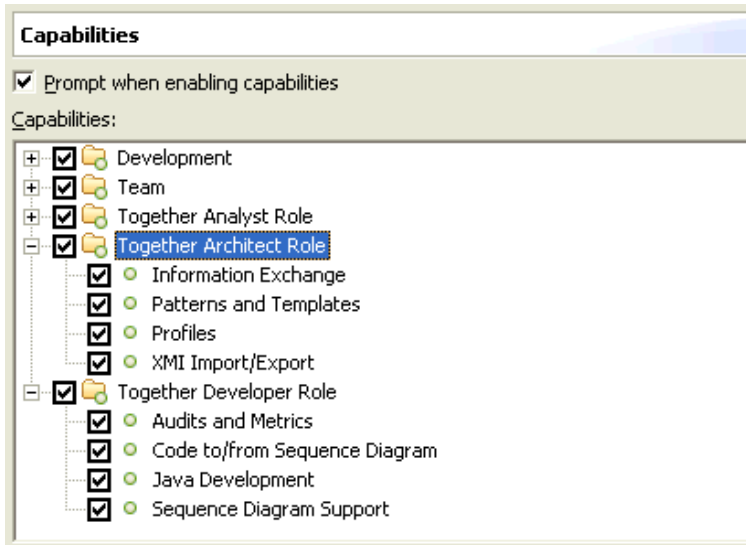
To enable Together Capabilities:

- 1 Choose **Window > Preferences** from the main menu. The Preferences dialog opens.
- 2 Expand "Workbench," and select "Capabilities" from the treeview list on the left of the Preferences dialog.
- 3 Expand the available Together Roles to view the capabilities available for each role.
- 4 Make your desired selections, and click **OK**.

Important For the remainder of this guide, we will be using most of the Together Capabilities. It is necessary to activate all of the features in the *Together Analyst, Architect, and Developer Roles* to follow this guide.

The Together Architect and Together Developer capabilities are shown in Figure 2.

Figure 2 Together Architect and Developer capabilities



Creating a project

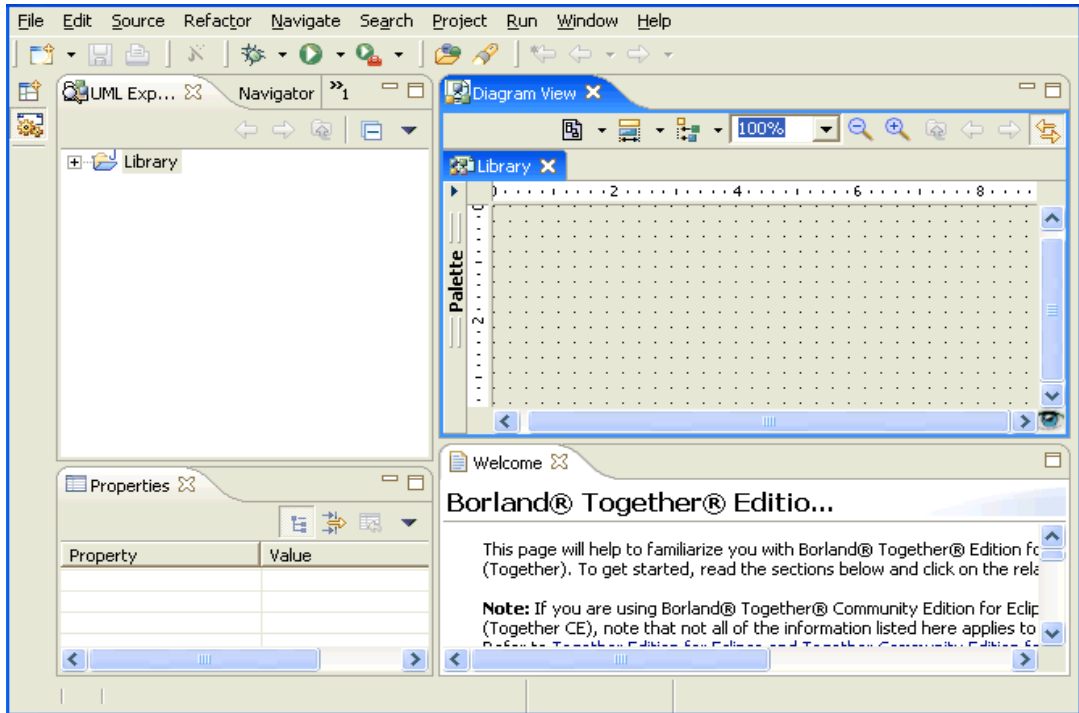
This guide models a simple library to explore the various features of TogetherEC.

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 3.

Figure 3 Library project as viewed from the Architect perspective



Working with the Diagram View

The Diagram View is visible when there is an open project. The Diagram View enables you to:

- Create new diagrams
- Draw diagram elements
- Open and close diagrams
- Zoom in or out on the diagram
- Apply layout formatting
- Resize diagram elements
- Navigate between open diagrams or open a parent diagram
- Manage diagrams and diagram elements using context menus

Tip Explore the context menus for the diagrams and diagram elements as you encounter them. The context menus make Together-specific and underlying IDE features available for diagrams and diagram elements.

Working with the Diagram View Toolbar

The Diagram view toolbar lies across the top of the Diagram view. The toolbar enables you to:

- Create new diagrams
- Resize diagram elements
- Align diagram elements (left, center, right, top, middle, bottom)
- Zoom in or out on the diagram
- Open the parent diagram
- Navigate between open diagrams
- Link code-generating elements to source code files

The table below shows the available toolbar buttons for the Diagram View and their associated tasks.

Table 1 Diagram View toolbar buttons and tasks




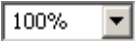






| Use this button to... | ...perform this task |
|---|---|
|  | Create a new diagram |
|  | Match the selected elements' width and height to the first selected element's width and height. |
|  | Reposition selected diagram elements (left, center, right, top, middle, bottom). |
|  | Set the desired magnification for the diagram. |
|  | Zoom out. |
|  | Zoom in. |
|  | Open the parent diagram. |

Table 1 Diagram View toolbar buttons and tasks

| Use this button to... | ...perform this task |
|---|---|
|  | Return to the previously-opened diagram. |
|  | Open the next diagram (first reached from the currently-opened diagram). |
|  | Synchronize source-generating diagram elements with their underlying source code and open the editor. |

Tip For more information on working with the Diagram View, refer to the online help user guide that ships with Together. From the main menu, choose **Help > Help Contents**, and select *Borland Together Edition for Eclipse User Guide* from the Contents pane.

Using the Tools Palette

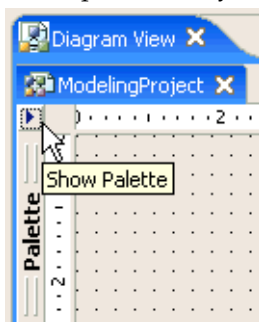
The tools for constructing diagram elements lie on a tools palette on the left side of the Diagram View. You use the tools to draw diagram elements on the Diagram View to represent the structural and behavioral elements and interactions of your model.

Hiding and showing the tools palette

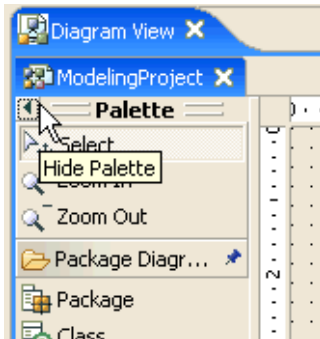
By default, the tools palette is hidden from the Diagram View.

To show or hide the tools palette:

- 1 To show the tools palette, click Show Palette as shown in the figure below. This action permanently displays the tools palette in the Diagram View.



- 2 To restore the tools palette to its default, hidden position, click Hide Palette as shown in the figure below.



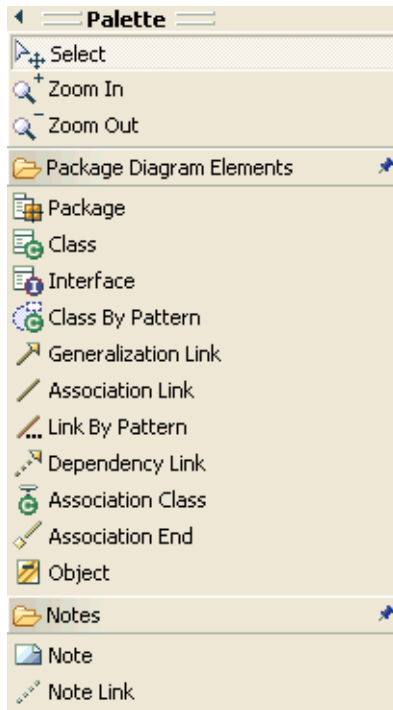
- 3 Before continuing with the remainder of this guide, be sure that the Tools Palette is showing on the diagram.

The Diagram View tools palette varies according to the type of focus diagram. These tools include principal elements defined by the UML for the focus diagram type as well as Together-specific enhancements (such as Class by Pattern in the class diagram). Note and Note Link elements are common to all diagram types. Tool tips display as you hover your cursor over the various diagram elements.

Hiding and showing groups of diagram elements

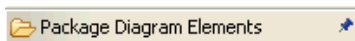
Once the tools palette is open, you can selectively hide or show the sets of diagram elements using the Pin Open button or by clicking on the main heading for a list of toolbar elements. In the figure below, the Package/Class Diagram Elements and Note Elements display on the tools palette.

Figure 4 Package/Class diagram elements



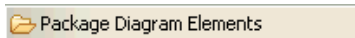
To hide or show a set of diagram elements on the tools palette:


- 1 Click the heading for the specific group of elements that you want to hide. For example, in the figure shown above, you would click



to hide all of the elements below that heading.

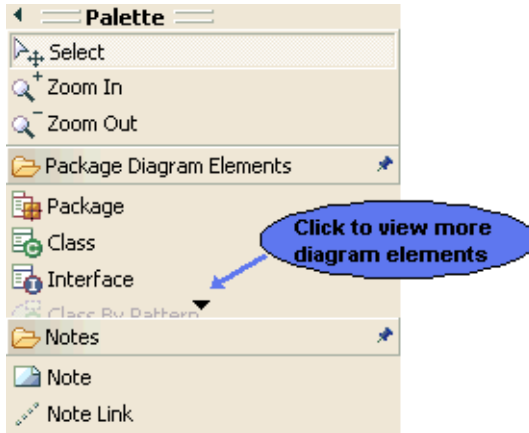
- 2 To show these hidden elements on the tools palette again, click



Tip You can also use the Pin Open button  to permanently display groups of specific diagram elements in the Diagram View.

Scrolling in the tools palette

If the Diagram View is smaller than the available height required to show all of the diagram elements in the tools palette, arrows display (▲ , ▼) on the tools palette to indicate more diagram elements are available. Use the arrows (▲ , ▼) to scroll through the available diagram elements. The figure below indicates that more Package Diagram Elements are available.



Creating a use case diagram

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

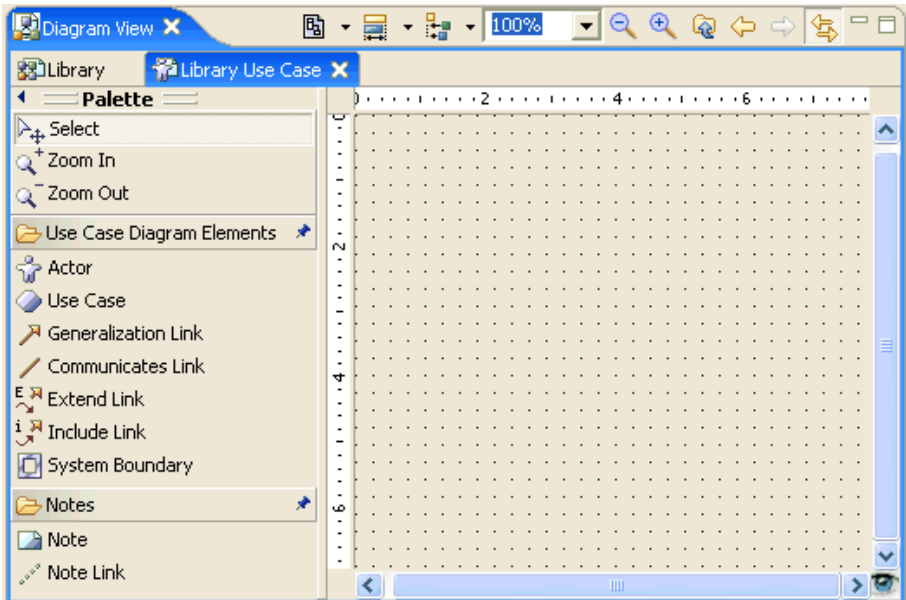
To create a use case diagram:

- 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 TogetherEC.

