Getting Started Guide

Borland Together Developer 2006 for Microsoft Visual Studio

Borland® Together®

Integrated and Agile Design Solutions

Borland®

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

Updated: May 15, 2006

Contents

CHAPTER

1

Introducing Together 2006 for Visual Studio

This chapter includes the following topics:

About Together 2006 for Visual Studio

Together 2006 for Visual Studio (hereafter "Together" or "Together for VS") is the extension to Microsoft Visual Studio 2005 for UML modeling. This product empowers the new generation of Visual Studio with the functionality of the award-winning, design-driven environment with features such as UML 2.0 modeling, OCL, design patterns, QA audits and metrics, IBM Rational Rose (MDL) and Rational XDE (MDX) import, XMI import and export, and automated documentation generation.

A key feature of Together, LiveSource™, keeps UML diagrams synchronized with your source code in the Visual Studio Editor.

Together is an integral part of a complete ALM (Application Lifecycle Management) solution provided by Borland Software Corporation. This version of Together is a part of the new generation of the Borland's ALM solution named SDO (Software Delivery Optimization). SDO is Borland's vision and strategy for transforming software delivery to an incorporated and disciplined approach that aligns teams, technology and process to maximize the business value of software.

Effective modeling with Together simplifies the development stage of your project. Smooth integration to Visual Studio provides developers with easy transition from models to C# and Visual Basic source code.

This guide assumes that you have installed Together Developer 2006 and Microsoft Visual Studio 2005 on your computer. For information about installing Together Developer, refer to the `install.html` file. For information about installing Visual Studio, refer to the respective product documentation.

About licensing

Together and Microsoft Visual Studio require separate licenses. For information regarding licensing for Visual Studio, refer to the Visual Studio documentation.

For information regarding licensing for Together, refer to the online help or the User Guide.

There are two types of licenses available for Together:

- Together Designer 2006 for Visual Studio (Together Designer)
- Together Developer 2006 for Visual Studio (Together Developer)

The feature set depends on the license acquired. Different features are intended for use by different team members. They play different roles in the development process.

Supported features

The following table indicates which features are available with each license type:

Table 1 The features supported by Together Designer and Together Developer

Feature	Together Designer	Together Developer
UML 1.4	●	●
UML 2.0	●	-
C# implementation projects	-	●[1]
Visual Basic implementation projects	-	●
Language-neutral design projects	●	-[2]
Quality Assurance audits	-	●
Quality Assurance metrics	-	●
OCL[3] 2.0	●	●

Table 1 The features supported by Together Designer and Together Developer (continued)

Feature	Together Designer	Together Developer
UML in color	●	●
Patterns (including GoF)	●	●
Transform design projects to C# and Visual Basic source code	●	●
Documentation generation	●	●
Search	●	●
XMI import[4]	●	-
XMI export[5]	●	-
IBM Rational Rose (MDL) files import	●	-
IBM Rational XDE (MDX) files import	●	-
Requirements management software integration using Borland CaliberRM	●	●
Version control integration using Visual Studio	●	●
Version control integration using StarTeam	●	●

1. For implementation projects, only UML 1.4 is supported, even if a license for Together Designer is available.
2. It is still possible to open design projects in Together Developer, but the Model View and the Diagram View are not available in this case.
3. Object Constraint Language.
4. This feature is available for UML 1.4 projects only.
5. This feature is available for UML 1.4 projects only.

Note All features will be available if you have licenses for both Together Designer and Together Developer.

About this guide

The goal of this guide is to provide an introduction to Together Developer 2006 and the Together modeling technology. It explains how to create a sample C# project. This project is used to exercise the primary features of Together. It is recommended that you read the chapters in sequential order.

For more task-oriented instructions, refer to the online help for Together. Choose [Help | Contents...] on the Visual Studio main menu, and select **Together 2006 for Visual Studio** in the Contents pane.

Note The instructions provided in this guide are specific to Together Developer.

This guide provides examples for:

- Setting global modeling options
- Creating a project and opening the Model and Diagram Views
- Creating a Use Case Diagram
- Creating a Class Diagram
- Creating a Sequence Diagram
- Working with diagrams
- Using the Model View
- Using the Properties Window
- Managing diagram views
- Using patterns
- Running quality assurance
- Generating documentation
- Opening projects created in other versions

Audience

The information in this guide is intended primarily for softwaredevelopers who use Together Developer to create models of their software applications.

The documentation assumes that you have experience with the following:

- The basics of the Unified Modeling Language (UML)
- The process your organization uses for designing and modeling object-oriented systems or components
- Designing applications in C# and/or Visual Basic
- Design patterns

Text conventions

This guide is presented in the Adobe PDF file format. Adobe Reader or any other Adobe Acrobat tool is required to view and print it.

The guide observes the following conventions:

- The names of buttons, dialog windows, and other interface elements are written in bold, for example: **Open**.
- Items selected on menus are separated by a vertical bar (“|”) between levels. For example: [Tools | Together for VS | Pattern Organizer] means: choose the **Tools** menu, then choose the **Together for VS** submenu, and then the **Pattern Organizer** menu item.

- `$TGH$` stands for Together Home, or the Together installation folder. By default, `C:\Program Files\Borland\Together for VS`

Together documentation set

The documentation set for Together consists of the following items:

Table 2 Together documentation set

Item	Description	Location
Release notes (ReadMe)	Late-breaking information including: • Last minute notes • System requirements • Known issues and limitations	• `$TGH$/Docs/readme.html` • http://info.borland.com/techpubs/together
Installing Together	Installing and starting Together Developer	• `$TGH$/Docs/install.html` • http://info.borland.com/techpubs/together
Getting Started Guides[1] *(you are reading now)*	Information for the first time user including: • Creating a sample project for use in Together Developer and Visual Studio • Working with Use Case, Sequence, and Class diagrams • Setting options in Together • Other information specific to using Together	• `$TGH$/Docs/gettingStarted_des.pdf` • `$TGH$/Docs/gettingStarted_dev.pdf` • http://info.borland.com/techpubs/together
Online help	General, context and dynamic help for Together includes comprehensive information most relevant to the user: • Introduction to Together • Setting personal preferences and options • Detailed instructions for using Together features • Working with UML diagrams in Together	The items on the Visual Studio main menu: • [Help | Contents/Index/Search...], filters: Together for VS Together for VS Designer Together for VS Developer • [Help | Dynamic Help]
User Guide	This guide in the Adobe PDF format includes all the information available in the online help.	• `$TGH$/Docs/userGuide.pdf` • http://info.borland.com/techpubs/together

1. There are different Getting Started Guides for Together Designer and Together Developer.

Other sources of information

If you need more information, try the following sources, or contact the Borland Software Corporation customer support service.

Tool tips

Together Developer displays brief descriptions for the program options in the lower part of the Options dialog window.

Borland web sites

The official Borland Software Corporation web site: http://www.borland.com/

The Borland Developer Network site: http://bdn.borland.com/

The Borland Support Center (Borland Answers): http://support.borland.com/

Third-party resources

If you are just getting started with UML, object-oriented technology, or distributed application development, the following books and web sites can help you:

UML Distilled: Applying the Standard Object Modeling Language by Martin Fowler. Addison-Wesley, 1997. ISBN: 0201325632

The Unified Modeling Language User Guide by Booch, Rumbaugh, and Jacobson. Addison-Wesley, 1998. ISBN 0-201-57168-4

Object Management Group site: http://www.omg.org/

Cetus Links site: http://www.cetus-links.org/

Dev-x Developer Exchange site: http://www.devx.com/

About modeling in Together

Programmers use Together to design UML-compliant models of their software applications. A model consists of a hierarchy of *namespaces (packages)* and a collection of interconnected *diagrams* within these namespaces (packages). A diagram is a scheme on which *design elements* (classes, actors, and so on) and their *relationships* are drawn.

