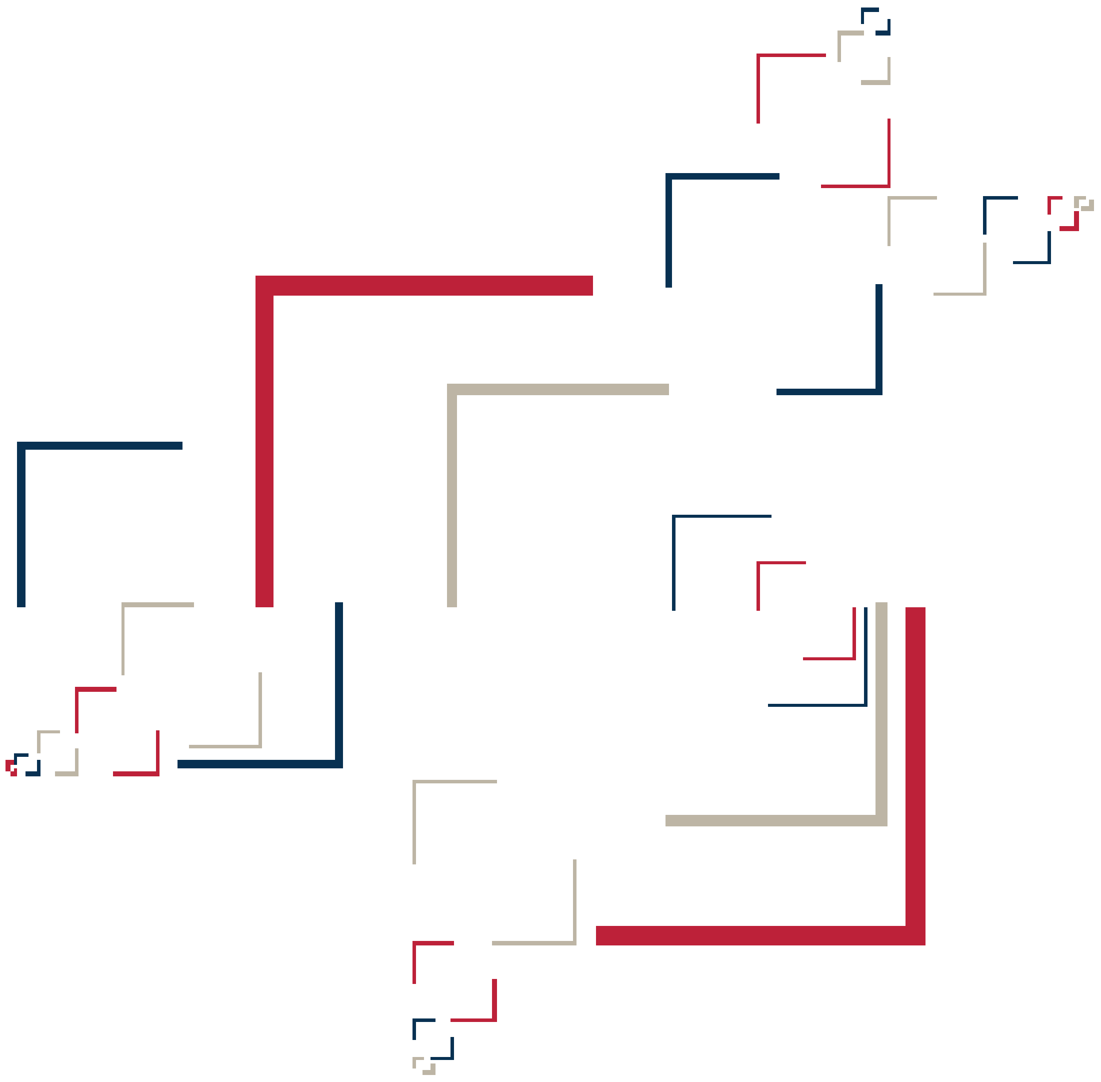


Micro Focus®

Modernization Workbench™

Analyzing Projects

MICRO FOCUS®

Contents

Preface

The Modernization Workbench is a suite of PC-based software products for analyzing, re-architecting, and transforming legacy applications. The products are deployed in an integrated environment with access to a common repository of program objects. Language-specific parsers generate repository models that serve as the basis for a rich set of diagrams, reports, and other documentation.

The Modernization Workbench suite consists of customizable modules that together address the needs of organizations at every stage of legacy application evolution: maintenance/enhancement, renovation, and modernization.

Audience

This guide assumes that you are a corporate Information Technology (IT) professional with a working knowledge of the legacy platforms you are using the product to analyze. If you are transforming a legacy application, you should also have a working knowledge of the target platform.