The Library Use Case tab appears as shown in Figure 5.

Figure 5 Library Use Case tab







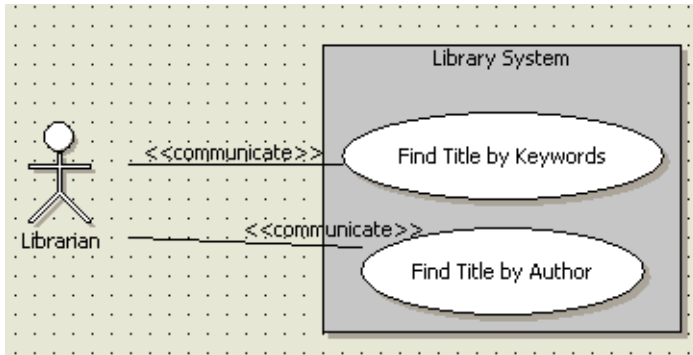
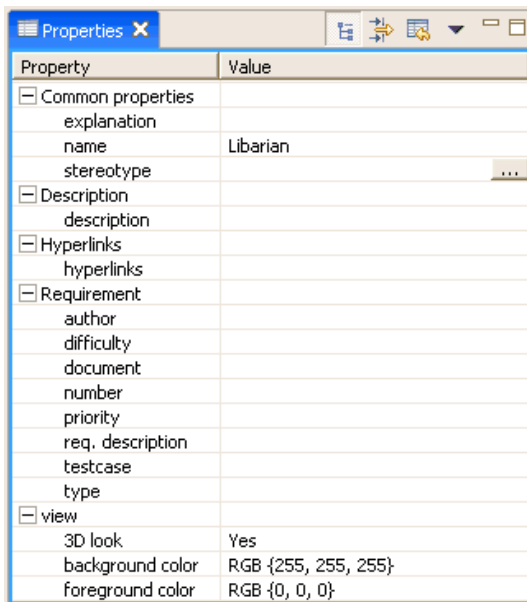
- 3 From the Diagram View Palette, click on the Actor button , and then click on the diagram background. Enter "Librarian" as the actor name.
- 4 Click the System Boundary button , and then click the diagram background to the right of the Librarian. Rename the system boundary "Library System."
- 5 Click 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 6.


Figure 6 Sample use case diagram



- 8 Select the **Librarian** actor to view the properties for Librarian in the Properties view as shown in Figure 7.

Figure 7 Properties view for Librarian



- 9 In the Properties view, click on the stereotype field. An information button  appears as shown in Figure 7. 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.



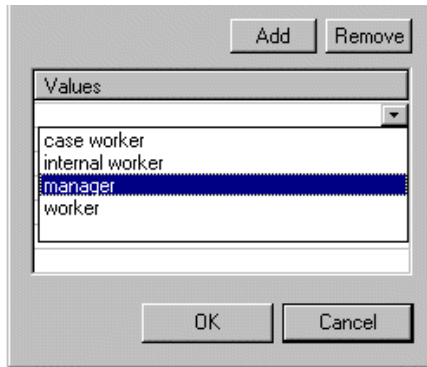
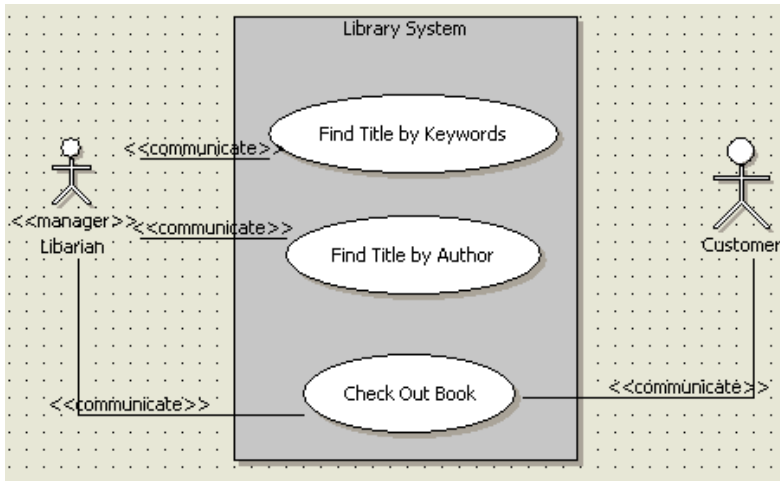
- 10 Click the information button  to open the Stereotype dialog shown in Figure 8.
- 11 Click **Add** and then click the drop down arrow . Select **manager** from the drop down list and click **OK**.

Figure 8 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 9.

Figure 9 Use case diagram for Library project



Using a class diagram

TogetherEC 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. Using the in-place editor, 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 10.

Figure 10 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 10. By default, TogetherEC opens class diagrams in this view. However, TogetherEC can also display icons in class and package diagrams as shown in Figure 11.

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 11 Class diagrams showing icons



Note For the remainder of this guide, we will work in the standard UML view shown in Figure 10.

As alternatives to the steps previously described, TogetherEC 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 2 lists the methods and attributes that you need to add to the classes you have created.

Table 2 Requirements for the Library sample project

| Class | Attributes | Methods |
|-----------|---|---|
| Library | address branchNumber hours phone | findByAuthor findByKeywords findByTitle |
| Librarian | librarian ID name accessLevel | |
| 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`.

Note By default, TogetherEC creates *private int* attributes and *public void* return methods. When you rename a class member on the diagram, you can also enter the type.

- 3 From the Library diagram, right click on the `Library` class and choose **New > Operation**.
- 4 Rename the operation `findByAuthor`.
- 5 Repeat the previous steps for the remaining attributes and methods for the `Library` class as listed in Table 2. Your `Library` class node should be similar to Figure 12.

Figure 12 Library class with attributes and methods

| |
|---|
| <<place>> Library |
| -address : String -branchNumber : String -hours : String -phone : String |
| +findByAuthor : void +findByKeywords : void +findByTitle : void |

- 6 Continue adding the attributes and methods listed in Table 2 to the `Librarian` and `Title` classes. For `getTitle`, specify **String** for type.
- 7 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*.
- 8 Select the `branchNumber` attribute of `Library`. In the Properties view, enter “Branch Number” as the alias value as shown in Figure 13.

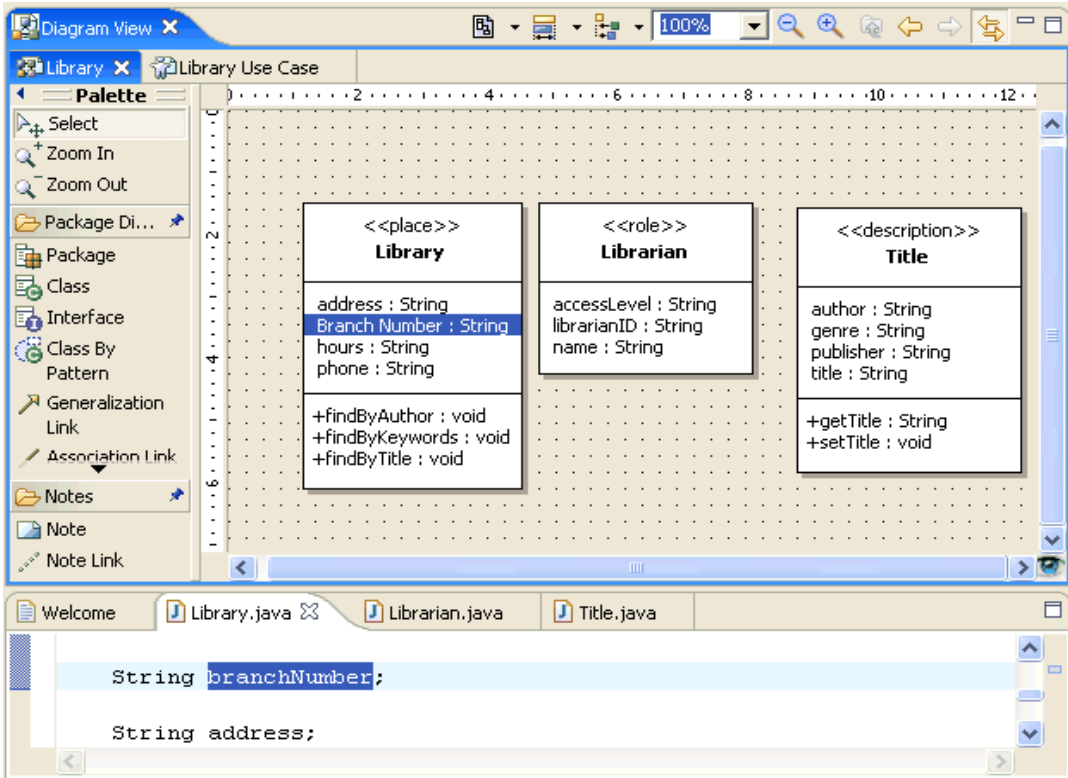
Note Some of the attributes in the table have names with spaces, which cannot serve as actual attribute names. For example, branch number becomes `branchNumber`. The *alias* property is useful for specifying a different name from that in the actual code.

Figure 13 Creating an alias for an attribute

| Property | Value |
|--|---------------|
| <input type="checkbox"/> Common properties | |
| alias | Branch Number |
| associates | |
| final | false |
| initial value | |
| name | branchNumber |
| static | false |
| stereotype | |
| transient | false |
| type | String |
| visibility | private |
| volatile | false |

The alias changes only the visual label for the attribute in the diagram, as shown in Figure 14. The attribute remains as `branchNumber` within the source code.

Figure 14 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 TogetherEC recognizes as a valid destination for dropping the end of the link.


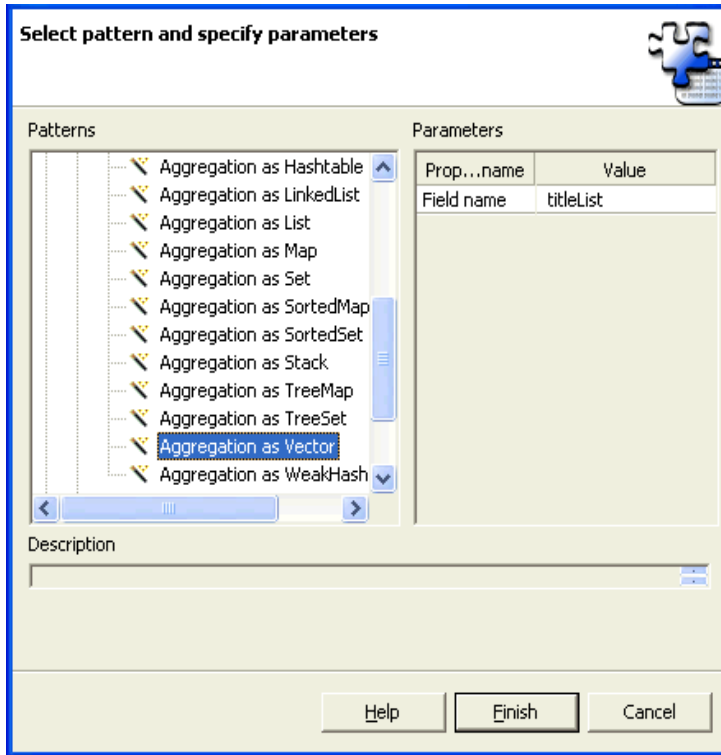
- 2 Click the **Link By Pattern** button . Click on **Library** and then drag-and-drop the link to **Title**. The Apply Pattern dialog opens.