There are two kinds of diagrams:

- Code-generating diagrams
- Not code-generating diagrams

There are only three types of code-generating diagrams: Class, Sequence, and Collaboration.

Other diagram types (for instance, Use Case) describe higher level behavior or supplementary sides of your model. They do not influence the source code of your application directly. Usually, software architects start modeling with Use Case or other not code-generating diagrams.

Supported modeling elements

With a Together Developer license, Together supports version 1.4 of UML.

With a Together Designer license, Together supports UML versions 2.0 and 1.4.

The following diagram types are supported:

- UML 1.4 diagrams
  - Class
  - Use Case
  - Sequence
  - Collaboration
  - Statechart
  - Activity
  - Component
  - Deployment

In addition to UML diagrams, Together Developer supports special namespace (package) diagrams.

When you create a new design project, you explicitly specify the version of UML.

For implementation projects, only UML 1.4 is supported.

Collaboration

There are many ways to collaborate. This section makes a few suggestions.

In general, communication is facilitated by UML diagrams, CaliberRM traces, comments in the code and perhaps even face-to-face conversations.

Analysts and architects collaborate through the design model. They may choose to create analysis-level class diagrams for high-level design decisions, validate use case models and requirements or to illustrate collaboration between objects.

Depending upon the development social environment and the particular development process used, collaboration on class diagrams may be limited to architects and developers. In this case, the project lead might decide that class diagrams will be created and elaborated only within the implementation project.

There will need to be a rather detailed assessment of how a particular group develops software in order to determine what Together 2006 for Visual Studio licenses are applicable and in what quantity.

Usage scenarios

This section provides some ideas about how to use Together products on a software development team. Practical examples that illustrate usage of these products are given in the following chapters.

It is possible to work with a model in two directions:

- From the model to the source code (see "Forward engineering" on page 8)
- From the source code to the model (see "Reverse engineering" on page 9)

Together with its LiveSource technology supports both methods, always keeping the model and the source code synchronized.

Forward engineering

Application modeling using forward engineering involves two major steps:

- Designing a model
- Developing an application based on the model

Usually different specialists perform these steps: software designers (or architects) and software developers, respectively.

Designing a model

The software designer creates models by using the Together Designer license, depending on the features required.

Numerous types of diagrams can be used for modeling (see "Supported modeling elements" on page 7). But in any case one or more class diagrams must be created as a basis for any object-oriented application.

Developing an application based on the model

There are two ways to start with a model and develop source code from it. You can:

- Create an application and a model simultaneously as a single implementation project (UML 1.4 is only available in this case, the Together Designer license is not required, LiveSource is constantly available)
- Complete your design project and then transform it to an implementation project (design project can be in UML 1.4 or 2.0 format, then it is converted to UML 1.4, design project and source code are not linked directly)

Reverse engineering

Reverse engineering can be required when you need to build a model around an existing C# or Visual Basic project (for example, the project was created in a previous version of Visual Studio or before installing Together). Such situations can be handled with Together Developer.

Together Developer analyzes your project and builds UML 1.4 Class, Sequence, and Collaboration diagrams for it. Once the model has been created, Together Developer uses the LiveSource technology to keep source code and diagrams synchronized.

Graphical user interface overview

Together Developer is integrated into your Microsoft Visual Studio 2005 development environment. Together complements Visual Studio by adding the following elements to its graphical user interface (GUI):

- New views: **Model View**, **Diagram View**, **CaliberRM Requirements**, **CaliberRM Traces**, **Manage Traces**, and **Trace Synchronizer**
- New option group named **Together for VS**
- New menu items: **Export Diagram to Image**, **XMI Import** and **XMI Export** on the **File** menu, **Together for VS Support...** on the **Project** menu, and **Together for VS** submenu on the **Tools** menu

The most important views

The following views are the most important when you work with Together Developer:

- **Model View** – The primary navigational view for Together Developer. You can use this view to manage diagram elements: open, create, delete, and so on. Explore the context menus of the different elements as you encounter them.
- **Diagram View** – Displays UML diagrams. This view provides a tab for each open diagram.
- **Properties Window** – Using Together Developer, this view shows the properties for an element selected from the Diagram or Model Views. Properties for each element are usually broken into three categories:
  - Comments – Add *Author*, *Since*, and *Version* tags to your source code
  - Design – Define foreground and background colors for design elements
  - General – UML properties for the selected element as well as source code properties for source code elements

Combined, these tools are central to designing projects in UML and creating source code for your project.

Standard actions with views

It is possible to rearrange the layout of a view in several ways. You can view, hide, move, resize, dock, and undock views.

Together Developer saves your layout configuration and restores it when you run it next time.

Viewing or hiding views

By default, only the **Solution Explorer** view is visible. Show or hide any view by choosing an appropriate item on the **View** menu.

Moving, stacking and resizing views

Move a pane to a new position on the screen by clicking the title bar of the view and dragging it.

When you move your view, Visual Studio at any moment highlights a new location of the view by

- Displaying a shadow rectangle of the new place of the view (*dockable* views only)
- Displaying navigational guide diamond in the center and at the borders of the screen

Any view can be stacked with others. Each view will be displayed as a tab. To unstack the view, click its tab and drag it to a new place.

To resize the view, drag its borders to a new position.

Floating, dockable, and tabbed views

Each view (except the **Editor** and the **Diagram View**) has always one of the following layout states:

- Floating (free position on the screen)
- Dockable (can be moved to a fixed position, arranged or stacked with other dockable views)
- Tabbed Document (the view is presented as another tab of the **Editor**)

A docked view can be displayed in the **Auto Hide** mode (rolled up to a small symbol at the border of the working area.)

To change the layout state or mode, select the pane and choose the appropriate command on the **Window** menu.

CHAPTER

2

Getting Started with Modeling

This chapter explains the basic features of Together Developer, including: how to navigate the Together environment, how to create a project, and how to create diagrams.

This chapter includes the following topics:

Setting global modeling options

There are numerous options for customizing the appearance and behavior of Together Developer.

To adjust global options:

1 From the **Tools** menu, choose **Options**. The **Options** dialog window opens.

2 Click the *Together for VS* folder to view Together-specific configuration options. These options are available at the following sublevels: *Default*, *Solution*, *Project*, and *Diagram*.

3 Select the *Default* | *General* subfolder. The General options appear.

4 Click the *Automatically enable Together for VS support for new projects* field. A drop-down arrow appears. You can globally deactivate Together Developer modeling support for all new projects (using this option) or for all of your open projects (using the click *Automatically enable Together for VS support for opened projects* option).

Note The *Automatically enable Together for VS support for opened projects* option works only for projects that have never been exposed to Together support.

5 Click the drop-down arrow and select *False* from the drop-down list. This disables Together modeling support. Be sure to set the value as *True* to complete this tutorial.

Note The Together modeling features are activated for C# and Visual Basic projects by default.

6 Select the *Default* | *Diagram* | *Appearance* subfolder.

7 Click the *UML in color* | *Enable UML in color* field. A drop-down arrow appears. Select *True* from the drop-down list. This enables the **UML in Color** feature.

8 Click the *Grid* | *Grid style* field in the same folder. A drop-down arrow appears. Select *Lines* from the drop-down list. You can customize color, size, and style of the grid lines on your diagrams.

9 Select the *Default* | *Diagram* | *Layout* subfolder.

10 Click the *General* | *Layout algorithm* field. A drop-down arrow appears. Select *Hierarchical* from the drop-down list. This algorithm will now be used for arranging elements on your diagrams.

11 Click **OK** to close the **Options** dialog window.

Creating a project and opening the Model and Diagram Views

This guide models a simple video store project to explore the various features of Together.

Note This guide models a C# project.