Organization

This guide contains the following chapters:

- Chapter 1, “Overview,” provides an overview of the Modernization Workbench project analysis tools.
- Chapter 2, “Analyzing Relationship Flows,” describes how to use the Diagrammer to perform high-level analysis of applications.
- Chapter 3, “Analyzing Global Data Flows,” describes how to use the Global Data Flow tool to perform low-level analysis of program data flows.
- Chapter 4, “Analyzing Batch Applications,”describes how to use the Batch Application Viewer to perform low-level analysis of batch processes.
- Chapter 5, “Analyzing Data Operations,” describes the workbench CRUD report and IMS port analysis feature.
- Chapter 6, “Estimating Complexity and Effort,” describes how to use the legacy estimation tools to estimate project complexity and effort.
- Chapter 7, “Identifying Classes of Data Items,” describes how to use the Change Analyzer to identify the class of data items used to perform a business function in a legacy application.
- Chapter 8, “Performing SOA Analysis,” describes how to assess legacy programs for their affinity with modern service-oriented architectures (SOA).
- Appendix A, “Repository Exchange Protocol Syntax,” describes the Repository Exchange Protocol (RXP) query syntax.
- Appendix B, “Common Diagramming Features,” describes features shared by all the workbench diagramming tools, except the Diagrammer itself. Use these features to edit diagrams, save diagrams in standard formats, print diagrams, and more.
- The Glossary defines the names, acronyms, and special terminology used in this guide.

Conventions

This guide uses the following typographic conventions:

- **Bold type**: indicates a specific area within the graphical user interface, such as a button on a screen, a window name, or a command or function.
- *Italic type*: indicates a new term. Also indicates a document title. Occasionally, italic type is used for emphasis.
- `Monospace type`: indicates computer programming code.
- **`Bold monospace type`**: indicates input you type on the computer keyboard.
- 1A/1B, 2A/2B: in task descriptions, indicates mutually exclusive steps; perform step A or step B, but not both.

Related Manuals

This document is part of a complete set of Modernization Workbench manuals. Together they provide all the information you need to get the most out of the system.

- *Getting Started* introduces the Modernization Workbench. This guide provides an overview of the workbench tools, discusses basic concepts, and describes how to use common product features.
- *Preparing Projects* describes how to set up Modernization Workbench projects. This guide describes how to load applications in the repository and how to use reports and other tools to ensure that the entire application is available for analysis.
- *Analyzing Programs* describes how to analyze applications at the program level. This guide describes how to use HyperView tools to view programs interactively and perform program analysis in stages. It also describes how to set up an application glossary and how to extract business rules.
- *Managing Application Portfolios* describes how to build enterprise dashboards that track survey-based metrics for applications in your

portfolio. It also describes how to use Enterprise View Express to browse Web-generated views of application repositories.

- *Creating Components* describes how to extract program components from a legacy application.
- *Transforming Applications* describes how to generate legacy application components in modern languages.
- *Error Messages* lists the error messages issued by Modernization Workbench, with a brief explanation of each and instructions on how to proceed.

Online Help

In addition to the manuals provided with the system, you can learn about the product using the integrated online help. All GUI-based tools include a standard Windows **Help** menu.

You can display:

- The entire help system, with table of contents, index, and search tool, by selecting **Help:Help Topics**.
- Help about a particular Modernization Workbench window by clicking the window and pressing the **F1** key.

Many Modernization Workbench tools have *guides* that you can use to get started quickly in the tool. The guides are help-like systems with hyperlinks that you can use to access functions otherwise available only in menus and other program controls.

To open the guide for a tool, choose **Guide** from the **View** menu. Use the table of contents in the **Page** drop-down to navigate quickly to a topic.

Overview

1

When you verify a legacy application, the parser generates a model of the application that defines the objects it uses and how they interact. The tools described in this guide help you analyze the model at the project level. You can view high- and low-level process and data flows, estimate project effort and complexity, identify classes of data items that may be affected by a change in a field, and assess legacy programs for their affinity with modern service-oriented architectures (SOA).

Analyzing Relationship Flows

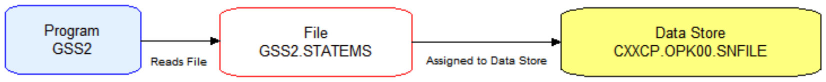
Modernization Workbench offers three related tools for analyzing relationship flows in legacy applications. The core Diagrammer tool lets you analyze high-level relationship flows. The other tools let you drill down deeper into data flows and batch processes All the tools let you browse source code interactively.

Note: Two other analysis tools, the Database Schema and User Interface tools, are described in *Transforming Applications*.

Analyzing Relationships with the Diagrammer

The Diagrammer lets you view the relationships between application objects interactively. These relationships describe the ways in which the objects interact. In Figure 1-1, for example, the GSS2 program *reads* the file CSS2.STATEMS. The file, in turn, is *assigned to* the CXX-CP.OPK00.SNFILE data store.

Figure 1-1 *Relationships in Diagrammatic Form*



The Diagrammer's high-level analysis of relationships lets you quickly trace the flow of information in an application. Use the Diagrammer to view:

- Program to program calls
- Program, transaction, and screen flows
- Program to table or data store flows
- Job, program, and data store flows

Analyzing Data Flows with the Global Data Flow Tool

The Global Data Flow tool lets you perform low-level analysis of program data flows. Use it to determine how changes to a variable may affect other variables and to trace assignments to and from a variable across programs.

Analyzing Batch Applications with the Batch Application Viewer

The Batch Application Viewer lets you perform low-level analysis of batch processes. Use it to determine whether jobs are dependent on one another, the programs that use a data store, and the flow of data into or out of a data store.

