EiffelCase User Guide

version 4.3

Eiffel Power™

from ISE

Interactive Software Engineering

Manual identification

EiffelCase User Guide, ISE Technical Report TR-EI-53/EA.

Publication history

First published in pre-release form: January 1995 (former title: *EiffelCase: Engineering Object-Oriented Software, Forward and Backward)*.

Previous published version: 3.3.7, July 1995.

This version: 4.3, March 1999. Title change. Corresponds to release 4.3 of the ISE Eiffel environment.

Author

This version: Paulette Le Blanc.

Other contributions from: Pascal Freund, Reda Kolli and Bertrand Meyer.

Previous version: Bertrand Meyer. Original version: Jean-Marc Nerson.

Software credits

See "SOFTWARE CREDITS AND ACKNOWLEDGMENTS" at the end of the Preface.

Cover design

Rich Ayling.

Copyright notice and proprietary information

Copyright © Interactive Software Engineering Inc. (ISE), 1993-1999. May not be reproduced in any form (including electronic storage) without the written permission of ISE. "Eiffel Power" and the Eiffel Power logo are trademarks of ISE.

All uses of the product documented here are subject to the terms and conditions of the ISE Eiffel user license. Any other use or duplication is a violation of the applicable laws on copyright, trade secrets and intellectual property.

Special duplication permission for educational institutions

Degree-granting educational institutions using ISE Eiffel for teaching purposes as part of the Eiffel University Partnership Program may be permitted under certain conditions to copy specific parts of this book. Contact ISE for details.

About ISE

ISE (Interactive Software Engineering) is dedicated to improving software quality and productivity through advanced methods, tools and languages, based on sound scientific principles and on the systematic application of object technology.

The company provides a complete line of development tools as well as on-site consulting, library development services, and a training program on all aspects of O-O technology: analysis, design, implementation techniques, graphics, library building, Eiffel language, project management, large system design etc.

ISE is the original designer of the Eiffel method and language and a member of NICE, the Nonprofit International Consortium for Eiffel.

For more information

Interactive Software Engineering Inc.
ISE Building, 2nd floor
270 Storke Road
Goleta, CA 93117 USA
Telephone 805-685-1006, Fax 805-685-6869

Internet and e-mail

ISE maintains a rich source of information at **http://eiffel.com**, with more than 1200 Web pages including online documentation, downloadable files, product descriptions, links to ISE partners, University Partnership program, mailing list archives, announcements, press coverage, Frequently Asked Questions, Support pages, and much more.

Write to **info@eiffel.com** for information about products and services. Write to **userlist-request@eiffel.com** to subscribe to the ISE Eiffel user list.

Support programs

ISE offers a variety of support options tailored to the diverse needs of its customers. Write to **info@eiffel.com** or check the support pages at **http://eiffel.com** for details.

Preface: Roundtrip engineering with EiffelCase

Organizations that have embraced object-oriented software development because of its promise of better quality and productivity need methods and tools that support both the *forward* and *backward* parts of the software process. The forward component is the production of software; the backward component, also called *reverse engineering*, is the analysis of previously produced software, necessary to understand it, improve it or adapt it. In the practice of software development, forward and backward tasks can seldom be separated into different phases of the software lifecycle; it is more realistic to think of these activities as interleaved and overlapping — inseparable companions on the road to software quality.

EiffelCase is an analysis, design and engineering workbench meant to help developers in both the forward and backward components of their work.

EiffelCase serves the needs of two communities:

- People involved in the modeling, analysis, understanding, documentation and design of possibly complex systems, usually (although not always) as part of software production.
- Eiffel developers in search of tools for producing high-level graphical views of the software they build or maintain.

EiffelCase will be useful to members of either one who do not also belong to the other. You can benefit from EiffelCase even if you are not an Eiffel developer and have no intention of producing Eiffel software; in that case you will use EiffelCase as a language-independent system description tool. If you *are* an Eiffel developer, you will additionally find EiffelCase precious as a set of powerful mechanisms to explore, document and reengineer Eiffel software produced by you or by others.

SCALING UP

A major goal of EiffelCase is to support the development of both small and large systems, as well as systems that start small and may grow large.

The practical challenge for a CASE (Computer-Aided Software Engineering) tool is whether it will "scale up" as the systems being described grow in size and complexity. EiffelCase provides to that effect a number of abstraction facilities allowing its users to limit and control complexity. You can view an entire system with its details hidden, or zoom into one of its subsystems. You can define and store multiple *views* of the same system, each focusing on some components and relations and excluding the rest. You can collapse some elements and expand others. These mechanisms are crucial to handling large systems.

Such large-system support remains compatible with an easy approach for small developments and an easy learning process for beginners and non-software experts. Simple things are done simply. Unlike some other CASE tools and methods, EiffelCase enables you to master the basics in a few moments and start using the tools right away. The simplicity of the graphical conventions also means that you can explain an EiffelCase model to a non-software professional, enabling him or her to concentrate on the problem at hand, not the tool or the notation.

REVERSIBILITY

EiffelCase supports a "yoyo" process of analysis and development that alternates between forward and backward directions. Some people like to start from visual diagrams rather than formal text; others prefer to start from text; but very few like to be restricted to either diagrams or text. If you make changes to the generated text, add new elements using text rather than diagrams, or add existing reusable components in text format, you will want to make sure that the visual representations reflect these changes.

Both textual and visual representations are useful; changes to one should be reflected in the other. EiffelCase helps users *reconcile* changes made on both sides, textual and graphical.

ABOUT VERSION 4.3

Version 4.3 of EiffelCase includes a set of major extensions and improvements on both Windows and Unix. See "What's new in version 4.3?", page 1 for details.

SOFTWARE CREDITS AND ACKNOWLEDGMENTS

Version 4.3: Pascal Freund, Reda Kolli.

Earlier versions: Jean-Marc Nerson, Gurvan Lullien, Jean-Pierre Sarkis, Dino Valente, Mario Mengen, Olivier Zendra.

The successive versions benefited from numerous user comments, in particular from: David Hollenberg, Mark Howard, Jean-Marc Nerson, Cliff Ritchie, Kim Waldén; for version 4.3: Thomas Beale, Sami Bengrine, Paul Ford, Darren Hiebert, Roy Phillips, Zoran Simic, Luc Taesch.

Contents

1

Introduction

1.1 Welcome to EiffelCase

Welcome to EiffelCase 4.3, the newest version of the graphical object-oriented analysis and design workbench that provides the methods and tools to support both the *forward* and *backward* segments of the software design process. The forward component produces the actual software, while the backward component, also known as *reverse engineering*, analyzes software created previously — either through EiffelCase itself or through any other tool. supports true roundtrip engineering.

EiffelCase prepares, documents and tracks object-oriented system elements and their relationships, from preliminary specifications to the programming phase. EiffelCase also supports the following multiple views:

- Informal charts, to plan a project and to interact with managers and customers.
- Graphical maps, to visualize system modifications quickly.
- Formal text, to create a precise and formal description.

Automatic update procedures guarantee that all views remain compatible.

1.2 What's new in version 4.3?

While experienced EiffelCase users will find all the benefits of previous versions, major effort has been put into improving and expanding EiffelCase, including a complete redesign of the user interface.

Several of the new features include:

- Generated Eiffel code that contains position, color and other visual information. As a result, you can forward- and reverse-engineer without losing any graphical information.
- EiffelCase retains all routine bodies.
- Automatic generation of *set* procedures from a feature, complete with preconditions and postconditions.

- *EiffelTips* — a feature that displays information about a class or iconified cluster when you point to that object.
- Importing and exporting individual clusters between EiffelBench and EiffelCase.
- Complete HTML and PostScript generation for a system.
- Choice of diagram layout: BON or UML.

1.3 ISE Eiffel

ISE Eiffel is a visual object-oriented software development environment for building enterprise-wide client-server applications heterogeneously. It provides a seamless approach by supporting:

- One set of concepts, tools and notations — from the analysis and design phases, through implementation and maintenance.
- Full source-code compatibility across major industry platforms: all UNIX variants, LINUX, all Windows variants, VMS, and Cray.
- Interface to C/C++, Corba, Java, COM, UML (through ObjectTeam).
- Cross-development: write your application once and move it to any supported platform without changing a single line of code.
- Database connectivity with Oracle, Sybase, and Matisse.
- Thousands of reusable library classes from ISE and third parties: graphics, client-server, parsing, communication, scientific computation, 3-D and multimedia, banking/finance, and so on.
- Run-time performance as good as C, without sacrificing any of the benefits of full object-orientation.

Most importantly, ISE Eiffel is easy to learn and understand, particularly by non-programmers. There are numerous books available from major publishers on Eiffel.

1.4 BON

EiffelCase supports the BON (Business Object Notation) analysis and design method, which is based on the same software engineering principles as the Eiffel language. BON provides a clear notation and a set of methodological guidelines for high-level analysis and design in the Eiffel spirit of precision, scalability and clarity.

BON presents a set of concepts for modeling object-oriented software and provides two type of notations — graphical and textual. BON also incorporates a well-defined set of conventions that support both semantics (such as assertions, contracts, class invariants) and structure. For more information on BON, see *Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems* (Prentice Hall, 1995).

1.5 UML

EiffelCase enables you to view diagrams in Unified Modeling Language (UML) format. UML is a widely accepted convention for representing the structure of object-oriented systems. You can switch between BON and UML at any time, using the **Preference** tool. This allows you to build a system in one notation, and then export it in another. For more information on the **Preference** tool, see chapter 6, "EiffelCase tools".

1.6 Customer support

If you have any problems running or using EiffelCase, email your questions to support@eiffel.com, or access ISE on the World Wide Web (WWW) at http://www.eiffel.com.

You can also contact ISE at (805) 685-1006 or FAX your questions to (805) 898-2452, Monday - Friday, 8:00 a.m. to 5:30 p.m. (PST).

1.7 Organization of this guide

This User Guide assumes that you have a basic understanding of BON, UML, or ISE Eiffel, but not necessarily all three.

Chapter 1, "Introduction", gives an overview of EiffelCase version 4.3, ISE Eiffel, BON, and UML. It also details customer support, the organization of this guide, and the tutorials.

Chapter 2, "Getting started", describes the system setup, installation, and registration procedures. It also defines the conventions used throughout this manual and the EiffelCase main window, as well as how to start and exit the program.

Chapter 3, "Tutorial A: forward engineering", walks you through the steps necessary to create a conference management system diagram and generate the Eiffel code for the system.

Chapter 4, "Tutorial B: reverse engineering", illustrates how EiffelCase and the EiffelBench development environment interact to implement the idea of reversible software development.

Chapter 5, "Menu bar and toolbar", provides detailed descriptions of the EiffelCase menus, submenus and commands, as well as toolbar buttons, holes and commands.

Chapter 6, "EiffelCase tools", explains how to create or retarget a tool. It also describes many of the tools available in EiffelCase.

Chapter 7, "System tool", Chapter 8, "Class tool", Chapter 9, "Relation tool", Chapter 10, "Feature tool", Chapter 11, "Editor tool", Chapter 12, "Documentation tool", Chapter 13, "Advanced Color tool", and Chapter 14, "Merging tool" give detailed descriptions of the menus, submenus, commands, holes, toolbars, buttons, and options available in these tools.

Appendix A, “System resources”, describes the system resource file and how to customize this file.

1.8 Tutorials

The tutorials included in this manual consist of two step-by-step lessons that show you how to use EiffelCase. The first example details the procedure for creating a system diagram, while the second employs many of the sophisticated features of interest to experienced users and professionals.

These chapters contain lessons and pictures to assist you. If you are familiar with EiffelCase, work through the tutorials to become comfortable with the new features. If you are new to Eiffel, you can read Appendix D of *ISE Eiffel: The Environment*.

2

Getting started

This chapter describes the system setup, installation, and registration procedures. It also defines the conventions used throughout this manual and the lay out of the EiffelCase main window, as well as how to start and exit the program.

2.1 System setup

EiffelCase requires the following minimum system configuration:

- Linux, UNIX, Sun Sparc with Solaris, Microsoft WIndows 95, 98, or NT operating system.
- 48 megabytes (MB) of RAM (64 MB recommended).
- 20 MB of free hard disk space (standard installation).
- VGA monitor using 256 colors (1024 x 768 recommended resolution).

Because you can port Eiffel to any platform, contact ISE for more information on supported platforms and operating systems. For more information on contacting ISE, see chapter 1, "Introduction".

2.2 Mouse conventions

This guide uses the following mouse conventions:

- **click** — to press and release the left mouse button.
- **double-click** — to press and release the left mouse button twice.
- **open** — to point and double-click an object; the effect of this operation depends on the object.
- **pick-and-drop** — to right-click, point to the new position, and then right-click again. For more information, see "Pick-and-drop operation", page 8, later in this chapter.
- **point** — to move the mouse to position the pointer on an area of the screen.
- **right-click** — to press and release the right mouse button.
- **select** — to point to an object and click; this usually highlights the object.

2.3 Installing EiffelCase

How you install EiffelCase depends on the operating system you are using. The program installs like most other applications:

To install EiffelCase:

1 Insert the Eiffel 4.3 CD in your CD-ROM drive.

2 Follow the on-screen prompts to complete the installation.

2.4 Starting EiffelCase

How you start EiffelCase depends on the operating system you are using. This section provides instructions for starting the UNIX and Windows versions of EiffelCase.

UNIX version

To start the UNIX version of EiffelCase:

- Open a shell tool or command window, and then type **ecase&**.

The main EiffelCase application windows appears.

For more information on starting the UNIX version of EiffelCase, see your System Administrator.

Windows version

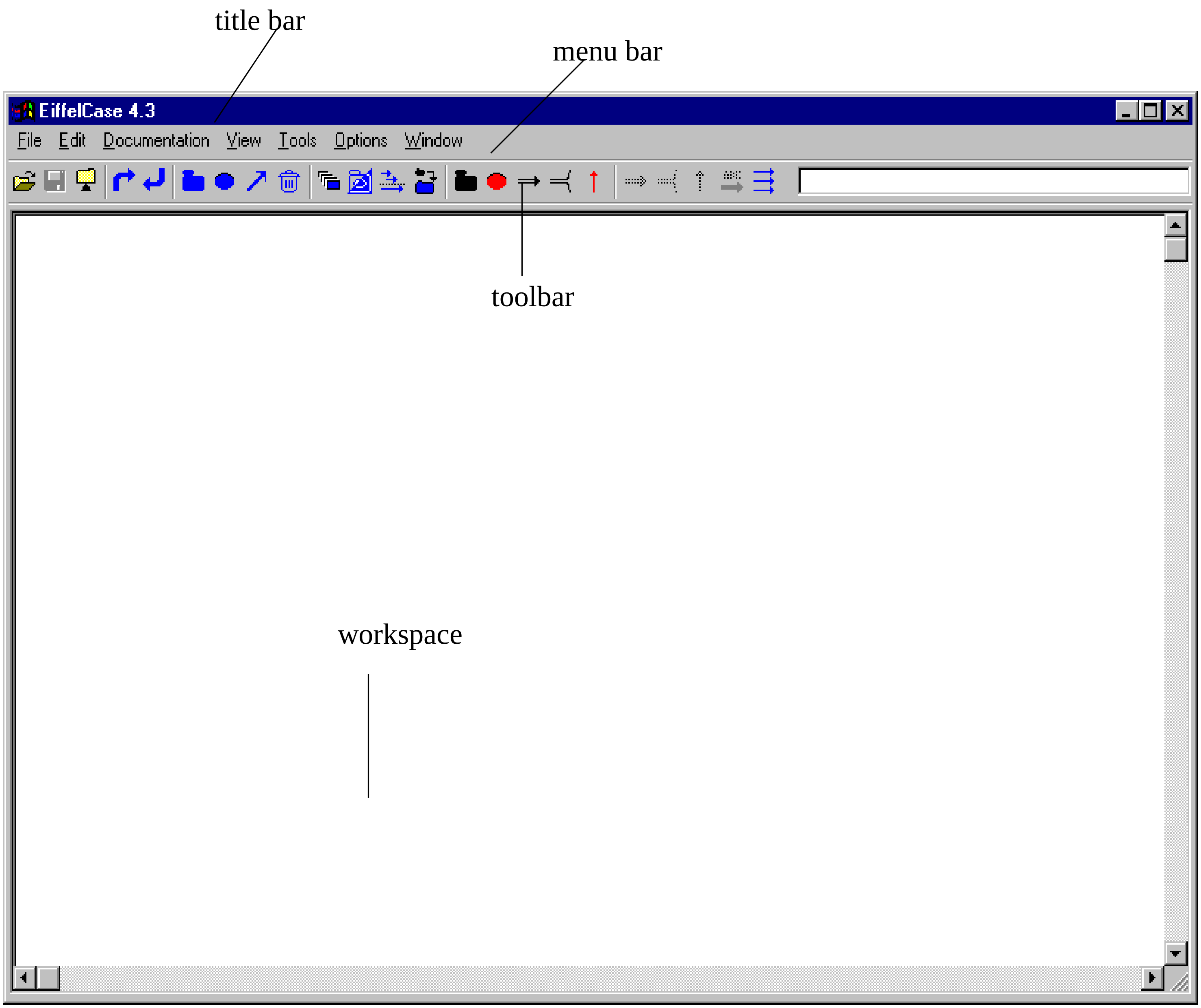
To start the Windows version of EiffelCase:

1 On the taskbar, click **Start**, and then point to **Programs**.

2 Point to the folder that contains EiffelCase, and then click **EiffelCase**.

The main EiffelCase application windows appears.

2.5 EiffelCase main application window



2.6 Development objects

EiffelCase provides a series of tools that correspond to the major abstractions relevant to modeling, analysis, documentation and design. These abstractions include:

- **System** — set of classes, one of which is the **root class** of the system; known as a program in traditional approaches.
- **Cluster** — set of related classes or subsystems.
- **Class** — set of data abstractions.

- **Feature** — set of properties attached to a class.

To simplify things, you can think of a system as a cluster that can contain other clusters, but is not contained in any other cluster.

A *development object* is concept of ISE Eiffel and is an instance of any of the abstractions. For the purposes of this user guide, the term development object represents either a cluster, class, feature or a relation link. For more information on relation links, see chapter 9, “Relation tool”.

The relationship between development abstractions and development objects is easily defined — development objects are instances of development abstractions. For example, a particular cluster in the active system is also a development object of the system: an instance of the cluster development abstraction.

2.7 Pick-and-drop operation

To use the Eiffel pick-and-drop operation:

1 Right-click a development object

The cursor changes to the shape of the selected object — a **pebble**.

2 Point to the new position for the object, and then right-click.

To cancel the operation, click anywhere.

2.8 Holes

Holes are icons that you drop pebbles on using the pick-and-drop operation, to create new tools, retarget existing tools, or execute other operations.

There are two types of holes in EiffelCase:

- **Tool holes** — a symbol for the corresponding development abstraction; located in the upper left corner of a tool.
- **Operation holes** — performs various operations on the target of a tool, when you drop the object on the hole.

If you target a tool, the hole displays with a dot in it, with the dot representing the target. The pebble and hole in a pick-and-drop operation must be compatible — not necessarily identical.

For example, the **Hidden Components Tool** hole accepts several different pebble types. For more information on the Hidden Components Tool hole, see chapter 5, “Menu bar and toolbar”.

2.9 Adding development objects and relation links

You can add classes or clusters to the workspace, as well as client, aggregation, and inheritance relation links. For more information on the toolbar, see chapter 5, “Menu bar and toolbar”.

Adding classes or clusters

To add a cluster or class to the workspace:

1 In the toolbar, right-click **Cluster** or **Class** .

2 Point to the position for the cluster or class, and then right-click.

3 In the **Editor** tool, type a name, and then click the **Check** button.

You can also click **Cluster** hole or **Class** hole, hold SHIFT, and then click anywhere in the workspace.

For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Adding inheritance relation links

To add a relation link between development objects:

1 In the toolbar, click **Inheritance link** .

2 Point to the source object, and then right-click.

3 Point to the target object, and then right-click.

Adding client or aggregation relation links

To add a relation link between development objects:

1 In the toolbar, click **Client link** or **Aggregation link** .

2 Point to the source object, and then right-click.

3 Point to the target object, and then right-click.

4 In the **Link Generation** tool, set options, click **Add**, and then click **Generate**.

For more information on the **Link Generation** tool, see chapter 6, “EiffelCase tools”.

2.10 Group operation

The group operation combines the development objects that you select into a single object. Once grouped, you can either move, delete or hide the object.

Grouping development objects

To group development objects:

1 Hold CTRL and click in the workspace.

2 Drag the rectangle until it contains all objects that you want to group.

The combined objects redisplay in cyan.

Hiding the grouped object

To hide the grouped object:

- Use the pick-and-drop operation to drop the grouped object on the **Hidden Components Tool** hole  on the toolbar.

For more information on the **Hidden Components Tool** hole, see chapter 5, "Menu bar and toolbar" chapter.

2.11 System file

When you create a new EiffelCase system, a **CASEGEN** subdirectory is added to the EiffelCase system directory, by default. The EiffelCase system directory also contains the system file (.ecr), whose default file name is system_architecture.ecr. The path, file name and view for the active system display in the EiffelCase title bar. For more information on the system file, see appendix A, "System resources".

2.12 Views

A *view* provides a different subset of the graphical properties of your system. At any time during an EiffelCase session, you are working on a specific view of your system. The name of the active view displays in the EiffelCase title bar.

If you change any of the graphical properties of your system, and then save the active state, only the active view changes. However, any change affecting an implementation property modifies all views. Examples include: adding a class; permanently removing a relation link; making a cluster a subcluster of another.

The **View** tool defines and changes the views available to the active system. For more information on the **View** tool, see chapter 6, "EiffelCase tools".

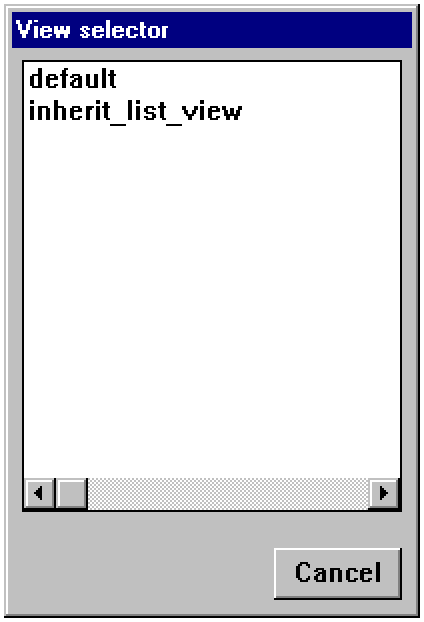
You must switch to the new view to work in that view and to save any changes made.

Switching views

To switch to a different view:

1. On the **View** menu, click **Change View**.

The **View selector** dialog box appears and lists of all defined views.



2 Click the name of the view that you want to display.

3

Tutorial A: forward engineering

This chapter walks you through the steps necessary to create the conference management system diagram from *Seamless Object-Oriented Software Architecture*, page 84, and generate the Eiffel text for cluster **CONFERENCE_DATA**.

3.1 Starting EiffelCase

How you start EiffelCase depends on the operating system you are using. For more information on starting EiffelCase, see chapter 2, "Getting started".

In this section, you will create your system.

3.2 Creating an EiffelCase system

To create an EiffelCase system:

1 On the **File** menu, click **Create**.

2 In the **Choice** box, type *drive*:**\tutorialA**, and then click **OK**.

The path, file name, and view for the active system display in the EiffelCase title bar. For more information on the system file, see chapter 2, "Getting started" and appendix A, "System resources".

3.3 Exploring the Cluster and Preference tools

When you start an EiffelCase session, the **Cluster** tool displays by default. The **Cluster tool** graphically defines the architecture of the active system, its classes, and clusters, as well as the relationship between all development objects — clusters and classes.

For more information on development objects and the **Cluster** tool, see chapter 6, "EiffelCase tools". Before you add development objects to your system, you will set display options using the **Preference** tool.

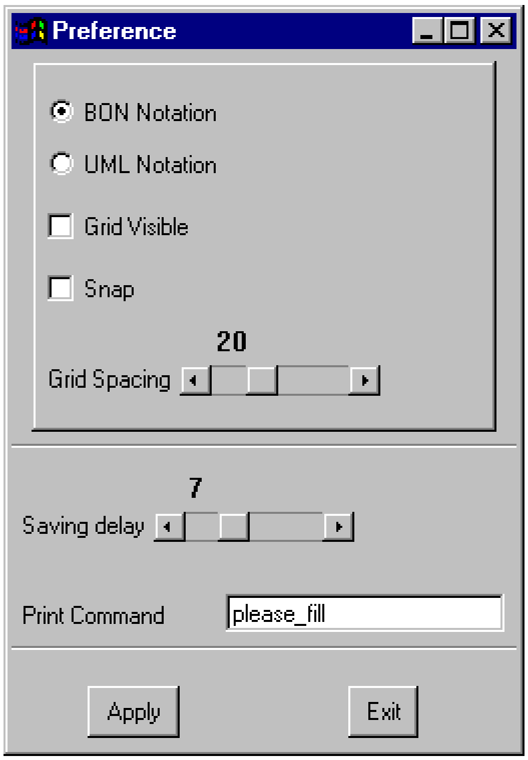
The **Preference** tool sets grid, display, saving delay, and print command preferences. For more information on the **Preference** tool, see chapter 6, "EiffelCase tools".

Preference tool

In order to accurately position the development objects, you will add a visible grid of 30 x 30 pixels, enable the invisible snap grid, and then set the automatic save time interval to 7 minutes.

To display the **Preference** tool and set display options:

1 On the **Tools** menu, click **Preference Tool**.



2 Select **Grid Visible**, and then select **Snap**.

3 Move the **Grid Spacing** slider to the right until 30 displays.

4 Move the **Saving delay** slider to the left until 7 displays, and then click **Apply**.

You can resize the workspace to extend it to the edges of the EiffelCase window.

Automatically resizing the workspace

To automatically resize the workspace:

- On the **Edit** menu, click **Automatic Resizing**.

Use the horizontal and vertical scrollbars to move through the workspace. You can now add development objects to your system.

3.4 Adding and modifying a cluster

There are several techniques to add and then modify development objects, in this case, a *cluster*. One way is to use the pick-and-drop operation. For more information on the pick-and-drop operation, see chapter 2, "Getting started".

Adding a cluster using the pick-and-drop operation

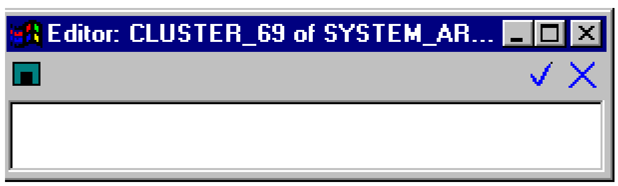
To add a cluster to the workspace using the pick-and-drop operation:

1 On the toolbar, right-click **Cluster**  .

The cursor changes to the shape of the selected object — a cluster.

2 Point anywhere in the workspace, and then right-click.

The **Editor** tool displays and sets the name of the active development object.



3 Type **ORGANIZATION**, and then press ENTER.

For more information on the **Editor** tool, see chapter 11, "Editor tool".

Before adding classes to the cluster, you will resize **ORGANIZATION** using the drag-and-drop operation.

Resizing a cluster

To resize **ORGANIZATION** using the drag-and-drop operation:

- Point to the lower-right corner of **ORGANIZATION**, and drag it until the cluster is nine grid squares high and twenty-four grid squares wide.

Resizing works only on the cluster, not on the label for the cluster.

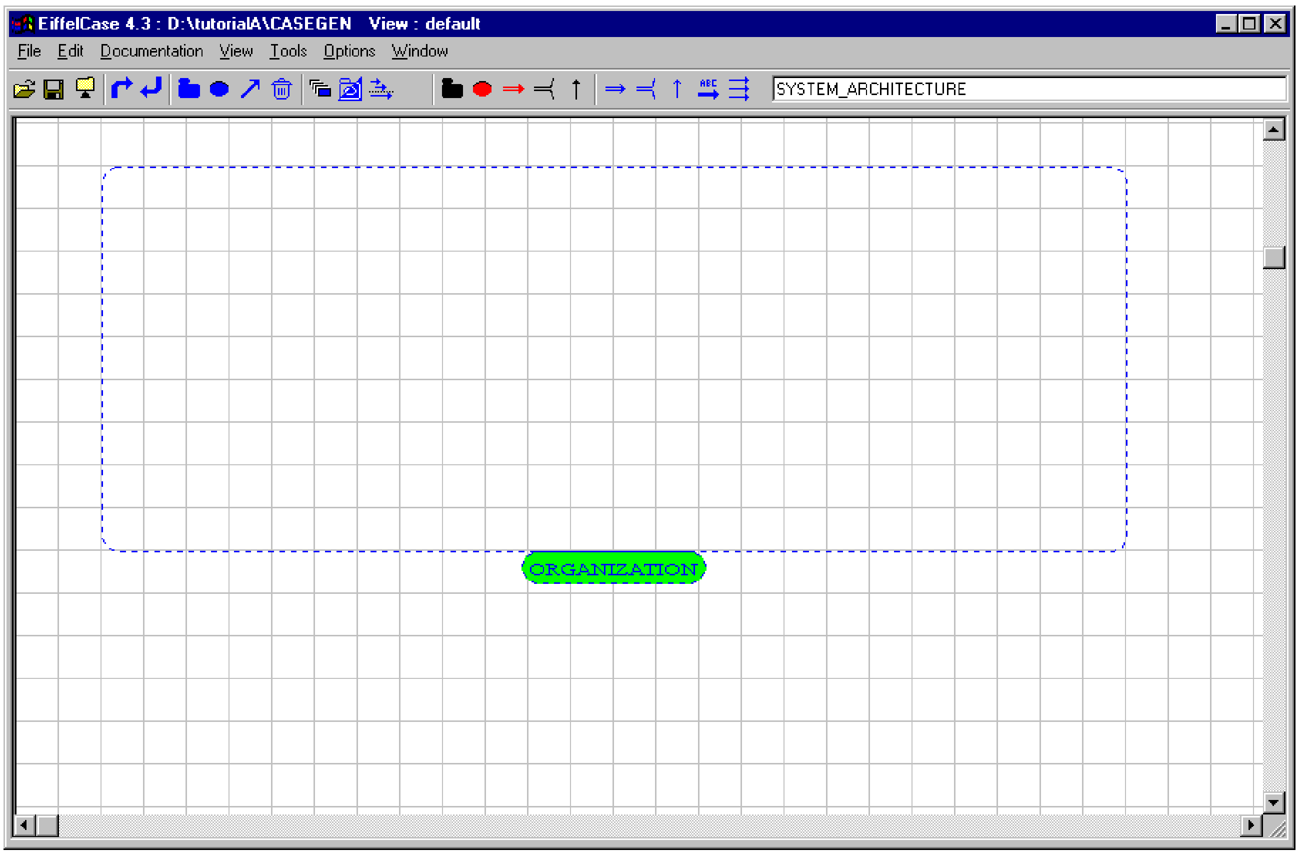
Moving a cluster

You can use the drag-and-drop operation to move development objects in the workspace. In this section, you will move **ORGANIZATION** to the upper-left corner of the workspace, and center the label below the cluster.