To create a new project:

1 From the main menu, choose [File | New | Project...]. The **New Project** dialog window opens.

2 From the **Project Types** pane, choose *Visual C# Projects*.

3 From the **Templates** pane, choose *Empty Project*.

4 Enter “Video Store” as the project and solution names.

5 Select a proper location for the project.

6 Click **OK**.

The project is created and displayed in the Solution Explorer.

Tip To quickly open the Solution Explorer, press **Ctrl+Alt+L**.

To open the Model and Diagram Views:

1 From the menu, choose [View | Together for VS Model View]. The **Model View** opens.

2 Expand the Video Store project node.

3 Double-click the *default* package diagram node  . The Diagram View opens.

Although the Model View opens initially as a free-floating window, it is a dockable window. The docking areas are any of the four borders of the Visual Studio window. You can position the Model View according to your preferences.

Creating a Use Case Diagram

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

To create a use case diagram:

1 In the Model View, right-click the **Video Store** root node  , and choose [Add | Other Diagram...] from the context menu.

2 In the resulting dialog window, choose **Use Case Diagram**. Enter “Video Store Use Case” as the name.

3 Click **OK**. The new use case diagram opens in the Diagram View.

Tip You can use this dialog window to create any of the diagrams supported by Together Developer.

Populating the diagram

Use case diagrams are scenarios describing *what* your system does. These diagrams contain actors, use cases, and communication links as three main items of interest. The Toolbox and diagram context menu contain the design elements to populate the diagram.

To populate the diagram:

1. Use the Toolbox to add design elements to the use case diagram. To open the Toolbox, choose [View | Toolbox] from the menu.
2. To view the model elements, click **UML Use Case Diagram** on the Toolbox.
3. Click the Actor button  .
4. Click the diagram background. The in-place editor automatically activates.
5. Type "Clerk" as the actor name. Press **Enter** to apply the name and close the in-place editor.

Tip To activate the in-place editor for diagram elements, double-click the textual caption on the element.

Adding a system boundary

Add a system boundary to the diagram to separate the inventory system from the external actors.

To add a system boundary:

1. Click the System Boundary button  .
2. Click the diagram background to the right of the Clerk.
3. Using the in-place editor, enter "Inventory System" as the system boundary name.

Tip Alternatively, click the diagram background and drag the mouse over the diagram background to draw the System Boundary to a specific size.

To resize the subject or any diagram element, select the subject node on the diagram. Drag the corner of the element to the desired size. The element expands or contracts in the direction of the cursor.

Adding use case elements and communication links

Use case elements represent a summary of scenarios for a single task, such as finding a movie by its title. Communication links show associations between actor and use case elements.

To add use case elements and communication links:

1. Click the Use Case button  and create two use cases inside the Inventory System. Name the use cases "Find item by keywords" and "Find item by title."
2. Click the Communicates  . Drag-and-drop from the actor to the use case. Repeat this step to create a link for each use case.

Adding a stereotype

Together supports the use of stereotypes. You can adhere to UML-defined stereotypes or customize stereotypes based on your requirements.

To add a stereotype:

1. Select the **Clerk** actor to view the properties for Clerk in the **Properties Window**. You can open the **Properties Window** by choosing **Properties Window** from the View menu. Alternatively, press **F4**.
2. In the **Properties Window**, click the *Stereotype* field. A drop down arrow appears.
3. In this field, select on the list or type in "*manager*".

Tip Once a stereotype has been added to a diagram element, you can make changes to the stereotype field using the in-place editor.

Completing the scenario

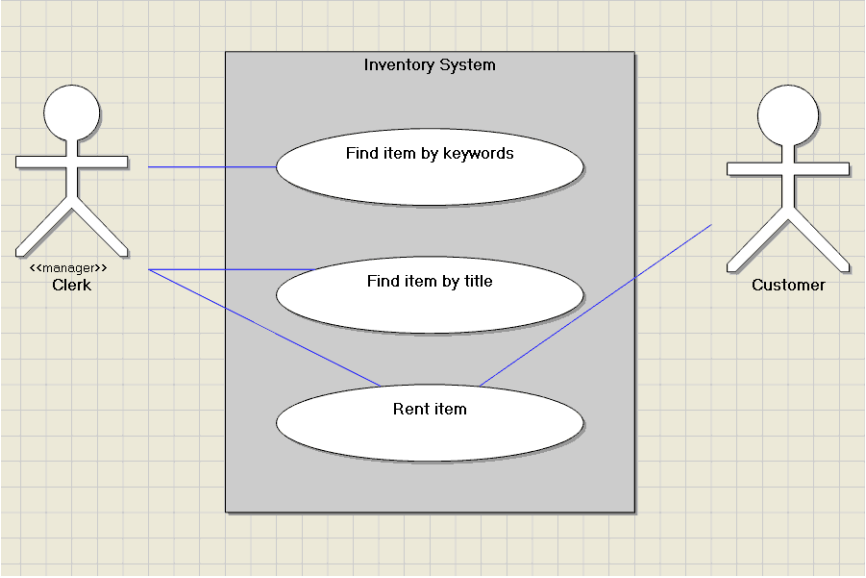
To complete the Video Store Use Case diagram, add additional use case and actor elements and the communication links between the appropriate elements.

To complete the diagram:

1. Add another use case inside *Inventory System*. Type "Rent item" as the use case name.
2. To the right of *Inventory System*, add another actor. Type "Customer" as the actor name.
3. Using the Communicates relationship, link the *Rent item* use case to both the Customer and Clerk actors.

Your diagram should be similar to the one shown in Figure 1:

Figure 1 Use case diagram for Video Store project



Creating a Class Diagram

Together Developer automatically creates a namespace (package) diagram for each namespace in a project. The Video Store project has a default namespace diagram that represents the project at the root level.

First, you need to create a new class diagram. Although it is possible to place classes on namespace (package) diagrams, it is recommended that you create separate class diagrams.

Creating the Base Classes diagram

To create the Base Classes diagram:

1. In the Model View, right-click the **Video Store** root node, and choose [Add | Class Diagram] from the context menu. The blank Class Diagram opens in the Diagram View.
2. In the **Properties Window**, edit the diagram **Name** field to rename the diagram to "Base Classes."

To populate the diagram:

1. Use the Toolbox to create a class. Click **UML Class Diagram** in the Toolbox to reveal the Class Diagram elements.

2 Click the Class button  .

3 Click the diagram background. Using the in-place editor, type “Store” as the class name.

4 Select the **Store** class to view its details in the **Properties Window**.

5 Select the *Stereotype* field, click the drop-down arrow, and select *place*. The class node turns green in accordance with the UML In Color profile.

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

6 Repeat the steps 1-5 to create classes for `Clerk` (stereotype = role) and `Item` (stereotype = description.)

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

- Right-click the diagram background and choose [Add | Class] from the context menu.

  or

- In the Model View, right-click the Base Classes diagram and choose [Add | Class] from the context menu.

  or

- Press **Ctrl+L**.

Setting compartment controls

You can collapse or expand compartments for the different members of class and interface elements. By default, compartments are displayed on diagrams. You can use the Options dialog window to set viewing preferences for compartment controls so that they are displayed as a straight line or with compartments. Compartment controls are particularly useful when you have large container elements with content that does not need to be visible at all times.