Analyzing Data Operations

The workbench parser generates relationships that show the data operations a program performs (Insert, Read, Update, or Delete) and the data objects on which the program operates (such as files, segments, tables, and so forth).

The workbench CRUD Report lets you view these relationships in a convenient format. The IMS Port Analysis feature determines the database segments or screens an IMS program operates on by tracing PSB usage through an entire application call sequence.

Estimating Complexity and Effort

If you are planning to implement a change request for a program, one thing you will want to know before you begin the work is how long it will take to complete the change. Modernization Workbench Legacy Estimation tools let you compare programs based on weighted values for selected complexity metrics. Based on the comparison, you can develop a credible estimate of the time required to make the change.

The complexity metrics used in the calculation are a combination of industry standard and Modernization Workbench-generated statistics (Table 6-1). If your own analysis shows that a given program is more or less complex than the weighted calculation would suggest, you can use a *change magnitude* to override the calculated value.

The program might have thousands of source lines, for example, increasing its calculated complexity, while actually being very easy to modify. When you use a change magnitude, your “subjective” estimate of the effort involved, Small, Medium, Large, Extra-large, becomes an input to the effort calculation, along with the weighted values.

Identifying Classes of Data Items

Suppose your organization is considering adding support for a new currency and that you are going to have to expand the existing exchange rate data field from 9(5)V9(3) to 9(5)V9(6) to accommodate the currency. You will need to know the data fields that are affected in the database, intermediate fields that may contain or use the exchange rate field in calculations, and so forth.

The Change Analyzer identifies the *class* of data items that may need to be modified to support the new currency: in our case, not only the exchange rate fields themselves, but any fields related to them by assignment. In this way it lets you answer the kinds of “What if?” questions posed in the recent past by the industry-wide changes for Y2K, Zip+4, and the Euro dollar: “What if I change the type of this variable, or the length of this field? What *other* fields will I also have to change?”

Use change analysis results to prepare project plans and technical specifications. You can generate reports showing source entities that may require modification, lines of code affected, and the like.

Performing SOA Analysis

Most legacy applications can be abstracted to an “ideal architecture,” in which screens, UI programs, data abstraction programs, and data access activities all are implemented in self-contained modules deployed in sharply demarcated layers.

The reality, of course, is usually very different. Abstract knowledge of the data, business logic, data access, and UI may be enmeshed in the same programs, making it extremely difficult to isolate candidates for reuse.

That’s where SOA Analyzer comes in. It categorizes programs according to the functions they perform in an abstract legacy architecture: UI, data abstraction, data access, and so forth. On the basis of the layering analysis it performs and the wealth of analysis tools it provides, users can identify legacy programs that now or with some rewriting can be exposed as Web services. It also provides facilities that make it easy to generate a user interface and navigation scheme on another platform.

Note: SOA Analyzer is a separately licensable add-on to Modernization Workbench.

What’s Next?

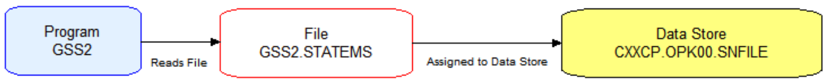
That’s all you need to know before you begin analyzing Modernization Workbench projects. Now let’s look at how you use the Diagrammer to perform high-level analysis of legacy applications.

Analyzing Relationship Flows

2

The Diagrammer lets you view the relationships between application objects interactively. These relationships describe the ways in which program objects interact. In Figure 2-1, for example, the GSS2 program *reads* the file CSS2.STATEMS. The file, in turn, is *assigned to* the CXXCP.OPK00.SNFILE data store.

Figure 2-1 *Relationships in Diagrammatic Form*



The Diagrammer's high-level analysis of relationships lets you quickly trace the flow of information in an application. Use the Diagrammer to view:

- Program to program calls
- Program, transaction, and screen flows

- Program to table or data store flows
- Job, program, and data store flows

Understanding Relationship Flow Diagrams

Relationship flow diagrams use boxes and lines to show the relationships between objects in an application. Depending on the Diagrammer mode, you can display the relationships for:

- All the objects in the selected project.
- Only the objects you copy and paste onto the canvas.

In each mode, all related workspace objects are included in the diagram. In copy-and-paste mode, you can add objects to an existing diagram and expand the diagram one level at a time.

Objects are color-coded, based on a color scheme of your choosing. If you have assigned business names to objects, their business names appear in the diagram as well as their technical names. You can “black-box” items in higher-level groupings that make it easy to visualize their roles in your application, and you can hide objects temporarily in folders.

Understanding Diagram Scopes

The *scope* of a diagram determines the relationships it displays. To view a diagram of the relationships, select the scope in the **Scope** drop-down and choose **View Scope** in the **Scope** menu.

The Diagrammer provides default scopes that you can use for most analysis tasks, but you can create your own scopes if you like. You might want to exclude data stores from a data flow, or include source files in a call map. For more information, see “Using the Scope Editor” on page 2-13.