To move **ORGANIZATION** and the corresponding label:

1 Point anywhere in **ORGANIZATION**, and drag it to one grid square over and two squares down from the upper-left corner for the workspace.

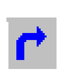
2 Point to the **ORGANIZATION** label, drag it below the cluster, and then center it.:



Correcting mistakes

If you made a mistake at any point and want to cancel your most recent operation:

- On the **Edit** menu, click **Undo**.

You can also click **Undo** in the toolbar .

3.5 Saving system

It is a good idea to save the active system after you make changes.

To save the active system:

- On the toolbar, click the **Save** button .

You can also click **Save** on the **File** menu.

3.6 Adding two more clusters

You will now add two more clusters, **CONFERENCE_DATA** and **CORRESPONDENCE**, below **ORGANIZATION** using the shift-click method. You must leave a least one grid square between the bottom of **ORGANIZATION** and the tops of the new clusters.

Earlier in this chapter, you used the pick-and-drop operation to add a cluster to the workspace. There is a faster technique for creating multiple instances of a development object or a relation link — the shift-click operation.

Adding clusters using the shift-click operation

To add clusters using the shift-click operation:

1 On the toolbar, click **Cluster** .

2 Hold SHIFT, and then click four grid squares below the lower-left corner of **ORGANIZATION**.

3 Type **CONFERENCE_DATA**, and then press ENTER.

4 Hold SHIFT, and then click seventeen grid squares to the right (same vertical position) of the left edge of **CONFERENCE_DATA**.

5 Type **CORRESPONDENCE**, and then press ENTER. Point to the lower-right corner of **CONFERENCE_DATA**, and drag it until the cluster is thirteen squares wide and eight squares high.

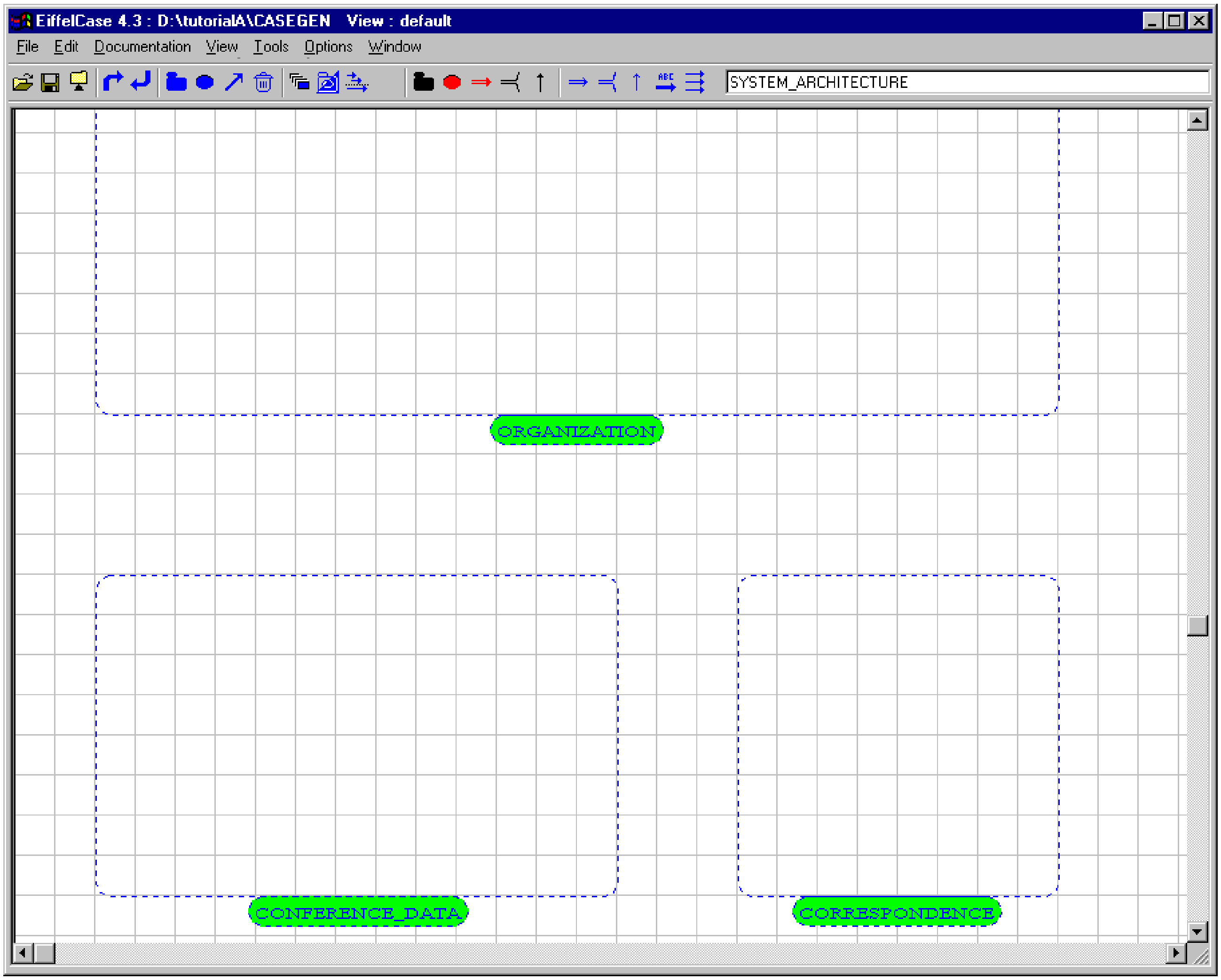
6 Point to the lower-right of **CORRESPONDENCE**, and drag it until the cluster is eight squares wide and eight squares high.

7 Point to the **CONFERENCE_DATA** label, drag it below the cluster, and then center it.

8 Point to the **CORRESPONDENCE** label, drag it below the cluster, and then center it.

9 On the **File** menu, click **Save**.

10 Make any changes, until the system diagram matches the following:



You can now add classes to the clusters.

3.7 Adding classes

If you were building a new system, you would probably not add all classes at once. More than likely, you would add a class, and then set all properties and relation links, before adding the next class.

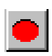
Since this exercise provides an overview of EiffelCase by having you reconstruct a known result, you will add all classes using the shift-click operation before setting properties and relation links.

You will add the following eight classes to the three clusters:

Cluster	**Class**
ORGANIZATION	PROGRAM_COMMITTEE
	CONFERENCE
	ORGANIZING_COMMITTEE
	COMMITTEE
CONFERENCE_DATA	CONFERENCE_EVENT
	REGISTRATION
	PERSON
CORRESPONDENCE	SEND_OUT

Adding classes using the shift-click operation

To add classes using the shift-click operation:

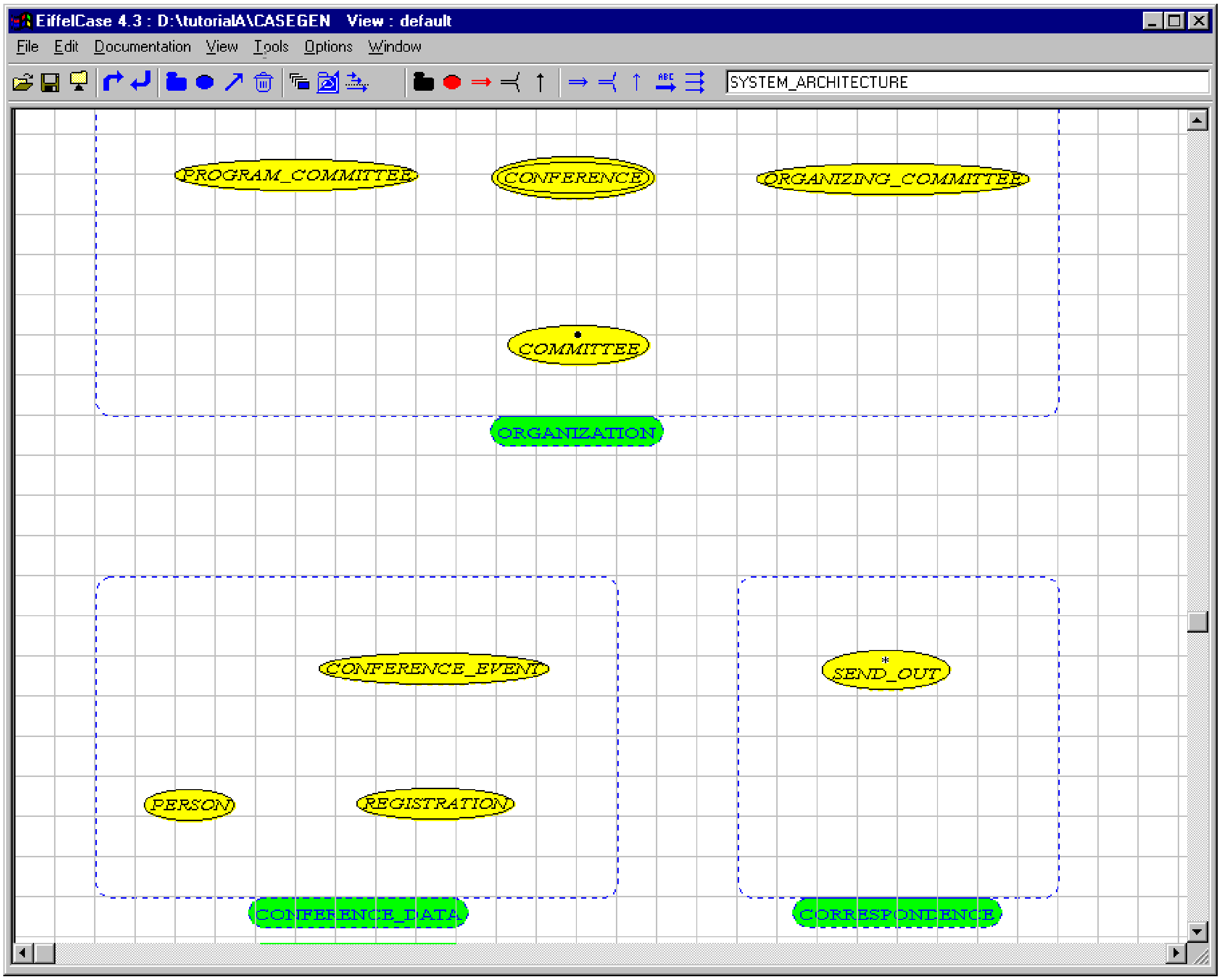
1 In the toolbar, click **Class**  .

2 Hold SHIFT, and then click anywhere in **ORGANIZATION**.

3 Type **PROGRAM_COMMITTEE**, and then press ENTER.

4 Repeat steps 2-3 for the remaining classes in the preceding table.

5 Move the classes as needed, until the system diagram matches the following:



You can now use the **Class** tool to set properties for the different classes.

3.8 Setting class properties

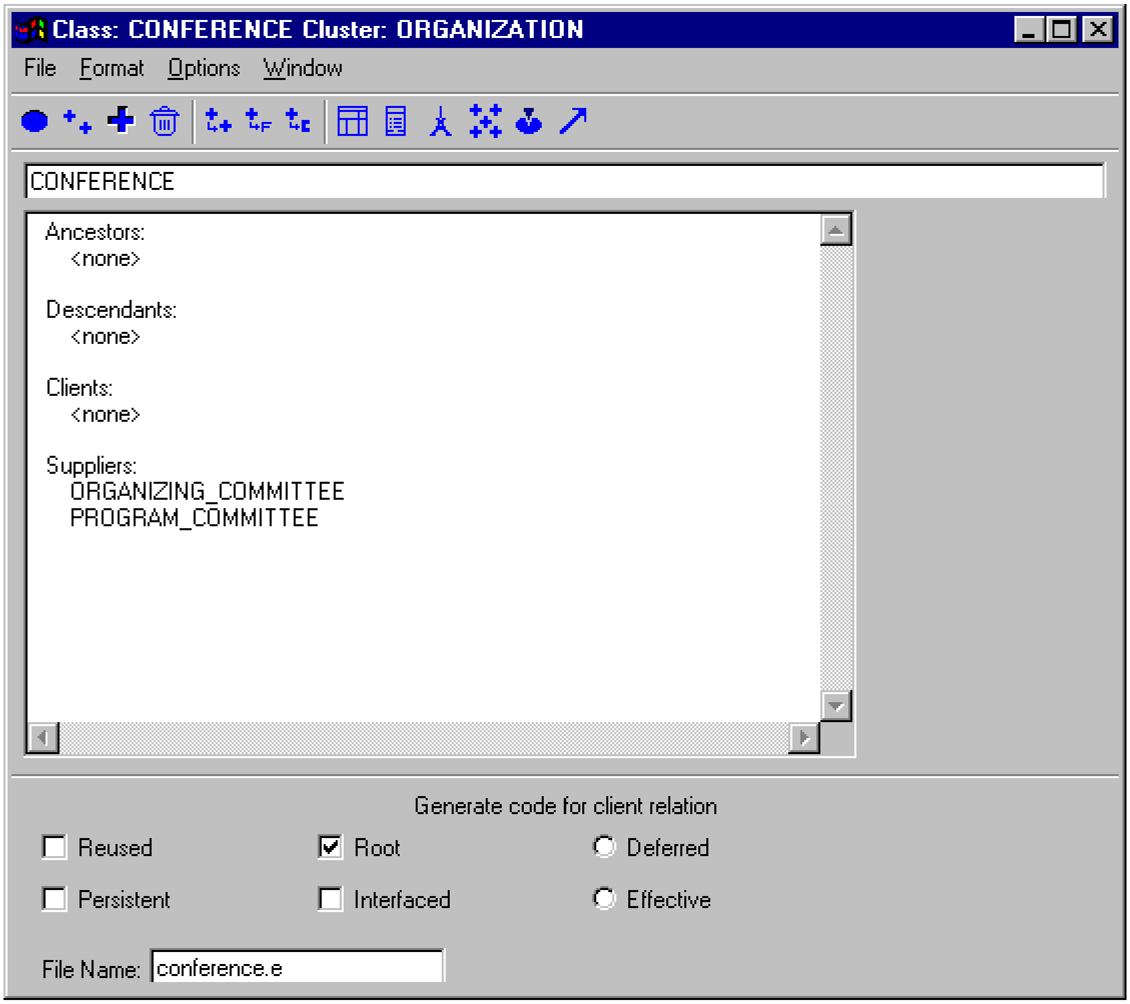
The **Class** tool sets properties, features, indexing information and constraints for the active class. For more information on the **Class** tool, see chapter 8, "Class tool".

In this section, you will use the **Class** tool to establish the *root class* (the master class in the system), and set properties for several other classes. A system can only have one root class.

Using the Class tool

To use the **Class** tool to set properties:

1 Hold down CTRL, and then right-click **CONFERENCE**.



2 Select **Root**, and then click the **Close** button.

Two lines appear around the border of **CONFERENCE**, indicating that it is the root class.

3 Use the pick-and-drop operation to drop **COMMITTEE** anywhere in the workspace.

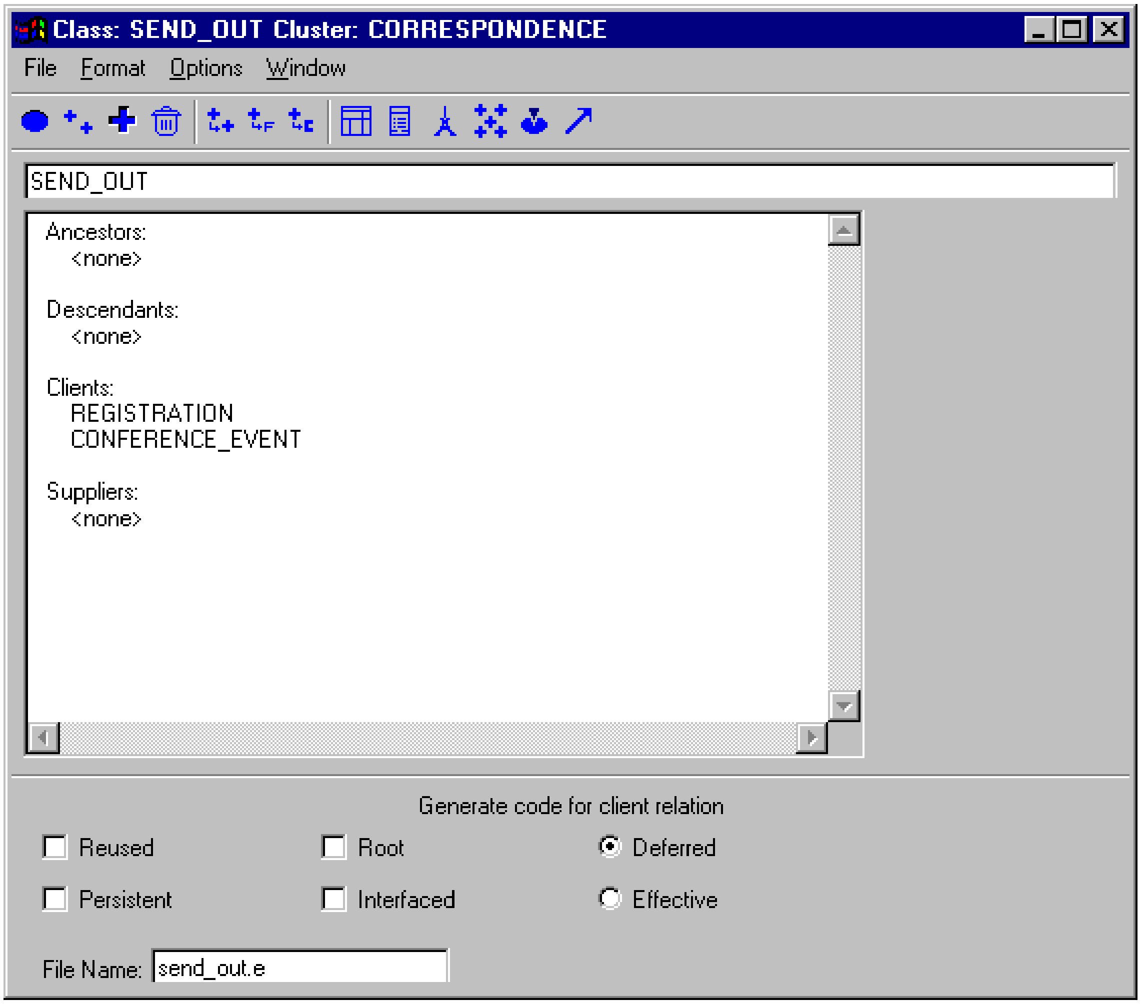
4 Select **Persistent**, and then click the **Close** button.

A solid black dot appears above **COMMITTEE**, indicating that it is *persistent* — instances of this class exist between sessions of the system.

5 On the **Tools** menu, click **Class Tool**.

6 In the **Class name** box, type **SEND_OUT**, and then press ENTER.

The **Class** tool retargets to **SEND_OUT**.



7 Select **Deferred**, and then click the **Close** button.

An asterisk appears above **SEND_OUT**, indicating that it is deferred.

You can now establish relation links and set properties for the links.

3.9 Establishing relation links between classes

Relation links are full-fledged development objects and exist between two classes, two clusters, or a class and a cluster.

There are three type of relation links:

- **client** — a class is a client of another if it uses a feature of the other (supplier) class.
- **inheritance** — a class is an heir of another if it incorporates the features of the other class, in addition to its own.
- **aggregation** — every object of a certain type is a combination (an aggregate) of zero or more objects of a certain type. For example, a “car” can be defined as an aggregation of “engine”, “body” and so forth.

To increase understanding of the system diagram, you can add labels to client and aggregation relation links.

For example, since **REGISTRATION** is a client of **PERSON** through feature **ATTENDEE**, a label named **attendee** is appropriate.

In the next section, you will create the following client relation links:

Source object	**Target object**	**Feature type**	**Label name**
REGISTRATION	PERSON	Attribute	attendee
REGISTRATION	SEND_OUT	Function	answer
CONFERENCE_EVENT	SEND_OUT	Attribute	calls:SET [SEND_OUT]
PROGRAM_COMMITTEE	PERSON	Attribute	referees:SET [PERSON]
COMMITTEE	PERSON	Attribute	chair,members:SET [PERSON]
COMMITTEE	CONFERENCE_EVENT	Attribute	conf_event

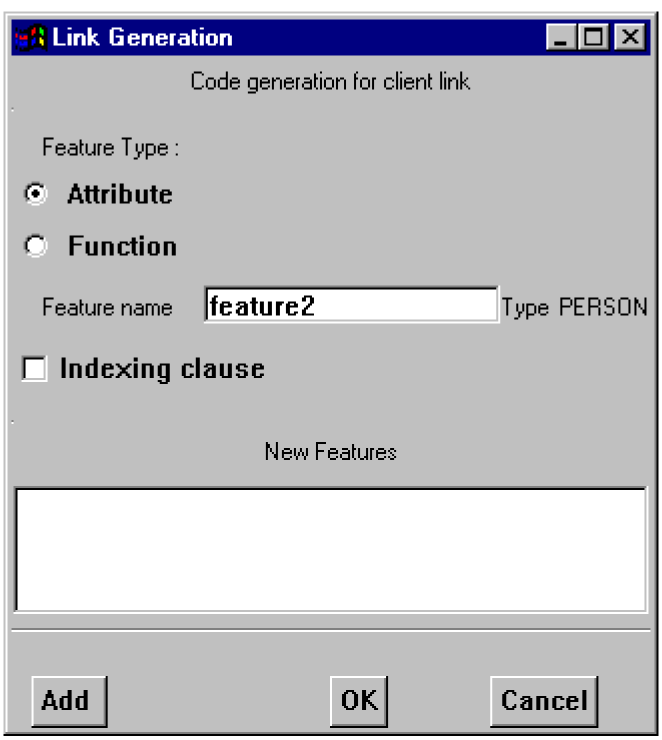
Creating client relation links and adding labels

To create client relation links:

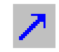
1 In the toolbar, click **Client link** .

2 Point to **REGISTRTION**, and then right-click.

3 Point to **PERSON**, and then right-click.



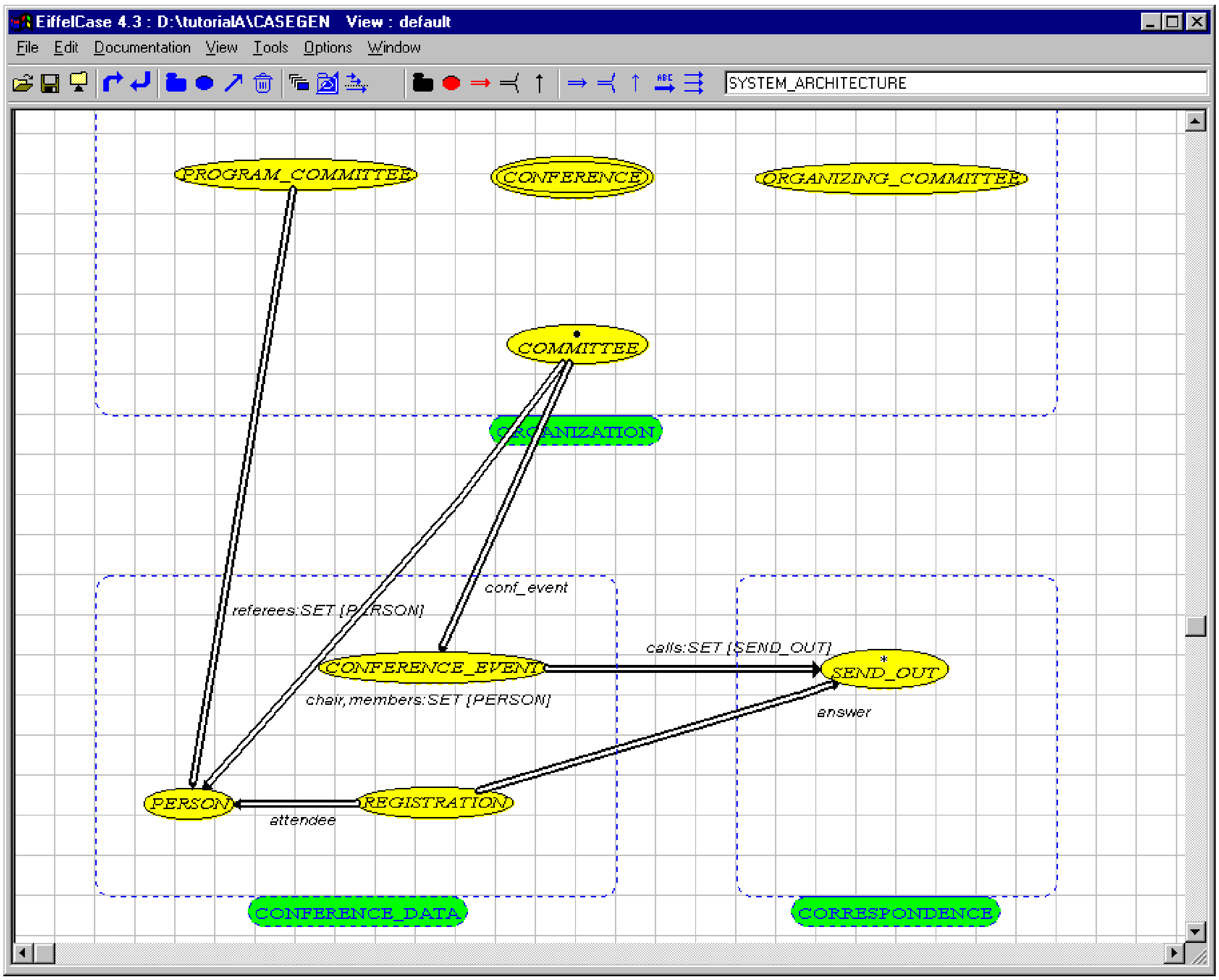
4 Click **Attribute**, and then click **OK**.