To view the compartment controls:

1 From the **Tools** menu, select **Options**. The Options dialog window opens.

2 Click the **Together for VS | Default | Diagram | Appearance** folder.

3 Select the **Nodes** group.

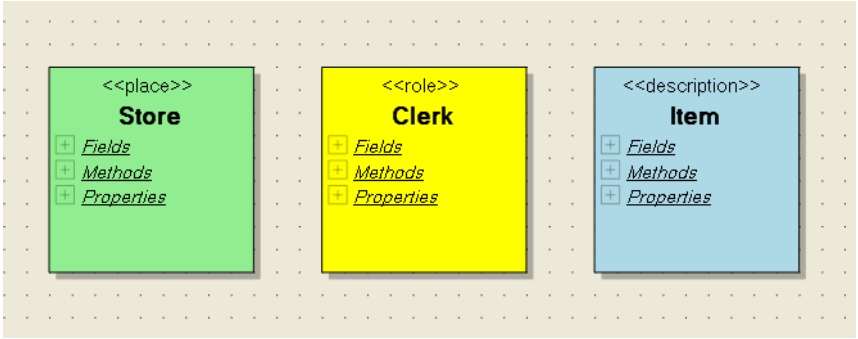
4 Click the *Show compartments as line* field.

5 Click the drop-down arrow and select *False.*

6 Click **OK**.

Your diagram elements should be similar to Figure 2:

Figure 2 The Base Classes diagram with the Store, Clerk, and Item classes



To collapse or expand compartments:

1 Select the class or interface on the diagram.

2 Click the "+" or "–" in the left corner of the compartment (it must have at least one member).

Adding methods and fields

As a part of the requirements identified in the use case diagram for the Video Store Inventory system, Table 2 lists the methods and fields that you need to add to the classes you have created:

Table 2 Requirements for the Video Store sample project

Class	Methods	Fields
Store	address storeNumber hours phone	findByTitle findByKeywords findByItemNumber
Clerk	employeeId name	
Item	itemNumber title	getItemNumber setItemNumber getTitle setTitle

To create members for classes:

1 Open the **Base Classes** diagram, right-click the **Store** class and choose [Add | Field] from the context menu. Alternatively, select the **Store** class on the diagram or in the Model View and press **Ctrl+W**.

2 The in-place editor activates. Using the in-place editor, name the field `address:string`. When you use the in-place editor to name a class member on the diagram, you can also enter its type.

Tip Alternatively, you can activate the in-place editor by selecting the field and pressing **F2** on your keyboard.

The "+" symbol indicates that the field is public.

Tip Use "+" for public, "–" for private, "#" for protected, and "~" for namespace (package) local.

3 In the **Properties Window**, click the *Visibility* field and choose *public* from the drop-down list. By default, Together creates *public* System.Int32 fields and *public* void methods.

Tip Once you click the *Visibility* field, press the `Tab` key on your keyboard and use the down-arrow on your keyboard to scroll through the different visibility values.

4 To add a method to the class, right-click the **Store** class and choose [Add | Method] from the context menu. Alternatively, select the **Store** class on the diagram or in the Model View and enter **Ctrl+M**.

5 Using the in-place editor, name the method `findByTitle:void`.

6 Repeat the previous steps to add the remaining fields and methods for the **Store** class as listed in Table 2.

7 Continue adding the fields and methods listed in Table 2 to the **Clerk** and **Item** classes. For the `getItemNumber` and `getTitle` methods, specify *string* as the return type.

Tip Select a class member such as a field or a method on the diagram or in the Model View and press the **Insert** key on your keyboard to quickly add another member of the same type to the current class. After the new member has been added, press **F2** on your keyboard to activate the in-place editor. You can also use the shortcut keys displayed from the context menus to quickly add members and diagram elements.

8 Select the `storeNumber` field of **Store**. In the **Properties Window**, enter "Store Number" as the alias value.

Tip The *alias* property is useful for specifying a different name from that in the actual source code.

The alias changes only the visual label for the field in the diagram. The field remains `storeNumber` within the source code.

Tip To view the source code, double-click the class node or right-click the field and choose **Go to Definition** from the context menu.

Creating relationships and links

The Video Store project includes two relationships:

- A `Clerk` is associated with a `Store`
- A `Store` consists of many `Items`

You can create an association link to establish the relationship between `Clerk` and `Store` as "client" and "supplier." For the second relationship, you can create an aggregation since `Store` has several `Items`. By using the **Link By Pattern** tool, you can easily implement the aggregation as a collection.

To create relationships and links:

1. Click the **Association** link button in the Toolbox. Click **Clerk** and then drag-and-drop the link to **Store**. This action establishes the relationship from client to supplier. Notice that as you move over a class, a blue border highlights the location that Together Developer recognizes as a valid destination for dropping the end of the link.
2. Click the **Link by Pattern** button in the Toolbox. Click **Store** and then drag-and-drop the link to **Item**. The Pattern Wizard opens.
3. In the Pattern Wizard, expand **Bundled Patterns (C#)** | **TCC code templates** | **Links** and select **Aggregation as Collection**.
4. Type "itemList" in the **Name** field to indicate a collection and click **OK**.

Together Developer automatically adds the link information to the code by placing tags above the field. In this example, the link is an aggregation linked to the **Item** class. By double-clicking a class in the Diagram or Model Views, you can open its source code in the Visual Studio Editor to view such tags. For example, the source for the `itemList` field in the **Store** class is as follows:

```
///<link>aggregation</link>
/// <supplierCardinality>0..*</supplierCardinality>
/// <associates>Item</associates>
private System.Collections.ArrayList itemList;
```

Creating hyperlinks

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.

Hyperlinking diagrams

The Video Store has a `findByKeywords` operation used for searching for items. The requirements for this operation are defined as the use case you created earlier, *Find item by keywords*.

To create a hyperlink between the use case and the operation:

1. Open the Video Store Use Case Diagram.
2. Right-click the **Find item by keywords** use case and choose [Hyperlinks | Edit...] from the context menu. The **Hyperlinks** dialog window opens.
3. The pane on the left includes two tabs: **Model Elements,** for the elements of the current solution, and **External Documents,** for any external files or URLs.
4. From the **Model Elements** tab, select [Video Store | Base Classes | Store | findByKeywords].

Tip You can use the **External Documents** tab to connect to an existing document. This is helpful for linking an element such as *Find item by keywords* to requirements specified in another location. When you select the hyperlink, Together Developer launches the default application associated with the file.

5. Click **Add >>** to add the element to the *Selected* pane on the right.
6. Click **OK**.

The use case with the newly created hyperlink is highlighted in blue font.

To test the hyperlink, right-click the use case and choose [Hyperlinks | Video Store.Store.findByKeywords] from the context menu. The class diagram containing the corresponding operation opens with the operation selected.

Creating a Sequence Diagram

Sequence Diagrams are used to design dynamic aspects of an object model. Together Developer provides the following modes for creating Sequence Diagrams:

- A simple "sketch pad"
- Two-way with source code and classes
- Generating a Sequence Diagram from an existing method

Creating the initial sequence

You can create a Sequence Diagram in the Video Store project to model some of the steps necessary to rent a video.

To create a sequence diagram:

1. In the Model View, right-click the **Video Store** root node and choose [Add | Other Diagram...] from the context menu.
2. In the resulting dialog window, choose **Sequence Diagram**.
3. Enter "Rent Movie Sequence" as the diagram name and click **OK**.

4 From the Toolbox, select **UML Sequence Diagram**.

5 Click the Actor button  .

6 Click the diagram background.

7 Using the in-place editor, type “Clerk” as theactor name.

8 Using the Toolbox, create an object  and name it “theStore.”

The steps that follow model the requirement of requesting an item from theStore.

Validating a customer. The Clerk (having already captured the scanned Customer ID from their membership card) will send a message to the Store asking for a validation of the Customer (at the same time it checks for any overdue rental fees).

9 To add a message link, click the Message button  , then drag-and-drop from *Clerk* to *theStore*.

Associating an object with a class

In Sequence or Collaboration Diagrams you can create associations between objects and classifiers.

Note You can associate an object with all classifiers including classes, interfaces, structures, modules (in Visual Basic projects only), enumerations, or delegates.