Understanding Diagram Layouts

Diagrammer layout styles are designed to produce aesthetically pleasing, easy-to-understand diagrams for a wide variety of needs. The following styles are available:

- The *hierarchical* layout style organizes nodes in precedence relationships, with levels for parent and child object types. Click the button on the Diagram pane tool bar to choose the hierarchical layout style.
- The *orthogonal* layout style organizes nodes in relationships drawn with horizontal and vertical lines only. Click the button on the Diagram pane tool bar to choose the orthogonal layout style.
- The *symmetric* layout style organizes nodes based on symmetries detected in their relationships. Click the button on the Diagram pane tool bar to choose the symmetric layout style.
- The *tree* layout style organizes nodes in parent-child relationships, with child nodes arranged on levels farther from the root node than their parents. Click the button on the Diagram pane tool bar to choose the tree layout style.
- The *circular* layout style organizes nodes in clusters of related objects. Click the button on the Diagram pane tool bar to choose the circular layout style.

Tip: Click a layout style button to change the style after a diagram is drawn.

About Random Layout Style

The *random* layout style organizes nodes randomly, so as to maximize drawing performance. When a very large diagram is being loaded for the first time, the Diagrammer notifies you of the large number of display objects and asks you if random layout style is acceptable. Click **Yes** to accept random layout style. Click **No** to draw the diagram with the layout style of your choosing.

Opening the Diagrammer

In the Modernization Workbench Repository Browser, select a project and choose **Diagram** in the **Analyze** menu. The Diagrammer window opens. The embedded Browser pane displays the projects in the workspace. The current project is expanded one level.

Generating Diagrams

You can generate relationship flow diagrams for:

- All the objects in the selected project.
- Only the objects you copy and paste onto the canvas.

In each mode, all related workspace objects are included in the diagram. In copy-and-paste mode, you can add objects to an existing diagram and expand the diagram one level at a time.

Generating a Diagram for the Selected Project

Follow the steps below to generate a relationship flow diagram for the selected project.

To generate a diagram for the selected project:

1. In the **Project** drop-down or the Browser pane, select the project for the diagram.
2. In the **View** menu, choose:
   - **Exclude Objects Outside Project** to limit the diagram to objects in the current project. Otherwise, all related workspace objects are included in the diagram.
   - **Potential Incomplete Composite Relationships** to show incomplete relationship chains in the diagram. Intermediate objects in the chain are displayed even if the final object in the chain is unresolved or otherwise unavailable. Relationships are displayed in red in the diagram.
   - **Project Boundary Entities** to colorize boundary objects in the diagram. Boundary objects are objects with relationships to ob-

jects outside the project. Boundary objects are displayed in the diagram with a red border. Related external objects are displayed in the diagram with a blue border.

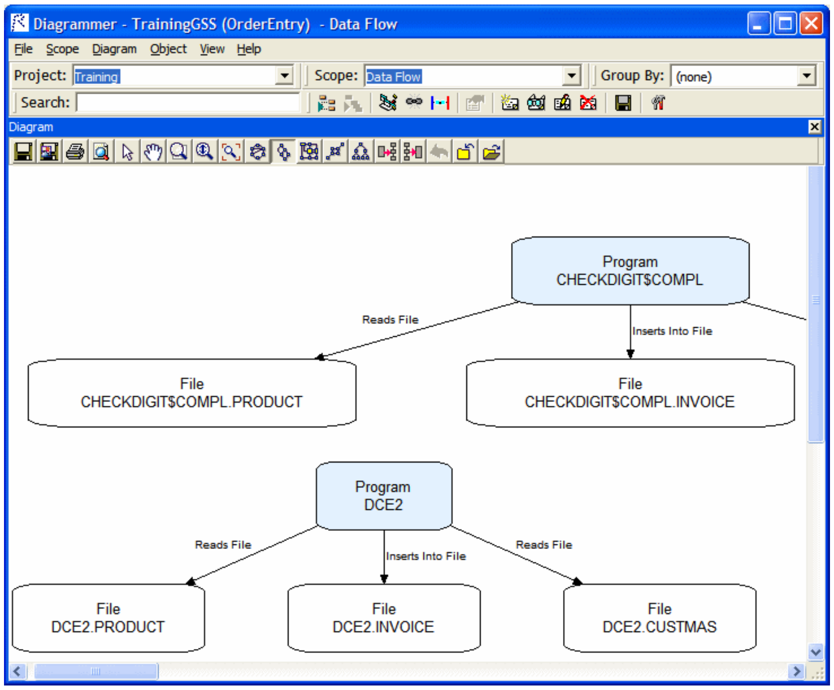
3 In the **Scope** drop-down, choose the scope of the diagram. You can prune the scope directly in the Diagrammer window after the diagram is drawn, as described in "Pruning a Scope" on page 2-25.

4 In the **Group By** drop-down, choose the tag you want to use to "black box" the diagram. Choose **Root Tags Only** in the **View** menu if you want the **Group By** drop-down to show top-level tags only.

5 On the Diagram pane tool bar, choose the layout style for the diagram.

6 Click the button on the Diagrammer tool bar. The Diagrammer draws the selected diagram (Figure 2-2).

Figure 2-2 *Data Flow Diagram for Selected Project*



Project: Training
Scope: Data Flow
Group By: (none)
Search:

Generating a Diagram for Objects Copied and Pasted onto the Canvas

Follow the steps below to generate a relationship flow diagram for objects copied and pasted onto the canvas from the embedded Browser pane. In this mode, you can expand the diagram one level at a time and add objects to an existing diagram.