5 Use the pick-and-drop operation to drop the new relation link on the **Relation Hole**  in the toolbar.

6 In the **Label** box, type **attendee**, and then click the **Close** button.

7 Repeat steps 2-6 for the remaining client relation links and labels defined in the preceding table.

8 Make any changes, until the system diagram matches the following:

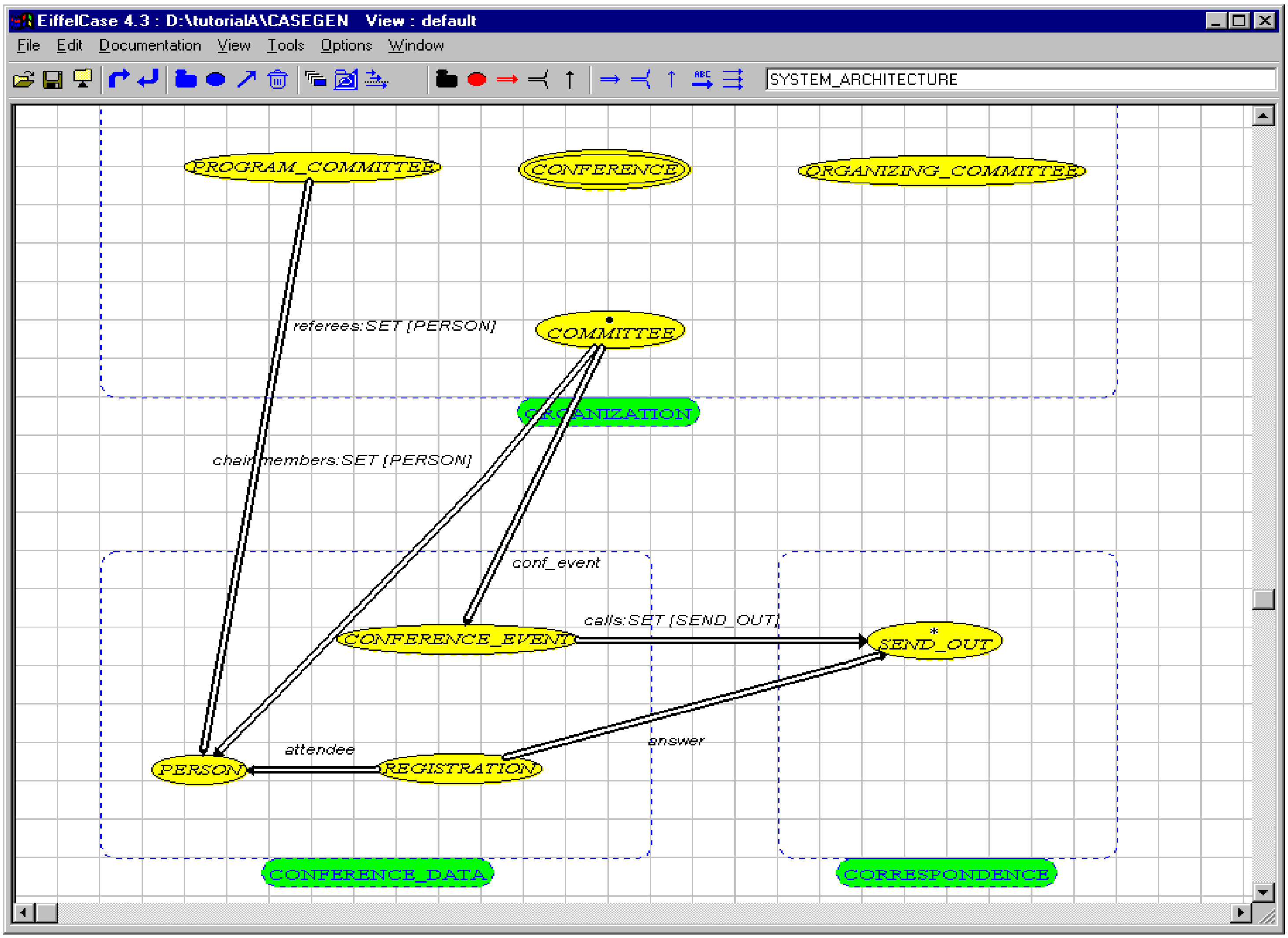


As you can see, the default positions of the relation link labels make the system diagram difficult to read. You can move the labels to increase readability.

3.10 Moving labels

You can use the drag-and-drop operation to move a label parallel to its corresponding relation link. You can also use this method to move a label from the right side of the relation link to the left, and vice versa.

Move the labels as needed, until the system diagram matches the following:



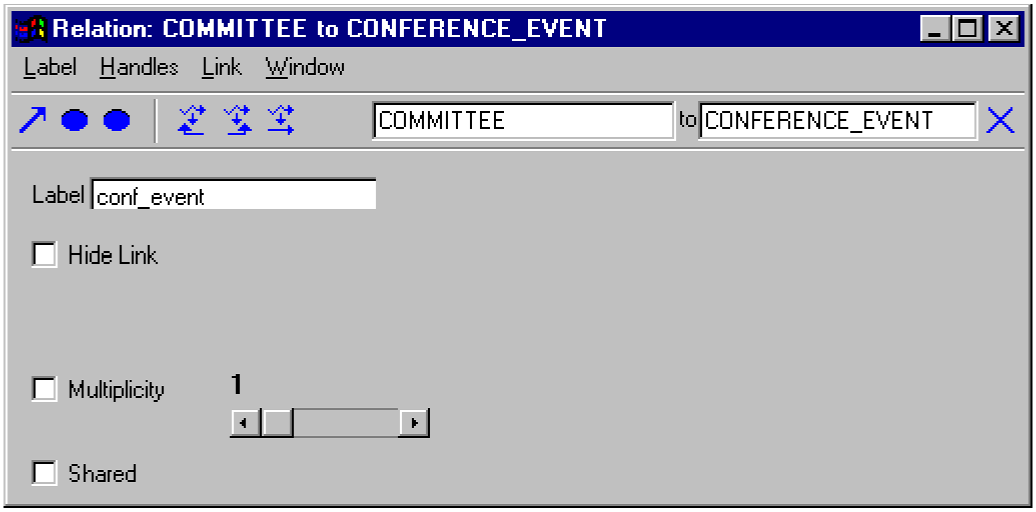
Because every **COMMITTEE** is dedicated to exactly one **CONFERENCE_EVENT**, you must add a multiplicity of 1 to this link using the **Relation** tool.

The **Relation** tool sets display properties for the active relation link, its classes and clusters. For more information on the **Relation** tool, see chapter 9, “Relation tool”.

Using the Relation tool

To use the **Relation** tool:

1 Use the pick-and-drop operation to drop the **COMMITTEE — CONFERENCE_EVENT** client relation link on the **Relation** hole in the toolbar.



2 Select **Multiplicity**, and then click the **Close** button.

A diamond with the number 1 appears in the middle of the link. This value corresponds to the number of development objects that participate in each instance of the relationship.

In the next section, you will create the following aggregation relation links:

Source object	Target object	Attribute or Function	Label name
CONFERENCE_EVENT	REGISTRATION	Attribute	registration:SET [REGISTRATIC
CONFERENCE	ORGANIZING_COMMITTEE	Attribute	organization
CONFERENCE	PROGRAM_COMMITTEE	Attribute	program

Creating aggregation relation links

To create aggregation relation links:

1 In the toolbar, click **Aggregation link**  .

2 Point to **CONFERENCE_EVENT**, and then right-click.

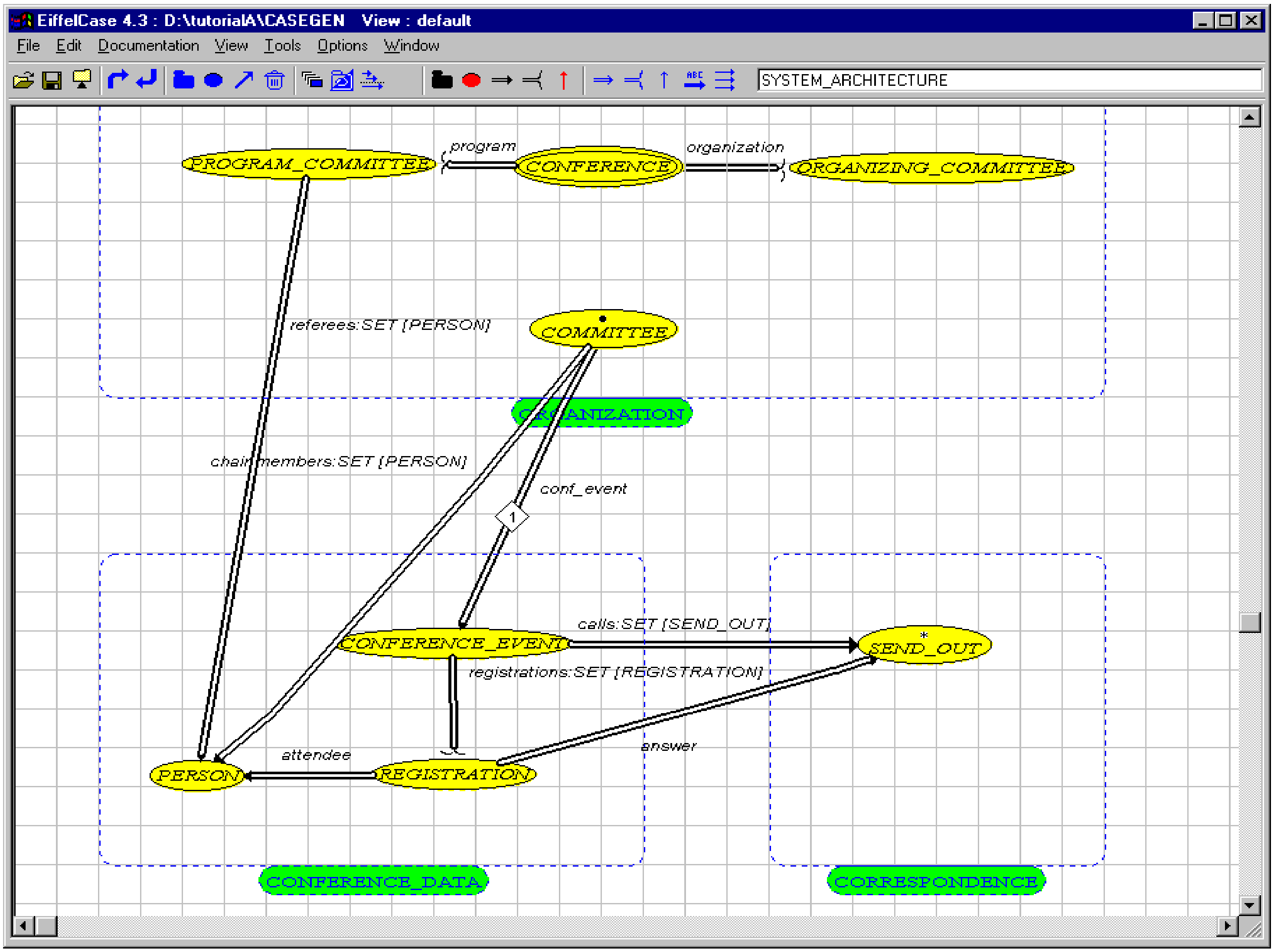
3 Point to **REGISTRATION**, and then right-click.

4 Click **Attribute**, and then type **registrations:SET [REGISTRATION]** in the **Feature name** box.

5 Click **Add**, and then click **OK**.

6 Repeat steps 2-5 for the remaining aggregation relation links in the preceding table, using the defined settings.

7 Move the relation labels until the system diagram matches the following:



In the next section, you will create the following inheritance relation links:

Source object	Target object
PROGRAM_COMMITTEE	COMMITTEE
ORGANIZING_COMMITTEE	COMMITTEE

Creating inheritance relation links

To create inheritance relation links:

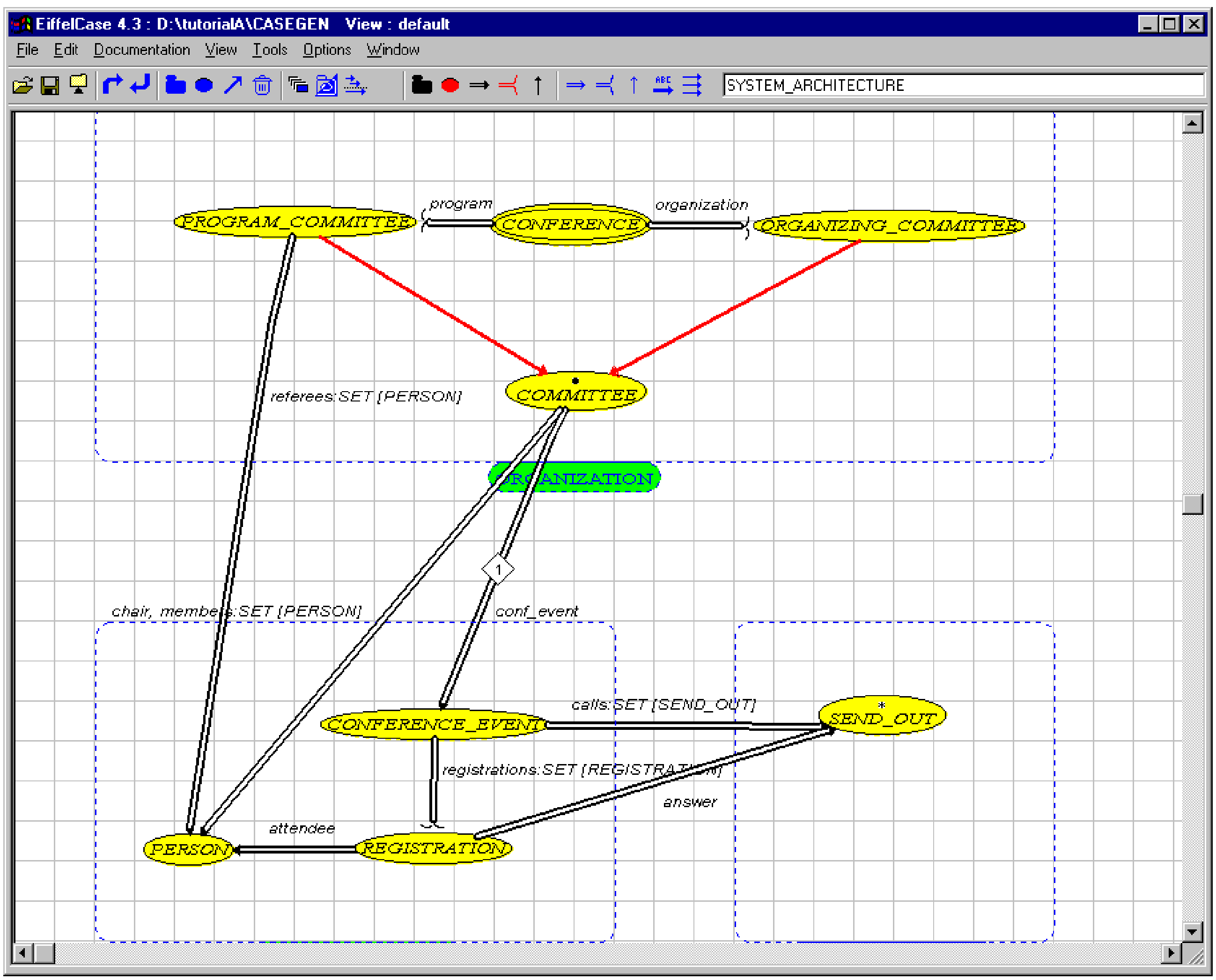
1 In the toolbar, click **Inheritance link**  .

2 Point to **PROGRAM_COMMITTEE**, and then right-click.

3 Point to **COMMITTEE**, and then right-click.

4 Point to **ORGANIZING_COMMITTEE**, and then right-click.

5 Point to **COMMITTEE**, and then right-click.



BON style rules require all relation links to be either vertical or horizontal.

To conform to these rules, you must add handles to the relation links between **REGISTRATION** and **SEND_OUT**, **PROGRAM_COMMITTEE** and **COMMITTEE**, and **ORGANIZING_COMMITTEE** and **COMMITTEE**.

The added angle or bend in the link is called a *handle*. A relation link can have as many handles as you want. Hold CTRL, and then click a relation link to display any existing handles. You can then use the drag-and-drop operation to move a handle.

Adding handles

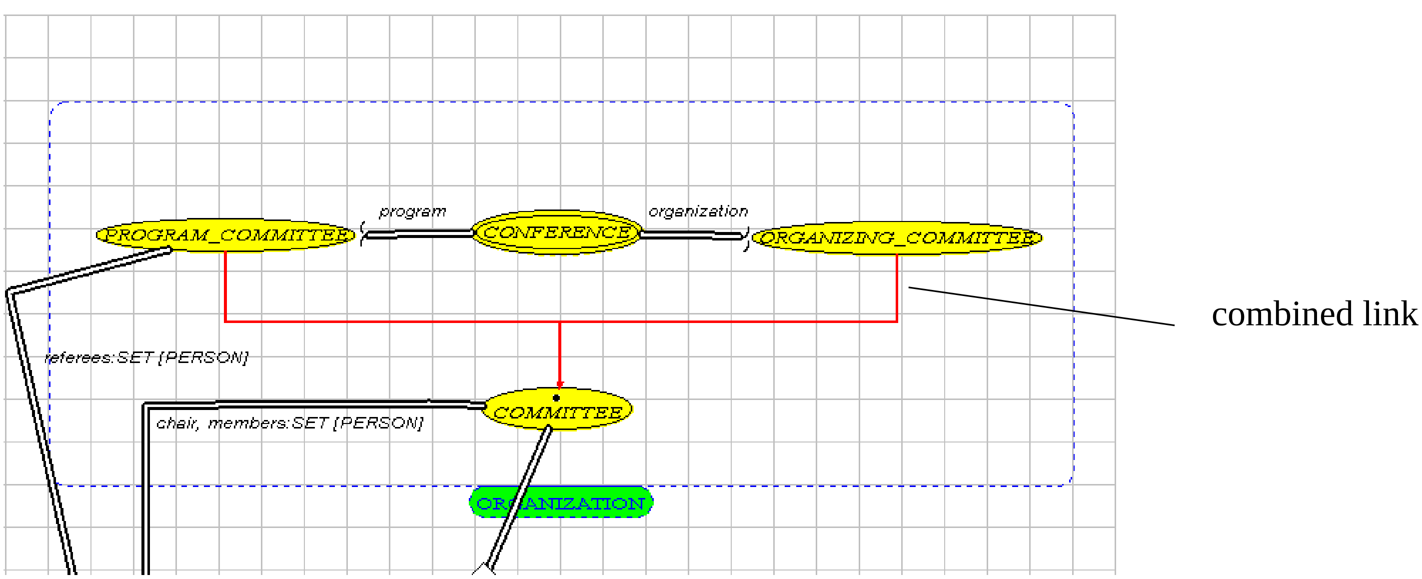
To add a handle to a relation link:

1 Point to the client relation link between **REGISTRATION** and **SEND_OUT**, and then drag the link to the right until it forms a ninety degree angle.

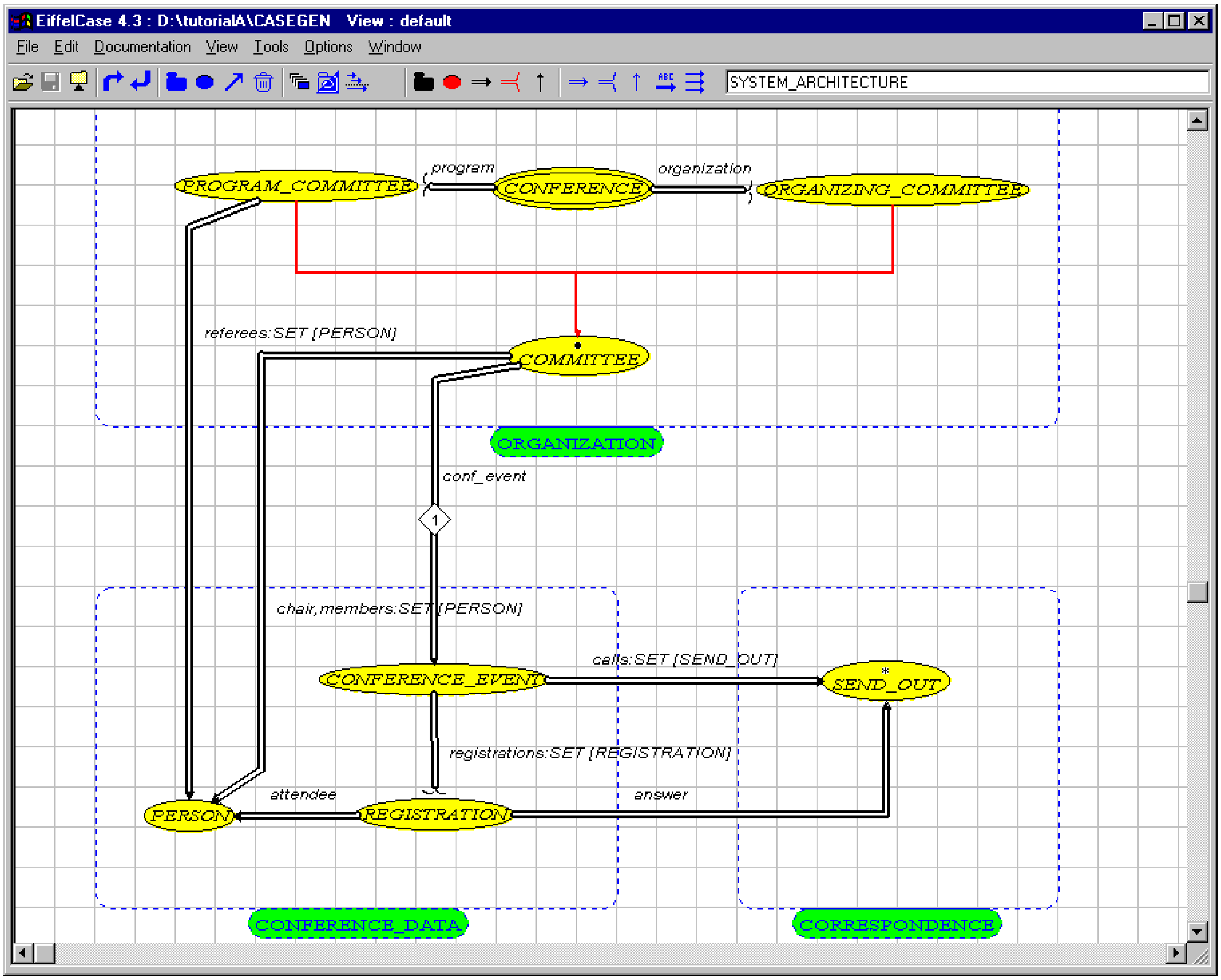
2 Point to the client relation link between **PROGRAM_COMMITTEE** and **PERSON**, and then drag the link to the left until it forms a ninety degree angle.

3 Point to the client relation link between **COMMITTEE** and **PERSON**, and then drag the link to the left until it forms a ninety degree angle.

4 Point to the relation links between **PROGRAM_COMMITTEE** and **COMMITTEE** and **ORGANIZING_COMMITTEE** and **COMMITTEE**, and then drag the links until they combine to match the following:



5 Move the labels as needed, until the system diagram matches the following:



The system diagram is now complete.

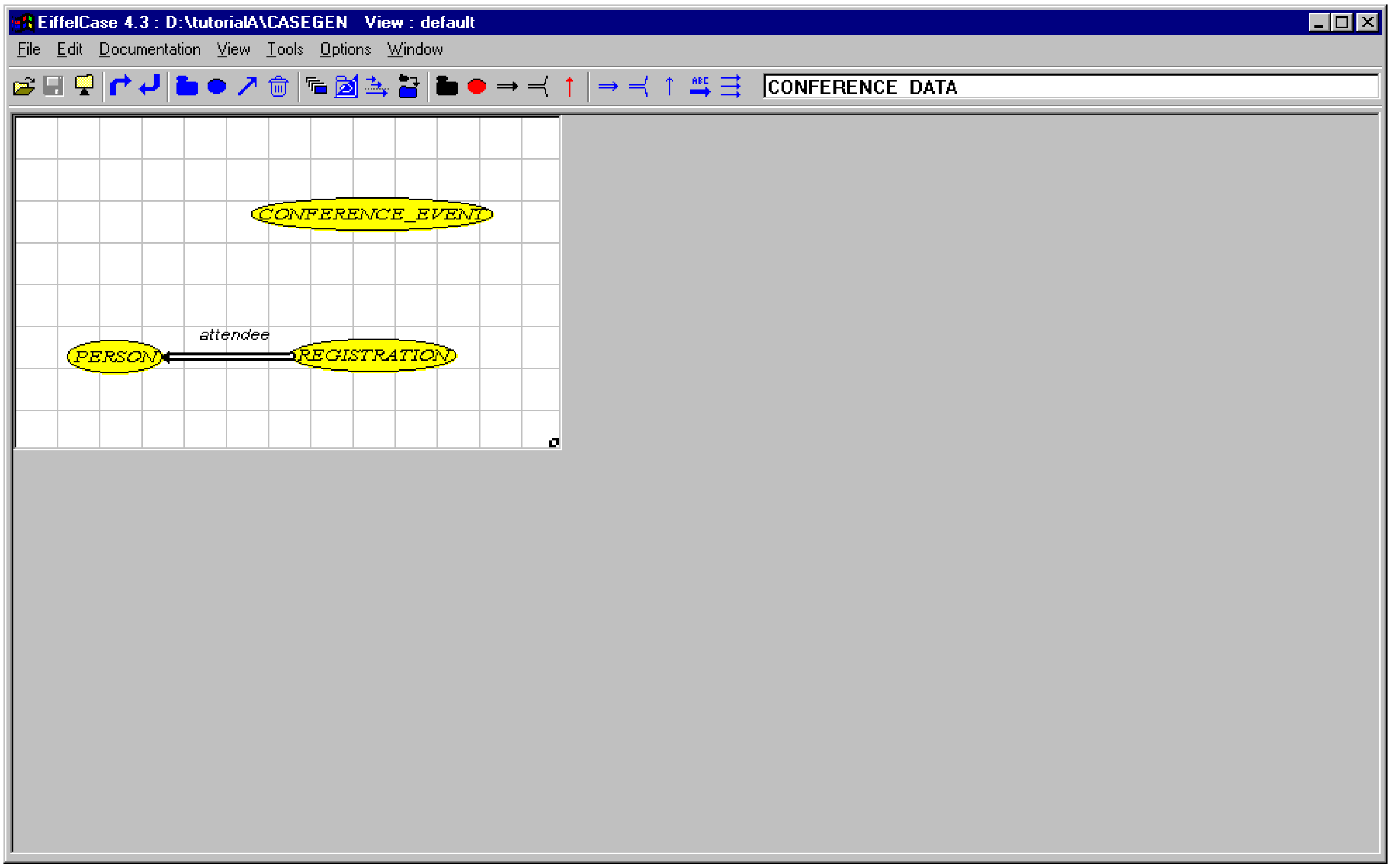
The final steps in the exercise are to display the contents of **CONFERENCE_DATA**, and then generate the Eiffel text for the cluster.

3.11 Displaying the contents of a cluster

To display the contents of **CONFERENCE_DATA**:

1 Right-click **CONFERENCE_DATA**.

2 Point anywhere in the workspace, and then right-click.



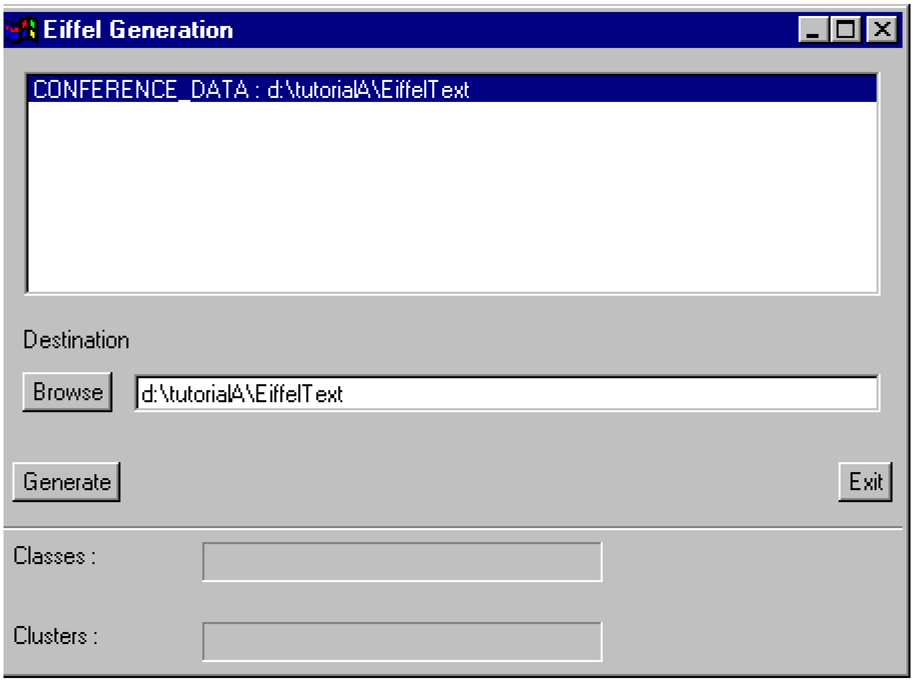
You will now generate Eiffel code for this cluster.

3.12 Generating Eiffel code

To generate the Eiffel code for the active cluster:

1 On the **File** menu, click **Generate Eiffel (this cluster).**

.



2 In the **Destination** box, type *drive:***\tutorialA\EiffelText**, and then click **Generate**.

The **Classes** and **Clusters** indicators display the percentage of generation completed (not available in UNIX).

You can use a text editor to view the generated file (.e) — drive:\tutorialA\EiffelText\conference_event.e.

3.13 Towards further study

You have now created your first EiffelCase system and generated the corresponding Eiffel text. This means that you have learned many of the basic EiffelCase procedures, enabling you to produce a system diagram. The next chapter details one of the more advanced features of EiffelCase — reverse engineering.

4

Tutorial B: reverse engineering

EiffelCase provides powerful tools to examine the structure of your software and to modify it using visual techniques. This chapter illustrates how EiffelCase and the EiffelBench development environment interact to implement the idea of reversible software development.

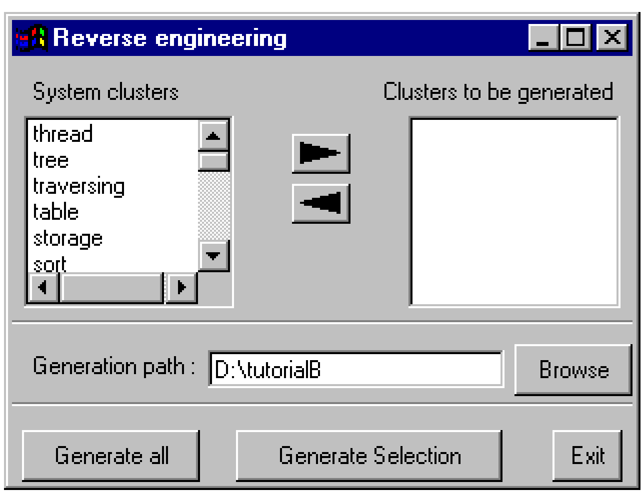
For the purposes of this example, it is assumed that you created an Eiffel system using EiffelBench and now want to produce a graphical system diagram, an HTML file, and then generate Eiffel text using EiffelCase.