Instantiated classifiers for an object can be selected from the model, or the classifiers can be created and added to the model. In the interaction diagrams there are two commands available on the lifeline context menu:

- **Add** that creates a new classifier in the model, and
- **Choose class** that provides a list of available classifiers.

In this example, we will associate `theStore` object with the `Store` class.

To associate an object with a classifier:

1 Select **theStore** object on the diagram.

2 In the **Properties Window**, select the **Instantiates** field and click the chooser  button. The **Choose Type to Instantiate** dialog window opens.

3 In this dialog window, choose the **Store** class and click **OK**.

The diagram is updated. Notice that the object name is followed by the name of the associated classifier.

Note After you associate a classifier to an object, you can right-click the object and click **Go to Class Definition** from the context menu to switch to the classifier source code.

Adding a new operation to a message link

Message links can be associated with the methods of the recipient class. You can select methods from existing methods in the recipient class or create new ones. This is accomplished by two commands provided by the message context menu, **Choose method** and **Add**.

In this example, we will associate the **1** link with the **findByTitle** operation of the **Store** class.

To do this:

1. Select the **1** message link on the **Rent Movie Sequence** diagram.
2. In the **Properties Window**, select the **Operation** field and click the chooser button. The "Choose operation name" dialog window opens.
3. In this dialog window, choose the **Store | findByTitle** operation.
4. Click **OK**.
5. Add a new message from **Clerk** to **Store**.
6. Right-click the new message link. Choose [Add | Method] from the context menu. A new operation named `Method1():void` is created in the **Store** class.
7. In the **Properties Window** for the message link, select the **Operation** field, to rename the new method, and enter, `validateUser():float`.
8. The **Rename Operation** dialog box opens indicating that the `validateUser` operation does not exist in the **Store** class. The dialog box gives you the option to either create a new method or rename the existing method. Select **Rename** to rename the existing operation.

You can also use the **Choose method** command on the message link context menu to associate the message link with an existing method in the object's recipient class.

Note After you associate a method to a message link, you can right click the object and select **Unlink method** from the context menu to remove the association.

Adding a parameter value to the validateUser method

You can easily add parameters to the `validateUser` method and reflect the parameters on the Sequence Diagram.

To add parameters to the validateUser method:

1. Open the Video Store *default* Class Diagram.
2. In the **Diagram View**, click the `validateUser` method in the `Store` class.
3. In the **Properties Window**, choose the **Params** field.
4. Enter `int ID` in the field.

5 Press `Enter` to complete the changes.

Return to the Rent Movie Sequence diagram.

Generating a Sequence Diagram

You can create Sequence and Collaboration Diagrams and populate them using buttons on the Toolbox. You can also generate Sequence Diagrams automatically from methods on a Class Diagram.

In this example, we will generate a simple Sequence Diagram using the Video Store project.

To do this:

1 Double-click the Base Classes diagram node in the Model View. The Base Classes diagram opens in the Diagram View.

2 Right-click the `findByKeywords` method on the `Store` class and choose **Generate Sequence Diagram...** from the context menu.

3 The *Generate Sequence Diagram* dialog box opens. Leave all the settings unchanged. Click **OK**.

The **Store.findByKeywords** Sequence Diagram is generated and opens in the Diagram View.

You can close this Sequence Diagram without saving any changes to it.

Tip To generate a collaboration diagram from a method, first generate a Sequence Diagram and then convert the diagram into a Collaboration Diagram.

Converting between Sequence and Collaboration Diagrams

You can convert between Sequence and Collaboration Diagrams. However, when you create a new diagram in Together Developer using the **Add New Diagram** dialog window, you must specify whether it is a Sequence or Collaboration Diagram.

To convert the Rent Movie Sequence Diagram to a Collaboration Diagram:

1 Open the **Rent Movie Sequence** diagram.

2 Right-click the diagram background and choose **Show as Collaboration** from the context menu. The **Rent Movie Sequence** collaboration diagram opens.

3 Open the Toolbox by choosing **Toolbox** from the **View** menu.

4 On the Toolbox, click **UML Sequence Diagram** to display the collaboration diagram elements.

5 Right-click the diagram background and choose [Layout | Do Full Layout] on the context menu.

6 To convert back to the Sequence Diagram, right-click the Collaboration Diagram background and choose **Show as Sequence**. The **Rent Movie** Sequence Diagram opens.

Working with diagrams

Any diagram is displayed as a field with nodes (representing model elements) and their relationships (link lines).

It is assumed that the **Base Classes** diagram is currently opened in the Diagram View. You may perform the following basic actions with the elements:

- Move or resize the nodes by dragging the nodes and their borders with your mouse in the Diagram View
- Cut, copy, paste, rename, or delete one node or a group of nodes by choosing appropriate commands from the context menu or on the **Edit** menu
- Rename or edit the content of the node by doing one of the following:
  - Double-clicking the name of the node

    or
  - Pressing **F2**

    or
  - Using the **Properties Window**
- Open subnamespace (subpackage) nodes by double-clicking the node.
- Align several nodes against each other. For example:

  a. Select **Store** and **Item** classes on the **Base Classes** diagram.

  b. Right-click the selection.

  c. Choose [Alignment | Top] from the context menu.

  Result: These two nodes are aligned. Alternatively, you may align to the left, right, bottom, vertical, or horizontal center.
- Arrange general layout of the diagram. For example:

  a. Right-click the **Base Classes** diagram background.

  b. Choose [Layout | Layout for Printing] from the context menu.

  Result: All nodes on the diagram are rearranged for better printing. Alternatively, you may layout nodes according to the selected algorithm[1], or optimize sizes of the selected elements.
- Manually reroute relationship links. For example:

1. The arrangement algorithm is configured at the “Layout” category of the project modeling properties. See “Setting global modeling options” on page 11.

- Click the link between the **Store** and **Clerk** nodes.
- Drag the line up. Together Developer automatically reshapes the link the way you want.

Searching and replacing

You can use all the commands under the [Edit | Find and Replace] menu in Visual Studio to locate elements or members on your diagrams. You can search with regular expressions and wildcards. You can adjust the search scope (current file, all open files, or current project).

To find occurrences of the Store class:

1. In the **Model View**, select the Video Store project root.
2. Choose [Edit | Find and Replace | Find...] on the main menu. The **Find** dialog window opens.
3. Type "Store" in the **Find what:** field. Choose **Current project** in the search options. Click **Find Next**.

The **Base Classes** diagram is opened with the **Store** class selected. If you click **Find Next** button again, the next occurrence of this class opens, and so on.

Searching for usages

You can search for usages of an element or a member in an implementation project.

To search for usages of the Item class:

1. Right-click the **Item** class on the **Base Classes** diagram.
2. Choose **Search for Usages...** from the context menu. The **Search for Usages** dialog window opens.
3. Check the **Include Usings/Imports** check box.
4. Click **Search**.

Together Developer begins searching. When the search is complete, the **Search for Usages** window opens. In this window, the search results are displayed as a tree view.

Undoing and redoing changes

Together Developer supports multiple undo/redo operations for any changes you make to your diagrams.

Choose [Edit | Undo/Redo] on the main menu or the toolbar, or press **Ctrl+Z/ Ctrl+Y**.

Printing diagrams

Use the context menu in the Model View or the File main menu (**File | Print...** or **File | Export Diagram to Image...**) to print diagrams or export diagrams to image files.

Zooming

Press +/– on the numeric keypad.

Using the Overview

The overview feature of the Diagram View provides a thumbnail view of the current diagram. The **Overview** button (

) is located in the bottom right corner of every diagram.

To use the overview feature:

1. On the **Base Classes** diagram, click the **Overview** button. The pane expands to show a thumbnail image of the current diagram.
2. Use the mouse to click the shaded area and drag it. This is a convenient way to scroll around the diagram.
3. Alter the size of the Overview pane by clicking the upper left corner of the pane and dragging to resize it.