To generate a diagram for copied-and-pasted objects:

1. In the **View** menu, choose:
   - **Exclude Objects Outside Project** to limit the diagram to objects in the current project. Otherwise, all related workspace objects are included in the diagram.
   - **Potential Incomplete Composite Relationships** to show incomplete relationship chains in the diagram. Intermediate objects in the chain are displayed even if the final object in the chain is unresolved or otherwise unavailable. Relationships are displayed in red in the diagram.
   - **Project Boundary Entities** to colorize boundary objects in the diagram. Boundary objects are objects with relationships to objects outside the project. Boundary objects are displayed in the diagram with a red border. Related external objects are displayed in the diagram with a blue border.
2. In the **View** menu, choose **Auto Expand** to show the entire relationship flow for the selected objects. Turn off **Auto Expand** if you want to expand the diagram one level at a time after it is drawn. Only immediate relationships of the selected objects are drawn initially. Expand the diagram as described in “Expanding and Collapsing Relationships” on page 2-8.
3. In the **Scope** drop-down, choose the scope of the diagram. You can prune the scope directly in the Diagrammer window after the diagram is drawn, as described in “Pruning a Scope” on page 2-25.
4. In the **Group By** drop-down, choose the tag you want to use to “black box” the diagram. Choose **Root Tags Only** in the **View** menu if you want the **Group By** drop-down to show top-level tags only.

5 On the Diagram pane tool bar, choose the layout style for the diagram.

6 In the Browser pane, select the startup objects for the diagram, then press CTRL-C to copy them onto the clipboard. In the **Diagram** menu, choose **Paste** (or press CTRL-V) to paste the objects onto the Diagrammer canvas.

Tip: Make sure to select startup objects in the diagram's intended scope. Most scopes include logical objects rather than source files.

7 Repeat step 6 for each object whose relationships you want to add to the existing diagram. The diagram is automatically redrawn.

Tip: Click the button on the Diagrammer tool bar to redraw the diagram after you have modified it.

Working in Diagrams

The Diagrammer window consists of a Diagram pane, Browser pane, List pane, Quick View pane, Overview pane, and Activity Log. You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Diagram Pane

The Diagram pane displays the diagram for the selected scope. Follow the instructions in "Generating a Diagram for the Selected Project" on page 2-4 or "Generating a Diagram for Objects Copied and Pasted onto the Canvas" on page 2-6 to generate diagrams.

Selecting an Object or Relationship Click the button on the Diagram pane tool bar to choose select mode. Select an object in the diagram by clicking it. Use Ctrl-click to select multiple objects. Click a blank area of the canvas, then drag the mouse over multiple objects to "lasso" them. Selected objects are displayed with handles that you can use to size the object.

Select a relationship by clicking it. The selected relationship is displayed in blue.

Searching for an Object Use the Search facility to find an object in the diagram. Enter the text you want to match in the **Search** field on the Diagrammer tool bar and press Enter. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

Navigating to an Object from a Relationship Select a relationship and choose **Locate Left** in the right-click menu to navigate to the object on the left side of the relationship. Choose **Locate Right** in the right-click menu to navigate to the object on the right side of the relationship.

Tip: Use the Overview pane to navigate to general locations in a diagram.

Moving an Object or Relationship Select an object and drag it to move it in the diagram. Select a relationship and drag it to move it in the diagram.

Resizing an Object Drag the handle of a selected object to resize it.

Viewing Source Select an object in the diagram to view its source in the Quick View pane.

Assigning a Business Name and Description to an Object Select an object and choose **Set Business Attributes:Name** or **Set Business Attributes:Description** in the right-click menu to assign a business name and description to the object. A dialog box opens, where you can enter the business name or business description.

Expanding and Collapsing Relationships Select an object and choose **Expand:Expand Incoming** in the right-click menu to expand incoming relationships. Choose **Expand:Expand Outgoing** to expand outgoing relationships.

Choose **Collapse:Collapse Incoming** in the right-click menu to collapse incoming relationships. Choose **Collapse:Collapse Outgoing** to collapse outgoing relationships.

Hiding Objects in a Folder Select an object or objects and click the button on the Diagram pane tool bar to hide the objects temporarily in a folder node ( ). Select the folder and click the button on the tool bar to restore the objects to the diagram.

Deleting an Object or Relationship Select an object and choose **Delete Node** in the right-click menu to delete the object from the diagram. Select a relationship and choose **Delete Edge** in the right-click menu o delete the relationship from the diagram.

Zooming Click the button on the Diagram pane tool bar to fit the diagram in the Diagram pane.

Click the button to zoom in interactive mode. Drag the mouse over the diagram to zoom on it.

Click the button to zoom in marquee mode. Drag the mouse over the the diagram to draw a marquee. The Diagrammer displays the portion of the diagram inside the marquee.

Moving a Diagram Click the button on the Diagram pane tool bar to choose panning mode. Hold down the left mouse button and drag the mouse to move the diagram in the Diagram pane.

Clearing a Diagram In the **Diagram** menu, choose **Clear** to delete the current diagram from the Diagram pane.

Saving a Diagram Click the button on the Diagram pane tool bar to save a diagram as an image. A Save As dialog opens, where you can specify the image type and characteristics. Click the button on the Diagram pane tool bar to save a diagram as a Tom Sawyer Visualization file (.tsv). A Save dialog opens, where you can specify the name and location of the file.