4.1 Reverse engineering in EiffelBench

To use the **Reverse engineer (case)** command in EiffelBench, you must first compile the example $EIFFEL4/examples/case. For more information on compiling a system, see *ISE Eiffel: The Environment*.

After you compile this system using EiffelBench:

1 On the **Special** menu, click **Reverse engineer (case)**.



2 Click **Generate all**.

The **Reverse Engineering Project** dialog box appears and displays the percentage of generation completed.

EiffelCase creates a **CASEGEN** subdirectory that contains a system file (.ecr) in the path you enter (default).

3 On the **File** menu, click **Exit**.

System file

When you create a new EiffelCase system, by default, a **CASEGEN** subdirectory is added to the EiffelBench system directory. The EiffelBench system directory also contains the system file (.ecr), whose default file name is the same as your EiffelBench system name.

For example, if **MY_PROJECT** is the name of your EiffelBench system, the generated file is **MY_PROJECT.ecr**. For more information on the system file, see chapter 2, “Getting started”, and appendix A, “System resources”.

4.2 Starting EiffelCase

How you start EiffelCase depends on the operating system you are using. For more information on starting EiffelCase, see chapter 2, “Getting started”.

4.3 Opening your system and viewing the contents

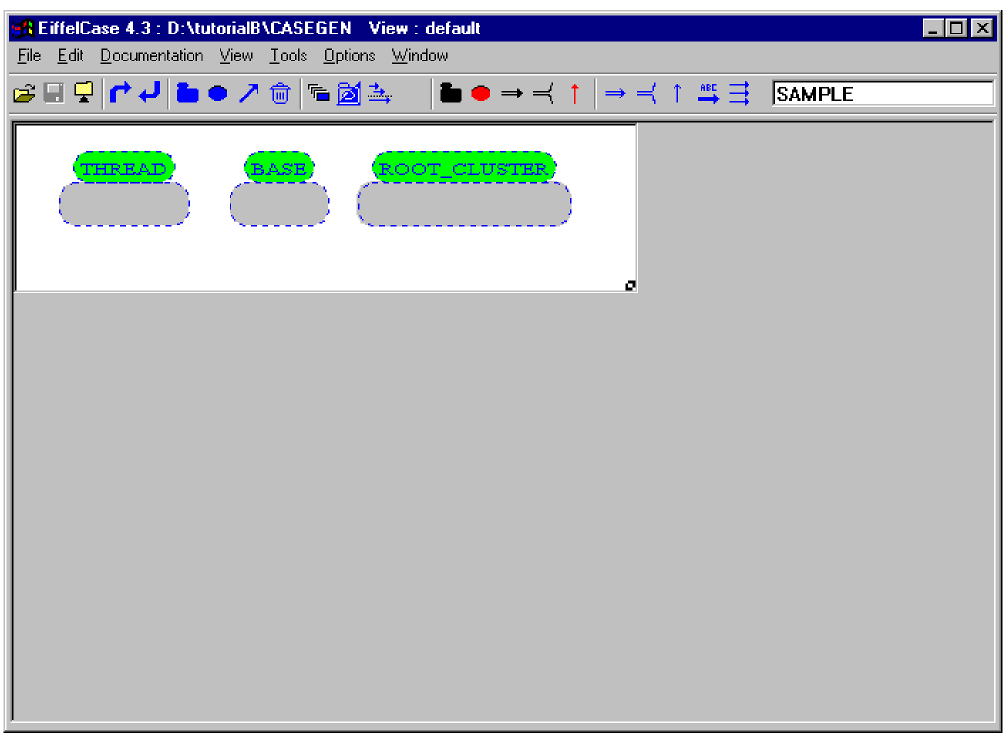
In this section, you will open your system and view the contents of a cluster.

To open your system:

1 On the **File** menu, click **Open**.

2 In the **Look in** list, click the folder that contains the **CASEGEN** subdirectory

3 Click sample.ecr, and then click **Open**.



The path, file name and active view for the system display in the EiffelCase title bar. By default, all active clusters in your system display iconified in the workspace.

Viewing the textual contents of a cluster

There are several ways to view the contents of a development object: in this case, cluster **BASE**. In the next section, you will use EiffelTips to display the contents of **BASE**. When you point to an iconified cluster or class, EiffelTips appear and display information about that object.

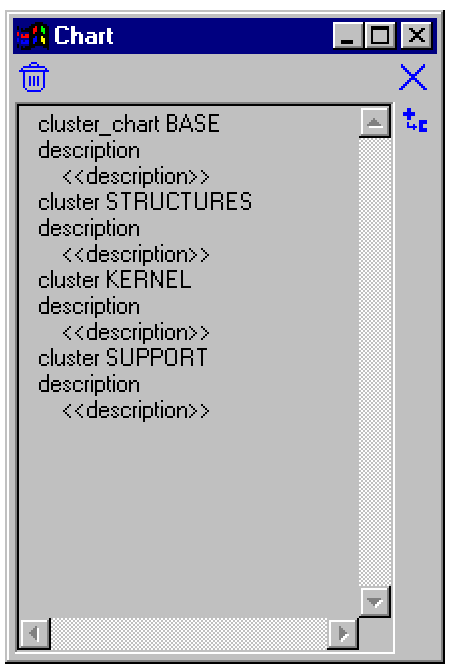
Using EiffelTips with clusters

To view the contents of cluster **BASE** using EiffelTips:

- Point to **BASE**.

After a few seconds, the **Chart** tool appears and displays information about **BASE** and its contents.

For more information on the **Chart** tool, see chapter 6, “EiffelCase tools”.



The **Chart** tool displays until you move the pointer or click the **Close** button.

To view the contents of **BASE** in more detail, you need to use another method. In this example, you will use the pick-and-drop operation.

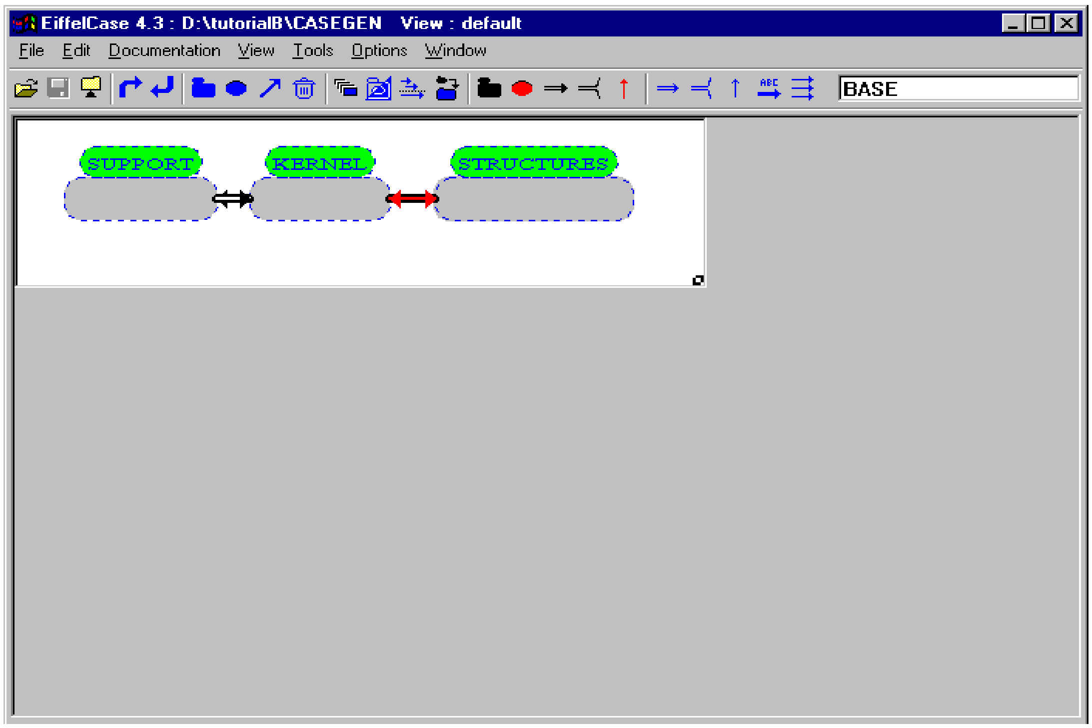
Viewing using the pick-and-drop operation

To view the contents of **BASE** using the pick-and-drop operation:

1 Right-click **BASE**.

2 Point anywhere in the workspace, and then right-click.

The subclusters of **BASE** display iconified.



To view the contents of a subcluster in more detail, you can open a new **Cluster** tool. For more information on the **Cluster** tool, see chapter 6, “EiffelCase tools”.

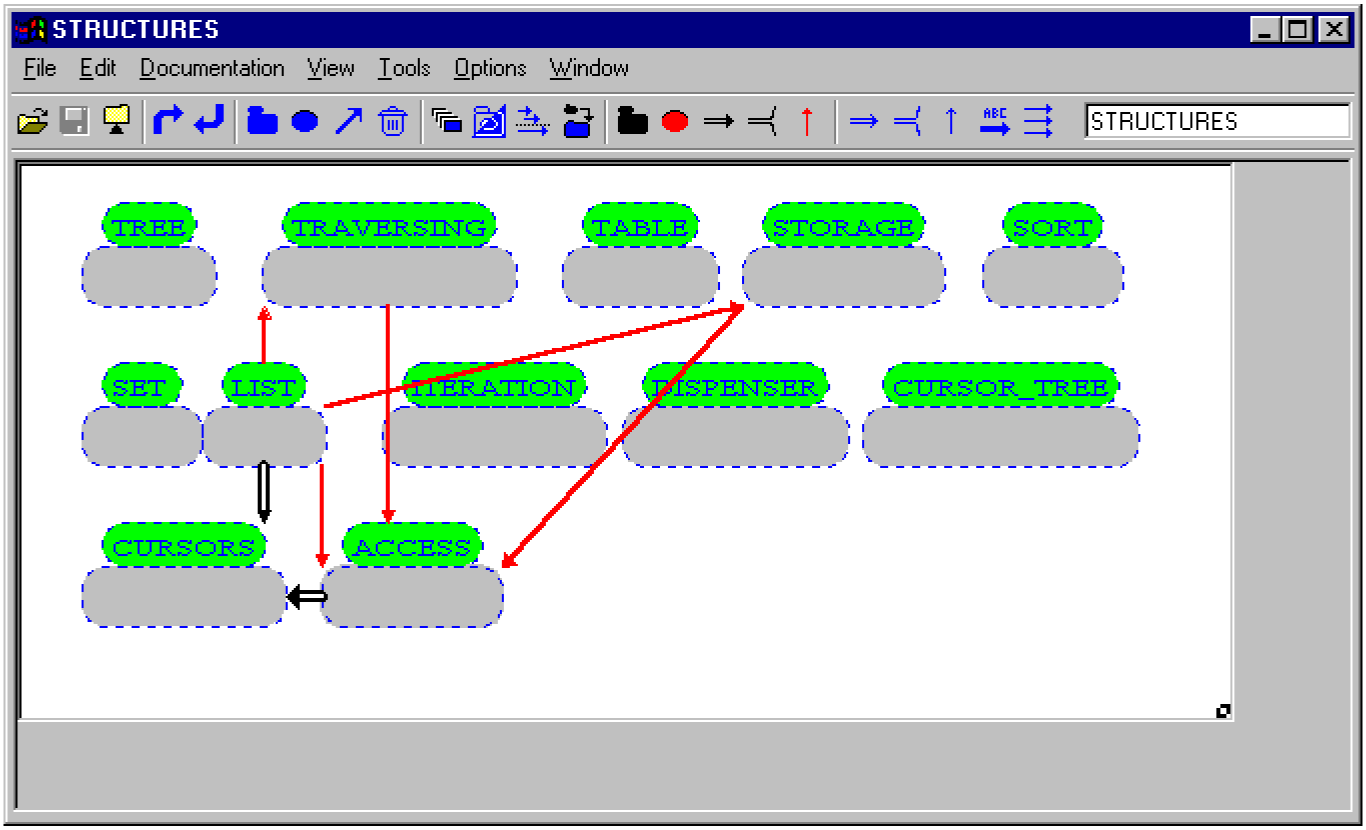
In the next section, you will view the contents of **LIST**, which is a subcluster of **STRUCTURES**, using the **Cluster** tool.

Opening a new Cluster tool

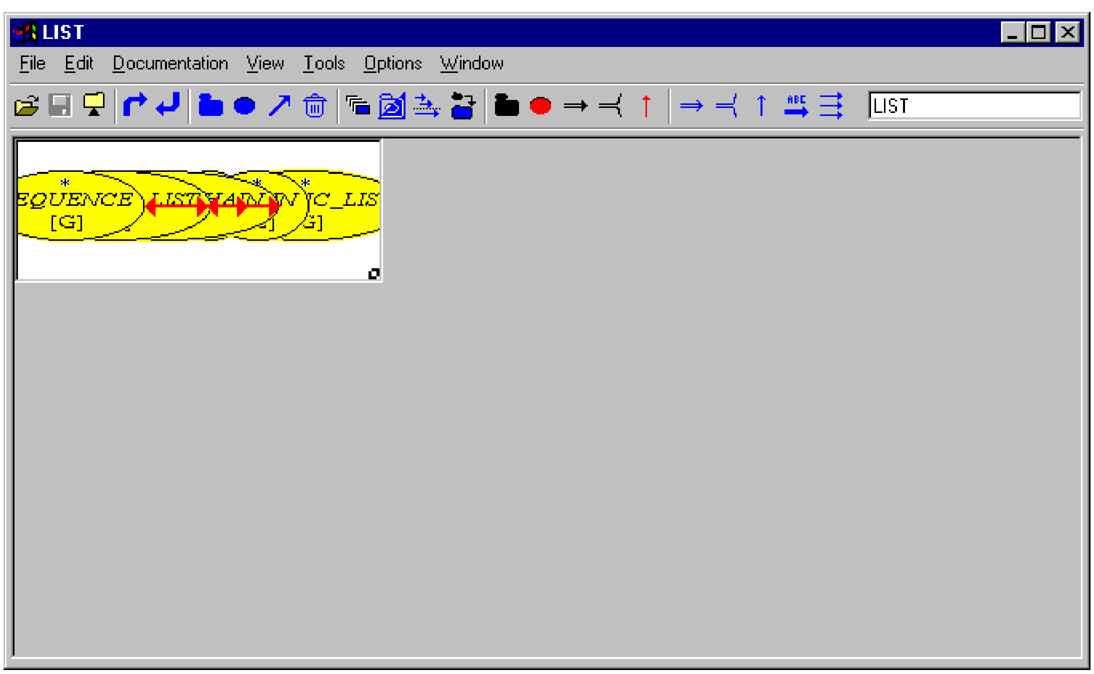
To open a new **Cluster** tool:

1 Hold CTRL, and then right-click **STRUCTURES**.

The **Cluster** tool appears with all subclusters in **STRUCTURES** iconified.



2 Right-click **LIST**, point anywhere in the workspace, and then right-click.



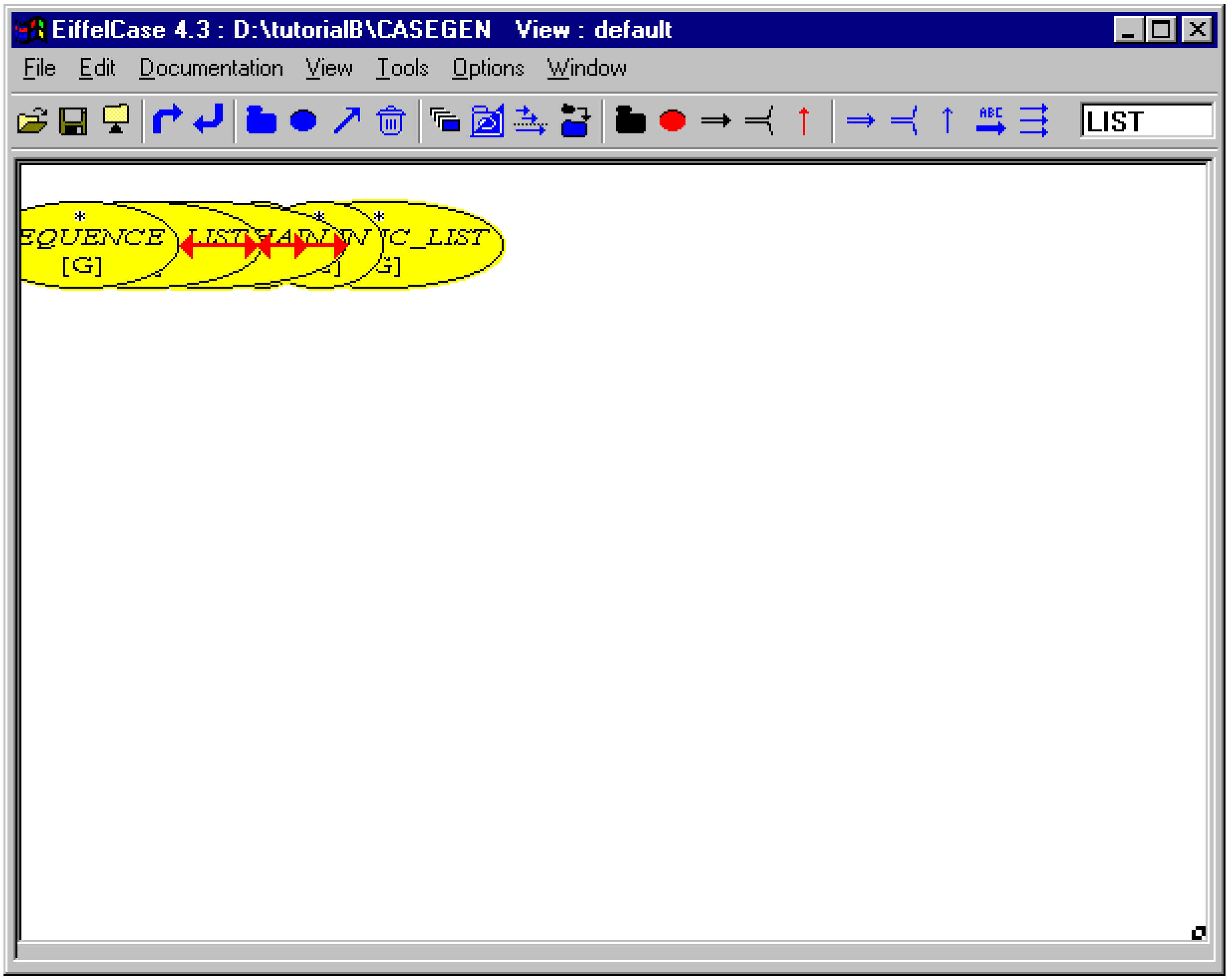
You can resize the window, and then use the drag-and-drop operation to move the classes to increase readability.

Resizing the window

To resize the window:

1 Click the **Maximize** button.

2 Point to the lower right corner of the workspace, and then drag it to the lower right corner of the application window.

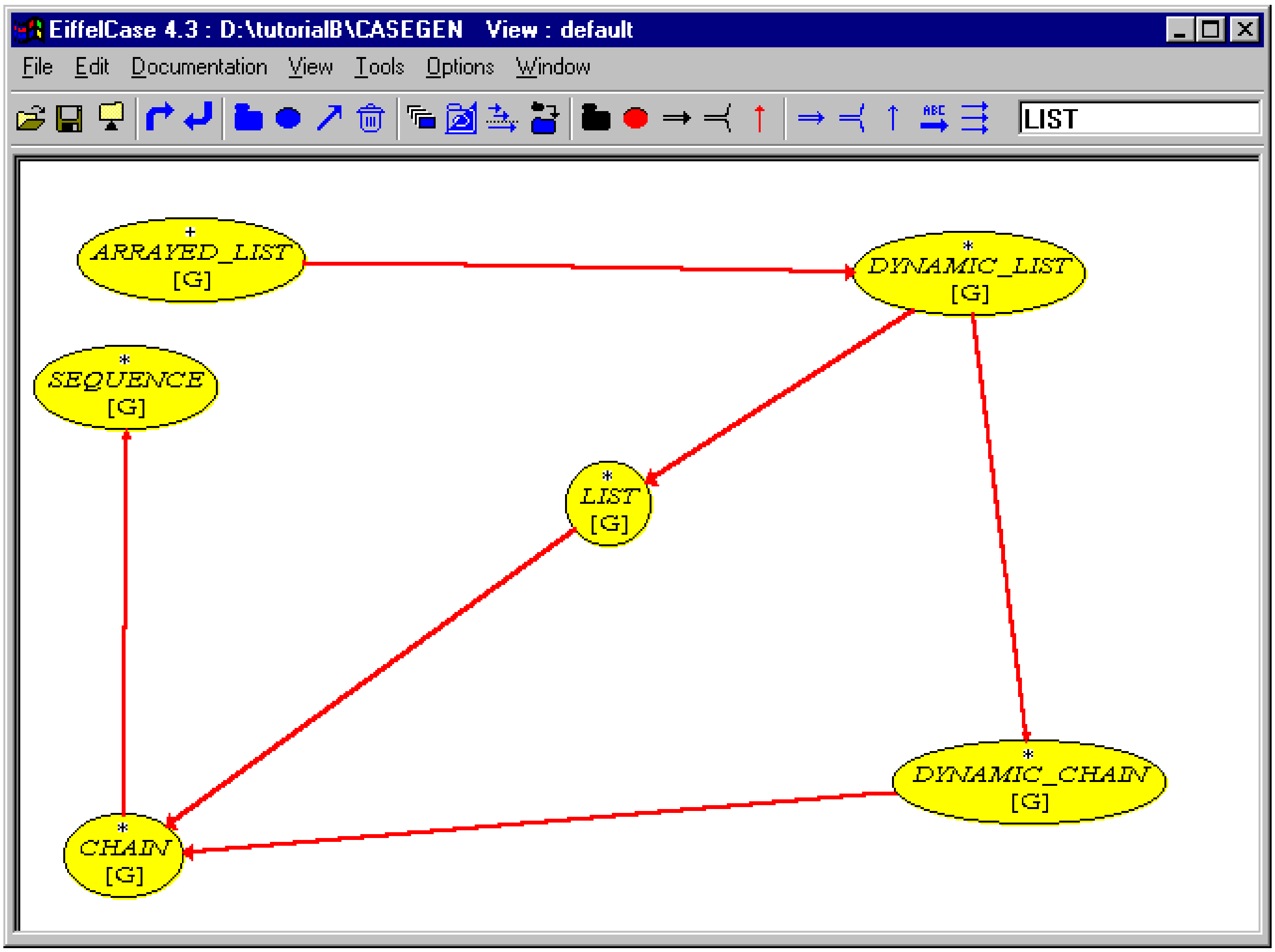


Moving classes

To move classes:

1 Point to **ARRAYED_LIST**, and then drag the class to a new position.

2 Repeat until all classes are visible and the system diagram matches the following:



3 On the **File** menu, click **Save**.

You will now create a view for the system.

4.4 Views and the View tool

A *view* provides a different subset of the graphical properties of your system. At any time during an EiffelCase session, you are working on a specific view of your system. The name of the active view displays in the EiffelCase title bar.

If you change any of the graphical properties of your system, and then save the active state, only the active view changes. However, any change affecting an implementation property modifies all views. Examples include: adding a class; permanently removing a relation link; making a cluster a subcluster of another.

The **View** tool defines and changes the views available to the active system. For more information on the **View** tool, see chapter 6, "EiffelCase tools".

In this example, you will create a new view that illustrates all inheritance relation links in **LIST** (**inherit_list_view**). *Relation links* are full-fledged development objects that exist between two classes, two clusters, or a class and a cluster.

There are three types of relation links:

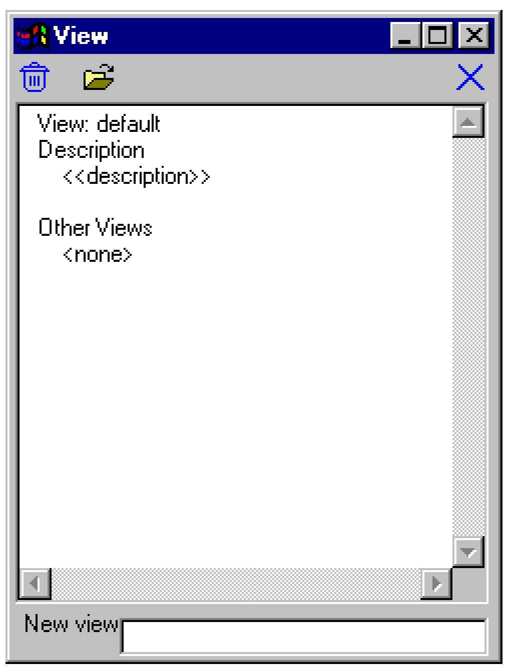
- **client** — a class is a client of another if it uses a feature of the other (supplier) class
- **inheritance** — a class is an heir of another if it incorporates the features of the other class in addition to its own
- **aggregation** — every object of a certain type is a combination (an aggregate) of zero or more objects of a certain type; For example, a "car" can be defined as an aggregation of "engine", "body" and so forth.

4.5 Creating inheritance relation link view

To create the inheritance relation link view:

1 On the **View** menu, click **View Tool**.

The **View** tool appears and lists all defined views.



2 In the **New view** box, type **inherit_list_view**, press ENTER, and then click the **Close** button.

3 On the **File** menu, click **Save**.

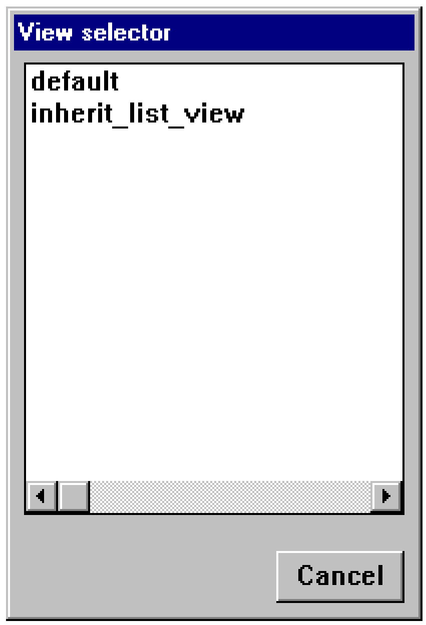
This only saves the active view as **inherit_list_view —** it does not mean you are working in the new view. You must switch to a view to work in that view.

In the next section, you will switch to the **inherit_list_view**, and then redisplay the contents of **LIST** using the pick-and-drop operation.

Switching to the inherit_list_view

To switch to the **inherit_list_view** view:

1 On the **View** menu, click **Change View**.



2 Click **inherit_list_view**.

Redisplaying the contents of LIST

1 Right-click **BASE**.

2 Point anywhere in the workspace, and then right-click.

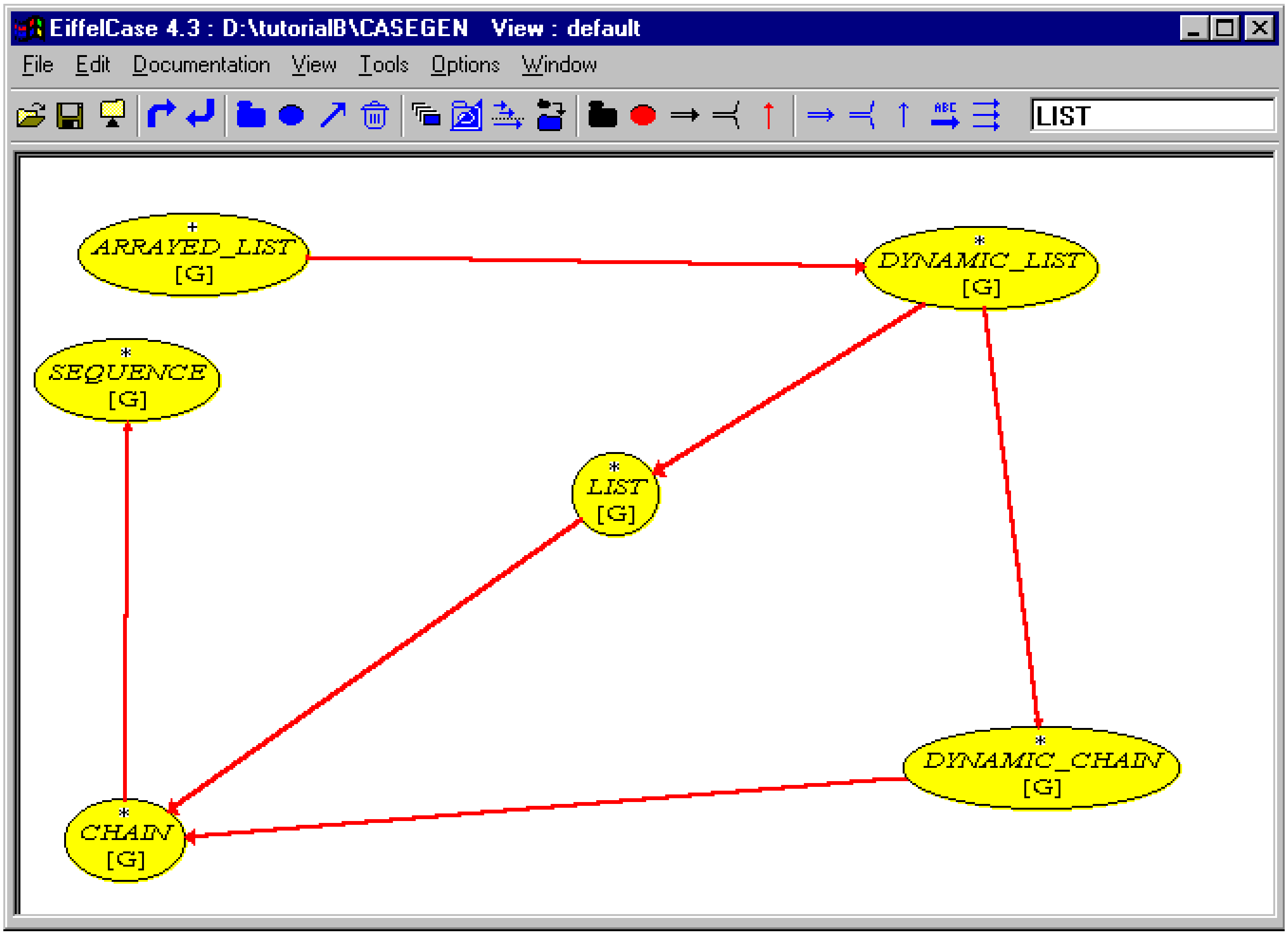
3 Right-click **STRUCTURES**.