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

Using the Model View

Use the Model View as your primary navigational view in Together Developer to manage diagram elements. Using the context menus in the Model View, you can open, create, and delete diagrams and design elements.

The Model View consists of a number of icons and context menu commands that you should become familiar with. Explore the context menus of the different elements and diagrams as you encounter them.

Using the context menu of your root project node  , you have access to the following features of Together Developer:

- Adding diagrams to your project
- Adding elements such as namespaces, classes and interfaces
- Importing from XMI/Exporting to XMI (for UML 1.4 projects)
- Generating documentation
- Transforming code from design project

- Managing CaliberRM requirements
- Using patterns
- Running QA audits and metrics

Viewing references

The Model View enables you to view class diagrams for references included in your project. You can add references to your project using the Solution Explorer.

To view the MsCorLib.dll in the Video Store project:

1. Expand the *References* node and the *mscorlib* node in the Model View.
2. Double-click the following node: [References | mscorlib | default]. The default diagram opens in the Diagram View.

You can expand the Microsoft and System folders to view other class diagrams as well.

Showing expandable diagram nodes

By default, the diagram nodes in the Model View are expandable. You can control whether you can expand nodes in the Model View to show their contents.

To make diagram nodes expandable:

1. From the Tools menu, choose **Options**. The Options dialog window opens.
2. Click the *Together for VS* folder to view Together-specific configuration options.
3. Select the *Default | Model View* subfolder. The Model View options appear.
4. Click the *Show diagram nodes expandable* field. A drop-down arrow appears.
5. Click the drop-down arrow and select *True* from the list.
6. Click **OK** to close the dialog window and apply the changes.

The diagrams in Model View display as expandable nodes. Expand the node for the **Base Classes** diagram to view its elements.

Finding diagram elements

When selecting a diagram element in the Model View, the element is not automatically selected in the Diagram View. To show an element selected from the Model View in the Diagram View, right-click the element in the Model View and choose **Select on Diagram**. Using this command, you can open the corresponding diagram of the element (for example, **Clerk**), and highlight the element on it.

Similarly, while working in the Diagram View, you can use the **Synchronize Model View** command to navigate directly to an element in the Model View. Right-click a diagram element, and choose **Synchronize Model View** from the context menu.

Using the Properties Window

You can use the **Properties Window** to view and edit values such as alias, name, visibility, and others.

To add details to the Item class:

1. Open the **Base Classes** diagram and select the **title** field of the **Item** class.
2. Open the **Properties Window** by choosing [View | Properties Window] from the menu. Alternatively, press **F4**.
3. In the *Initial* field, type "Title" (include the quotes).

The source code automatically updates to reflect this change. You can easily navigate from the Diagram View to the source code.

To open the source code editor for the title field:

1. Right-click the **title** field of the **Item** class on the diagram.
2. Choose **Go to Definition** from the context menu.

The source code editor opens, highlighting the appropriate code. Notice that the initial value for the title field is defined as specified within the **Properties Window**.

Managing diagram views

View management controls which diagrams and elements are displayed in Together Developer.

This section explains the following features for specifying diagram views: detail level, showing/hiding information, and global filters.

Specifying a detail level

You can use a 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. On the Tools menu, select **Options**. The Options dialog window opens.
2. Select the **Together for VS | Default | Diagram | Appearance** folder, the **General** group.

3 Select the *Diagram detail level* field.

4 Click the drop-down arrow near this field and choose the detail level.

5 Click **OK** to close the dialog window.

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

Showing and hiding information

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

The following example demonstrates how to hide and show information on the Base Classes diagram created earlier in this guide.

To hide some elements:

1 Right-click the **Store** class node in the diagram and choose **Show/Hide...** from the context menu. The **Show Hidden...** dialog window opens.

2 In this dialog window, the following lists are displayed: **Diagram Elements** and **Hidden Elements**. On the first list, select the elements you want to hide and click **Add >>**.

3 Click **OK**.

The selected elements disappear from the diagram.

To show the hidden elements:

1 Open the **Show Hidden...** dialog window and click **<< Remove All**.

2 Click **OK**.

To hide the entire `Store` class, right-click this class and choose **Hide**.

To show the `Store` class, right-click the diagram background and choose **Show/Hide...** from the context menu. The **Show Hidden...** dialog window opens.

The `Store` class appears in the *Hidden Elements* list. Select the `Store` class, click **<< Remove**, and then click **OK**. The `Store` class displays on the diagram again.

Note All hidden parts will be still processed by Visual Studio and Together Developer.

View filters

For global control over the diagram view, you can use the filters in the Options dialog window. Choose [Tools | Options...] from the menu. In the options dialog window, choose the [Together for VS | Default | Diagram | View Filters] folder.

To specifically filter members, you can set the *Show members* property to *False*.

To filter members:

1. Click the *Show members* field.
2. Click the drop-down arrow and select *False.*
3. Click **OK**.

This results in disabling the fields, methods, non-public members as well as any inner classifiers.

Since inner classifiers are treated as members of the container element, the following filters do not filter inner classifiers:

- Show classes
- Show delegates
- Show enumerations
- Show interfaces
- Show structures

CHAPTER

3

Using Together Developer Features

The information in this chapter is specific to using the special features of Together Developer. For further information, refer to the online help for Together Developer.

This chapter includes the following topics:

Using patterns

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

For the following example, assume that the **Item** class requires a dynamic system so that after an **Item** has been returned, the counter is updated and other systems are notified. The **Observer** pattern is useful for designing such a system. To apply the **Observer** pattern information to the **Item** class, use the Pattern Wizard.

Note For instructions on how to create a link based on an existing pattern by using Create Link by Pattern, see "Creating relationships and links" on page 20.

Applying a pattern

To use the Pattern Wizard:

1. Right-click the **Base Classes** diagram background and choose **Create by Pattern**. The Pattern Wizard opens.
2. From the *Patterns pane* on the left, choose the **Bundled Patterns (C#) | GoF | Behavioral | Observer** pattern.
3. In the *Pattern Properties* pane on the right, change the *Class Subject* name to `Item`.
4. Accept the other default properties settings and click **OK**.

The diagram is updated with the pattern.

Item is updated with the notification and observer methods (`attach` and `detach`). The other classes and interfaces have been created ready for use. Together Developer recognizes the pattern and visualizes the element on the diagram as an oval shape. In addition, it lists the participants of the pattern and the pattern links. You can expand the *Participants* node on the oval **Observer** element to view pattern participant information.

The pattern elements on the diagram have specific pattern actions that you can choose relevant to the pattern. Right-click the **Observer** element and choose **Add** from the context menu to view the specific pattern actions available.

Creating a pattern

To use the Create Pattern Wizard:

1. On the **Base Classes** diagram, select the Store and Item classes and the link between them.
2. Choose **Save as Pattern** from the context menu. The Create Pattern Wizard opens.
3. Type the name, filename and description for your new pattern.
4. Click **Next >>**.
5. In the pattern parameters, uncheck the **Use Current** check boxes.
6. Click **Next >>**.
7. Select the Custom Patterns folder.
8. Click **Finish**.

Running quality assurance

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

Running audits

Audits help you unobtrusively enforce company standards and conventions and improve what you do. When you run audits, you select specific rules to which your source code should conform to. The results display the violations of those rules so that you can examine each problem and decide whether to correct the source code or not. Together provides a wide variety of audits to choose from, ranging from design issues to naming conventions, along with online descriptions of what each audit looks for and suggestions on how to fix violations.