Saving Diagram-Based Reports Choose **Save *Type* Report** in the **File** menu to display a printable report based on a diagram. Choose **Save Blackbox Interface Report** in the **File** menu to display a printable report showing the relationship of black boxes in a diagram to each other and to other diagram objects. In the printable report, click **Print** to print the report. Click **Save** to export the report to HTML, Excel, RTF, Word, or formatted text.

Printing a Diagram In the **Diagram** menu, click the button on the Diagram pane tool bar to print a diagram. A Print dialog opens, where you can specify the properties of the print job. Click the button to preview the printed diagram.

Browser Pane

The Browser pane displays the workspace repository in tree form. Follow the instructions in "Generating a Diagram for Objects Copied and Pasted onto the Canvas" on page 2-6 to generate a diagram for objects in the Browser pane. For Browser usage information, see *Getting Started* in the workbench documentation set.

Note: To improve Diagrammer startup performance in your next session, hide the Browser pane before you end your current session. Click the close box in the upper righthand corner to hide the pane. When the Browser pane is hidden, select the project you want to diagram in the **Project** drop-down.

List Pane

The List pane displays the relationships in the selected scope, as they appear from left to right in the diagram. Select the relationship for an object in the List pane to navigate to the object in the Diagram pane.

The list includes *relationship chains*, such as Program[IsDefinedInCobol]Cobol[Includes]Copybook[GeneratesTargetXml]TargetXML. If you selected **Partial Relationships** in the **View** menu when you drew the diagram, the list shows incomplete relationship chains, in which the final object in the chain is unresolved or otherwise unavailable.

Sorting Entries Click a column heading in the List pane to sort the relationship entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Quick View Pane

The Quick View pane lets you browse HyperView information for the object selected in the Diagram pane. The information available depends on the type of object selected. You see only source code for a copybook, for example, but full HyperView information for a program. You can also view the properties of the selected object.

Select an object in the Diagram pane to view it in the Quick View pane. Choose the information you want to view for the object from the **Source** drop-down. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set. For Properties pane usage information, see *Getting Started* in the workbench documentation set.

Overview Pane

The Overview pane lets you navigate to a general location in a diagram. It displays the entire diagram, with a frame surrounding the portion of the diagram visible in the Diagram pane. Drag the frame to the area of the diagram you want to view in the Diagram pane. Diagrammer displays the selected area in the Diagram pane.

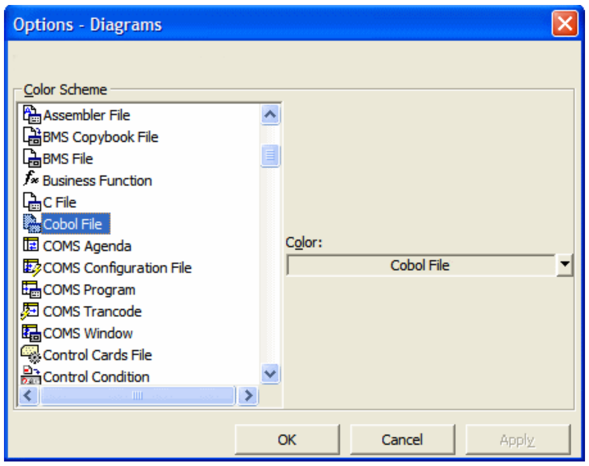
Setting Diagrams User Preferences

Object types are color-coded in relationship flow diagrams so that you can distinguish one type from another: programs from transactions, transactions from screens, and so forth. You can use the default color scheme or create your own. Color settings apply across workspaces.

To set Diagrams user preferences:

1 In the **View** menu, choose **Options**. The Options window opens (Figure 2-3).

Figure 2-3 *Options Window*


Options - Diagrams

Color Scheme

Assembler File
BMS Copybook File
BMS File
Business Function
C File
Cobol File
COMS Agenda
COMS Configuration File
COMS Program
COMS Trancode
COMS Window
Control Cards File
Control Condition

Color: Cobol File

2 In the Color Scheme pane, click the object type whose color you want to edit. The current background color of the type (if any) is displayed in the **Color** drop-down.

3 Click the arrow beside the **Color** drop-down and choose **ForeColor** from the pop-up menu if you want to edit the color of the caption for the object type, or **BackColor** if you want to edit the color of the background for the object type.

4 A standard Windows color control is displayed. Use the **Palette** tab to select the color of the caption or background from the Windows palette. Use the **System** tab to match the color of the caption or background with the color of standard Windows elements.

Moving or Copying Source Files for Diagrammed Objects into Projects

You can use the Diagrammer to move or copy the source files for the objects in a diagram to different projects.

To move or copy source files for diagrammed objects into projects:

1 In the **Diagram** menu, choose **Include in Project**. The Select Project window opens.

2 In the Select Project window, select the project you want to move or copy source files into. Click **New** to create a new project.

3 Select the **Include All Related Objects** check box if you want to move or copy the objects in the workspace related to the diagram objects, the Cobol copybooks included in a Cobol program file, for example.