4 Point anywhere in the workspace, and then right-click.

5 Right-click **LIST**.

6 Point anywhere in the workspace, and then right-click.

The classes of **LIST** redisplay.



You can use the **Advanced Color** tool to add color to the workspace. For more information on the **Advanced Color** tool, see chapter 6, "EiffelCase tools".

Using the Advanced Color tool

To use the **Advanced Color** tool:

1 On the **Options** menu, click **Advanced Color Tool**.

2 In the **Classes** box, type **LIST**, and then press ENTER.

3 Under **Select type of entities**, select **Heir**, and then clear **Supplier**.

4 Select **Override previous coloring**, and then move the **Class level** slider to the right until 1 displays.

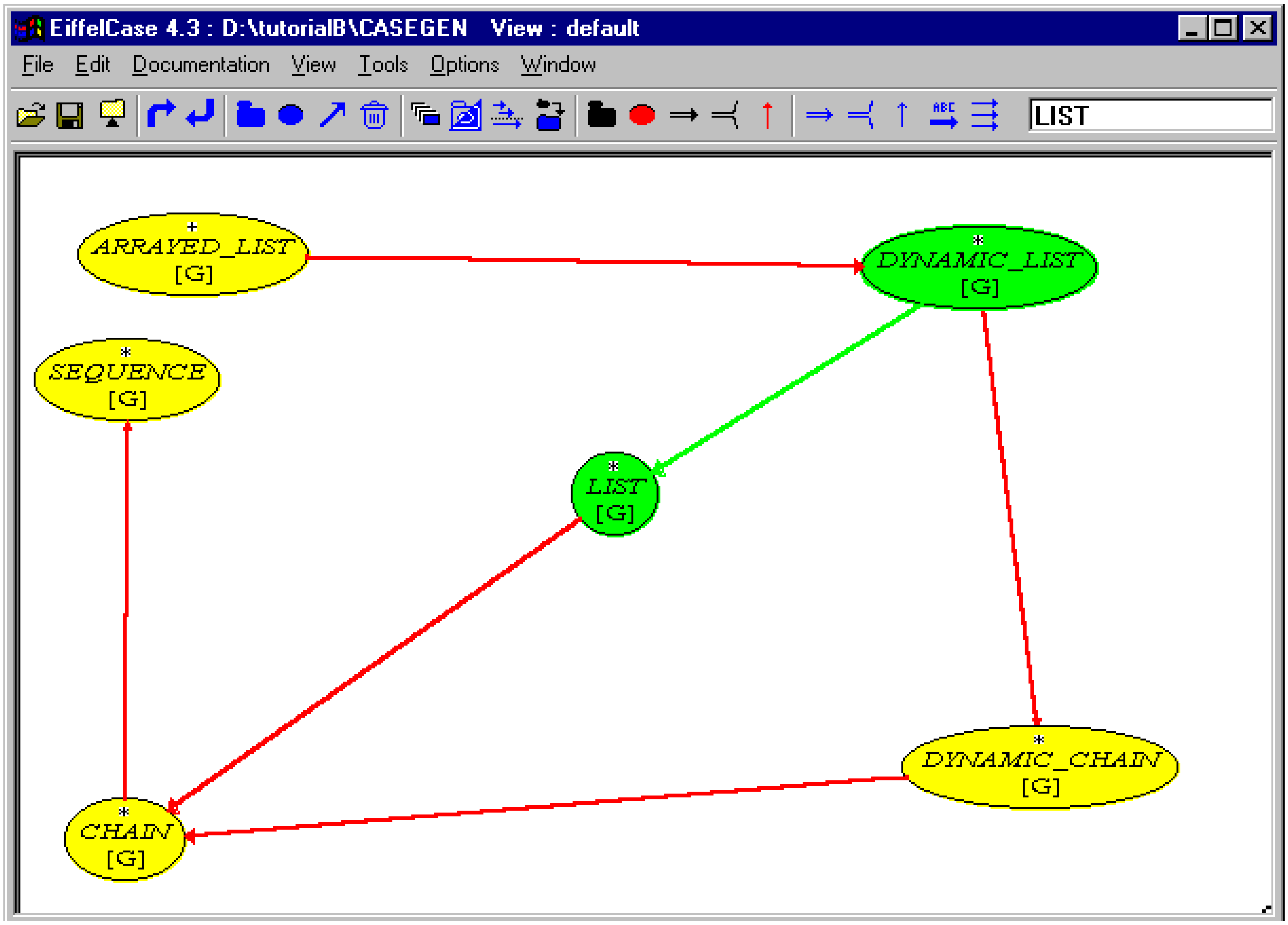
5 Move the **Link level** slider to the right until 1 displays.

6 Type **green** in the **Color** box, and then press ENTER

7 Click **OK**, and then click **EXIT**.

All parents and relation links associated with **LINK** redisplay in green:

.



There are several methods of viewing the contents of a class, including EiffelTips and using the **Class** tool.

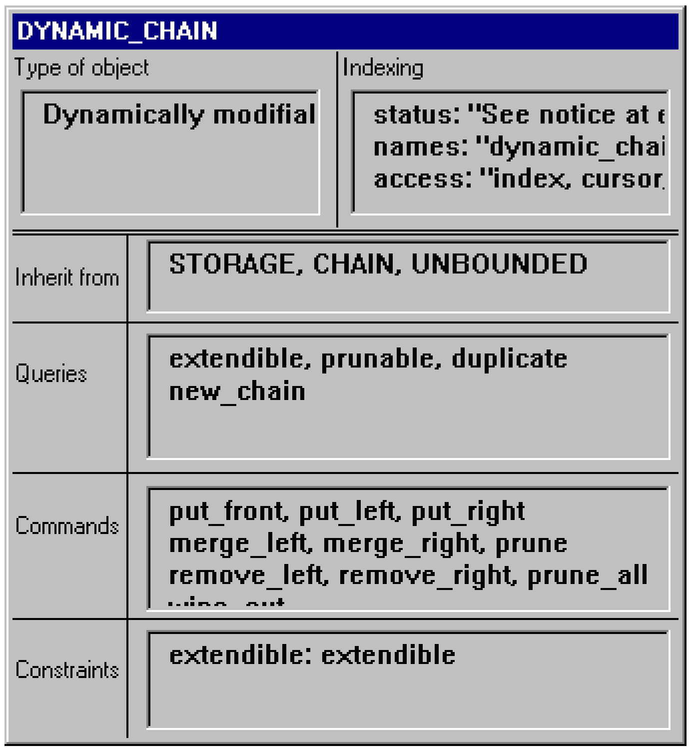
In the following sections, you will explore the contents of a class (**DYNAMIC_CHAIN**) using both EiffelTips and the **Class** tool.

Using EiffelTips with classes

To view the contents of **DYNAMIC_CHAIN** using EiffelTips:

- Point to **DYNAMIC_CHAIN**.

A dialog box that contains textual information about **DYNAMIC_CHAIN** and its contents in the BON Class Chart format appears.



This dialog box displays until you move the pointer. To view the contents of **DYNAMIC_CHAIN** in more detail, you need to use another method — the **Class** tool.

4.6 Classes and the Class tool

The **Class** tool sets properties, features, indexing information and constraints for the active class.

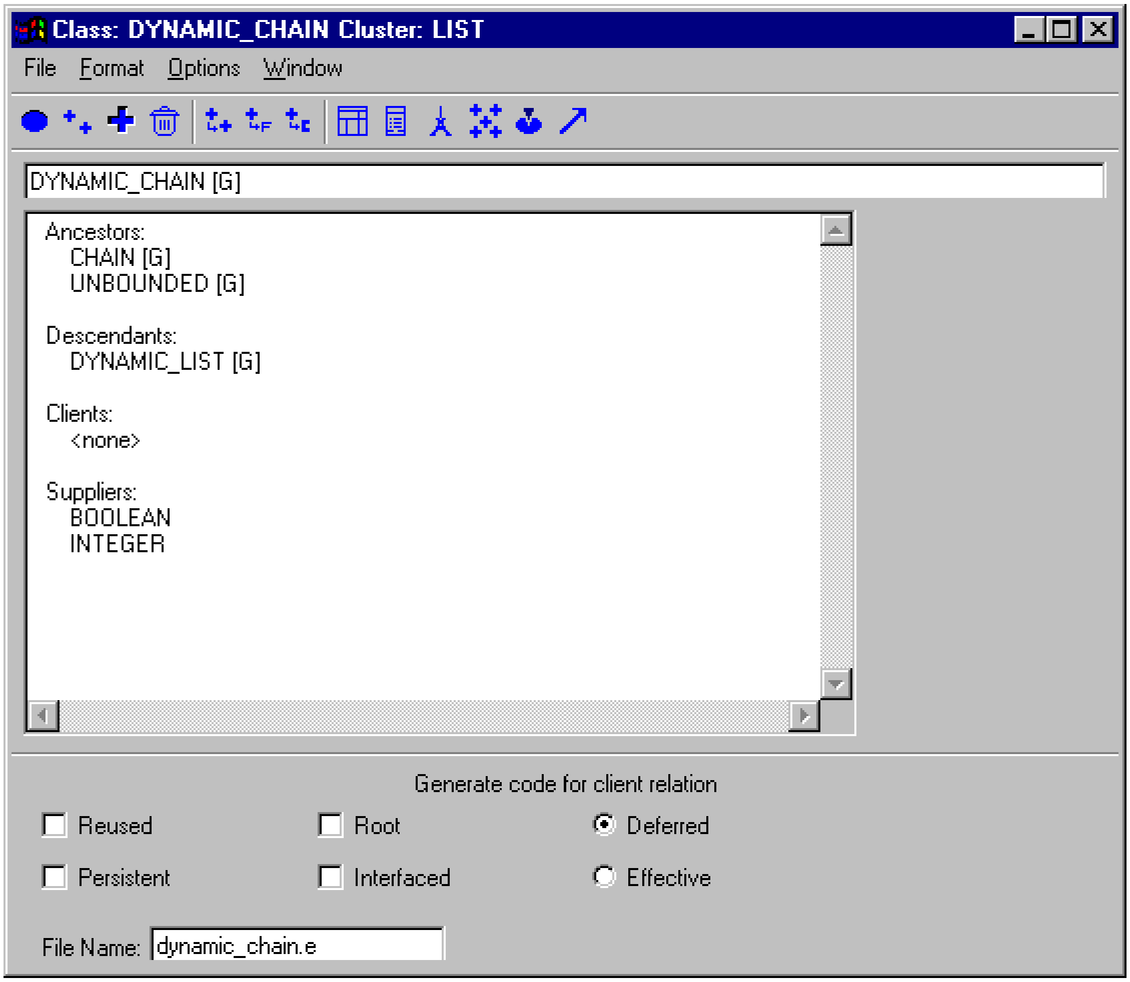
Using the Class tool

To view the contents of **DYNAMIC_CHAIN** using the **Class** tool:

1 Right-click **DYNAMIC_CHAIN**.

2 Point anywhere in the workspace, and then right-click.

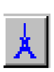
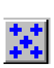
The **Class** tool appears with the relational format (default) displaying in the tool window.



You can use the commands on the Class toolbar to view the contents of **DYNAMIC_CHAIN** in different formats.

Viewing different formats

To view a format below, click the corresponding button on the **Class** toolbar:

- **Chart**  : displays a textual representation of **DYNAMIC_CHAIN** information.
- **Specifications**  : displays the *specification* for **DYNAMIC_CHAIN** — the part of the class visible to other classes.
- **Eiffel Code**  : displays **DYNAMIC_CHAIN** in syntactically-correct Eiffel form.
- **Features**  : lists all features in **DYNAMIC_CHAIN**.

- **Modified features** : lists all features in **DYNAMIC_CHAIN** that were added or changed during the active session.

- **Relations** : lists all relation links for **DYNAMIC_CHAIN**.

For more information on the **Class** tool, see chapter 8, “Class tool”.

Exiting the Class tool

To exit the **Class** tool:

- On the **File** menu, click **Exit**.

The last feature of EiffelCase you will explore before generating documentation and Eiffel text is the class dictionary in the **System** tool.

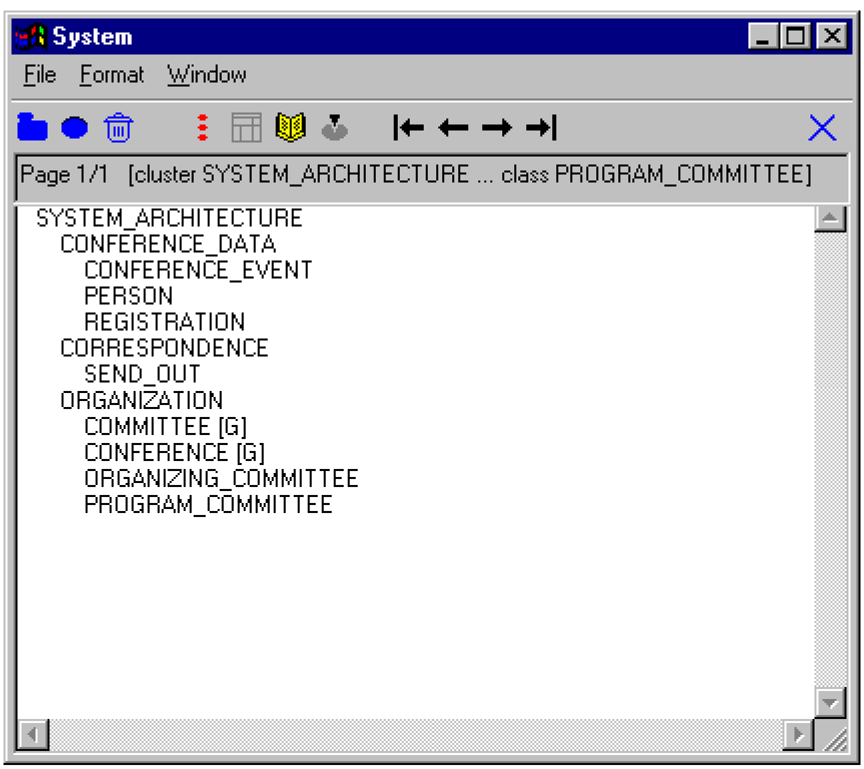
The **System** tool sets, adds and modifies textual documentation information for the active system, and the corresponding clusters and classes. Only one **System** tool can be open per EiffelCase session.

For more information on the **System** tool, see chapter 7, “System tool”

4.7 System tool

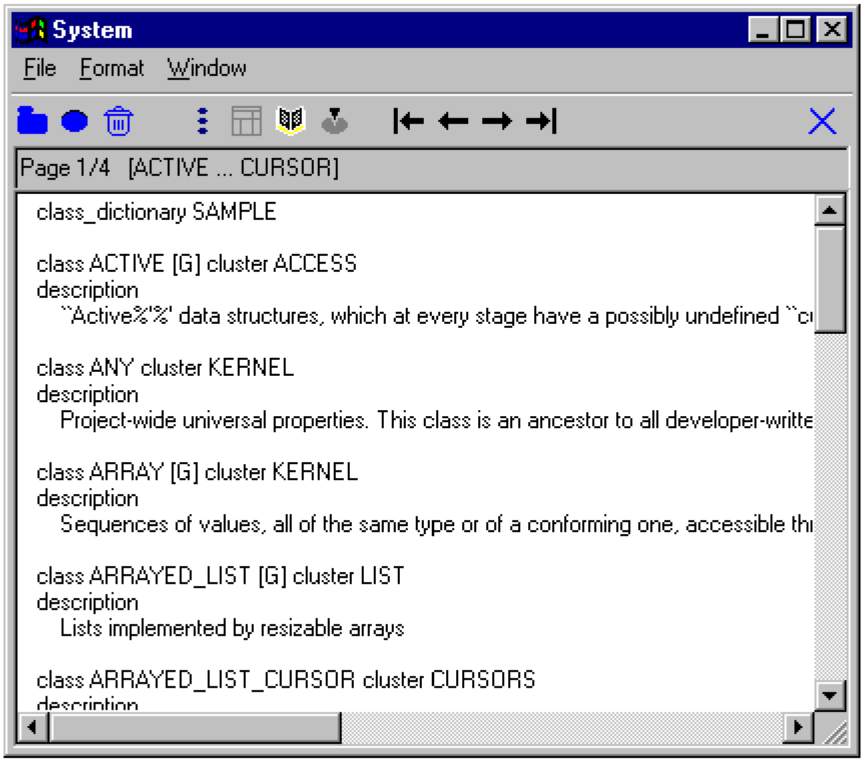
To display the **System** tool and the class dictionary:

1 On the **Tools** menu, click **System Tool**.



2 On the **System** toolbar, click **Show class dictionary**  .

The class dictionary lists all active classes in your system alphabetically, and includes descriptive information.



3 On the **File** menu, click **Exit**.

The next step is to generate HTML for the active system.

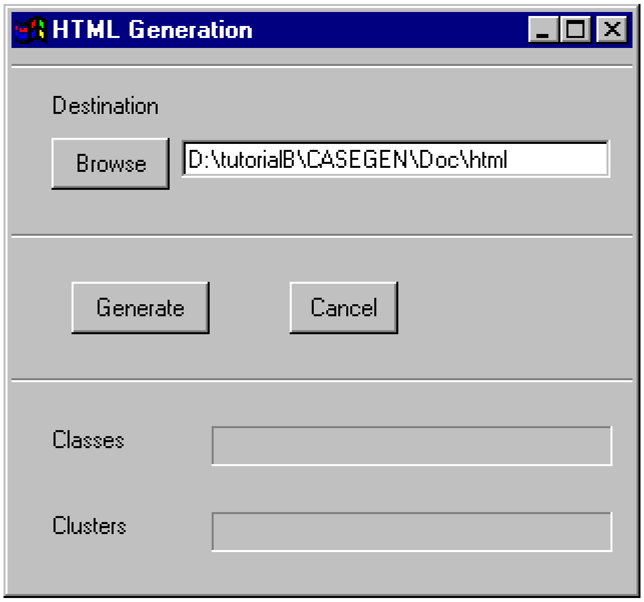
4.8 Report generation

There are several different types of reports you can create using EiffelCase. In this section, you will generate an HTML file.

Generating the HTML file

To generate the HTML file:

1 On the **Documentation** menu, click **HTML (whole system)**.



2 Click **Generate**.

By default, EiffelCase generates an HTML file in $CASEGEN\Doc\html.

The **Clusters** and **Classes** progress indicators display the percentage of generation completed (not available in UNIX).

Viewing the HTML file

To view the HTML file:

- Start your Internet browser, and then open the HTML file $CASEGEN\Doc\html\index.html.

For more information on using your browser, see your Internet browser documentation.

Because the cluster that you will use to generate Eiffel text (**ROOT_CLUSTER**) is not the active cluster (**LIST**), you must redisplay the initial cluster diagram using the **Retarget to parent** command.

This command replaces the active cluster diagram with the cluster diagram of its parent. It is not available at the System level, since the System level is the highest level diagram.

4.9 Redisplaying the initial cluster diagram

To return to initial cluster diagram:

- On the toolbar, click **Retarget to parent**  three times.

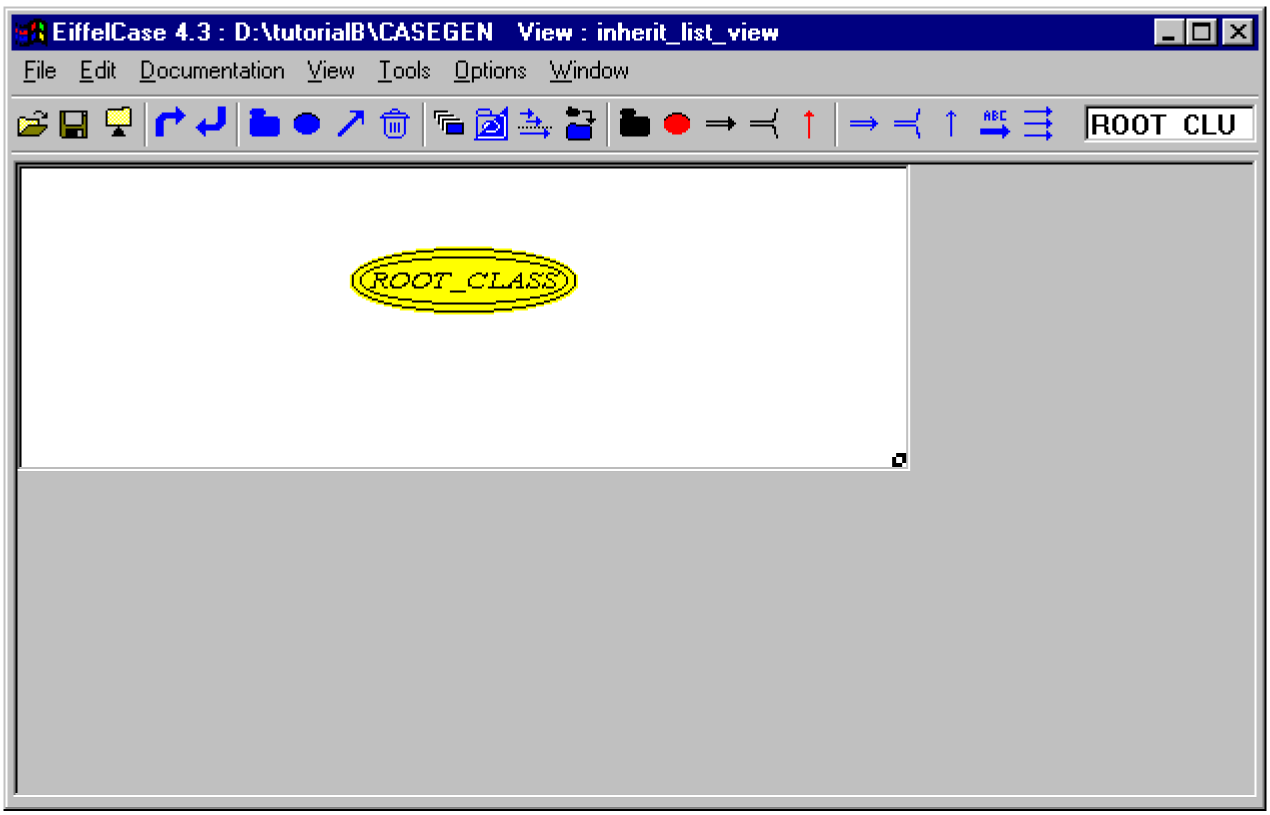
For more information on Retarget to parent, see chapter 5, “Menu bar and toolbar”.

Displaying the contents of ROOT_CLUSTER

To display the contents of **ROOT_CLUSTER**:

1 Right-click **ROOT_CLUSTER**.

2 Point anywhere in the workspace, and then right-click.



4.10 Generating Eiffel text

To generate the Eiffel text for **ROOT_CLUSTER**:

1 On the **File** menu, click **Generate Eiffel (this cluster)**.

2 Click **Browse**, in the **Directories** list, and then click the folder that contains the **CASEGEN** sub-

directory.

3 Click **OK**, and then click **Generate**.

The **Classes** and **Clusters** indicators display the percentage of generation completed (not available in UNIX). You can use a text editor to view the generated file (.e).

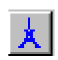
As mentioned earlier, EiffelCase supports both forward and reverse engineering. When performing either of these operations, modifications can be made to the same class in two different ways. If this happens, the **Merging** tool appears and allows you to reconcile the differences.

To illustrate this tool, you will add a feature to **ROOT_CLASS** using the **Class** tool. You will then use the **Merging** tool to reconcile the differences, before generating Eiffel text a second time.

4.11 Adding a feature

To add a feature to **ROOT_CLASS**:

1 Hold down CTRL, and then right-click **ROOT_CLASS**.

2 In the Class toolbar, click **Show Eiffel code**  .

3 Click **Add command**, and then on the **File** menu, click **Exit**.

4.12 Regenerating Eiffel text

To regenerate the Eiffel text for **ROOT_CLUSTER**:

1 On the **File** menu, click **Generate Eiffel (this cluster)**.

2 In the **Eiffel Generation** dialog box, click **Generate**.

Because there are differences between the saved version of **ROOT_CLASS** and the active one, the **Merging** tool appears. For more information on the **Merging** tool, see chapter 14, “Merging tool”.

Differences detected by the **Merging** tool include inheritance properties, individual features, and invariant clauses. For each one of these elements, you can choose the text that you want to use — from either the active EiffelCase system or the Eiffel text.

In order to incorporate the changes made, you will select the EiffelCase text version for every difference.

Using the Merging tool

To use the Merging tool:

- In the **Select a version** area, under **Everything**, click **EiffelCase**, and then click **Generate**.

4.13 Towards further study

You have now learned how to reverse engineer an EiffelBench system and produce the corresponding graphical system diagram, HTML file, and Eiffel text. During this process, many of the new or enhanced features of EiffelCase were also introduced, including view definition, EiffelTips, HTML generation, and the **Merging** tool. The remaining chapters in this guide explore the complete functionality of EiffelCase, in great detail.

5

Menu bar and toolbar

This chapter provides detailed descriptions of the EiffelCase menus, submenus and commands, as well as toolbar buttons, holes and commands. Where applicable, you will find references to other chapters or sections in this chapter for more information.

5.1 Menu bar

The Menu bar contains the **File**, **Edit**, **Documentation**, **View**, **Tools**, **Options**, and **Window** menus.

5.2 File menu

The **File** menu contains the following commands.

Create CTRL+N

Creates a new system file (.ecr) and view. In the **Choice** box, type the path and file name or click a folder in the **Directories** list. The path, file name, and view for the active system display in the EiffelCase title bar.

Open CTRL+O

Opens a previously-created system. In the **File name** box, type the path and file name or click a folder in the **Look in** list. The path, file name, and view for the active system display in the EiffelCase title bar.

Recent Systems

Lists the most recently opened system files. In the submenu, click to open a file.

Save CTRL+S

Saves the active system and all related resources in the active view.

If you change the name of the active system using the **Target Name** tool, a new system file is created in the EiffelCase system directory under the name you enter when you save the system.

For example, if you type **MY_SYSTEM** in the **Target Name** tool and then save the system, a file named MY_SYSTEM.ecr is created in the EiffelCase system directory.

Automatic Save

Saves the active system and all related resources at the time interval entered in the **Preference** tool. For more information on the **Preference** tool, see chapter 6, “EiffelCase tools”.

Save As

Creates a new view for the active system. In the **Choice** box, type the path and file name or click a folder in the **Directories** list. For more information on views, see chapter 2, “Getting started”.

Import Cluster

Imports all clusters from a previously-created EiffelCase system into the active system. In the **File name** box, type the path and file name or click a folder in the **Look in** list.

Import Glossary

Imports the text file (.crv) that contains cluster, class, and indexing structure information into the active system. In the **File name** box, type the path and file name or click a folder in the **Look in** list.

Generate Eiffel (whole system)

Generates Eiffel text for the active system.

In the **Eiffel Generation** dialog box, do one of the following, and then click **Generate**:

- Type the path and file name in the **Destination** box.
- Click **Browse**, and then in the **Choice** box, type a path or click a folder in the **Directories** list.

The **Classes** and **Clusters** indicators display the percentage of generation completed (not available in UNIX).

Generate Eiffel (this cluster)

Generates Eiffel text for the active cluster.

In the **Eiffel Generation** dialog box, do one of the following, and then click **Generate**:

- Type the path and file name in the **Destination** box.

- Click **Browse**, and then in the **Choice** box, type a path or click a folder in the **Directories** list.

The **Classes** and **Clusters** indicators display the percentage of generation completed (not available in UNIX).

Exit CTRL+Q

Quits EiffelCase.

5.3 Edit menu

The **Edit** menu contains the following commands.

History Tool CTRL+H

Lists the operations performed in the active session and corrects any mistakes. For more information on the **History** tool, see chapter 6, “EiffelCase tools”.

Redo CTRL+Y

Cancels your most recent **Undo** operation.

Undo CTRL+Z

Cancels your most recent operation.