To run audits on the Video Store project:

1. Right-click the **Base Classes** diagram background and choose **QA Audits**. The **QA Audits** dialog window opens.
2. In the *Scope* drop-down list, choose Model.
3. On the left side of the dialog window, expand the nodes of the audit categories (Coding Style, Declaration Style, and so on) to view the available audits. Check or clear the appropriate check boxes to indicate which categories and concrete audits to run. As you click an audit, its description displays in the lower pane of the dialog window. Categories that have names in gray are partly selected.
4. For each audit, the severity level and other audit-specific options are displayed on the right pane of the dialog window. Change the settings if necessary.
5. For this example, accept the default settings. Click **Start**. The *Operation in progress* message box opens displaying a status bar that indicates the progress completed, until the process finishes.

Results: The Audits dialog window with a table is displayed. In this table, all the violation cases are listed.

Working with the audit results view

Using the audit results view, you can perform several tasks, such as:

- Sorting or grouping audit results
- Opening the corresponding source code for an audit violation
- Viewing the description of an audit
- Printing and saving audit results
- Refreshing (recalculating) the audit violations that are currently displayed
- Restarting the audit calculation

Although the audit results window opens initially as a free-floating window, it is a dockable window. The docking areas are any of the four borders of the Visual Studio window. You can position the audit results window according to your preferences.

Sorting or grouping audit results

When viewing audit results, you might want to compare and organize the items in the results report. You can sort and group the results as follows:

- ***Sorting by one column:*** To sort all the items according to the values for a specific column, click the column heading. Click once to sort in ascending order. Click a second time to sort in descending order.
- ***Grouping audit results:*** To group items according to the current column, right-click the Audit results table and choose [Group By... | <command>] from the context menu. This enables you to organize the results by changing the relationship of rows and columns. Audit results display in tabbed pages. To ungroup the results, right-click the table, and choose [Group by... | Ungroup] from the context menu.

Opening source code and viewing an audit description

The results report is tightly connected with the diagram elements and the source code. Using the report, you can navigate to the specific location in the source code where the violation actually takes place.

To open the source code for an audit violation:

In the Audit results view, double-click any entry. Together Developer opens the source in the Visual Studio Editor, and highlights the line of code.

To view an audit description:

In the **Audit results** view, right-click an entry and choose **Show Description** from the context menu. This opens a description of the corresponding audit.

Printing audit results

You can print the entire table of audit violations or select specific rows and columns to print.

To print audit results:

1. Select the rows of the table that you want to print. If you want to print the entire list, do not select anything.
2. Click the **Print** button  on the toolbar. The **Print audit** dialog window opens.
3. Choose the scope of the results to print using the *Select View* list box.
   Note: Unless the results have been grouped using the **Group by...** commands, the Active Group option is not enabled in the dialog window. The possible view options are:
   - *All Results* — If the results are *grouped*, choosing *All Results* prints a report for all groups in the current tabbed page. If the results are not *grouped*, then all results print for the current tabbed page.

- *Active Group* — If the results are *grouped*, then you can select a group in the current tabbed page. The printed report will contain the results from the selected group.
- *Selected Rows* — You can select single or multiple rows in the audit results report view. Choosing *Selected Rows* prints a report for such selections.

4 If desired, specify the print zoom factor in the *Print zoom* field, or check *Fit to page* if you want to print the results on a single page. If checked, the *Print zoom* field is disabled.

5 If necessary, adjust the page and printer settings:

- Click the *Print* list box, and choose the *Print dialog* command to select the target printer.
- When the **Print audit** dialog window is closed, choose [Tools | Options...] on the main menu. In the options dialog window, choose the folder [Together for VS | Default | Diagram | Print], and set up the paper size, orientation, and margins.
- Click the drop-down arrow to the right of the Preview option to open the preview pane. Use the *Preview zoom (auto)* slider, or *Auto preview zoom* check box as required. Click the up arrow to the right of the **Preview** option to close the preview pane.

6 Click **Print** to open the system print dialog window, and send the file to the printer.

Saving audit results

Export audit results to an XML or HTML file so you can share them with team members or review them later.

To save audit results:

1 Select the rows of the table that you want to save. If you want to save the entire list, do not select anything.

2 Click the **Save** button  on the toolbar.

3 In the *Save Audit Result* dialog window that opens, choose the scope of the results to export using the Select View list box.

Note: Unless the results have been grouped using the **Group by** commands, the Active Group option is not enabled in the dialog window. The possible view options are:

- *All Results* — If the results are *grouped*, choosing *All Results* generates a report for all groups in the current tabbed page. If the results are not *grouped*, then all results for the current tabbed page are exported.
- *Active Group* — If the results are *grouped*, then you can select a group in the current tabbed page. The exported report will contain the results from the selected group.
- *Selected Rows* — You can select single or multiple rows in the audit results report view. Choosing *Selected Rows* generates a report for such selections.

4 In the *Select Format* list box, select the format for the exported file:

- *XML* — Generates an XML-formatted report.
- *HTML* — Generates an HTML-formatted report.

Selecting HTML format activates the following check boxes:

- *Add Description* — This adds hyperlinks to the descriptions (stored under the Together Developer installation folder) from the results file.
- *Launch Browser* — This option opens the generated HTML file in the default browser.

5 Click **Save** to save the results in the specified location.

Refreshing and restarting audits

You can update or refresh the results table using the toolbar:

- Click the **Refresh** button to recalculate the results that are currently displayed.
- Click the **Restart** button to open the **QA Audits** dialog window, where you can change the settings as necessary and run the audits again. The new results replace the results that are currently displayed.

Close the **Audits** view by clicking the **Close** button in the upper-right corner of the dialog window.

Choosing specific elements to run audits

Using the Audits feature of Together Developer, you can selectively determine which classes to run a set of audits against.

To run audits on a particular class element:

1 Go to the Base Classes diagram, right-click the `Item` class and choose **QA Audits**. This restricts the *Scope* of the audit to the `Item` class.

2 Accept all other default settings, and click **Start**. The Audit results view opens. As you run audits on your project, the results display in the Audit results view in a tabbed page format. The most recent audits ran have focus.

3 You can close the audit results (individual tabbed pages) by right-clicking the corresponding tab and choosing **Close**. Click the Close button in the upper right corner of the results dialog window to close the audit results.

Running metrics

Metrics evaluate object model complexity and quantify your code. When you run metrics, you select measures to apply to your source code.

Metrics results can highlight parts of code that may need to be redesigned, or they can be used for creating reports and for comparing the overall impact of changes in a project.

Together provides a wide variety of metrics to choose from, ranging from numbers of entities to complexity analysis, along with online descriptions of what each metric looks for and suggestions on how to fix violations.

To run metrics on the Video Store project:

1. Right-click the Base Classes diagram background and choose **QA Metrics**. The **QA Metric** dialog window opens.
2. In the *Scope* drop-down list, choose Model.
3. On the left side of the dialog window, expand the nodes of the metrics categories (Basic, Cohesion, and so on) to view the available metrics. Check or clear the appropriate check boxes to indicate which categories and concrete metrics to run. As you click a metric, its description displays in the lower pane of the dialog window. Categories that have names in grey are partly selected.
4. For each metric, the aggregation type and other metric-specific options are displayed in the right side of the dialog window. Change the settings if necessary.
5. For this example, accept the default settings and click **Start**. The **Operation in progress...** message box opens displaying a status bar that indicates the progress completed, until the process finishes.

Results: The **Metrics** dialog window is displayed. On the left side of this dialog window, the model structure is displayed. On the right side of the dialog window, the table with metric results for the selected node is displayed. In this table, rows are for the selected model nodes, and columns are for the selected metrics.