4 Choose one of the following:

- **Copy** to copy the source files to the specified project.
- **Move From Current Project** to move the source files to the specified project.
- **Move From All Projects** to move the source files from all projects to the selected project.

5 Click **OK** to move or copy the selection.

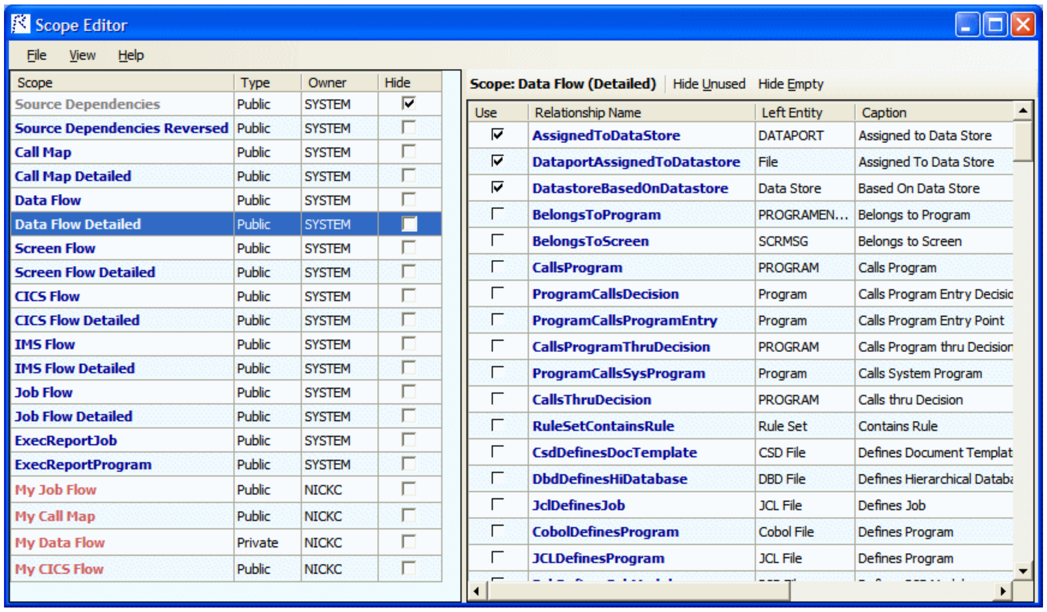
Using the Scope Editor

Use the *Scope Editor* to manage the scopes in your repository, create custom composite relationships, and define relationship filters. For background on scopes, see “Understanding Diagram Scopes” on page 2-2.

Understanding the Scope Editor WIndow

The Scope Editor displays the scopes in your repository and their relationships. To open the Scope Editor, choose **Scope Editor** in the **Scope** menu. The Scope Editor window opens (Figure 2-4).

Figure 2-4 *Scope Editor Window*



Scope	Type	Owner	Hide
Source Dependencies	Public	SYSTEM	✓
Source Dependencies Reversed	Public	SYSTEM	
Call Map	Public	SYSTEM	
Call Map Detailed	Public	SYSTEM	
Data Flow	Public	SYSTEM	
Data Flow Detailed	Public	SYSTEM	
Screen Flow	Public	SYSTEM	
Screen Flow Detailed	Public	SYSTEM	
CICS Flow	Public	SYSTEM	
CICS Flow Detailed	Public	SYSTEM	
IMS Flow	Public	SYSTEM	
IMS Flow Detailed	Public	SYSTEM	
Job Flow	Public	SYSTEM	
Job Flow Detailed	Public	SYSTEM	
ExecReportJob	Public	SYSTEM	
ExecReportProgram	Public	SYSTEM	
My Job Flow	Public	NICKC	
My Call Map	Public	NICKC	
My Data Flow	Private	NICKC	
My CICS Flow	Public	NICKC	

Scope: Data Flow (Detailed) Hide Unused Hide Empty

Use	Relationship Name	Left Entity	Caption
✓	AssignedToDataStore	DATAPORT	Assigned to Data Store
✓	DataportAssignedToDatastore	File	Assigned To Data Store
✓	DatastoreBasedOnDatastore	Data Store	Based On Data Store
	BelongsToProgram	PROGRAMEN...	Belongs to Program
	BelongsToScreen	SCRMSG	Belongs to Screen
	CallsProgram	PROGRAM	Calls Program
	ProgramCallsDecision	Program	Calls Program Entry Decisic
	ProgramCallsProgramEntry	Program	Calls Program Entry Point
	CallsProgramThruDecision	PROGRAM	Calls Program thru Decision
	ProgramCallsSysProgram	Program	Calls System Program
	CallsThruDecision	PROGRAM	Calls thru Decision
	RuleSetContainsRule	Rule Set	Contains Rule
	CsdDefinesDocTemplate	CSD File	Defines Document Templat
	DbdDefinesHiDatabase	DBD File	Defines Hierarchical Databa
	JclDefinesJob	JCL File	Defines Job
	CobolDefinesProgram	Cobol File	Defines Program
	JCLDefinesProgram	JCL File	Defines Program

The lefthand pane of the Scope Editor window lists every scope in the repository. Table 2-1 describes the columns in the lefthand pane.