Figure 15 Apply Pattern dialog



- 3 Choose **java.util Collections > Aggregations > Aggregation as Vector**. Rename the field name to "titleList" to indicate a collection as shown in Figure 15. Click **Finish**.

TogetherEC 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 37.

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.

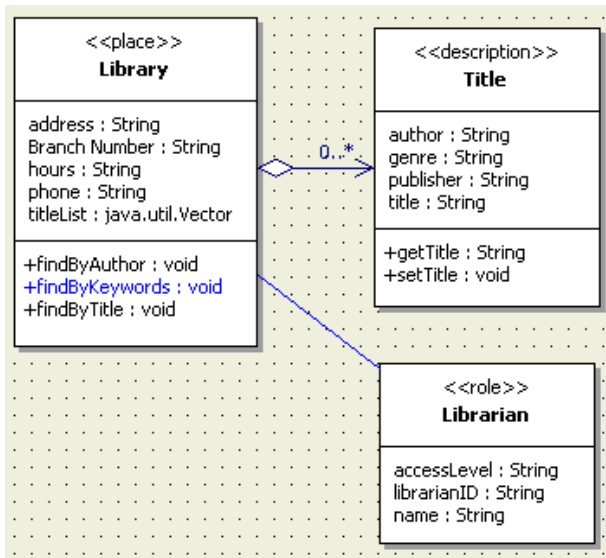
Note Creating hyperlinks is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

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 16. The corresponding use case is also highlighted in blue.

Figure 16 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.

- Note** Hyperlinks are bi-directional. The hyperlinked method on the class diagram is also highlighted in [blue font](#). You can navigate from the method on the class diagram to the use case diagram and vice versa using the Hyperlinks command for the hyperlinked element.
- 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, TogetherEC 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. TogetherEC provides three modes for creating sequence diagrams:

- A simple “sketch pad”
- Two-way with source code and classes
- Generating a sequence diagram from an existing method

- Note** Generating code to/from a sequence diagram is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

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:



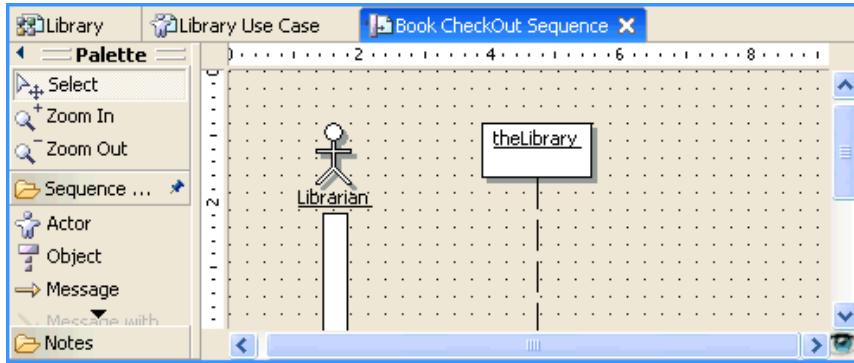
- 1 In the Library class diagram, right click the background of the project and choose **New Diagram > Sequence**.
- 2 Click the new diagram to select it. Then go to the Properties view and change the name of the diagram to “Book Checkout Sequence.”
- 3 From the Diagram view toolbar, click the Actor 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 17.

Figure 17 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 a 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).


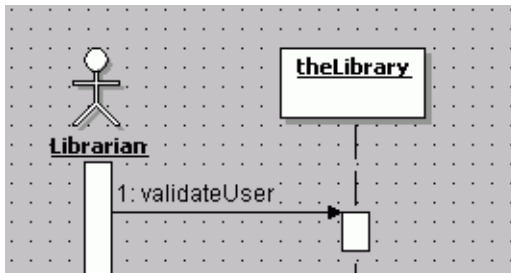
- 5 To add the first message, click the Message button  and then drag and drop from the Actor to theLibrary lifeline. The in-place editor activates. Rename the message "validateUser," as shown in Figure 18.

Figure 18 Validate User message



Associating an object with a class

In sequence or collaboration diagrams you can create associations between objects and classes (or interfaces).

Instantiated classes (or interfaces) for an object can be selected from the model, or the classes (or interfaces) can be created and added to the model. In the sequence and collaboration diagrams there are two commands available on the object right-click menu:

New that creates a new class or interface in the model, and

Select class that provides a list of available classes and interfaces.

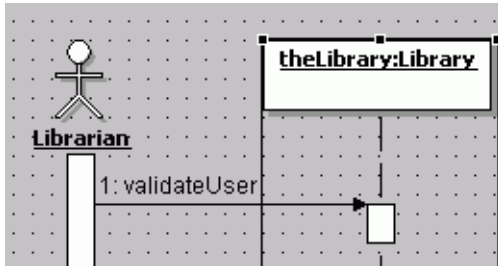
In this example, we will associate the Library object with the Library class.

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 19. Notice that the object name is followed by the name of the associated class.

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

Figure 19 An object associated with a class




- 2 Select the validateUser message and edit the Properties view with the following changes as shown in Figure 20:
 - Add the argument `ID`.
 - Add the return type `float` (in case any money is due).

Figure 20 Properties for validateUser message

| Property | Value |
|-------------------------------------|-------|
| <input type="checkbox"/> Properties | |
| arguments | ID |
| condition | |
| creation | false |
| destruction | false |
| iteration | |
| non-atomic delivery | false |
| return | float |
| return message | false |
| sequence number | 1 |
| synchronization | call |

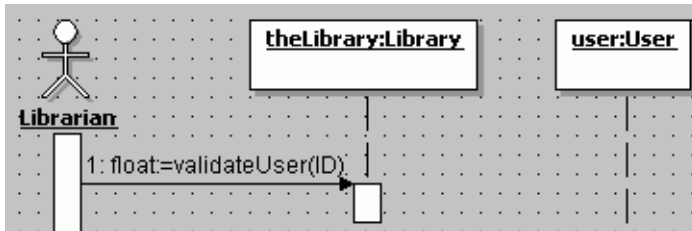
The diagram immediately updates to show the changes.

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**. Your sequence diagram should be similar to the one shown in Figure 21.

Figure 21 Sequence diagram with User class




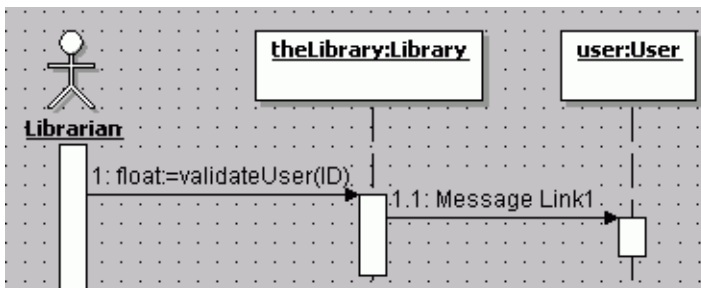
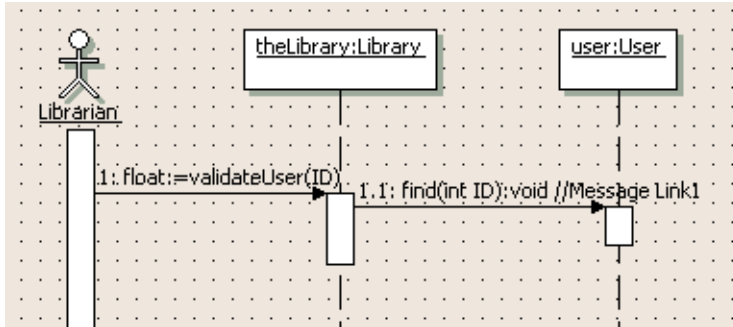
- 6 Add a new message (more specifically, a static finder or constructor). Instead of using “sketch” mode, create a real operation in the new `User` class. Click the Message button , and then drag and drop from the `theLibrary` activation bar to the `user` lifeline. A second message line appears as shown in Figure 22.

Figure 22 Sequence diagram with two message lines



- 7 Right click on the new message line and choose **New > Operation**. The New dialog opens.
- 8 For Name, enter “find” and then click **Finish**.
- 9 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
- 10 Return to the sequence diagram to see the updated parameter as shown in Figure 23.

Figure 23 Updated sequence diagram

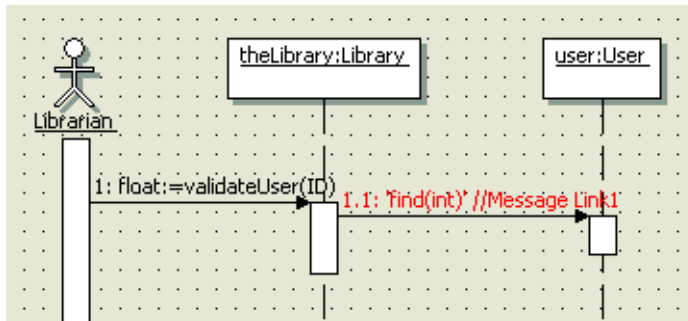


Resolving sequence diagram problems

The previous section demonstrated that TogetherEC 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 24. This signifies that the operation in the `User` class is no longer available.

Figure 24 Sequence diagram that requires resolution

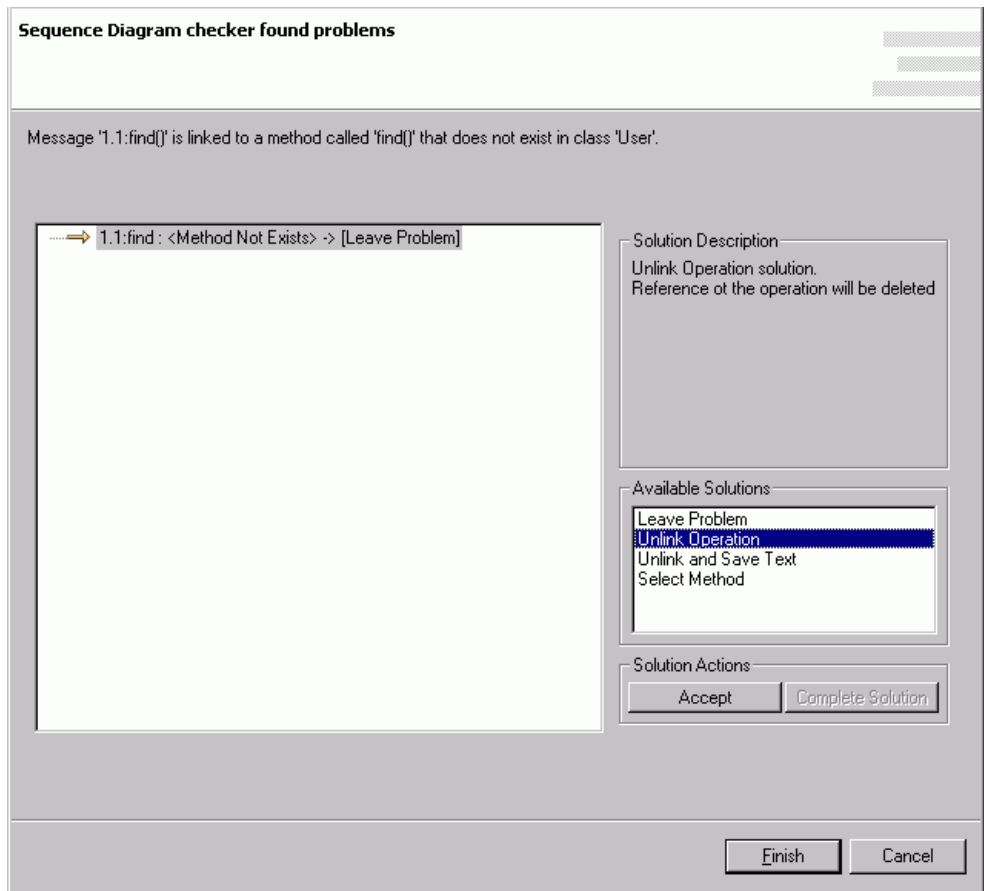


- 3 To resolve the sequence diagram, right click on the diagram background and choose **AutoFix** to display the Solve Problems dialog. The Solve Problems dialog indicates the problem, suggests a default solution, and lists other available solutions as shown in Figure 25.
- 4 Choose the **find** method.
- 5 From Available Solutions choose **Unlink Operation**.