Resizing

Resizes your workspace by the value (in pixels) that you click in the submenu:

Modify height

Height adjustment value: **+100**, **+50**, **-50**, or **-100**.

Modify width

Width adjustment value: **+100**, **+50**, **-50**, or **-100**.

Automatic resizing CTRL+R

Automatically resizes the workspace for optimum performance and viewing.

5.4 Documentation menu

The **Documentation** menu contains the following commands.

Documentation Tool CTRL+D

Sets report types and any printing commands.

Print Current Graph CTRL+P

Prints the graphical view of the active system.

In the **Print Graph** dialog box, do one of the following, and then click **OK**:

- Select the **Print to printer** check box, click **Setup**, and then set printing defaults.
- Select the **Print to file** check box, click **Browse**, and then in the **File name** box, type the path and file name or click a folder in the **Save in** list.

When printing to file, Windows generates a bitmap, UNIX a PostScript file.

HTML (whole system)

Generates a Hypertext Markup Language (HTML) file for the active system, which includes MAP definitions for each cluster in the system and generated graphics.

In the **HTML Generation** dialog box, do one of the following, and then click **Generate**:

- Type the path and file name in the **Destination** box.
- Click **Browse**, and then in the **Choice** box, type the path and file name or click a folder in the **Directories** list.

The **Classes** and **Clusters** indicators display the percentage of generation completed (not available in UNIX). By default, EiffelCase generates an HTML file in $CASEGEN\Doc\html.

Viewing the HTML file

To view the HTML file:

- Start your Internet browser, and then open the HTML file $CASEGEN\Doc\html\index.html.

For more information on using your browser, see your Internet browser documentation.

5.5 View menu

The **View** menu contains the following commands.

View Tool CTRL+G

Defines and changes views available to the active system.

Change View CTRL+E

Switches between all defined views for the active system. For more information on views, see chapter 2, “Getting started”.

Changing views

To switch to a different view:

- In the **View selector** dialog box, click a view.

Retarget to Parent

Replaces the active cluster diagram with the cluster diagram of its parent. This command is not available at the System level.

Hide/Show Labels

Hides all active class and cluster labels. Click a second time to redisplay.

Hide/Show Implementation

Hides all active relation links. Click a second time to redisplay.

Hide/Show Inheritance

Hides all active inheritance relation links. Click a second time to redisplay.

Hide/Show Client Links

Hides all active client relation links. Click a second time to redisplay.

Hide/Show Aggregation

Hides all active aggregation relation links. Click a second time to redisplay.

5.6 Tools menu

The **Tools** menu contains the following commands. For more information on the following tools, see chapter 6, “EiffelCase tools”, chapter 7, “System tool”, chapter 8, “Class tool”, chapter 9, “Relation tool” and chapter 10, “Feature tool”.

System Tool CTRL+I

Sets, adds and modifies textual documentation information for the active system, its related clusters and classes.

Chart Tool CTRL+F

Displays cluster information in the chart format defined in *Seamless Object-Oriented Software Architecture*.

Color Tool CTRL+W

Sets default and display colors for the active development objects and associated relation links.

For clusters, the color applies to both the delimiting dotted lines and the label. For relation links, the color applies to all segments of the link. For a class, the background color changes. Class names always display in black for readability.

Preference Tool CTRL+K

Sets grid, display, saving delay, and print command preferences.

Hidden Components Tool

Hides and redisplays development objects in the workspace.

Class Tool

Sets properties, features, indexing information and constraints for the active class.

Relation Tool

Sets display properties for the active relation link, its classes and clusters.

Feature Tool

Sets properties for the active feature and its signature (number and type of arguments).

5.7 Options menu

The **Options** menu contains the following commands.

Gather

Groups all active development objects together and moves them in the direction that you click: **Up**, **Left**, or **Up_Left**.

Advanced Color Tool

Detects and adds color to all active links and classes; useful for reverse engineering. For more information, see chapter 6, "EiffelCase tools" and chapter 4, "Tutorial B: reverse engineering".

Right Angles

All subsequently added relation links of the type that you click display as a ninety degree right angle: **Client**, **Inheritance**, or **Both**. Click a second time to add relation links as straight lines.

5.8 Window menu

The **Windows** menu contains the following command.

Save size

Sets and saves the active workspace size as the default in the resource file. For more information on the resource file, see appendix A, “System resources”.

5.9 Toolbar

The toolbar displays across the top of the workspace and contains the following buttons, holes, and commands. For more information on the pick-and-drop operation, see chapter 2, “Getting started”.

Open

Opens a previously-created system. In the **Choice** box, type or select a directory or click one in the **Directories** list. The path, file name and view for the active system display in the EiffelCase title bar.

Save

Saves the active system and all related resources in the active view.

Save As

Creates a new view for the active system. In the **Choice** box, type or select a directory or click one in the **Directories** list. For more information on views, see chapter 2, “Getting started”.

Undo

Cancels your most recent operation.

Redo

Cancels your most recent **Undo** operation.

Cluster Hole

Displays the contents of the cluster you drop on the hole using the pick-and-drop operation.

Class Hole

Retargets and displays the **Class** tool for the class you drop on the hole using the pick-and-drop operation. For more information, see chapter 8, “Class tool”.

Relation Hole

Retargets and displays the **Relation** tool for the relation link you drop on the hole using the pick-and-drop operation. For more information, see chapter 9, “Relation tool”.

Deletion Hole

Deletes the development object you drop on the hole using the pick-and-drop operation.

Iconify/deconify cluster

Iconifies the clusters you drop on the hole using the pick-and-drop operation. Drop the iconified cluster on the hole to deconify it.

Hidden Components Tool

Hides the development object you drop on the hole using the pick-and-drop operation. Click to display the **Hidden Components** tool, where you can redisplay hidden objects.

For more information on the **Hidden Components** tool, see chapter 6, “EiffelCase tools”.

Compress/uncompress link

Compresses all intercluster relation links in the cluster you drop on the hole using the pick-and-drop operation. Drop the compressed link on the hole to redisplay all relation links.

Retarget to Parent

Replaces the active cluster diagram with the cluster diagram of its parent. This command is not available at the System level.

Cluster

Adds a cluster to the workspace using the pick-and-drop operation. For more information on adding clusters, see chapter 2, “Getting started”.

Class

Adds a class to the workspace using the pick-and-drop operation. For more information on adding classes, see chapter 2, “Getting started”.

Client Link

Adds a client relation link between the active cluster or class and the one you select using the pick-and-drop operation. Client relation links display in red. For more information on adding client links, see chapter 2, "Getting started".

You can only have one inheritance, one client, and one aggregation relation link between clusters.

Aggregation Link

Adds an aggregation relation link between the active cluster or class and the one you select using the pick-and-drop operation. Aggregation relation links display in blue. For more information on adding aggregation links, see chapter 2, "Getting started".

Inheritance Link

Adds an inheritance relation link between the active cluster or class and the one you select using the pick-and-drop operation. Inheritance relation links display i black. For more information on adding inheritance links, see chapter 2, "Getting started".

Hide/Show client links

Hides all active client relation links. Click a second time to redisplay.

Hide/Show aggregation links

Hides all active aggregation links. Click a second time to redisplay.

Hide/Show inheritance links

Hides all active inheritance relation links. Click a second time to redisplay.

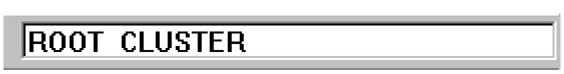
Hide/Show link labels

Hides all active client and aggregation relation link labels. Click a second time to redisplay.

Hide/Show all links

Hides all active relation links. Click a second time to redisplay.

Target Name Tool



Displays the name of the active system (default) or cluster, which you can rename. You can also use this tool to display the contents of a cluster.

For more information on the **Target Name** tool, see chapter 6, "EiffelCase tools".

5.10 Class Menu

The **Class Menu** provides a way to quickly modify a class.

To display the **Class Menu**:

- Use the pick-and-drop operation to drop a class on itself.

The **Class Menu** contains the following commands.

Client link to self

Adds a client relation link between the active class and itself.

Rename

Renames the active class using the **Editor** tool.

Renaming the active class

To rename the active class:

- Type a name, and then click the **Check** button.

For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Associated Tool

Retargets and displays the **Class** tool for the active class. For more information, see chapter 8, “Class tool”.

Hide

Hides the active class. You must use the **Hidden Components** tool redisplay the hidden class.

For more information on the **Hidden Components** tool, see chapter 6, “EiffelCase tools”.

Remove

Deletes the active class.

6

EiffelCase tools

EiffelCase tools are object-oriented and provide interaction with the development objects. For more information on development objects, see chapter 2, “Getting started”.

The following is a list of the tools included in EiffelCase:

- **Advanced Color** tool — Adds color to the classes and relation links in the active cluster. For more information, see chapter 13, “Advanced Color tool”.
- **Chart** tool — Displays cluster information in the chart format defined in *Seamless Object-Oriented Software Architecture*. For more information, see “Chart tool”, page 65, later in this chapter, and *Seamless Object-Oriented Software Architecture*.
- **Class** tool — Sets properties, features, indexing information and constraints for the active class. For more information, see chapter 8, “Class tool”.
- **Cluster** tool — Graphically defines the architecture of the active system, its classes, and clusters, as well as the relationship between all development objects. When you start an EiffelCase session, it displays by default. All functionality described in this chapter applies to the **Cluster** tool.
- **Color** tool — Sets default and display colors for the active development objects. For more information, see “Color tool”, page 66, later in this chapter.
- **Documentation** tool — Specifies report types and any printing commands. For more information, see chapter 12, “Documentation tool”.
- **Editor** tool — Sets name, comment and documentation information for the active development object. For more information, see chapter 11, “Editor tool”.
- **Feature** tool — Sets properties for the active feature and its signature (number and type of arguments). For more information, see chapter 10, “Feature tool”.
- **Feature Clause** tool — Sets accessibility permission and comments for the active feature. For more information, see “Feature Clause tool”, page 68, later in this chapter.

- **Hidden Components** tool — Hides and redisplays development objects in the workspace. For more information, see "Hidden Components tool", page 69, later in this chapter.
- **History tool** — Lists the actions performed during the active session, and allows you to corrects any mistakes. For more information, see "History tool", page 70, later in this chapter.
- **Link Generation** tool — Sets feature types and adds labels to client and aggregation relation links. For more information, see "Link Generation tool", page 71, later in this chapter.
- **Merging tool -** Displays and then reconciles the differences between the active EiffelCase system and the Eiffel text to be generated. Differences detected by the **Merging** tool include inheritance properties, individual features, and invariant clauses. For more information, see chapter 14, "Merging tool".
- **Preference tool** — Sets grid, display, saving delay, and print command preferences. For more information, see "Preference tool", page 73, later in this chapter.
- **Relation tool** — Sets display properties for the active relation link, its classes and clusters. For more information on relation and relation links, see chapter 9, "Relation tool", and chapter 2, "Getting started".
- **System tool** — Sets, adds and modifies textual documentation information for the active system, its related clusters and classes. Only one **System** tool can be open per EiffelCase session. For more information, see chapter 7, "System tool".
- **Target Name tool** — Displays the name of the active system (default) or cluster, which you can rename. You can also use this tool to view the contents of a cluster. For more information, see "Target Name tool", page 74, later in this chapter.
- **View tool** — Defines and changes views available to the active system. For more information on views, see "View tool", page 75, later in this chapter, and chapter 2, "Getting started".

An active tool has two modes — **void**, or **targeted** to a specific development object called the **target** of the tool. During an EiffelCase session, you can open as many **Class**, **Cluster**, **Feature** or **Relation** tools as you want. Since you can only open one system per session, you are limited to one **System** tool.

A tool and its target are always the same type: for example, a **Class** tool targets a class; a **Feature** tool, a feature. You can change information about a development object, as well as to perform various operations on the object in a tool.

To modify a development object using a tool, you can either create a new tool or retarget an existing tool.

6.1 Creating a tool

To create a new tool, do one of the following:

- Use the pick-and-drop operation to drop a development object on the corresponding tool hole.
- Click a hole.
- Hold down CTRL, and then right-click a development object.
- Create a new tool, and then retarget that tool.
- Use the pick-and-drop operation to drop a development object anywhere in the tool window.

For more information on the pick-and-drop operation, see chapter 2, "Getting started".

6.2 Retargeting a tool

To retarget an existing tool, do one of the following:

- Use the pick-and-drop operation to drop a development object on the corresponding tool hole.
- Type the name of the target in the **Target Name** tool in the **Cluster**, **Class**, **Relation,** or **Feature** tool.
- Hold down CTRL, and then right-click a development object.
- Retarget a previously-created tool.

For more information on the **Class**, **Relation** and **Feature** tools, see chapter 8, "Class tool", chapter 9, "Relation tool" and chapter 10, "Feature tool" respectively. For more information on the **Target Name** tool, see "Target Name tool", page 74, later in this chapter.

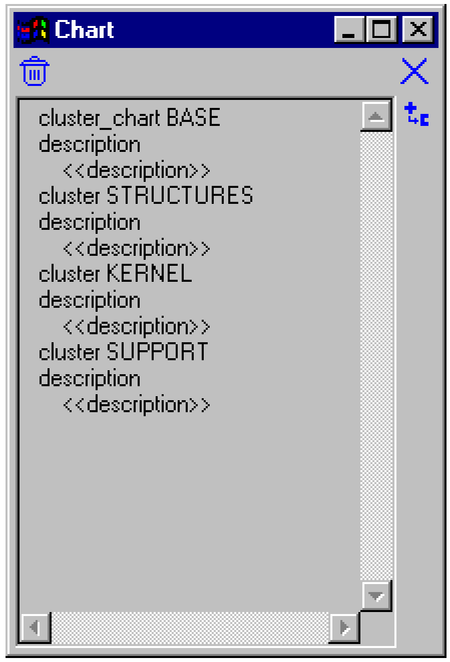
6.3 Chart tool

The **Chart** tool displays textual information for the active cluster in the class chart format defined in *Seamless Object-Oriented Software Architecture*. For more information on the class chart format, see chapter 7, "System tool".

To display the **Chart** tool, do one of the following:

- On the **Tools** menu, click **Chart Tool**.
- Press CTRL+F.

- Point to an iconified cluster.



The **Chart** tool contains the following hole and buttons.

Deletion Hole

Deletes the information you drop on the hole using the pick-and-drop operation.

Close

Closes the **Chart** tool.

Add index

Adds an index clause entitled **index"nothing_yet"** to the active cluster chart. You can then change the default information for the index clause using the **Editor** tool. For more information, see chapter 11, "Editor tool".

Changing default information for an index clause

To change default information for an index clause:

1 Hold down SHIFT, and right-click to select the index clause.

2 In the **Editor** tool window, type text, and then click the **Check** button.

6.4 Color tool

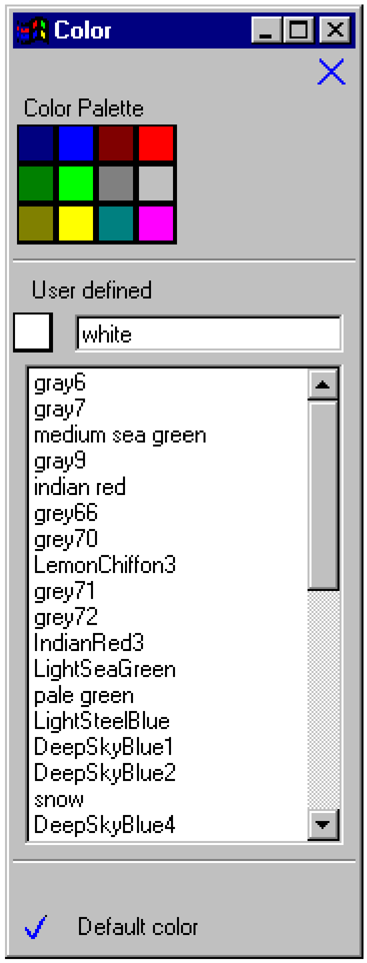
The **Color** tool sets default and display colors for the active development objects.

For clusters, the color applies to both its delimiting dotted lines and the label. For relation links, the color applies to all segments of the link. For a class, the background color changes. Class names always display in black for readability.

Color properties set in a view do not affect other views.

To display the **Color** tool, do one of the following:

- On the **Tools** menu, click **Color Tool**.
- Press CTRL+W.



The **Color** tool contains the following buttons and options.

Close

Closes the **Color** tool.

Color Palette

Displays a series of predefined colors.

Adding color to a development object

To add color to a development object:

- Use the pick-and-drop operation to drop one of the **Color Palette** buttons on the development object.

User defined

Displays a list of custom colors.

Adding a user-defined color to a development object

To add a user-defined color to a development object:

1. In the **User defined** list, click a color.
2. Use the pick-and-drop operation to drop the custom color button on the development object.

Default color

Sets the color selected in the **Color Palette** or **User defined** list as the default for the active development object.

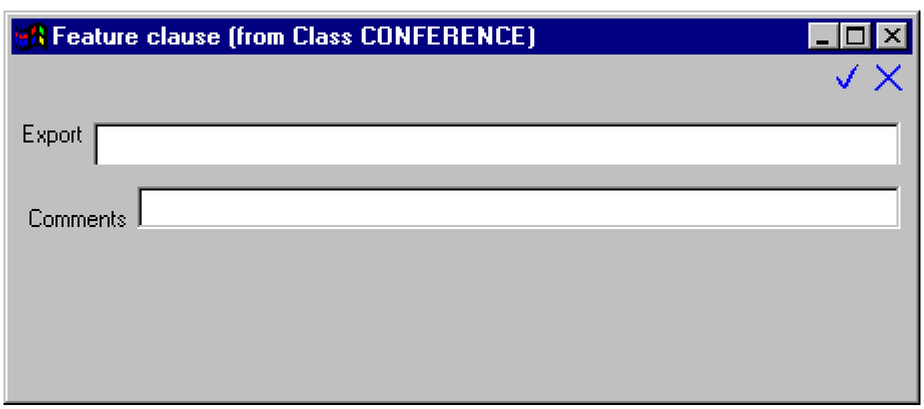
6.5 Feature Clause tool

The **Feature Clause** tool sets accessibility permission and comments for the active feature.

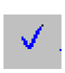
To display the **Feature Clause** tool:

- Use the pick-and-drop operation to drop the feature you want to change on the **Feature Clause** hole ADD ICON on the toolbar in the **Class** tool.

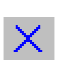
For more information on the **Class** tool, see chapter 8, “Class tool”.



The **Feature Clause** tool contains the following buttons and options.

OK 

Closes the **Feature Clause** tool, and saves all changes.

Close 

Closes the **Feature Clause** tool.

Export

Type the name of the classes that have permission to access the feature. By default, a feature is accessible to all classes in the active system.

Comments

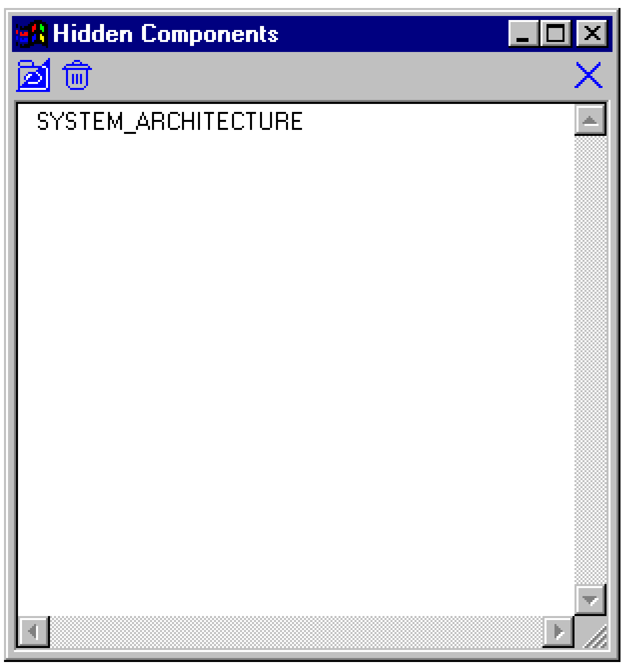
Adds descriptive information to the active feature clause.

6.6 Hidden Components tool

The **Hidden Components** tool hides and redisplays development objects in the workspace. It also lists all hidden development objects. Hidden objects are not deleted from the system, they just do not display.

To display the **Hidden Components** tool:

- On the **Tools** menu, click **Hidden Components Tool**.
- Use the pick-and-drop operation to drop the development object you want to hide on the **Hidden Components Tool** hole on the toolbar.



For more information on the **Hidden Components Tool** hole, see chapter 5, “Menu bar and toolbar”.

The **Hidden Components** tool contains the following holes and button.

Hidden Components Tool Hole 

Redisplays the development object you drop on the hole using the pick-and-drop operation.

Deletion Hole

Deletes the development object you drop on the hole using the pick-and-drop operation.

Close

Closes the **Hidden Components** tool.

Redisplaying a hidden development object

To redisplay a hidden development object:

1 On the **Tools** menu, click **Hidden Components Tool**.

2 Use the pick-and-drop operation to drop the object on the **Hidden Components Tool** hole, and then click the **Close** button.

6.7 History tool

The **History** tool lists all actions performed during the active session. The most recent action displays and the system does not save this list.

To display the **History** tool, do one of the following:

- On the **Edit** menu, click **History Tool**.
- Press CTRL+H.

The **History** tool contains the following buttons:

Undo 

Cancels the selected actions. Most EiffelCase actions are undoable, except for commands in the **File** menu.

When you undo one or more actions, you can redo the same actions, as long as you have not carried out any other command (except **Redo**). You can only redo actions that were undone once.

Using the Undo button

To cancel the effect of one or more actions, do one of the following:

- Highlight to select the actions to undo, and then click **Undo**.
- Click the most recent action you do *not* want to undo. This cancels all commands after the selected action.

The EiffelCase tools update to reflect the cancellations.

Redo 

Carries out actions undone using the **Undo** button.

Using the Redo button

To carry out actions undone with the **Undo** button, do one of the following:

- Highlight to select the actions to redo, and then click **Redo**.
- Click the most recent action you want to redo. This carries out all actions undone upto and including the selected action.

The EiffelCase tools update to reflect the re-executions.

Close 

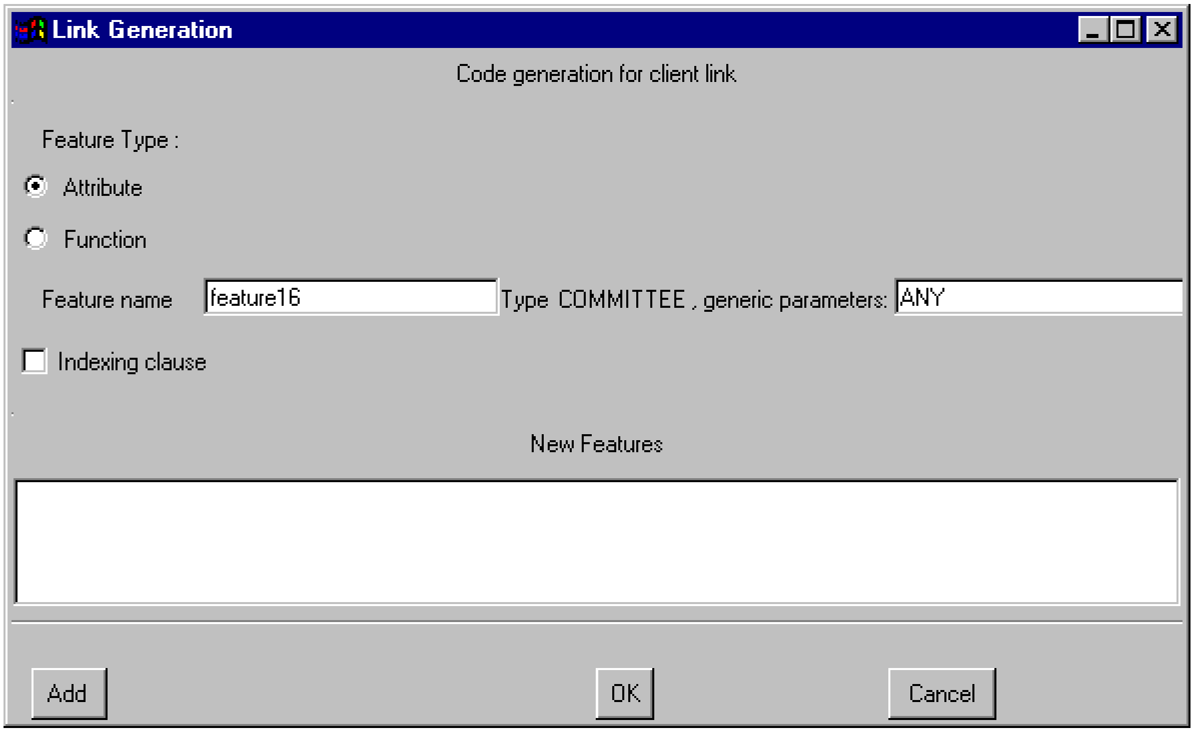
Closes the **History** tool.

6.8 Link Generation tool

The **Link Generation** tool sets feature types and adds labels to client and aggregation relation links. It appears when you add client or aggregation relation links between development objects.

Because a feature is either an attribute of a class or performs a routine, the **Attribute** and **Function** options are mutually exclusive: a relation link is either one or the other, never both. For more information on features, see chapter 10, “Feature tool”.

Options may or may not display, depending on if the active class contains generic parameters. The number of **generic parameter** text boxes that display directly correspond to the number of generic parameters in the active class. The maximum number of generic parameters that you can set for a class is three.



The **Link Generation** tool contains the following options and buttons. Select to include an option.

Feature type

Click to select the type of feature. Options may or may not display, depending on the parameters associated with the feature.

Attribute

Sets the name of the attribute and any generic parameters (maximum 3). Type the name of the attribute in the **Feature name** box, the name of the generic parameters in the **generic parameters** box, and then click **Add**. The attribute and any generic parameters display in the **New Features** list.

Function

Sets the name of the function and any generic parameters (maximum 3). Type the name of the function in the **Feature name** box, the name of the generic parameters in the **generic parameters** box, and then click **Add**. The function and any generic parameters display in the **New Features** list.

Indexing clause

Creates entries in the indexing clause in the **Class** tool window for all items in the **New Features** list.

For more information on the **Class** tool, see chapter 8, “Class tool”.

New Features

Lists all new features (attributes or functions) for the active class, including any generic parameters.

Add

Adds the attribute or function to the **New Features** list.

OK

Creates the features in the **New Features** list.

Cancel

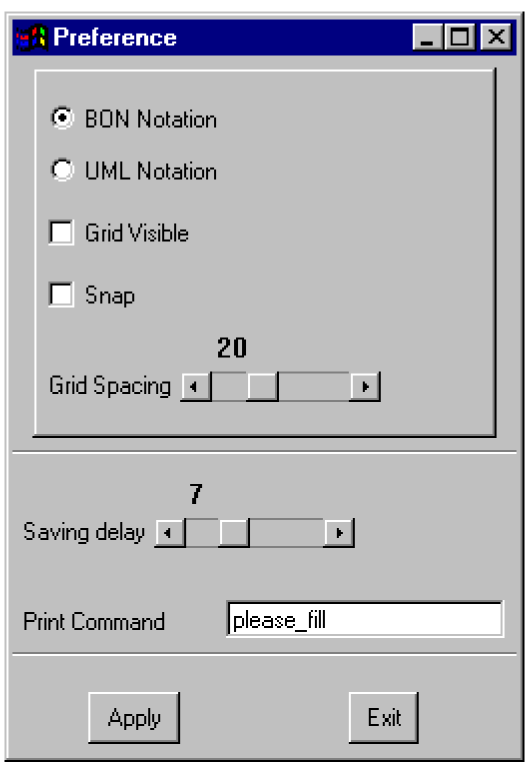
Closes the **Link Generation** tool.

6.9 Preference tool

The **Preference** tool sets grid, display, saving delay, and print command preferences.

To display the **Preference** tool, do one of the following:

- On the **Tools** menu, click **Preference Tool**.
- Press CTRL+K.



The **Preference** tool contains the following check boxes, options and command buttons.

BON Notation

Displays all active development objects using the shapes defined by the BON style rules.

UML Notation

Displays all active development objects using the shapes defined by UML style rules.

Grid Visible

Displays the visible grid, which displays as horizontal and vertical lines. It does not print. Set units of display in **Grid Spacing**.

Snap

Applies the invisible snap grid to the workspace. As you add objects to the workspace, they attach to the closest point on the invisible grid. This ensures regular spacing between objects.

Grid Spacing

Sets the line spacing value (in pixels) for the visible grid. Move the slider to the right to increase the spacing.

Saving delay

Sets the time interval to save the active system and all related resources. Move the slider to the right to increase the saving interval.

Print Command

Type the command for printing from the DOS console (Windows) or for printing from a shell (LINUX, UNIX).

Apply

Applies the settings in the **Preference** tool to the workspace.

Exit

Closes the **Preference** tool.

6.10 Target Name tool

The **Target Name** tool displays the name of the active system (default), cluster or class, which you can rename. You can also use it to view the contents of a cluster.

The **Target Name** tool displays in the **Cluster**, **Class** (**Class name**), **Relation** (**Class/cluster name to Class/cluster name**), and **Feature** (**Feature name from Class name**) tools. For more information on the **Class**, **Relation**, and **Feature** tools, see chapter 8, “Class tool”, chapter 9, “Relation tool”, and chapter 10, “Feature tool”.