Working with the metrics results view

Using the metrics results view, you can perform several tasks, such as:

- Sorting nodes in the table
- Showing or hiding some of the metric results
- Opening the source code corresponding to the selected node
- Viewing the description of a metric
- Saving metric results
- Creating Kiviat and Bar charts
- Refreshing (recalculating) the metric results that are currently displayed
- Restarting the metric calculation

Although the metric results window opens initially as a free-floating window, it is a dockable window. The docking areas are any of the four borders of the Visual Studio window. You can position the metric results window according to your preferences.

Sorting metric results

When viewing metric results, you might want to sort and compare the items in the results report. To sort all the items according to the values for a specific column, click the column heading. Click once to sort in ascending order. Click a second time to sort in descending order.

Showing or hiding some of the metric results

You can show or hide some of the information displayed in the metric results table by using the following buttons on the toolbar:

- **Show Namespaces**  — click this button to show/hide namespace(s) under the selected node as separate row(s) with aggregated values of its classes (sum, minimum, maximum and so on, depending on the **Aggregation** metric parameter)
- **Show Classes**  — click this button to show/hide class(es) under the selected node as separate row(s) with aggregated values of its methods
- **Show Methods**  — click this button to show/hide method(s) under the selected node
- **Show Children**  — click this button to show content of subnodes of the selected node

Opening source code and viewing a metric description

The results report is tightly connected with the diagram elements and the source code. Using the report, you can navigate to the specific location in the source code where the corresponding node is described.

To open the source code for a selected node:

In the Metric results view, double-click any class or method node. Together Developer opens the corresponding *.cs* file in the Visual Studio Editor. The text cursor is positioned at the corresponding line of code.

To view a metric description:

In the Metric results view, right-click anywhere in the column of the required metric. Choose **Show Description** from the context menu. This opens a description of the corresponding metric.

Saving metric results

Export metric results to an XML or HTML file so you can share them with team members or review them later.

To save metric results:

1 Select the rows of the table that you want to save. If you want to save the entire list, do not select anything.

2 Click the Save button  on the toolbar.

3 In the *Save Metric Result* dialog window that opens, choose the scope of the results to export using the Select View list box. The possible view options are:

- *Active List* — The exported report will contain the results for the selected node and (optionally) its children.
- *Selected Rows* — You can select single or multiple rows in the metric results report view. Choosing *Selected Rows* generates a report for such selections.

4 In the *Select Format* list box, select the format for the exported file:

- *XML* — Generates an XML-formatted report.
- *HTML* — Generates an HTML-formatted report.

Selecting HTML format activates the following check boxes:

- *Add Description* — This adds hyperlinks to the descriptions (stored under the Together Developer installation folder) from the results file.
- *Launch Browser* — This option opens the generated HTML file in the default browser.

5 Click **Save** to save the results in the specified location.

Refreshing and restarting metrics

You can update or refresh the results table using the toolbar:

- Click the **Refresh** button  to recalculate the results that are currently displayed.
- Click the **Restart** button  to open the **QA Metric** dialog window, where you can change the settings as necessary and run the metrics again. The new results replace the results that are currently displayed.

Close the **Metrics** view by clicking the **Close** button  in the upper-right corner of the dialog window.

Choosing specific elements to run metrics

Using the metrics feature of Together Developer, you can selectively determine which classes to run a set of metrics against.

To run metrics on a particular class element:

1 Go to the Base Classes diagram, right-click the **Store** class and choose **QA Metrics** from the context or main menu. This restricts the *Scope* of the metrics to the **Store** class.

2 Accept all default settings and click **Start**. The **Metrics** results view opens. As you run metrics on your project, the results display in the **Metrics** results view in a tabbed page format. The most recent metrics ran have focus.

3 You can close the metrics results (individual tabs) by right-clicking the corresponding tab and choosing **Close**. Click the **Close** button  in the upper right corner of the results dialog window to close the metric results.

Generating documentation

Together Developer features a wizard that you can use to generate HTML documentation for the Video Store project.

To generate HTML documentation for a project:

1 Choose [Tools | Together for VS | Generate Documentation...] on the main menu. The **Generate Documentation** dialog window opens.

2 Select your preferred *Scope* and *Options* settings.

The *Scope* section at the top of the dialog window has radio buttons to indicate the parts of the project to be parsed and included in the generated documentation:

- **Current namespace:** Generated output includes only the current namespace selected in the Model View.
- **Current namespace with descendant namespaces:** Generated output includes the current namespace selected in the Model View and any descendant namespaces under it.
- **Current diagram:** Generated output for the current diagram that is in focus in the Diagram View.
- **All:** Generated output covers the entire project. Select this option for our sample project.

The *Options* section of the dialog window has options to specify the destination and other optional actions:

- **Output folder:** Enter the location for the generated output or select it using the file chooser.
- Check boxes:
  - **Include diagrams:** Check to include diagram images in the output.
  - **Include navigation tree:** Check to include a navigation tree in the output.
  - **Launch HTML browser:** Check to load the documentation in the default web browser for your operating system. This starts the application if necessary. If you do not select this option, you can open the documentation later by navigating to your designated output folder and accessing the `index.html` file.

Leave all these options checked.

3 Click **OK** to generate documentation.

4 The **Confirmation** dialog window opens. You are prompted to create the `out\doc` directory in the Video Store project. Click **Yes**.

When Together Developer finishes generating the documentation, it opens in the default web browser for your system. The browser opens with a frameset to display the generated documentation. Expand the `Video Store` node in the tree in the lower left frame. Notice that clicking a class name in the lower left frame opens the documentation in the lower right pane.

You can further explore the generated documentation with the following exercises:

- Use the **Project** tab to navigate through the project.
- Click the `Store` class to display its corresponding documentation.
- Click `findByKeywords` to jump to that section in the documentation.
- Use the hypertext to navigate through the documentation and access the index.

Opening projects created in other versions

You may want to import diagrams designed with other products of the Together family. If you do not have any projects to be reused in Together Developer, skip this section.

If your project was created using a previous version of Together, you can open it in Together 2006 for Visual Studio in the regular way.

If your project was created using another Together Edition, follow these steps:

1. Choose [File | New | Project...] on the main menu. The **New Project** dialog window opens.
2. The project type should correspond to the type of the source project.
   - For a C# project, choose [Visual C# Projects | Empty Project].
   - For a Visual Basic project, choose [Visual Basic Projects | Empty Project].

Note Diagrams in the project must be in the new XML/TXV format (diagrams in the legacy DF format are not supported).
Diagram elements must be embedded (standalone elements are not supported).

3. The project name should be exactly equal to the source project name.
4. Adjust the remainder of the settings on your own. Click **OK** to create a project.
5. Copy all model files including subfolders from the source project to the `ModelSupport` folder under your new project root[1]. These files are located under `diagrams` or `Model Folder` directories, depending on the version of Together.

Note For some projects (for example, Visual Basic projects designed using Borland Together Architect) these files are located in the same folders as the source code files. In this case you will have to pick out the modeling files manually. Basically, you need all files with `.txv*` and `.txa*` extensions. Copy these files to the `ModelSupport` folder.

Tip You can share your project with another edition of Together, and adjust the setting **Short name of the folder where model support files are stored**. This setting is available under the option category [Together for VS | (level) | General | Together for VS support].

1. For copying, you can use Microsoft Explorer or any other file management tool.

6 If you have an implementation project and you need to keep your source code, copy it from the source project to the new one keeping the folder structure.

7 In Visual Studio, open the Solution Explorer. Click the **Show All Files** button on the toolbar of the Solution Explorer.

8 You will see the new files and folders in the Solution Explorer. For each item, choose **Include In Project** from the context menu.

Visual Studio and Together process your files. When completed, the imported project is displayed in the Model and Diagram Views.

Congratulations!

You finished reading the *Getting Started Guide*!

If you successfully performed all actions described in this guide, you are ready to work with real-life Together Developer projects. In case of questions consult the online help or the User Guide.

Borland®

Borland Software Corporation
May 15, 2006