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

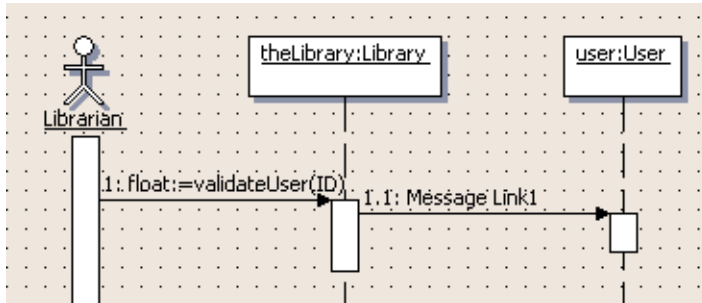
7 Click **Finish**.

Figure 25 Solve Problems dialog



The message link remains displayed but the operation is now unlinked as shown in Figure 26.

Figure 26 Sequence diagram with an unlinked operation



Generating a sequence diagram from existing source code

You can create sequence and collaboration diagrams and populate them using buttons in the Diagram view toolbar. You can also generate sequence diagrams from methods on a class diagram.

Note Generating sequence diagrams is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

In this example, we will generate a simple sequence diagram using the Library project.

To generate a sequence diagram:

- 1 Open the *Library default* class diagram.
- 2 Right click on the `findByKeywords` method on the *Library* class, and choose **Generate Sequence Diagram**.
- 3 The first page of the *Generate Sequence Diagram* wizard opens. Accept the default settings on this page, and click **Next**. The second page of the Generate Sequence Diagram wizard opens.
- 4 Accept the default settings in the second page of the Generate Sequence Diagram wizard, and click **Finish**. For a description of the Generate Sequence Diagram settings, see “Specifying options for sequence diagram generation” on page 33.

The sequence diagram is generated and opens in the Diagram view.

Tip To generate a collaboration diagram from an operation, first generate the sequence diagram and then convert the diagram into a collaboration diagram. For more information, see “Converting between sequence and collaboration diagrams” on page 35.

Specifying options for sequence diagram generation

The first page of the wizard allows you to specify options for the generated diagrams, such as:

- **Depth of call nesting:** During source code parsing, this value limits how deep the parser traverses the source code calling sequence. In other words, you may request a sequence diagram from the `main()` method of a complex application. This value can keep the generated sequence diagram from becoming so large that it is unusable.

In general, the higher the number, the greater the risk of producing a very complex sequence diagram. Also, in such cases, the length of time required to parse the source code increases.

Tip You may even want to use this feature as a means for quickly generating a high-level sequence diagram for a complex operation by using a low call nesting value (1-3). Then, based on the results in that diagram, you can choose to look deeper and/or create additional sequence diagrams for each of the major methods uncovered.

- **Exclude messages to self:** This option specifies whether to show messages to self on the generated sequence diagram.
- **Create multiple diagrams:** This option specifies how many sequence diagrams will be created if you select multiple operations before you invoke the Generate Sequence Diagram command. If this option is checked, multiple sequence diagrams are generated (one diagram for each operation). If this option is not checked, one diagram is generated with multiple initial messages.
- **Show multiple diagrams:** If you generated multiple sequence diagrams (see Create multiple diagrams above), this option specifies whether to show all generated diagrams immediately.

Tip If this option is not checked, you can still open any generated diagram from the UML Explorer or UML Navigator views. If you choose to create a hyperlink (see Create hyperlinks below), you can also use the Hyperlinks command on the context menu of the operation to open the diagram.

- **Create hyperlinks:** Check this option to create a Hyperlink from the selected operation to the generated sequence diagram. The operation displays in **blue font** to indicate that a hyperlink exists.

On the second page of the Generate Sequence Diagram wizard, you can select the packages and classes that you want to display in the generated diagram. All packages and classes are selected by default. However, some Java packages/classes may not be relevant, for example, `java.lang.integer`. You can increase the meaningfulness of the generated diagram by removing anything that does not help explain the sequence of operations.

Show on diagram/Show implementation options: In the Package/Class list, for those elements that you decide to show in the diagram, check whether to show implementation detail in the generated diagram.

Converting between sequence and collaboration diagrams

You can convert between sequence and collaboration diagrams. However, when you create a new diagram in TogetherEC, you must specify whether it is a sequence or collaboration diagram.

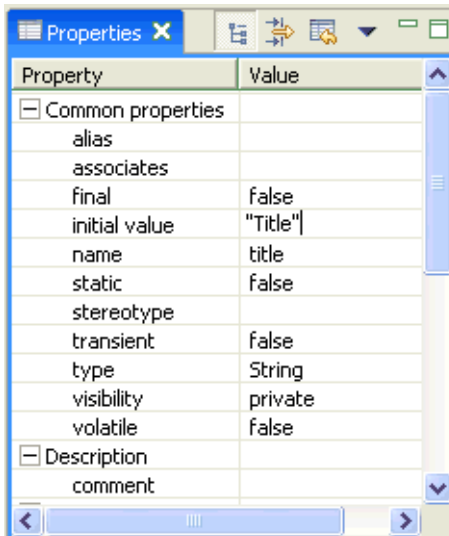
To convert the *Book Checkout Sequence* diagram to a collaboration diagram:

- 1 Open the *Book Checkout Sequence* diagram.
- 2 Right click on the diagram background, and choose **Show as Collaboration** from the context menu. The collaboration diagram displays.
- 3 To convert back to the sequence diagram, right click on the collaboration diagram background, and choose **Show as Sequence**.

Using the Properties view

You can use the Properties view to view and edit values of diagram elements such as alias, name, stereotype, and others as shown in Figure 27.

Figure 27 Properties view

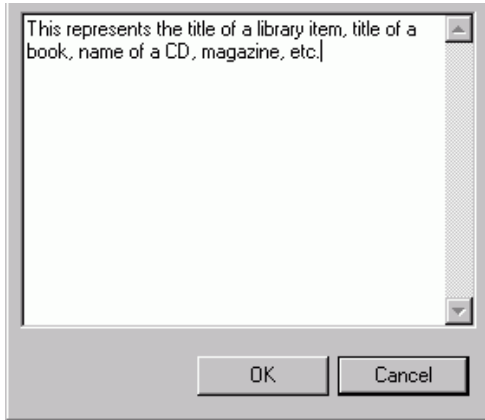


To add details to the *Title* class:

- 1 In the Properties view of the Library diagram, select the `title` attribute.
- 2 For *initial value*, enter `"Title"` (include the quotes) in the Properties view, as shown in Figure 27.
- 3 Also in Properties, expand *Description*, 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 28, and then click **OK**.

Figure 28 Enter text dialog



Using the Overview


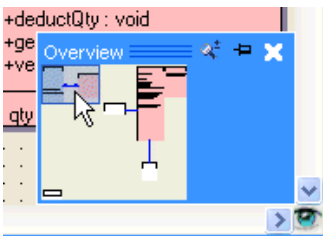

The overview feature of the Diagram view provides a thumbnail view of the current diagram. You can click the **Overview**  button located in the bottom right hand corner of every diagram, to open the Overview pane as shown in Figure 29.

Figure 29 Overview pane



To use the overview:

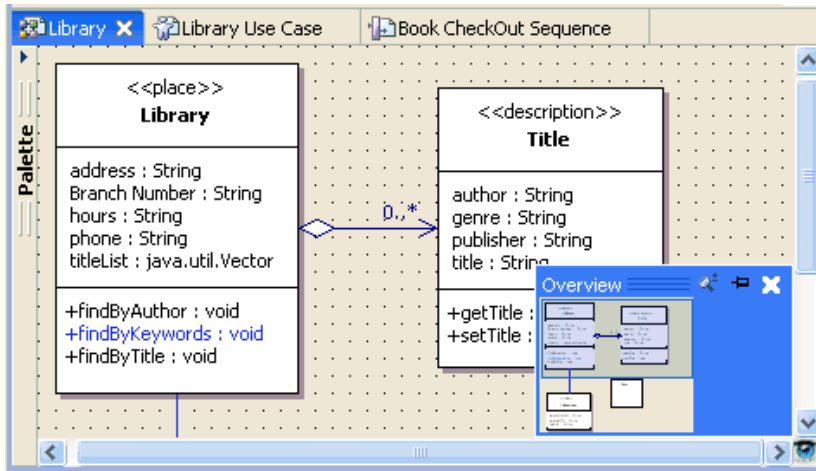
- 1 On the Library diagram, click the **Overview**  button. The pane expands to show a thumbnail image of the current diagram as shown in Figure 30.
- 2 Use the mouse to click on the shaded area and drag it. This is a convenient way to scroll around the diagram.

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 30 Thumbnail of Library diagram



Using and creating patterns

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

Note Patterns and templates are an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

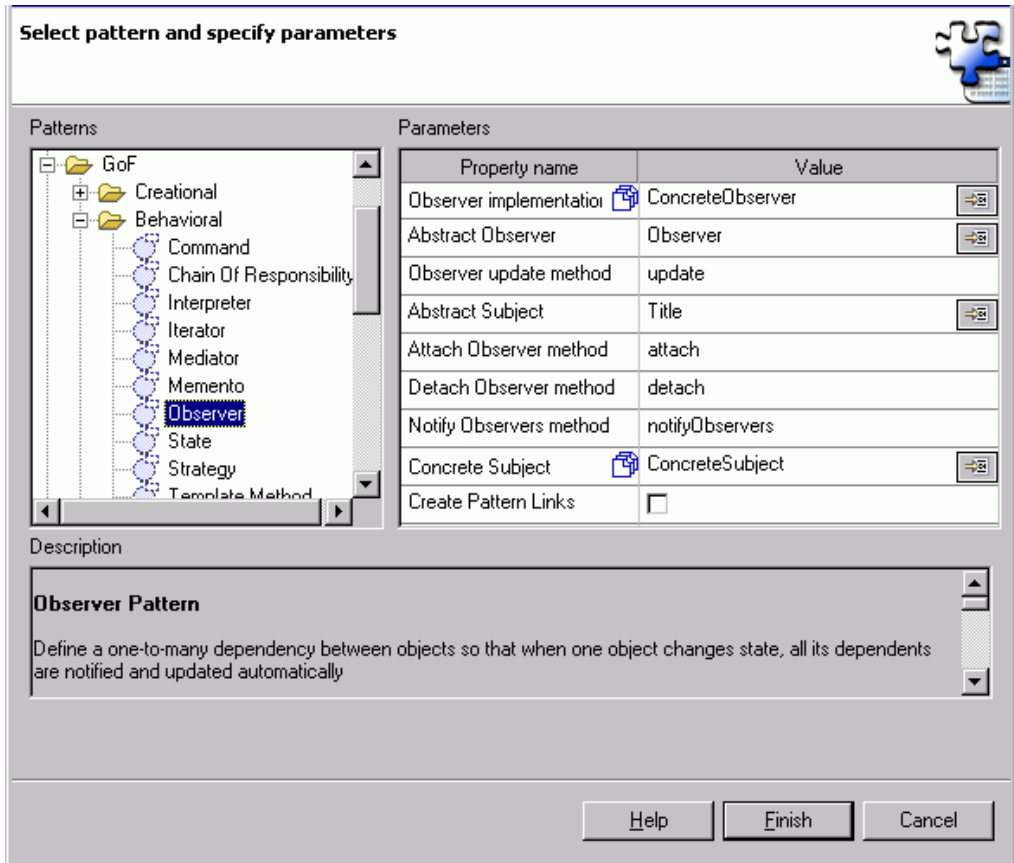
For the following example, assume that the Title class requires a dynamic system so that after a Title has been returned, the counter is updated and other systems are notified. The Observer pattern is useful for designing such a system.

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 24.

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 31 shows the corresponding Apply pattern dialog.