Renaming the active system, cluster, or class

To rename the active system, cluster, or class:

1 In the **Target Name Tool** box, type the new name, and then press ENTER.

2 Click **— rename —**.

3 On the **File** menu, click **Save**.

Displaying the contents of a cluster or class

To display the contents of a cluster or class:

- In the **Target Name Tool** box, type the cluster or class name, and then press ENTER.

The contents of the cluster or class appear in the workspace.

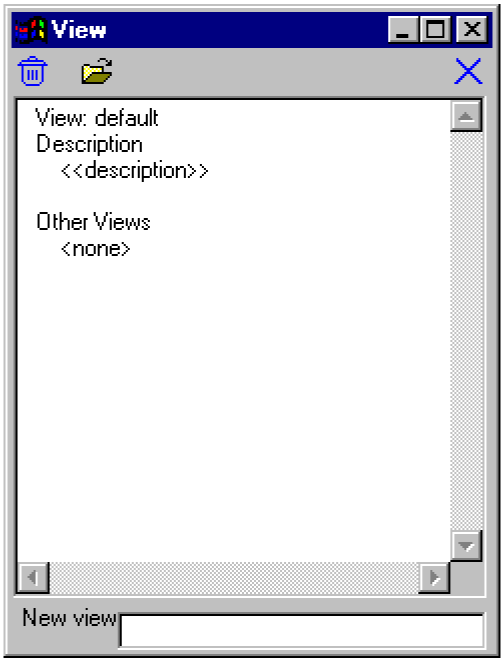
6.11 View tool

The **View** tool defines and changes the views available to the active system. For more information on views, see chapter 2, “Getting started”. The name of the active view displays in the EiffelCase title bar.

To display the **View** tool, do one of the following:

- On the **View** menu, click **View Tool**.

- Press CTRL+G.



The **View** tool contains the following hole, command, button and option.

Deletion Hole 

Deletes the view you drop on the hole using the pick-and-drop operation. For more information on the pick-and-drop operation, see chapter 2, "Getting started".

Change View 

Switches between all defined views in the active system.

Changing views

To switch to a different view:

- In the **View selector** dialog box, click a view.

Close 

Closes the **View** tool.

New View

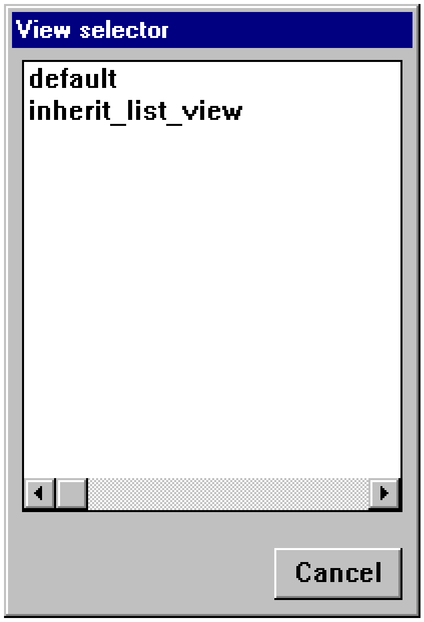
Creates a new view for the active system. Type a name for the new view, press ENTER, and then click the **Close** button.

You must switch to the new view to work in that view, and to save any changes made.

Switching views

To switch to a different view:

1 On the **View** menu, click **Change View**.



2 Click the name of the view that you want to display.

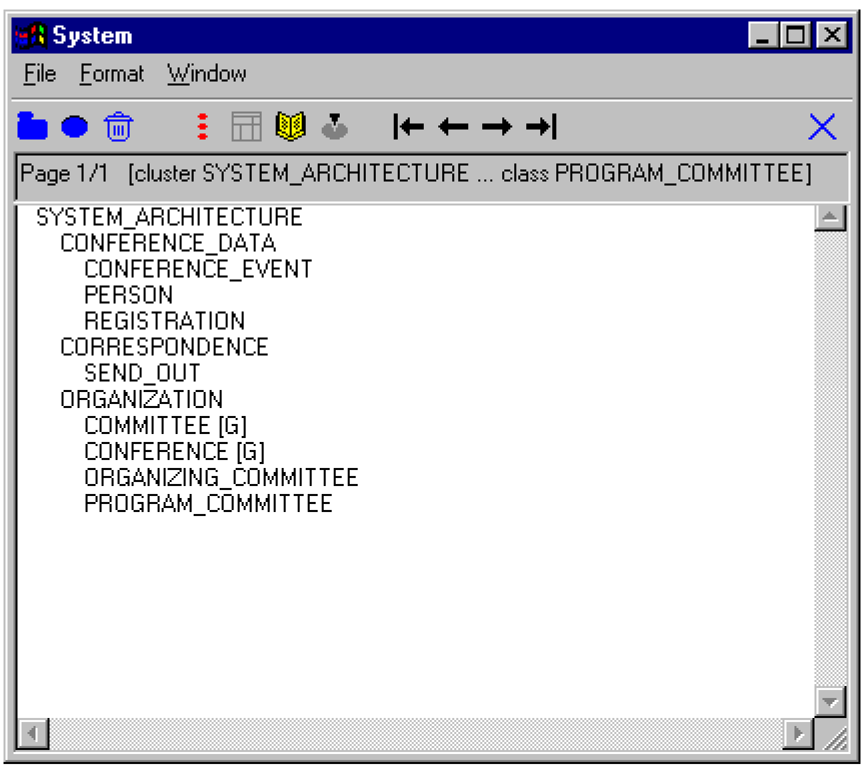
7

System tool

The **System** tool sets, adds and modifies textual documentation information for the active system, its related clusters and classes. Only one **System** tool can be open per EiffelCase session.

To display the **System** tool, do one of the following:

- On the **Tools** menu, click **System Tool**.
- Press CTRL+L.



The **System** tool contains the following menus, commands, toolbar, holes and buttons.

7.1 File menu

The **File** menu contains the following commands.

Print

Prints the active system to the destination selected.

In the **Documentation generation** dialog box, do one of the following, and then click OK:

- Select the **File** check box, type the path, and then click an output filter in the **Select a filter** list. You can also click **Browse**, and then in the **Choice** box, type the path or click a folder in the **Directories** list.
- Select the **Printer** check box, click **OK**, and then set printing defaults.

Exit

Closes the **System** tool.

7.2 Format menu

The **Format** menu contains the following commands.

Show Components

Lists the components of the active system.

Show System Chart

Lists all active clusters, including any descriptive information.

Show Class Dictionary

Lists all active classes in the system alphabetically, including any descriptive information. For more information on adding descriptive information, see "Chart", page 81, later in this chapter.

Show Modified Classes

Lists all classes in the system that were added or changed during the active session.

7.3 Window menu

The **Window** menu contains the following command.

Save size

Sets and saves the active window size as the default in the resource file. For more information on the resource file, see appendix A, "System resources".

7.4 System toolbar

The **System** toolbar displays across the top of the tool window and contains the following buttons and commands. For more information on the pick-and-drop operation, see chapter 2, “Getting started”.

Cluster

Displays the contents of the cluster you drop on the hole using the pick-and-drop operation. You can also click the hole to display a new **Cluster** tool.

Class

Retargets and displays the **Class** tool for the class you drop on the hole using the pick-and-drop operation. You can also click the hole to display a new **Class** tool. For more information, see chapter 8, “Class tool”.

Deletion Hole

Deletes the development object you drop on the hole using the pick-and-drop operation.

System classes/clusters

Displays the components of the active system in textual format.

An indentation represents class and cluster inclusion. All subclusters and classes of a cluster display indented below the parent class. All displaying objects are available for use with the pick-and-drop operation.

Chart

Displays a textual representation of the active class information.

You can use the **Editor** tool to enter descriptive information for any field displaying between double angle brackets; for example **<<explanation>>**. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Show class dictionary

Lists all active classes in the system alphabetically, including any descriptive information.

Modified class

Lists all classes in the system that were added or changed during the active session.

Go to first page

Displays the first page.

Go to previous page ←

Displays the previous page (if applicable).

Go to next page →

Displays the next page (if applicable).

Go to last page →|

Displays the last page (if applicable).

Close ✕

Closes the **System** tool.

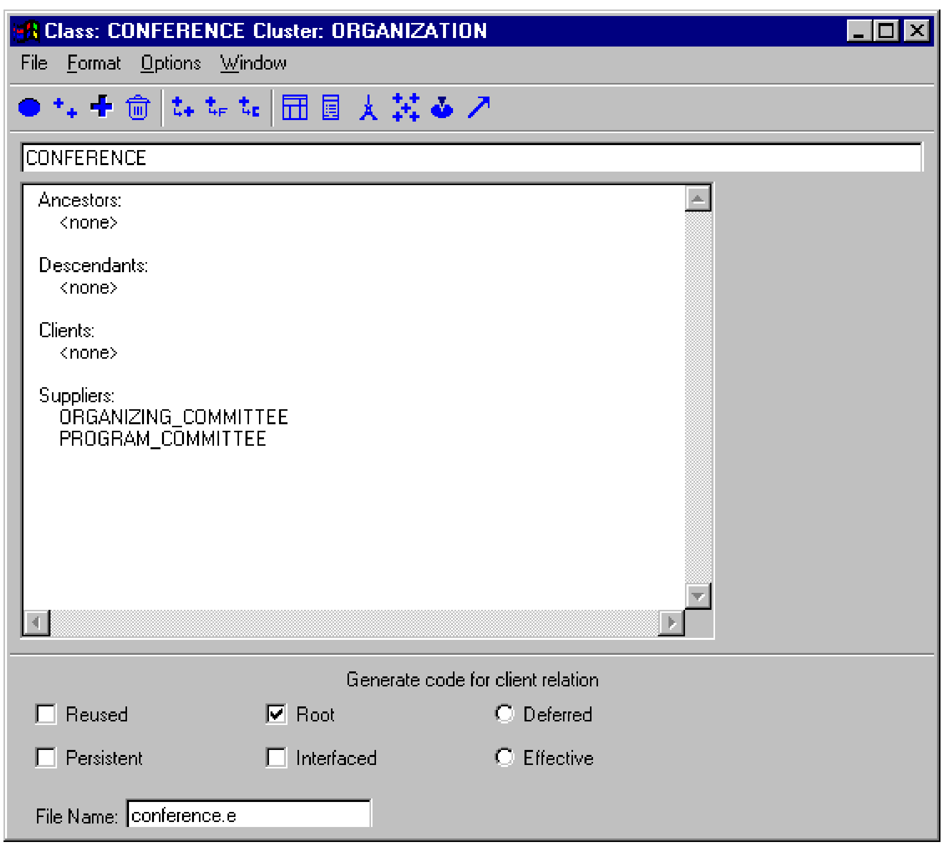
8

Class tool

The **Class** tool sets properties, features, indexing information and constraints for the active class.

To display the **Class** tool, do one of the following:

- Hold down CTRL, and then right-click a class.
- Use the pick-and-drop operation to drop a class anywhere in the workspace.
- Use the pick-and-drop operation to drop a class on itself, and then on the **Class Menu**, click **Associated Tool**.
- On the **Tools** menu, click **Class Tool**.
- In the **System** tool, click the **Class** hole.



For more information on the pick-and-drop operation, see chapter 2, “Getting started”. For more information on the **Class Menu**, see chapter 5, “Menu bar and toolbar”. For more information on the **System** tool, see chapter 7, “System tool”.

The **Class** tool appears with the relational format (default) displaying in the tool window and contains the following menus, commands, toolbar, holes and buttons.

8.1 File menu

The **File** menu contains the following commands and submenu.

Print

Prints the format that displays in the tool window to the destination selected. Printing to file creates a text file (.e).

Formats available for printing include **Chart**, **Specifications**, and **Eiffel Code**. For more information on these formats, see “Format menu”, page 84, and “Class toolbar”, page 86, later in this chapter.

In the **Documentation generation** dialog box, do one of the following, and then click OK:

- Select the **File** check box, type the path and file name, and then click an output filter in the **Select a filter** list. You can also click **Browse**, and then in the **Choice** box, type the path and file name or click a folder in the **Directories** list.
- Select the **Printer** check box, click **OK**, and then set printing defaults.

Import indexing

Imports the text file, which contains the indexing structure, that you click in the submenu:

Default

Applies the EiffelCase default indexing structure.

Browse

Imports the text file that you select. In the **File name** box, type the path and file name or click a folder in the **Look in** list.

Exit

Closes the **Class** tool.

8.2 Format menu

The **Format** menu contains the following commands.

Chart

Displays a textual representation of the active class information.

You can use the **Editor** tool to enter descriptive information for any field displaying between double angle brackets; for example **<<explanation>>**. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Specification

Displays the *specifications* for the active class — the part of the class visible to other classes.

Eiffel Code

Displays the active class in syntactically-correct Eiffel form. This provides a preview of what happens when you click **Generate Eiffel (this system)** or **Generate Eiffel (this cluster)** on the **File** menu.

Features

Lists all features in the active class.

Relations

Lists all relation links for the active class.

Modified features

Lists all features in the active class that have been added or changed during the active session.

8.3 Options menu

The **Options** menu contains the following command.

Repercussion on graph

Displays changes made to the text in the **Class** tool window graphically in the system diagram.

8.4 Window menu

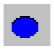
The **Window** menu contains the following command.

Save size

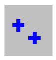
Sets and saves the active tool window size as the default in the resource file. For more information on the resource file, see appendix A, “System resources”.

8.5 Class toolbar

The **Class** toolbar displays across the top of the tool window and contains the following holes, buttons, and commands.

Class Hole 

Retargets and displays the **Class** tool for the class you drop on the hole using the pick-and-drop operation.

Feature Clause Hole 

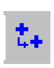
Retargets and displays the **Feature Clause** tool for the feature you drop on the hole using the pick-and-drop operation. You can also click the hole to display a new **Feature Clause** tool. For more information on the **Feature Clause** tool, see chapter 6, “EiffelCase tools”.

Feature Hole 

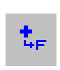
Retargets and displays the **Feature** tool for the feature you drop on the hole using the pick-and-drop operation. You can also click the hole to display a new **Feature** tool. For more information on the **Feature** tool, see chapter 10, “Feature tool”.

Deletion Hole 

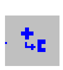
Deletes the development object you drop on the hole using the pick-and-drop operation.

Cloning Hole 

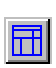
Adds a copy of the feature that you drop on the hole using the pick-and-drop operation to the **Class** tool window. This is useful when multiple **Class** tools are open.

Add associated function 

Adds a function to the feature that you drop on the hole using the pick-and-drop operation.

Add associated command 

Adds a command to the feature that you drop on the hole using the pick-and-drop operation.

Chart 

Displays a textual representation of the active class information.

You can use the **Editor** tool to enter descriptive information for any field displaying between double angle brackets; for example <<**explanation**>>. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Specifications

Displays the specifications for the active class - the part of the class visible to other classes.

Show Eiffel code

Displays the active class in syntactically-correct Eiffel form. This provides a preview of what happens when you click **Generate Eiffel (this system)** or **Generate Eiffel (this cluster)** on the **File** menu.

Features

Lists all features in the active class.

Modified features

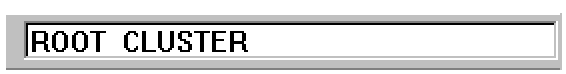
Lists all features in the active class that were added or changed during the active session.

Relations

Lists all relation links for the active class.

8.6 Class name (Target Name tool)

Displays the name of the active class, which you can rename. You can also switch to view a different class.



For more information on the **Target Name** tool, see chapter 6, “EiffelCase tools”.

8.7 Command buttons

Buttons may or may not display, depending on the format you click.

Formats available include **Chart**, **Specifications**, **Eiffel Code**, **Features**, **Modified features**, **Relations**, and **Browsing Mode**. For more information on these formats, see “Format menu”, page 84, and “Class toolbar”, page 86, earlier in this chapter.

Add index

Adds an index entry to the indexing clause for the active class. By default, the entry is named **index:”nothing_yet”**.

You can use the **Editor** tool to change the name and add the body of the index entry. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Changing an index entry

To change an index entry:

1 Hold SHIFT, and then right-click **index:”nothing_yet”.**

2 In the **Editor** tool, type a name in the left box, type the body of the entry in the right box, and then click the **Check** button.

Add inheritance

Adds a class name entry to the inheritance clause for the active class. By default, the entry is named **CLASS_#**, where # is an automatically generated number.

You can use the **Editor** tool to rename the class. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Add creation

Adds the name of a routine to the creation routine list for the active class. By default, the entry is named **feature#**, where **#** is an automatically generated number.

You can use the **Editor** tool to rename the routine. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Add clause

Adds a feature clause to the active class.

In the **Feature Clause** tool, set accessibility permissions, add a name, any comments for the feature, and then click the **Check** button. For more information on the **Feature Clause** tool, see chapter 11, “Editor tool”.

Add attribute

Adds an attribute to the active class. By default, the attribute is named **feature#:ANY**, where **#** is an automatically generated number and the type is **ANY**.

You can use the **Feature** tool to set properties for the attribute. For more information on the **Feature** tool, see chapter 10, “Feature tool”.

Setting attribute properties

To set properties for an attribute:

1 Use the pick-and-drop operation to drop **feature#:ANY** anywhere in the tool window.

2 In the **Feature** tool, set properties, and then click the **Close** button.

Add query

Adds a query to the active class. A *query* is a routine that returns a result. By default, the query has the following format:

feature#: ANY is
— comment
do

end

where **#** is an automatically generated number, **ANY** is the result type, and **do** is the body of the query.

You can use the **Editor** tool to rename the query, change the result type, add comments, and add Eiffel code to the query. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

For more information on coding in Eiffel, see *Eiffel: The Language*.

Add command

Adds a command to the active class. A *command* is a routine that does not return a result. By default, the command has the following format:

feature# is
— comment
do

end

where **#** is an automatically generated number, and **do** is the body of the command.

You can use the **Editor** tool to rename the command, add comments, and add Eiffel code to the command. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

For more information on coding in Eiffel, see *Eiffel: The Language*.

Add generic

Adds a generic parameter to the active class. By default, the first generic parameter is named **G**. Subsequently added parameters are named alphabetically — the second generic parameter is named **H**, the third **I**, and so on.

You can use the **Editor** tool to rename generic parameters. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Add invariant

Adds an invariant to the constraints list for the active class. By default, the invariant is named **tag**.

You can use the **Editor** tool to define the invariant. For more information on the **Editor** tool, see 11, “Editor tool”.

8.8 Properties

Sets the properties of the class. Select to include a property.

Reused

Class belongs to a library of reusable components. Reused classes display with the class name underlined.

Persistent

Instances of the class exist between sessions of the system. Persistent classes display with a solid black dot above the class name.

Root

Sets the class as the root class. A system can have only one root class. Root classes display with two lines around the border.

Interfaced

Class interfaces with development objects outside the system proper. Interfaced classes display with a solid black triangle above the class name.

Deferred

Defers implementation of the class — opposite of **Effective**. Deferred classes display with an asterisk (*) above the class name.

Effective

Completely implements the class — opposite of **Deferred**. Effective classes display with a plus sign (+) above the class name.

8.9 File name

Displays the name of the text file (.e) that contains all information about the active class.

9

Relation tool

Relation links are full-fledged development objects and exist between two classes, two clusters, or a class and a cluster.

There are three type of relation links:

- **client** — a class is a client of another if it uses a feature of the other (supplier) class
- **inheritance** — a class is an heir of another if it incorporates the features of the other class in addition to its own
- **aggregation** — every object of a certain type is a combination (an aggregate) of zero or more objects of a certain type; For example, a “car” can be defined as an aggregation of “engine”, “body” and so forth.

In order to alleviate confusion in the future, it is important to understand the difference between a relation and a relation link. A *relation* is a general concept that models an association between objects, and is independent of the objects to which it applies. For example, the interaction between a parent and a child is a parent-child relation.

On the other hand, a *relation link* defines the connection that exists between two or more development objects — it is an instance of the concept. Therefore, it is correct to say that a relation link connects two development objects.

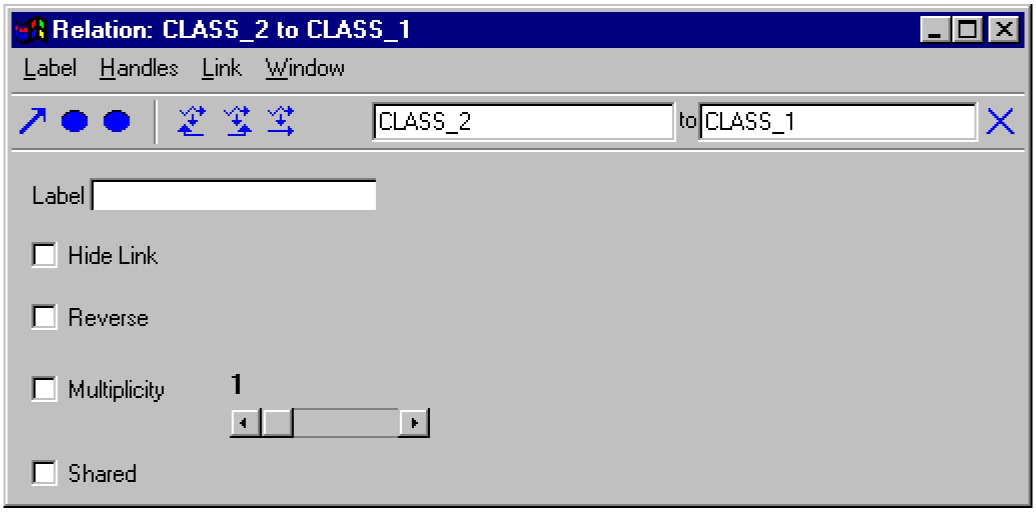
For example, there is a relation link of the inheritance type from class **SQUARE** to class **RECTANGLE** (in a graphical library). For more information on adding relation links, see chapter 2, “Getting started”.

The **Relation** tool sets display properties for the active relation link, its classes and clusters.

To display the **Relation** tool, do one of the following:

- Use the pick-and-drop operation to drop a relation link on the **Relation** hole on the EiffelCase toolbar.
- Use the pick-and-drop operation to drop a relation link anywhere in the workspace.

- Hold down CTRL, and then right-click a relation link.
- On the **Tools** menu, click **Relation Tool**.



For more information on the pick-and-drop operation, see chapter 2, "Getting started".

The **Relation** tool contains the following menus, commands, toolbar, holes and buttons. Commands may or may not display, depending on the type of link selected.

9.1 Label menu

The **Label** menu contains the following commands. Only client and aggregation relation links can have labels.

For more information on adding labels, see "Adding a label to a relation link", page 95, later in this chapter.

Change side

Moves the label to the other side of the relation link. Labels on the left side, move to the right, and vice versa.

Hide/Show

Hides all relation link labels. Click a second time to redisplay.

9.2 Handles menu

The **Handles** menu contains the following commands.

For more information on handles, see "Resizing relation links", page 95, later in this chapter.

Place left-bottom corner

Relation link displays as a ninety degree left and down arrow.

Place right-bottom corner

Relation link displays as a ninety degree down and left arrow.

Remove

Restores the relation link to its default display.

9.3 Link menu

The **Link** menu contains the following commands.

Hide/Show Link

Hides all relation links. Click a second time to redisplay.

Remove reverse link

Separates the active reverse link into two client or aggregation links.

9.4 Window menu

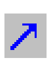
The **Window** menu contains the following command.

Save Size

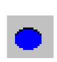
Sets and saves the active window size as the default in the resource file. For more information on the resource file, see appendix A, “System resources”.

9.5 Relation toolbar

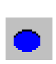
The **Relation** toolbar displays across the top of the tool window and contains the following holes, buttons and commands.

Relation Hole 

Displays the **Relation** tool for the relation link you drop on the hole using the pick-and-drop operation.

Class at the beginning of link 

Displays the **Class** tool for the source class you drop on the hole using the pick-and-drop operation.

Class at the end of link 

Displays the **Class** tool for the target class you drop on the hole using the pick-and-drop operation.

Pull Handle Left 

Relation link displays as a ninety degree left and down arrow.

Pull Handle Right 

Relation link displays as a ninety degree down and left arrow.

Remove Handles 

Restores the relation link in its default display.

Class/cluster name to Class/cluster name (Target Name tool)

Displays the name of the source cluster or class (left text box) and the source cluster or class (right text box) for the active relation link. For more information on the **Target Name** tool, see chapter 6, "EiffelCase tools".

Close

Closes the **Relation** tool.

9.6 Label

Type the name of the label, and then press ENTER. By default, the label displays to the right of the relation link.

9.7 Properties

Sets the properties of the relation link. Select to include a property.

Hide link

Hides the active relation link.

Reverse

Groups client or aggregation relation links together. Once grouped, the relation link can either be moved or deleted.

When you edit a relation link that has **Reverse** selected, you actually edit the opposing link. For example, if you are editing a relation link from CLASS_1 to CLASS_2 and you select **Reverse**, you switch editing the relation link from CLASS_2 to CLASS_1.

Multiplicity

Sets the number of development objects that participate in each instance of the given relationship. Move the slider to the right to increase the value.

A diamond with a number appears in the middle of the link, to denote multiplicity. This value corresponds to the number of development objects that participate in the relationship.

Shared

A circle appears in the middle of the link, to denote that it is a shared link (BON).

9.8 Adding a label to a relation link

To add a label to a relation link:

1. Use the pick-and-drop operation to drop the relation link on the **Relation** hole on the toolbar.
2. In the **Label** box, type a name for the label, and then press ENTER.

9.9 Moving labels

Use the drag-and-drop operation to move a label parallel to its corresponding relation link. You can also use this method to move a label from the right side of the relation link to the left, and vice versa.

9.10 Resizing relation links

BON rules require all relation links to be either vertical or horizontal. For display purposes, this may not provide optimum readability — particularly in complicated diagrams. You can add a handle to resize a relation link and increase legibility.

The added angle or bend in the link is called a *handle*. A relation link can have as many handles as you want. You can use the drag-and-drop operation to move an existing handle.

Adding a handle

To add a handle to a relation link:

- Point to a relation link, and then drag the relation link to a new position.

You can also add predefined handles to client or inheritance relation links.

Adding predefined handles

To add a predefined handle to a client or inheritance relation link:

- On the **Options** menu, click **Right Angles**, and then click **Client** or **Inheritance**.

All subsequently added relation links of the type that you click display as a ninety degree angle.

10

Feature tool

The features of a class include both the inherited and immediate features. A feature is of two possible kinds:

- **attribute** — contains information stored with each instance of the class and returns a result. Features that are attributes cannot have arguments.
- **routine** — describes an algorithm applicable to every instance of the class. If the routine returns a result, it is called a **function**; if not, it is called a **procedure**.

This is a classification based on the feature's implementation technique as seen from the class itself. As viewed from the perspective of a client class, a feature can also be classified as either:

- a **command** — the feature can change its target object, but does not return a result. A command may only be implemented as a *procedure*.
- a **query**— the feature returns a result, but should not change the object. A query may be implemented as either a *function* or an *attribute*.

and its implementation status in the class is one of two kinds:

- **effective** — the feature is completely implemented.
- **deferred** — the feature is *specified* in the current class, but not implemented yet; implementations may be provided in proper descendants.

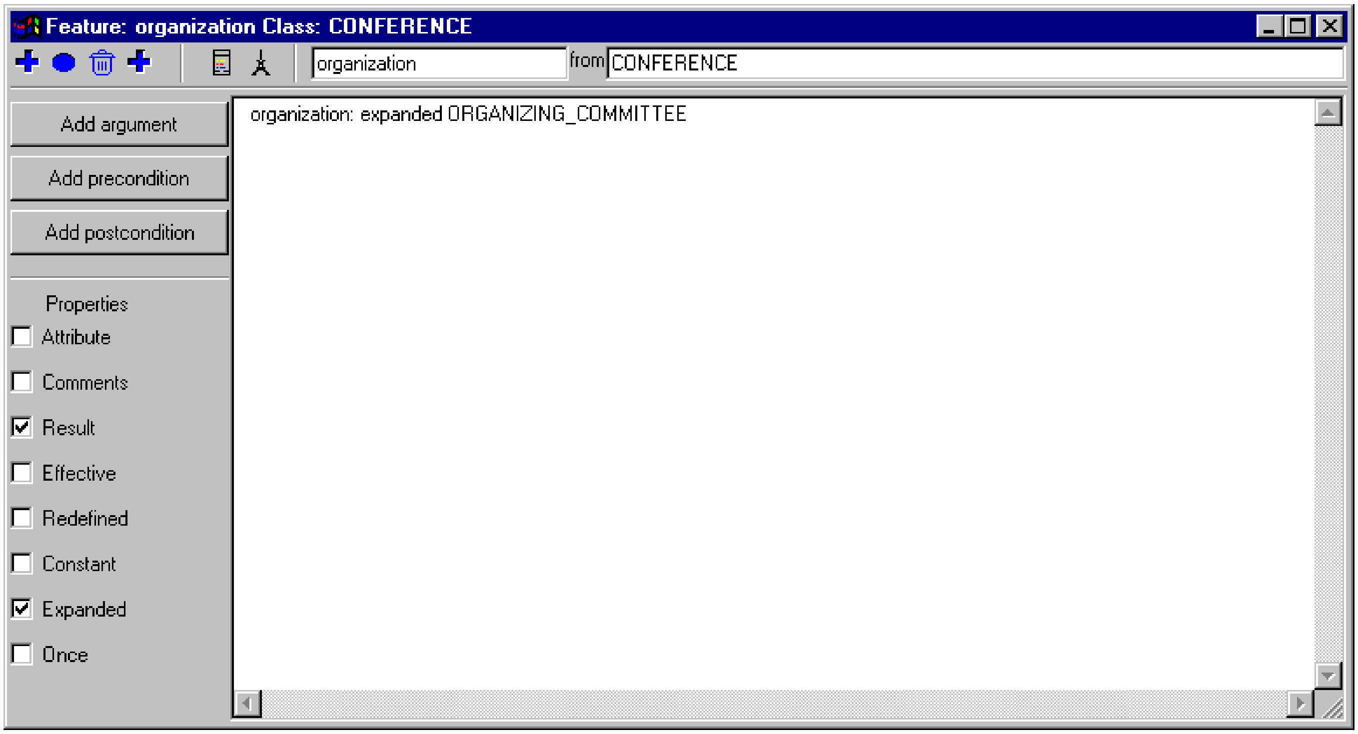
For more information, see "Add argument", page 99, and "Properties", page 100, later in this chapter.