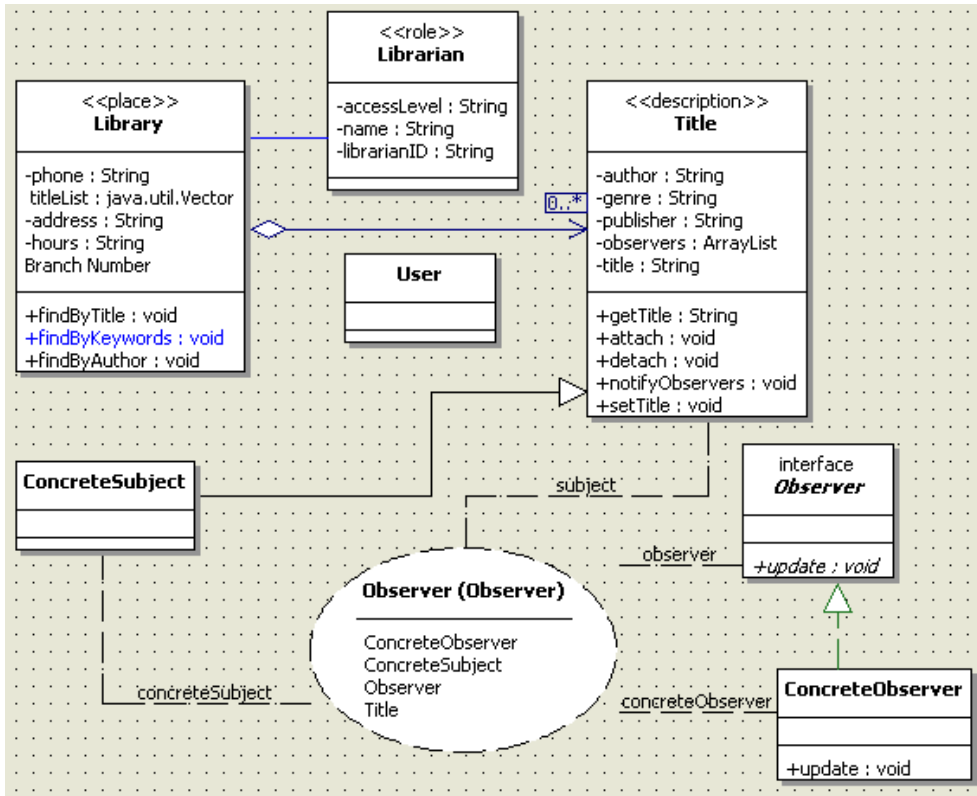
Figure 31 Apply pattern



3 Click Finish.

The diagram is updated as shown in Figure 32. Title is updated with the notification and observer methods (attach and detach). The other classes and interfaces have been created ready for use. TogetherEC 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 32 Applying the Observer pattern to a diagram

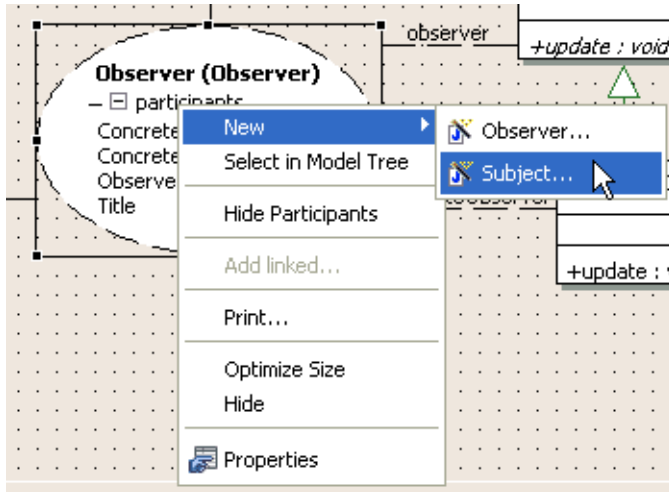


The pattern elements on the diagram have specific pattern actions that you can choose relevant to the pattern as shown in Figure 33.

If the Observer pattern “bubble” doesn’t appear in your diagram, you may need to enable patterns.

- 4 In the menubar, select **Window > Preferences**.
- 5 Expand **Modeling > Patterns**.
- 6 Check the *Recognize patterns when building project* checkbox on the General tab.
- 7 In the Patterns Recognizer view, right click on the project and select **Rebuild Patterns**.

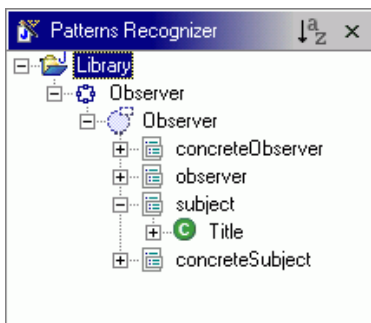
Figure 33 Applying pattern-specific options



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 34.

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 34 Patterns Recognizer view



Managing diagram views

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

Specifying a detail level

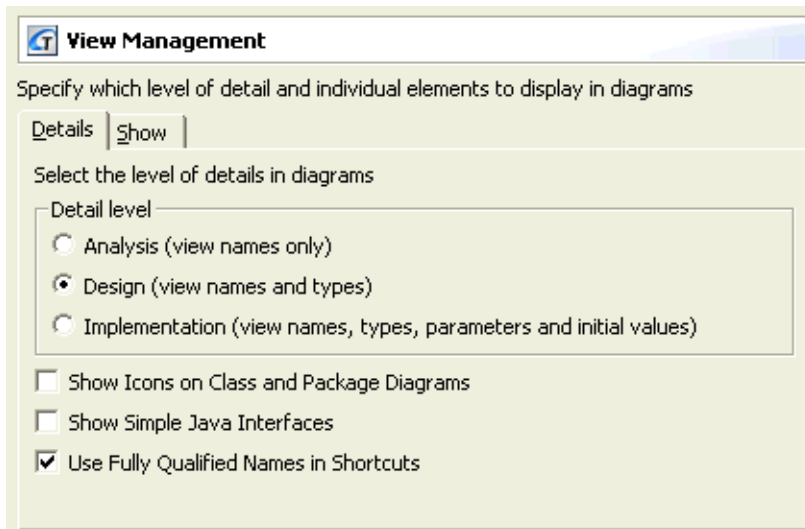
You can use detail level to view the different levels of detail shown in class diagrams. Detail levels include Analysis, Design, and Implementation.

To specify a detail level:

- 1 Choose **Window > Preferences** from the main menu to view the Preferences dialog.
- 2 On the left of the dialog, expand **Modeling > View Management**, and select the **Details** tab shown in Figure 35.
- 3 Choose the detail level (Analysis, Design, or Implementation) 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 35 Modeling View Management Preferences



Hiding/Showing Information

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

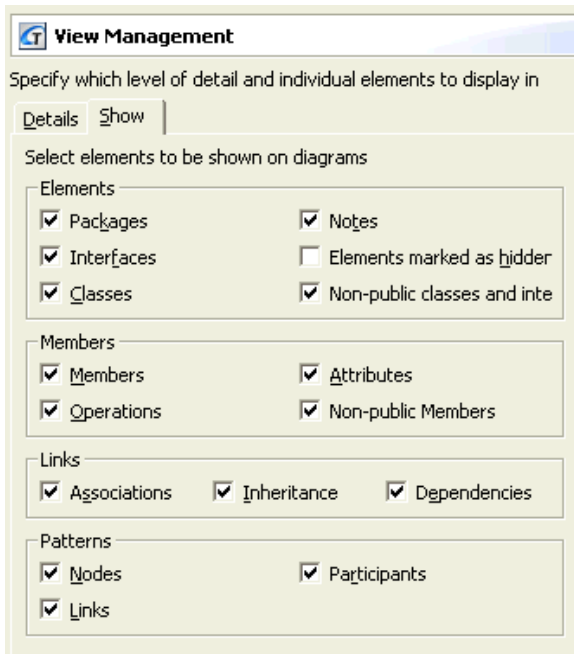
The UML Explorer has filters to show or hide information in the tree. You can also hide and show elements on diagrams. The following example demonstrates how to hide and show information on the Library class diagram created earlier in this guide.

To hide/show elements:

- 1 Right click on the `Library` class node in the diagram and choose **Hide/Show > Attributes**. The attributes node in the `Library` class is hidden on 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 36.

Figure 36 Preferences dialog for view management



The filters shown in Figure 36 are global filters. To specifically filter members from your classes, 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.

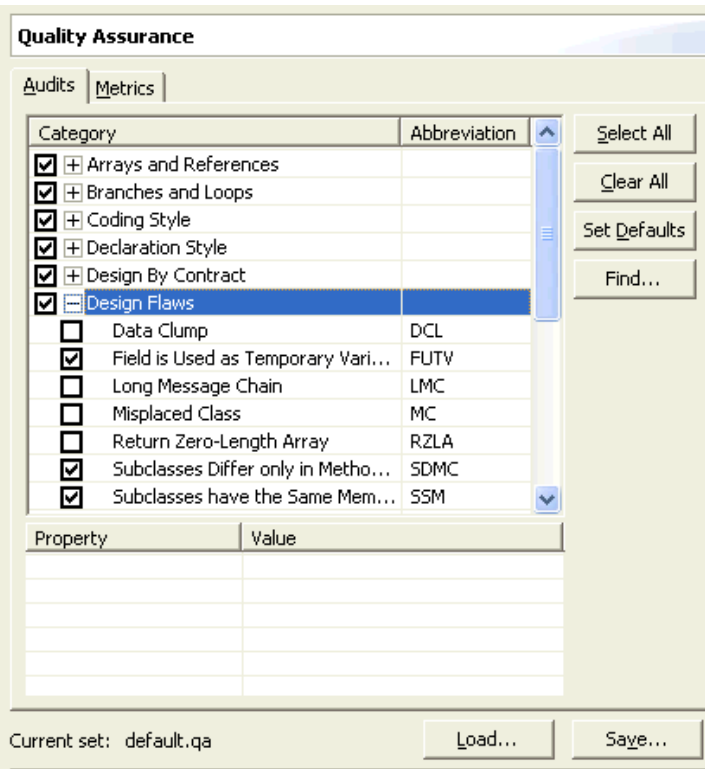
Note Quality assurance is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

Running audits

To run audits for the Library project:

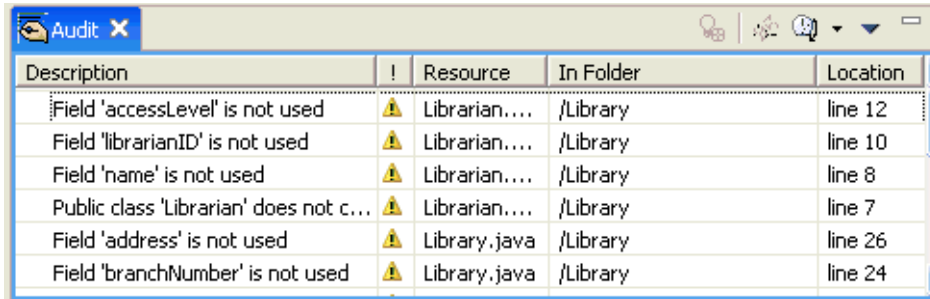
- 1 In the UML Explorer, right click on the Library class diagram and choose **Quality Assurance > Audits**.
- 2 The Run QA, Audits Execution dialog opens. Click **Preferences**. The Audit preferences dialog opens as shown in Figure 37. You can use this dialog to choose the specific audits you want to run.

Figure 37 Audit preferences



- 3 Click **OK** to accept the preferences. Click OK again to run the audits. TogetherEC generates the audits in the Audits view, as shown in Figure 38.

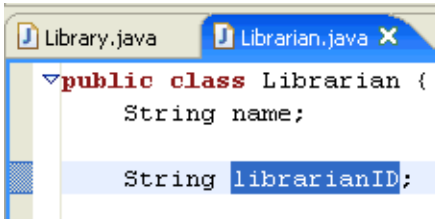
Figure 38 Audits view (your results may differ)



| Description | ! | Resource | In Folder | Location |
|--|---|---------------|-----------|----------|
| Field 'accessLevel' is not used | ⚠ | Librarian.... | /Library | line 12 |
| Field 'librarianID' is not used | ⚠ | Librarian.... | /Library | line 10 |
| Field 'name' is not used | ⚠ | Librarian.... | /Library | line 8 |
| Public class 'Librarian' does not c... | ⚠ | Librarian.... | /Library | line 7 |
| Field 'address' is not used | ⚠ | Library.java | /Library | line 26 |
| Field 'branchNumber' is not used | ⚠ | Library.java | /Library | line 24 |

- 4 In the Audit view, double click the second entry for *Field 'librarianID' is not used*. TogetherEC opens the source in the Java Editor, and highlights the line as shown in Figure 39.


Figure 39 Java Editor showing problematic source code



```
Library.java Librarian.java X
public class Librarian {
    String name;

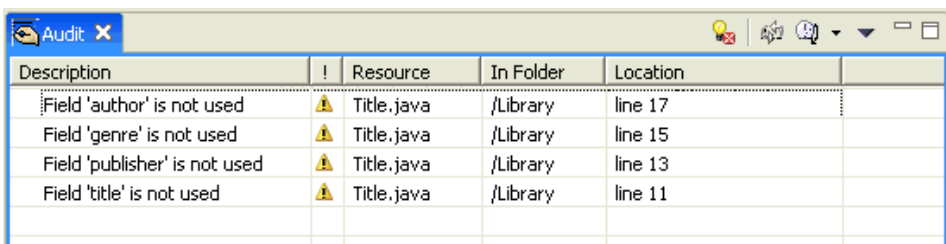
    String librarianID;
```

- 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.

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 .

- 6 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 40.

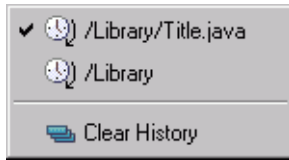
Figure 40 Audit view showing Title class



| Description | ! | Resource | In Folder | Location |
|-------------------------------|---|------------|-----------|----------|
| Field 'author' is not used | ⚠ | Title.java | /Library | line 17 |
| Field 'genre' is not used | ⚠ | Title.java | /Library | line 15 |
| Field 'publisher' is not used | ⚠ | Title.java | /Library | line 13 |
| Field 'title' is not used | ⚠ | Title.java | /Library | line 11 |

- 7 Click on the down arrow on the history button to see the list of audits you can run as shown in Figure 41.

Figure 41 List of audits



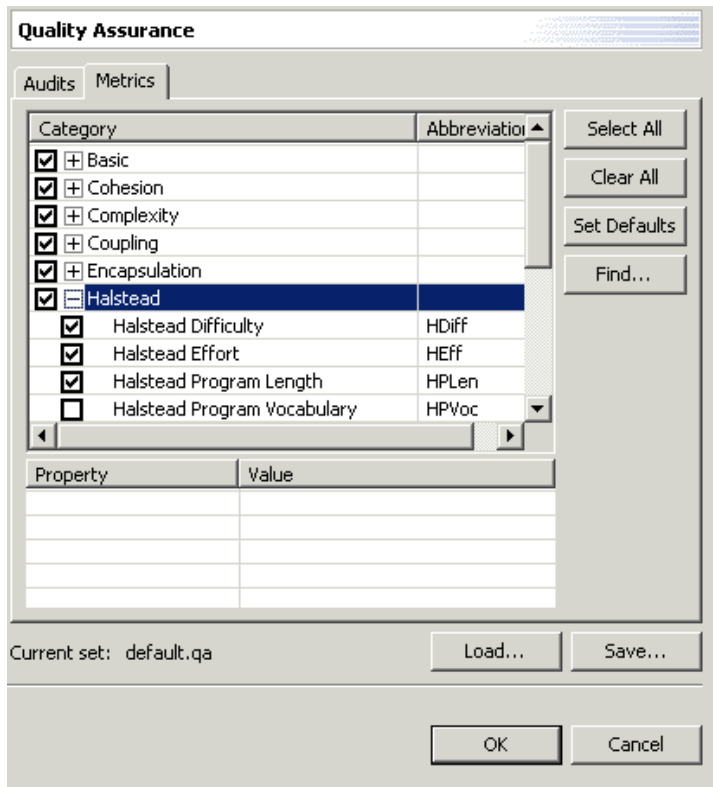
A checkmark next to an audit indicates that it is currently displayed in the audit table. This audit history exists during the lifetime of your current TogetherEC session.

Running metrics

To run metrics on the Library project:

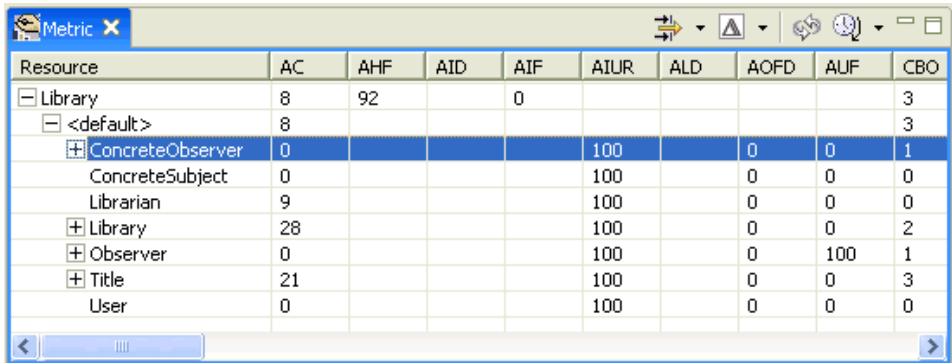
- 1 In the UML Explorer, right click on the Library project and choose **Quality Assurance > Metrics**.
- 2 The Run QA, Metrics Measurement dialog opens. Click **Preferences** to open the Metrics Preferences dialog as shown in Figure 42. Here, you can specify the metrics you want to generate.

Figure 42 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 43.

Figure 43 List of classes



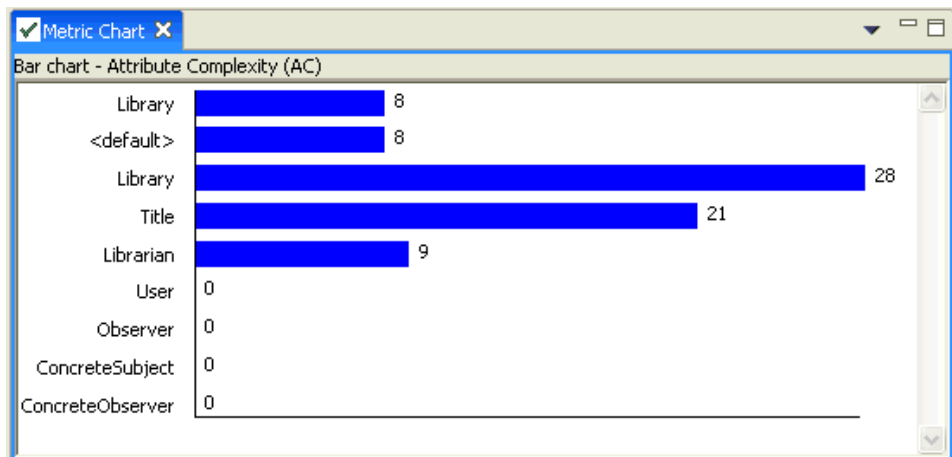
| Resource | AC | AHF | AID | AIF | AIUR | ALD | AOFD | AUF | CBO |
|------------------|----|-----|-----|-----|------|-----|------|-----|-----|
| Library | 8 | 92 | | 0 | | | | | 3 |
| <default> | 8 | | | | | | | | 3 |
| ConcreteObserver | 0 | | | | 100 | | 0 | 0 | 1 |
| ConcreteSubject | 0 | | | | 100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

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

Note The Metric table has a History button like the Audit table as described in “Running audits” on page 43. 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 44 shows the bar graph for the AC column.

Figure 44 Bar graph



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

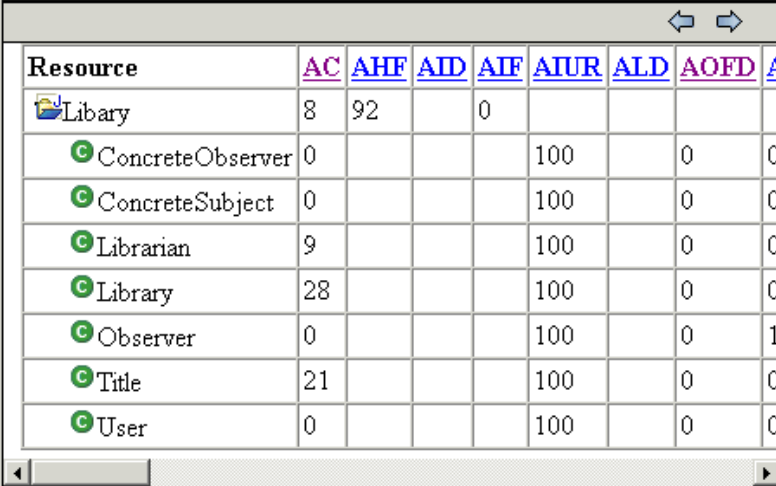
7 To generate a report, right click on the table in the Metric view and choose **Export**. This opens the Export QA results to file dialog.

8 Enter a location for saving the file.


9 For Type, choose **HTML file**.

- 10 Make sure the option *Open in browser* is checked and then click **OK**. TogetherEC generates the report, saves it, and opens it in a browser, as shown in Figure 45.

Figure 45 Sample QA report



| Resource | AC | AHF | AID | AIF | AIUR | ALD | AOFD | A |
|------------------|----|-----|-----|-----|------|-----|------|---|
| Library | 8 | 92 | | 0 | | | | |
| ConcreteObserver | 0 | | | | 100 | | 0 | 0 |
| ConcreteSubject | 0 | | | | 100 | | 0 | 0 |
| Librarian | 9 | | | | 100 | | 0 | 0 |
| Library | 28 | | | | 100 | | 0 | 0 |
| Observer | 0 | | | | 100 | | 0 | 1 |
| Title | 21 | | | | 100 | | 0 | 0 |
| User | 0 | | | | 100 | | 0 | 0 |

Note The report includes the same elements shown in the Metric results table. To change the elements included in the report, use the Filters button .

Saving and loading metric results

To save metric results:

- 1 Right click on the table in the Metric view.
- 2 Choose **Export**.
- 3 The Export Metric Results dialog opens.
- 4 Enter the path and name for the saved metrics, or accept the default.
- 5 From the *Type* dropdown list, choose *Save in loadable format*.
- 6 The file is saved as a *.mtbl file.

Note Audits are saved the same way, but as *.atbl files. You may also want to note the other formats available. The HTML choice creates a browser-compatible .html file that includes descriptions of each audit and metric. The text options are best for using in a spreadsheet.

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.

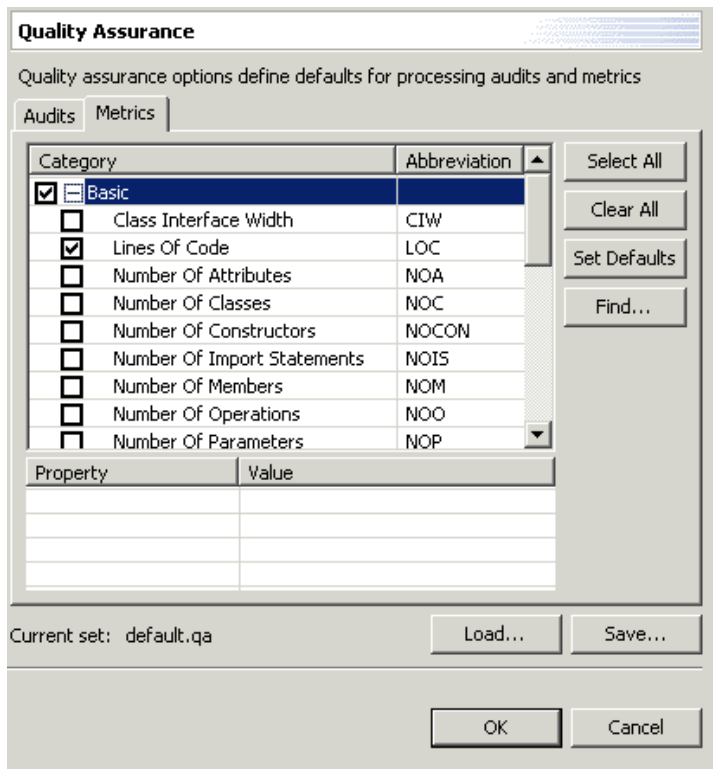
Setting rules for quality assurance

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

To set rules for QA:

- 1 From the main menu, choose **Window > Preferences** to open the Preferences dialog.
- 2 Expand **Modeling > 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 46.