The **Feature** tool sets properties for the active feature and its signature (number and type of arguments).

To display the **Feature** tool, do one of the following:

- In the **Class** tool, hold down CTRL, and then right-click the name of a feature.
- On the **Tools** menu, click **Feature Tool**.

For more information on the **Class** tool, see chapter 8, "Class tool".



The **Feature** tool contains the following toolbar, commands, holes and buttons.

10.1 Feature toolbar

The **Feature** toolbar displays across the top of the tool window and contains the following buttons, holes and commands.

Feature Hole

Displays the contents of the feature you drop on the hole using the pick-and-drop operation.

Class Hole

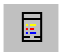
Displays the contents of the class you drop on the hole using the pick-and-drop operation.

Deletion Hole

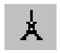
Deletes the development object you drop on the hole using the pick-and-drop operation. For more information on the pick-and-drop operation, see chapter 2, "Getting started".

Cloning Hole

Creates a copy of the feature you drop on the hole using the pick-and-drop operation. The new feature contains all the properties of the original one.

Specification 

Displays the active feature in syntactically-correct BON form (default display).

Show Eiffel code 

Displays the active feature in syntactically-correct Eiffel text. This provides a preview of what happens when you click **Generate Eiffel (this system)** or **Generate Eiffel (this cluster)** on the **File** menu.

Feature name from Class name (Target Name tool)

Displays the name of the active feature (left text box) and the corresponding class (right text box).

You can retarget the **Feature** tool to a feature in the active class or a different class by typing in these text boxes. For more information on the **Target Name** tool, see chapter 6, "EiffelCase tools".

Retargeting the Feature tool to a different feature

To retarget the **Feature** tool to a different feature in the active class:

- Type the name of the feature in the left text box, and then press ENTER.

Retargeting the Feature tool to a different class

To retarget the **Feature** tool to a feature in a different class:

- Type the name of the class in the right text box, and then press ENTER.

Instead of typing a complete feature or class name, type the wildcard operator (*) and then press ENTER to display a menu of matching possibilities. For example, typing **APP*** returns words such as APPLE, APPLET, APPLICATION and so on.

10.2 Add argument

Adds an argument to the active feature. By default, the argument is named **arg**, and the type is **ANY**.

You can use the **Editor** tool to change the name and type of an argument. For more information on the **Editor** tool, see chapter 11, "Editor tool".

Changing the name and type of an argument

To change the type of an argument:

1 Hold down SHIFT, and then right-click to select **arg**.

2 In the **Editor** tool, type a new name in the left text box and a new type in the right text box, and then click the **Check** button.

Changing the type of an argument

To change the type of an argument:

1 Hold down SHIFT, and then right-click to select **ANY**.

2 In the **Editor** tool window, enter a new type, and then click the **Check** button.

10.3 Add precondition and Add postcondition

Adds a precondition or postcondition clause to the feature.

A *precondition clause* contains any requirements that the client must satisfy before a routine is called. A *postcondition clause* contains the requirements that the routine must satisfy on return, assuming the precondition clause is true.

You can use the **Editor** tool to change the precondition or postcondition clause. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Changing a precondition or postcondition

To change a precondition or postcondition:

1 Hold down SHIFT, and then right-click to select the precondition or postcondition.

2 In the **Editor** tool, type information, and then click the **Check** button.

10.4 Properties

Sets the properties of the feature. Select to include a property.

Attribute

Defines the feature as an *attribute* — a data item associated with the corresponding class.

Comments

Adds descriptive information for the active feature. By default, the comment is named **comment**.

You can use the **Editor** tool to add comments for the active feature. For more information on the **Editor** tool, see chapter 11, “Editor tool”.

Changing default comment information

To change default comment information:

1 Hold down SHIFT, and then right-click to select **comment**.

2 In the **Editor** tool window, type text, and then click the **Check** button.

Result

Defines the feature as either an attribute or a function. If clear (default), sets the feature as a procedure.

Effective

Completely implements the feature — opposite of **Deferred**.

Redefined

Redefines (effects) the feature.

Deferred

Defers implementation of the feature — opposite of **Effective**.

Expanded

Defines the feature as an expanded type.

Once

The value returned depends on the type of routine:

- **procedure** — algorithm executes only once.
- **function** — returns the same value each time the routine executes.

11

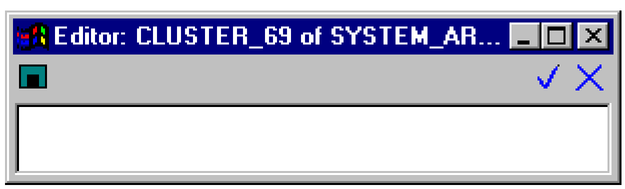
Editor tool

The **Editor** tool sets name, comment and documentation information for the active development object. It appears when you do one of the following:

- Add a development object or relation link to the workspace.
- Hold SHIFT, and then right-click a development object in the **System**, **Class**, **Chart**, **Hidden Components**, or **Feature** tool windows.

For more information on adding a development object or relation link to the workspace, see chapter 2, “Getting started”.

For more information on the **System**, **Class**, **Chart**, **Hidden Components**, and Feature tools, see chapter 6, “EiffelCase tools”, chapter 7, “System tool”, chapter 8, “Class tool”, and chapter 10, “Feature tool”.



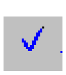
The **Editor** tool contains the following toolbar.

11.1 Editor toolbar

The **Editor** toolbar displays across the top of the tool window and contains the following the following hole and buttons.

Name hole 

Retargets the tool to the development object you drop on the hole using the pick-and-drop operation.

OK 

Closes the **Editor** tool, and saves all changes.

Close

Closes the **Editor** tool.

11.2 Adding descriptive information

To add descriptive information:

1 Hold down SHIFT, and then right-click to select the field.

2 In the **Editor** tool window, type text, and then click the **Check** button.

All information entered converts to **description** entries in the Indexing clauses for the classes. If you do not enter any descriptive information, the default placeholders are deleted before printing.

Note: Entered text must contain only alphanumeric characters.

11.3 Renaming a development object

To rename a development object:

1 Hold SHIFT, and then right-click the name of the object to rename**.**

2 In the **Editor** tool, type a new name, and then click the **Check** button.

11.4 Adding Eiffel code to a query

To add Eiffel code to a query:

1 Hold SHIFT, and then right-click **do**.

2 In the **Editor** tool, type the Eiffel code, and then click the **Check** button.

For more information on coding in Eiffel, see *Eiffel: The Language*.

11.5 Defining an invariant

To define an invariant:

1 Use the pick-and-drop operation to drop **tag** anywhere in the tool window.

2 In the **Editor** tool, type a new name in the left text box, type the condition for the invariant to be true in the right text box, and then click the **Check** button.

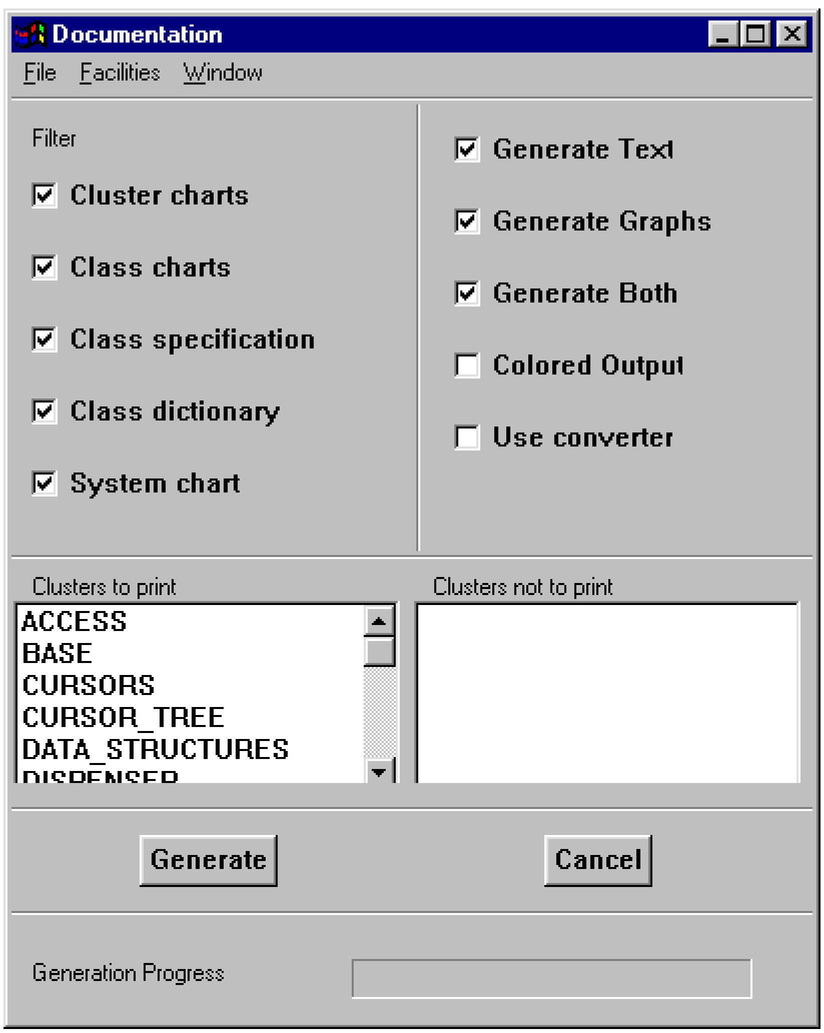
12

Documentation tool

The **Documentation** tool specifies report types and any printing commands.

To display the **Documentation** tool, do one of the following:

- On the **Documentation** menu, click **Documentation Tool**.
- Press CTRL+D.



The **Documentation** tool contains the following menus, commands, check boxes and command buttons.

12.1 File menu

The **File** menu contains the following commands.

Generate textual doc

Prints the text of the active system to the selected destination.

In the **Documentation generation** dialog box, do one of the following, and then click OK:

- Select the **File** check box, type the path, and then click an output filter in the **Select a filter** list. You can also click **Browse**, and then in the **Choice** box, type the path or click a folder in the **Directories** list.
- Select the **Printer** check box, click **OK**, and then set printing defaults.

Generate graphical doc

Prints the graphical representation of the active system to the selected destination. When printing to file, Windows generates a bitmap, UNIX a PostScript file.

In the **Documentation generation** dialog box, do one of the following, and then click OK:

- Select the **File** check box, type the path, and then click an output filter in the **Select a filter** list. You can also click **Browse**, and then in the **Choice** box, type the path or click a folder in the **Directories** list.
- Select the **Printer** check box, click **OK**, and then set printing defaults.

We recommend using the PostScript filter to generate all graphical documentation.

Exit

Closes the **Documentation** tool.

12.2 Facilities menu

The **Facilities** menu contains the following commands.

Add all clusters to generate list

Adds all active clusters to the **Generate these clusters** list.

Move all clusters to exclusion list

Moves all active clusters from the **Generate these clusters** list to the **Exclude these clusters**.

12.3 Window menu

The **Window** menu contains the following command.

Save size

Sets and saves the active tool window size as the default in the resource file. For more information on the resource file, see appendix A, "System resources".

12.4 Filter

Sets the development objects and information to include in the output. Select to include an option.

Cluster charts

Prints a textual representation of all active cluster information.

Class charts

Prints a textual representation of all active class information.

Class specification

Prints the Eiffel code for all active classes to one file.

Class dictionary

Prints all active classes in the system alphabetically, including any descriptive information. For more information on adding descriptive information, see chapter 7, "System tool".

System chart

Prints a list of all active clusters, including descriptive information.

12.5 Format

Sets the output format. Select to include an option.

Generate Text

Select to print textual information.

Generate Graphs

Select to print graphical information.

Generate Both

Select to print the structure of the active system in both graphical and textual formats.

Colored Output

Prints the EiffelCase output in color, if supported by your PostScript printer. For more information, see your PostScript printer documentation.

12.6 Generate these clusters

Lists all active clusters available to include in the generated documentation. Click to move an individual cluster to the **Exclude these clusters** list.

12.7 Exclude these clusters

Lists all active clusters to exclude from the generated documentation. Click to move an individual cluster to the **Generate these clusters** list.

12.8 Generate

Prints the active system using all values set in the **Documentation** tool to the selected destination. When printing to file, Windows generates a bitmap, UNIX a PostScript file.

In the **Documentation generation** dialog box, do one of the following, and then click OK:

- Select the **File** check box, type the path, and then click an output filter in the **Select a filter** list. You can also click **Browse**, and then in the **Choice** box, type the path or click a folder in the **Directories** list.
- Select the **Printer** check box, click **OK**, and then set printing defaults.

12.9 Cancel

Closes the **Documentation** tool.

12.10 Generation Progress

Displays the percentage of documentation generation that has been completed (not available in UNIX).

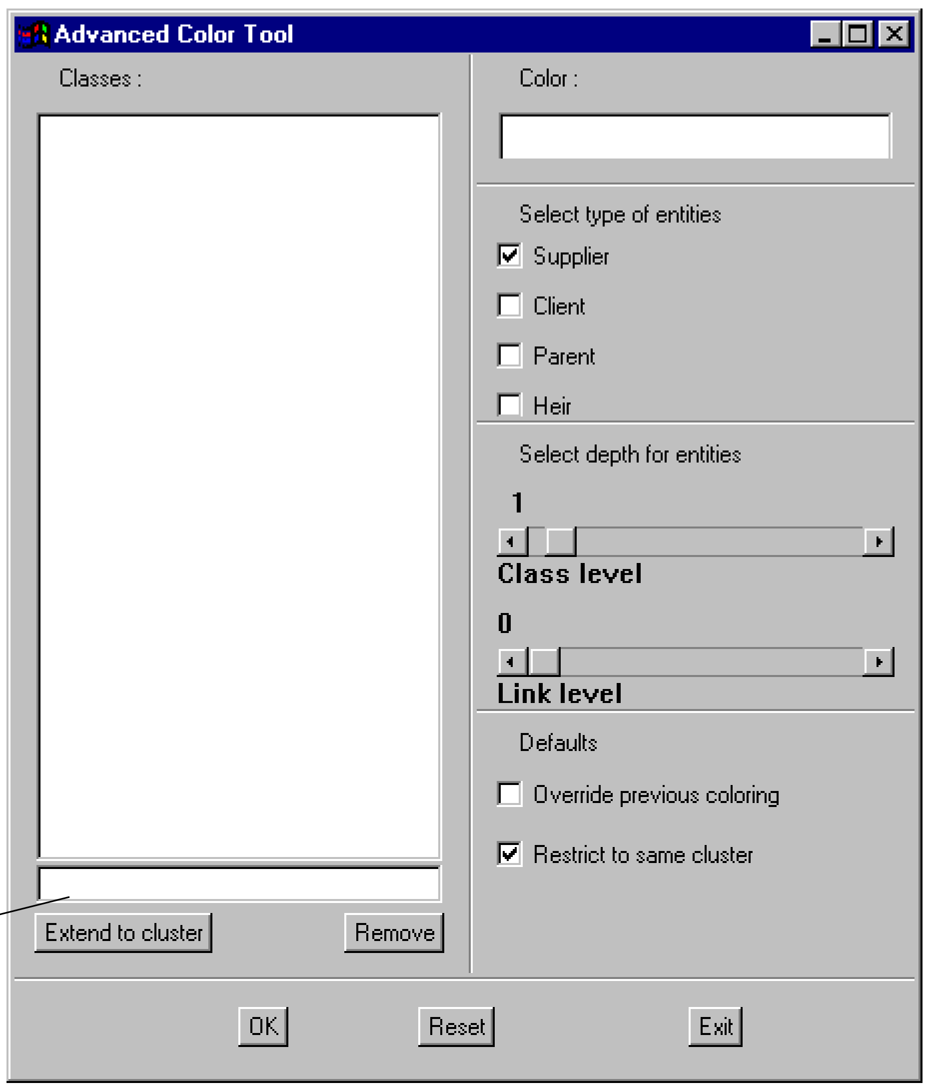
13

Advanced Color tool

The **Advanced Color** tool adds color to the classes and relation links in the active cluster.

To display the **Advanced Color** tool:

- On the **Options** menu, click **Advanced Color Tool**.



Classes box

The **Advanced Color** tool contains the following commands, buttons and options.

13.1 Classes

Lists the classes to recolor. You can add classes to this list manually or using the pick-and-drop operation.

Adding classes to the Classes list

To add classes to the **Classes** list:

- Type the name of the class to add in the **Classes** box, and then press ENTER.

13.2 Extend to cluster

Adds all classes from the same cluster as the class selected in the **Classes** list to the **Classes** list.

13.3 Remove

Removes the selected class from the **Classes** list.

13.4 Color

Sets the color to use to recoloring the active classes and associated relation links.

Setting the recoloring color

To set the recoloring color:

1. Type the name of the color to add in the **Color** box, and then press ENTER.
2. Under **Defaults**, click **Override previous coloring**, and then click OK.

13.5 Select type of entities

Lists the types of relation links that you can recolor: **Supplier**, **Client**, **Parent**, or **Heir**. Select to recolor relation links of that type.

13.6 Select depth for entities

Move the slider to the right to increase the following values: **Link level** or **Class level**.

13.7 Defaults

Click to select an option.

Override previous coloring

Recolors classes and relation links that are already colored.

Note: A class displaying in the default class display color (yellow) is not considered “already colored”.

Restrict to same cluster

Recolors only the classes and relation links in the same clusters as the classes in the **Classes** list.

13.8 OK

Recolors the classes in the **Classes** list using the color set in the **Color** box. All associated relation links of the type selected in the **Select type of entities** area also recolor to the link and class levels set in the **Select depth for entities** area.

13.9 Reset

Restores the **Advanced Color** tool to its default values.

13.10 Exit

Closes the **Advanced Color** tool.

14

Merging tool

EiffelCase supports both forward and reverse engineering. This means that when working on a system you can:

- Modify the active EiffelCase system, and then use EiffelCase to generate the corresponding software text when you click **Generate Eiffel (whole system)** or **Generate Eiffel (this cluster)** on the **File** menu.

- Modify the software text, and then use EiffelBench to reverse-engineer the text, updating the active EiffelCase system.

When performing either of these operations, modifications can be made to the same class in two different ways. If this happens, the **Merging** tool appears and reconciles the differences.

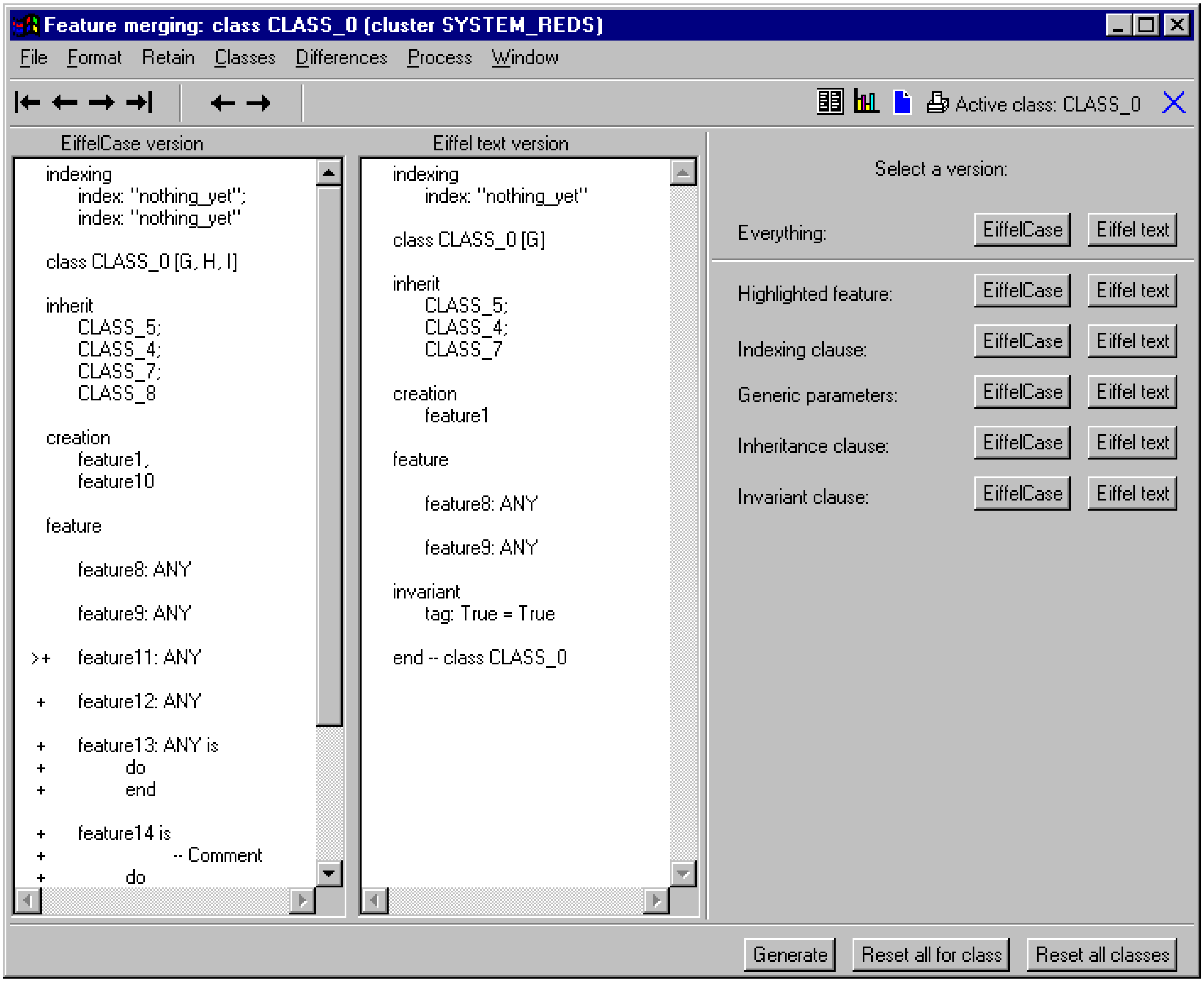
Differences detected by the **Merging** tool include inheritance properties, individual features, and invariant clauses. For each one of these elements, you can choose the text that you want to use from either the active EiffelCase system (**EiffelCase version** list) or the Eiffel text (**Eiffel text version** list).

There are two different display formats for the **Merging** tool:

- **Class differences** — differences display in the **EiffelCase version** list and the **Eiffel text version** list.

- **Statistics** — the **Classes already reconciled** list and the **Classes remaining** list display.

For more information on the display formats, see “Format menu”, page 115, later in this chapter.



The **Merging** tool contains the following menus, commands, toolbar, and buttons. Commands may or may not be available for use, depending on the display format selected.

14.1 File menu

The **File** menu contains the following commands.

Print to file

Prints the selected software text to the selected destination.

In the **Merging Text Generation** dialog box, do one of the following, and then click **Generate**:

- Type the path name in the **Destination** box.
- Click **Browse**, in the **Choice** box, type the path or click a folder in the **Directories** list.

By default, EiffelCase generates the file in $CASEGEN\Doc.

Print

Prints the selected software text to your printer.

Exit

Closes the **Merging** tool.

14.2 Format menu

The **Format** menu contains the following commands.

Show class differences

Displays the difference between the classes in the **EiffelCase version** list and the **Eiffel text version** list.

Show statistics

Displays the classes whose differences have been reconciled (**Classes already reconciled** list), as well as the classes remaining to be reconciled (**Classes remaining** list).

14.3 Retain menu

The **Retain** menu contains the following commands.

Retain EiffelCase version

Generates Eiffel text using the feature or feature clause text selected in the **EiffelCase version** list.

Retain Eiffel text version

Generates Eiffel text using the feature or feature clause text selected in the **Eiffel text version** list.

Retain EiffelCase versions for all

Generates Eiffel text using all text in the **EiffelCase version** list.

Retain Eiffel text versions for all

Generates Eiffel text using all text in the **Eiffel text version** list.

14.4 Classes menu

The **Classes** menu contains the following commands.

First class

Displays the first class that contains differences in the **EiffelCase version** or **Eiffel text version** list.

Previous class

Displays the previous class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Next class

Displays the next class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Last class

Displays the last class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list.

14.5 Differences menu

The **Differences** menu contains the following commands.

Next differing item in class

Displays the next feature or feature clause that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Previous differing item in class

Displays the previous feature or feature clause that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

14.6 Process menu

The **Process** menu contains the following commands.

Apply selections (no undo)

Generates Eiffel text using the selections made.

Reset all classes

Discards all changes made to the active class since the beginning of merging.

14.7 Window menu

The **Window** menu contains the following command.

Save size

Sets and saves the active window size as the default in the resource file. For more information on the resource file, see appendix A, “System resources”.

14.8 Merging toolbar

The **Merging** toolbar displays across the top of the tool window and contains the following buttons, commands, and tool.

First class

Displays the first class that contains differences in the **EiffelCase version** or **Eiffel text version** list.

Previous class

Displays the previous class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Next class

Displays the next class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Last class

Displays the last class in the same cluster that contains differences in the **EiffelCase version** or **Eiffel text version** list.

Previous feature

Displays the previous feature that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Next feature

Displays the next feature that contains differences in the **EiffelCase version** or **Eiffel text version** list, if applicable.

Show class differences

Displays the differences between the classes in the **EiffelCase version** list and the **Eiffel text version** list.

Show statistics

Displays the classes whose differences have been reconciled (**Classes already reconciled** list), as well as the classes remaining to be reconciled (**Classes remaining** list).

Print to file

Prints the software text to the selected destination.

In the **Generation Tool** dialog box, do one of the following, and then click **Generate**:

- Type the path and file name in the **Destination** box.
- Click **Browse**, in the **Choice** box, type the path and file name or click a folder in the **Directories** list.

By default, EiffelCase generates the file in $CASEGEN\Doc.

Print

Prints the software text to your printer.

Active class

Displays the name of the active class.

Close

Closes the **Merging** tool.

14.9 Display formats

There are two formats that display in the **Merging** tool, depending on the format selected — **Show class differences** or **Show statistics**:

- **EiffelCase version** list and **Eiffel text version** list (**Show class differences**)
- **Classes already reconciled** list and **Classes remaining** list (**Show statistics**)

EiffelCase version

Displays the Eiffelcase version of the software text for the active system.

Eiffel text version

Displays the Eiffel text version of the software text for the active system.

Classes already reconciled

Lists the classes that have been reconciled.

Classes remaining

Lists the remaining classes to be reconciled.

14.10 Select a version

The **Select a version** area contains the following buttons. Buttons may or may not display, depending on the differences found.

Click the button corresponding to the version of the text to include in the generated software text — **EiffelCase** or **Eiffel text** — for the following:

- **Everything**
- **Highlighted feature**
- **Indexing clause**
- **Generic parameters**
- **Inheritance clause**
- **Invariant clause**

14.11 Generate

Generates Eiffel text using the selections made.

14.12 Reset all for class

Discards all changes made to the active class since the beginning of merging.

14.13 Reset all classes

Discards all changes made to all classes since the beginning of merging.

A

System resources

When you create a new EiffelCase system, a CASEGEN subdirectory is added to the EiffelCase system directory, by default. Under Windows, the EiffelCase system directory also contains the system file (.ecr), whose default file name is system_architecture.ecr.

On operating systems such as Linux and Unix that support the concept of a home directory, you can use the **eifinit** subdirectory of the root directory as a personal resource directory. Resources defined in the personal directory override those in the default directory.

You can set the default properties of EiffelCase sessions by defining any of the resources in the following table:

Resource	**Default value**
automaticc_saving_delay	**10**
background_color	**“gray“**
class_color	**“black”**
class_interior_color	**"yellow"**
cluster_color	**“blue”**
cluster_title_interior_color	**"green"**
delay_for_sneak	**1500**
drawing_background_color	**“white”**
foreground_color	**"black"**
grid_color	**"LightGray"**
link_apart	**1**
link_color	**"black"**
link_aggreg	**"blue"**
link_Inherit	**"red"**

Resource	**Default value**
link_label_space	**1**
link_supplier	**“black”**
nbr_color_per_line	**Windows: 4 UNIX: 5**
printer_command	**"please_fill"**
resize_coin_color	**"black"**
selected_interior_color	**"cyan"**
selected_invert_color	**"black"**
show_bon	**True**
sneak_window_display	**1**
tab_length	**8**

Index