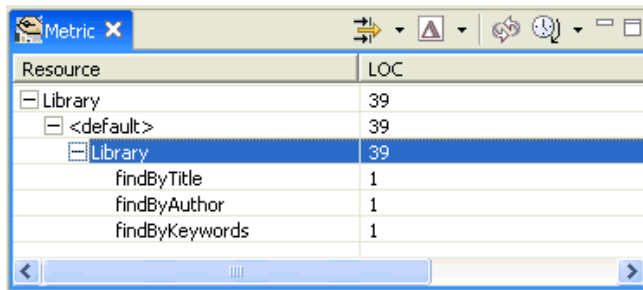
Figure 46 Metric preferences



- 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 47.

Figure 47 Results for saved set of rules (your numbers may differ)



| Resource | LOC |
|----------------|-----|
| Library | 39 |
| <default> | 39 |
| Library | 39 |
| findByTitle | 1 |
| findByAuthor | 1 |
| findByKeywords | 1 |

Comparing metrics results


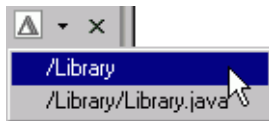
After you have run metrics on your project more than once, or loaded some existing metric results, the down arrow of the Compare With button  contains a list of the results as shown in Figure 48.

Figure 48 Compare With menu on the Metric view



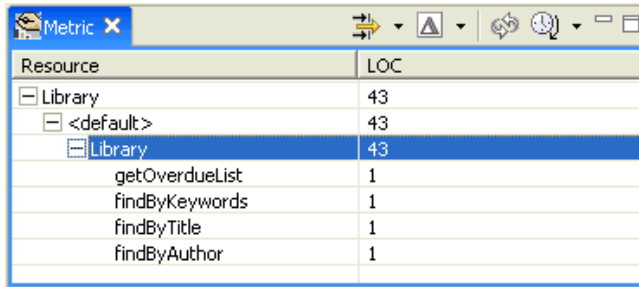
Compare With compares the current set of results with the one you choose on the menu.

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 49 shows similar results.

Figure 49 Results of metrics for Library class with the new method (your numbers may differ)



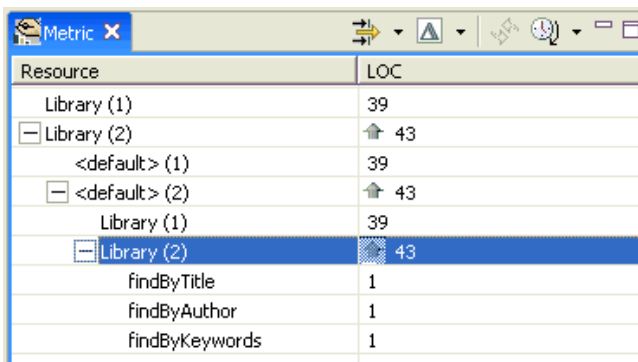
The screenshot shows a window titled 'Metric' with a table of results. The table has two columns: 'Resource' and 'LOC'. The 'Resource' column is expanded to show a tree structure: 'Library' (43), '<default>' (43), and 'Library' (43). The 'Library' (43) row is selected, and its details are shown in the bottom section of the table: 'getOverdueList' (1), 'findByKeywords' (1), 'findByTitle' (1), and 'findByAuthor' (1).

| Resource | LOC |
|----------------|-----|
| Library | 43 |
| <default> | 43 |
| Library | 43 |
| getOverdueList | 1 |
| findByKeywords | 1 |
| findByTitle | 1 |
| findByAuthor | 1 |

- 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 50.

Figure 50 Comparing metric results (your numbers may differ)



The screenshot shows a window titled 'Metric' with a table of results. The table has two columns: 'Resource' and 'LOC'. The 'Resource' column is expanded to show a tree structure: 'Library (1)' (39), 'Library (2)' (43), '<default> (1)' (39), '<default> (2)' (43), 'Library (1)' (39), and 'Library (2)' (43). The 'Library (2)' (43) row is selected, and its details are shown in the bottom section of the table: 'findByTitle' (1), 'findByAuthor' (1), and 'findByKeywords' (1).

| Resource | LOC |
|----------------|-----|
| Library (1) | 39 |
| Library (2) | 43 |
| <default> (1) | 39 |
| <default> (2) | 43 |
| Library (1) | 39 |
| Library (2) | 43 |
| findByTitle | 1 |
| findByAuthor | 1 |
| findByKeywords | 1 |

In the example above, the LOC for `Library` originally had a value of 35. However, the latest metric results show the number of lines of code increasing to 39 (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

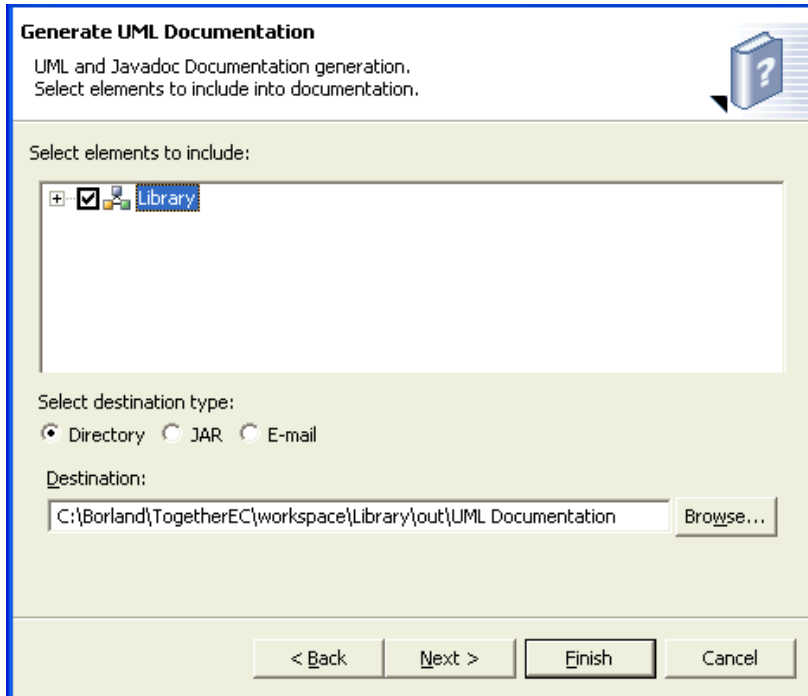
TogetherEC features a UML documentation wizard that you can use to generate documentation for the `Library` project. You can easily generate documentation across multiple projects and from CaliberRM requirements.

Note Generating documentation is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

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 51.

Figure 51 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 project tree at the top of this page and unchecking the parts you do not want included in the documentation.

- 4 On the first page of the wizard, you can accept the default name and path for the generated documentation or enter your own. (There should be no spaces in the name of your destination folder.) Then click **Next**.
- 5 On this page, you can specify the content of the report, such as whether the report should include diagrams. You can also choose the diagram image format (SVG, BMP, BMP (RLE), GIF, or JPG). Accept the defaults for a full report. Click **Next**.

- Click the box next to the *Document title* field and enter “Initial Library Documentation” as shown in Figure 52. Leave the other options unchanged. (The location for “referenced archives and projects” in the lower half of the dialog will differ from that shown in Figure 52.)

Figure 52 Documentation Wizard

Generate UML Documentation

UML and Javadoc Documentation generation.
Configure Javadoc arguments for standard doclet.

☒ Document title: Initial Library Documentation

Basic Options

- ☒ Generate use page
- ☒ Generate hierarchy tree
- ☒ Generate navigator bar
- ☒ Generate index
- ☒ Separate index per letter

Document these tags

- ☒ @author
- ☒ @version
- ☒ @deprecated
- ☒ deprecated list

Create Javadoc for members with visibility:

☐ Private ☐ Package ☐ Protected ☒ Public

Select referenced archives and projects to which links should be generated:

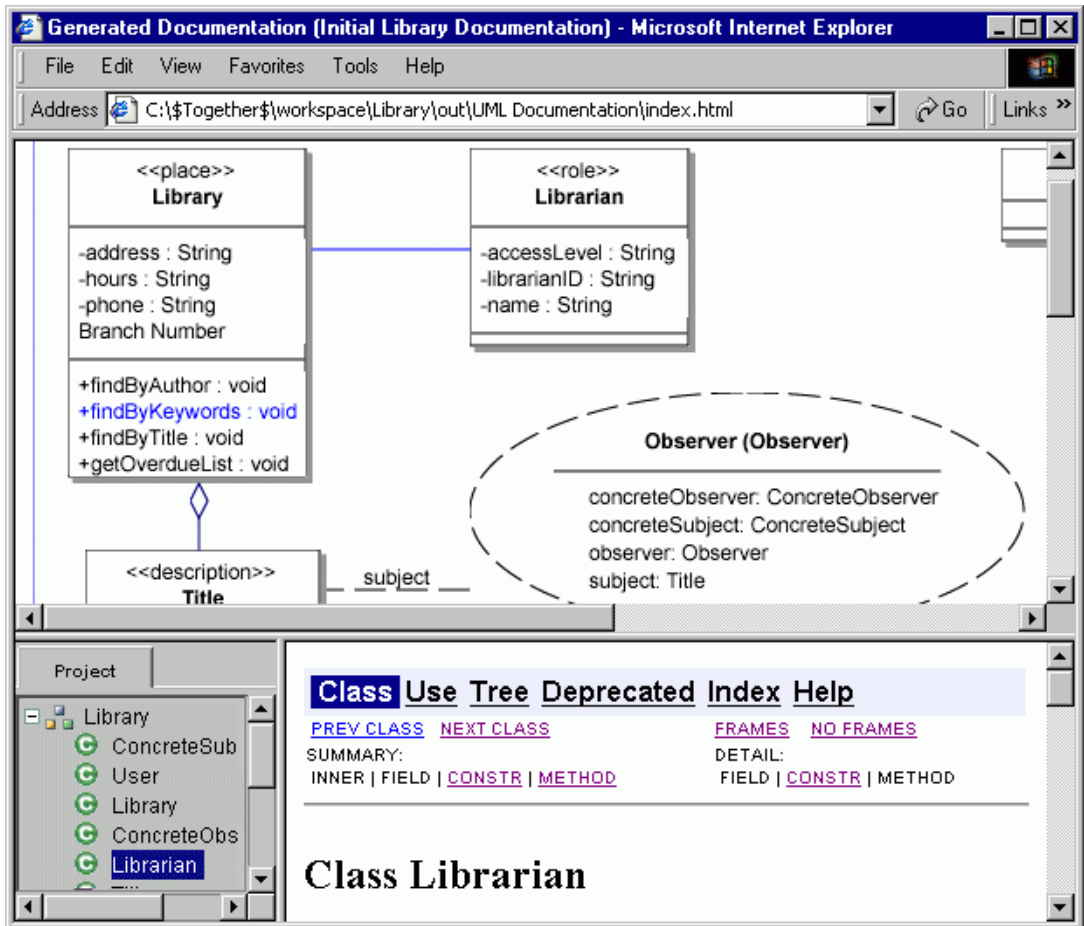
- ☐ rt.jar - C:/jdk1.4/jre/lib/rt.jar
- ☐ sunrsasign.jar - C:/jdk1.4/jre/lib/sunrsasign.jar
- ☐ jsse.jar - C:/jdk1.4/jre/lib/jsse.jar
- ☐ jce.jar - C:/jdk1.4/jre/lib/jce.jar
- ☐ charsets.jar - C:/jdk1.4/jre/lib/charsets.jar
- ☐ dnsns.jar - C:/jdk1.4/jre/lib/ext/dnsns.jar
- ☐ ldance.jar - C:/jdk1.4/jre/lib/ext/ldance.jar

☐ Style sheet: Browse...

< Back Next > Finish Cancel

- Click **Finish**.
- When TogetherEC finishes generating the documentation, a dialog opens. Check *Open generated documentation in a browser*, and click **OK**.
- 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 53. Notice that clicking a class name in the lower left frame opens the documentation in the lower right pane.

Figure 53 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 TogetherEC.

- 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

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

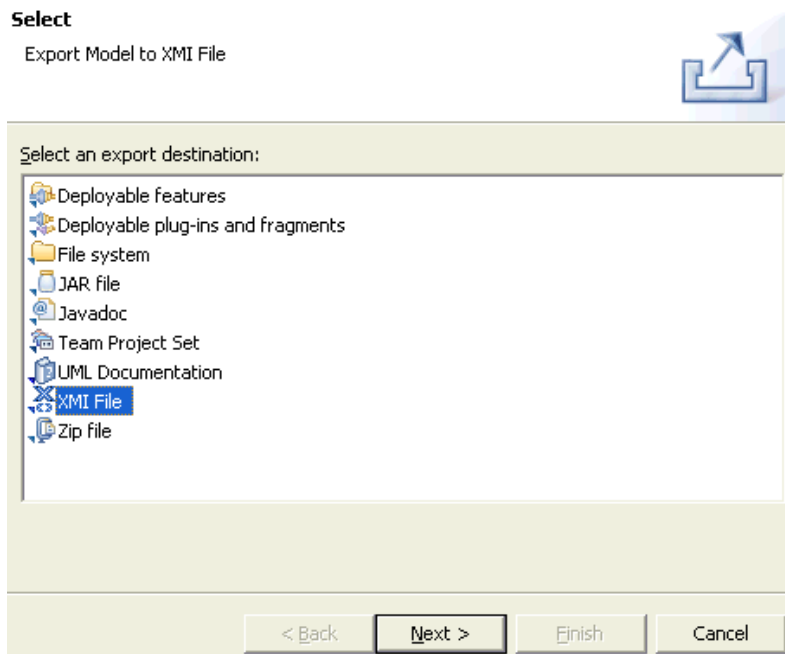
Note XMI import/export is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

Exporting XMI projects

To export a project as an XMI file:

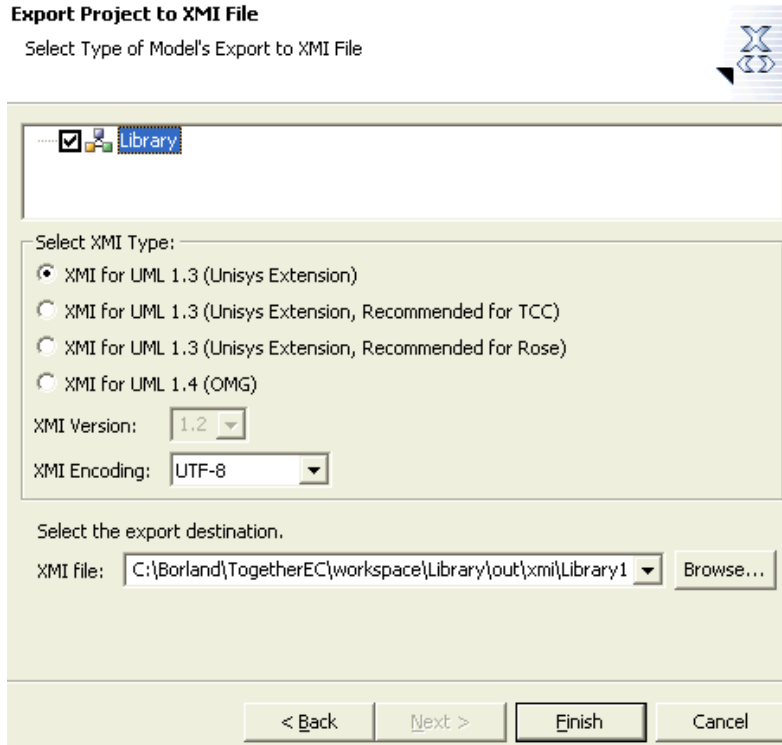
- 1 In the UML Explorer, right click on the Library project and choose **Export**.
- 2 In the Export wizard dialog, click **XMI File**. Click **Next**.

Figure 54 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 55 The XMI Export wizard
Export Project to XMI File
Select Type of Model's Export to XMI File



- 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. By default, it is saved to the following location:
\$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 Import dialog opens, choose **XMI file** from the list and follow the prompts from the XMI Import wizard.

Note Only XMI 1.1/1.2 import is supported. Attempting to import an XMI 1.0 file results in an empty project.

Creating a Project from an MDL model

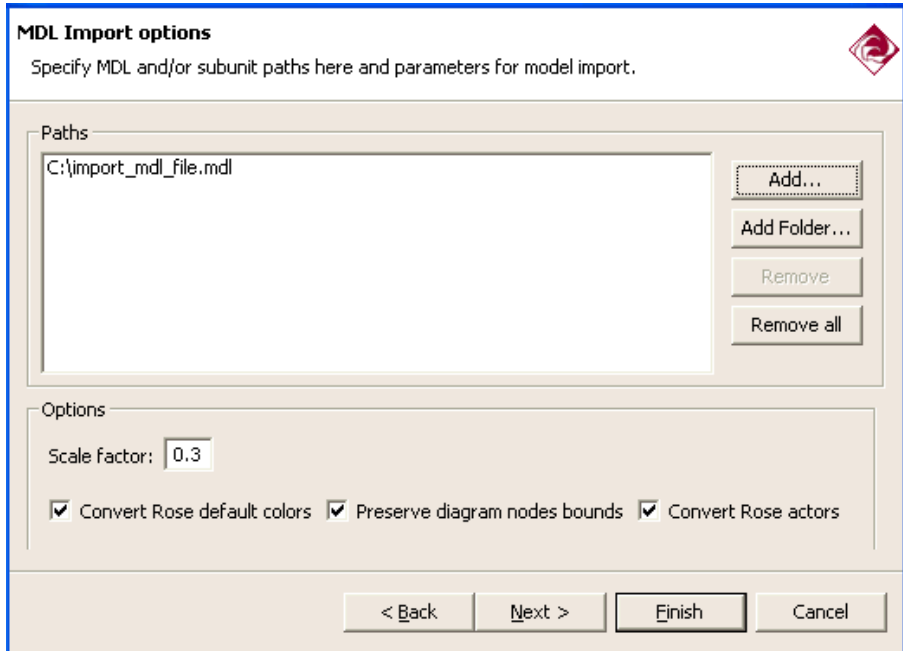
Together enables you to create projects around IBM® Rational® Rose model files (.mdl, .ptl, .cat, .sub). You can import a set of petal and subunit files.

Together projects created on the basis of the imported MDL models always comply with the UML 1.4 specification.

Note Importing an MDL model is an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

To create a project from an MDL model:

- 1 From the menubar, select File > New > Project. The New Project wizard displays.
- 2 Expand the Together node in the tree view list, and select Create Project from MDL. Click Next. The first screen of the wizard is shown in the figure below:



- 3 In the first screen of the wizard:

- Click either **Add** or **Add Folder** to designate the MDL project path. This step specifies the name (or names) of the Rational Rose project file (or files) to be imported (several model files can be imported at once). Click **Remove** to delete the selected file or files from the Paths list. Click **Remove all** to delete all files from the Paths list.

Important Avoid adding a model file along with its subunit to the import list, since it results in invalid project.

- Use the *Scale factor* field to specify the element dimensions coefficient. By default, the scale factor is 0.3.
- You can also check the following options for the project:

- *Convert Rose default colors*: if this option is checked, the default Rational Rose colors will be replaced with the default Together colors.
 - *Preserve diagram nodes and bounds*: if this option is checked, user-defined bounds are preserved in the resulting diagrams. Otherwise the default values are applied.
 - *Convert Rose actors*: this option enables you to choose mapping for the Rose actors.
 - If the option is checked, the Rose actors are mapped to Together actors.
 - If the option is not checked, the Rose actors are mapped to the classes with the Actor stereotype, such as: Actor, Business Actor, Business Worker, or Physical Worker.
 - Click Finish to create the project accepting the remainder of the default settings.
- 4 If the imported project contains path aliases, supply real paths for each recognized path alias in the Virtual Path Map dialog box. Note that specifying aliases is an iterative process. If a file or subunit contains other files with aliases, the dialog will show up several times, depending on the nesting level of the documents being imported. For more information, see “About Path Aliases” on page 58.

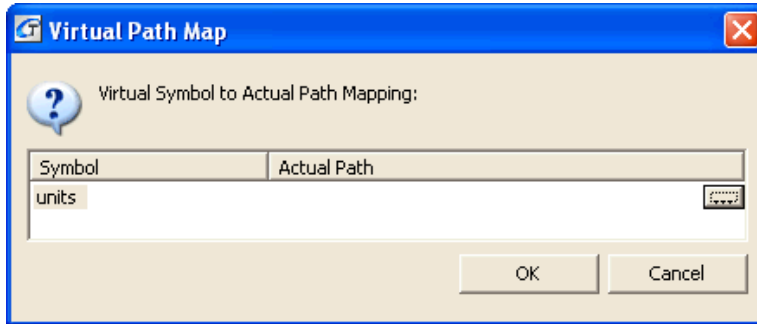
Once the import process completes, you can view the project structure in the UML Explorer view. The mdlimport.log is generated by default and lists any errors encountered during the import process.

Note After entering a project name, you can click Finish without completing the remaining steps of the wizard. The project is created using the remainder of default settings.

About Path Aliases

Rational Rose model files may contain path aliases that need to be converted to real paths. Together recognizes path aliases and displays the Virtual Path Map dialog box, shown in the figure below, that enables you to supply a real path for each path alias. To specify the actual path, click the Browse button. This opens the Select Actual Path dialog box. Navigate to the desired path and click OK when ready.

Figure 56 Virtual Path Map dialog



Tip If new aliases are encountered in course of the file or subunit parsing, the Virtual Path Map dialog will be displayed again.

MDL Import Notes

A single state/activity element in a Rose model can be put into several different swimlanes. However, state/activity elements in Together can belong to only one swimlane; therefore, when importing a Rose project with a single state/activity element placed into several different swimlanes, Together places the state/activity element in one swimlane only.

Using Rose, it is possible to create nested diagrams for class, use case, activity, and state elements. When using MDL import, the relationship between the element and the nested diagram is shown by a hyperlink that is created from the element to the diagram.

Setting UML profiles

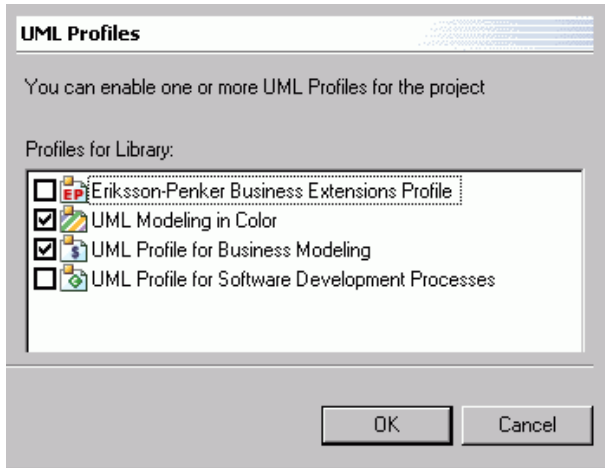
TogetherEC uses profiles to support various extensions of UML. A single project can have multiple profiles. You can manage profiles through the project properties.

Note UML profiles are an activatable Together Capability. For more information, see “Activating Together Capabilities” on page 9.

To enable a profile for a project:

- 1 From the UML Explorer view, right click on the project root and choose **Properties**.
- 2 In the resulting dialog, select **UML Profiles**.
- 3 Check the desired profile or multiple profiles from the list (you can enable more than one) as shown in Figure 57, and click **OK**.

Figure 57 Properties for Library dialog (Your profile list may differ.)



Activating profiles enables several new stereotype options for specific diagram elements associated with the activated profile, such as class, interface, package, and activity elements.

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