Table 2-1 *Scope Editor Window Lefthand Columns*

Column Name	Description
Scope	The name of the scope. Scopes are color-coded as follows: • Blue means that the scope is available in the Diagrammer **Scope** drop-down. • Gray means that the scope is not available in the **Scope** drop-down, because it is hidden or because it is a private scope owned by another user. Make a private scope owned by another user available by copying it, as described in “Exporting Scopes” on page 2-18. • Red means that the scope has no relationships associated with it. Specify the relationships in a scope as described in “Specifying the Relationships in a Scope” on page 2-17.
Type	The type of the scope, public or private. The type determines availability of the scope in the Diagrammer **Scope** drop-down: • A public scope is available to every user of the workspace. • A private scope is available only to its owner. Private scopes are listed in the Scope Editor, where you can copy them as described in “Exporting Scopes” on page 2-18.
Owner	The owner of the scope. SYSTEM denotes a default scope.
Hide	Whether the scope is hidden in the Diagrammer **Scope** drop-down. A check mark means that the scope is hidden.

The righthand pane of the Scope Editor window lists every relationship in the repository. Table 2-2 describes the columns in the righthand pane.

Table 2-2 *Scope Editor Window Righthand Columns*

Column Name	Description
Use	Whether the relationship is used in the selected scope. A check mark means that the relationship is used.
Relationship Name	The name of the relationship. Relationships for which a filter has been defined are color-coded magenta.
Left Entity	The entity on the left end of the relationship. Entities for which a relationship filter has been defined are color-coded magenta. A tool tip displays the text of the condition.
Caption	The caption of the relationship.
Right Entity	The entity on the right end of the relationship. Entities for which a relationship filter has been defined are color-coded magenta. A tool tip displays the text of the condition.
Class	The class of the relationship, basic or composite: • A basic relationship defines the direct interaction of two objects: a program and a program entry point, for example. • A composite relationship defines the indirect interaction of two objects. If a job runs a program entry point that belongs to a program, for example, the relationship between the job and program is said to be composite: defined by the chain of relationships between the job and program.
Occurs	The number of times a basic relationship occurs in the repository. For composite relationships, N/A.

Managing Scopes

Use the Scope Editor to edit the Diagrammer **Scope** drop-down, view scope diagrams, create, edit, copy, and delete scopes, import and export scopes, and more.

Editing the Scope Drop-Down Remove a scope from the Diagrammer **Scope** drop-down (but not from the repository) by selecting the scope in the Scope Editor and putting a check mark next to it in the Hide column.

Displaying Only Relationships Used in a Scope Display only the relationships used in a scope by selecting the scope in the Scope Editor and clicking the **Hide Unused** button in the righthand pane.

Hiding Empty Relationships Hide relationships that have no instances in the repository by clicking the **Hide Empty** button in the righthand pane.

Viewing a Diagram of a Scope View the diagram of a scope by selecting the scope in the Scope Editor and choosing **Diagram** in the **File** menu. You can also view a scope diagram by selecting the scope in the Diagrammer **Scope** drop-down and choosing **View Scope** in the **Scope** menu.

Use the buttons described in "Understanding Diagram Layouts" on page 2-3 to view the scope diagram in different layouts. Use the features described in "Working in Diagrams" on page 2-7 to manipulate the diagram.

Viewing a Diagram of a Composite Relationship View the diagram of a composite relationship by selecting the relationship in the Scope Editor and choosing **Diagram** in the right-click menu.

Use the buttons described in "Understanding Diagram Layouts" on page 2-3 to view the relationship diagram in different layouts. Use the features described in "Working in Diagrams" on page 2-7 to manipulate the diagram.

Creating a Scope Create a scope by choosing **New Scope** in the **File** menu. A dialog box prompts you to enter the name, optional caption, class, and owner of the scope. Enter the requested information and click **Save**. For a description of scope details, see Table 2-1.

Specifying the Relationships in a Scope Specify the relationships in a scope by selecting the scope in the Scope Editor and putting a check mark in the Use column next to each relationship you want to include in the scope. Choose **Save** in the **File** menu to save the scope. You can specify relationships only for a scope you own.

Create custom composite relationships as described in "Creating Composite Relationships" on page 2-18. Define relationship filters as described in "Defining a Relationship Filter" on page 2-20.

Editing Scope Details Edit the name, caption, class, and owner of a scope by selecting the scope in the Scope Editor and choosing **Edit** in the **File** menu. A dialog box displays the current details for the scope. Enter the new information and click **Save**. You can only edit details for a scope you own.

Copying a Scope Copy a scope by selecting the scope in the Scope Editor and choosing **Copy** in the **File** menu. A dialog box prompts you to enter the name, optional caption, class, and owner of the new scope. Enter the requested information and click **Save**. You can copy default scopes and public or private scopes owned by another user, then edit them as you would your own scope.

Deleting a Scope Delete a scope by selecting the scope in the Scope Editor and choosing **Delete** in the **File** menu. A dialog box prompts you to confirm that you want to delete the scope. Click **OK**. You can only delete a scope you own.

Deleting a Composite Relationship Delete a composite relationship by selecting the relationship in the Scope Editor and choosing **Delete** in the right-click menu. A dialog box prompts you to confirm that you want to delete the composite relationship. Click **OK**. You can only delete a composite relationship you own.

Importing Scopes Import scopes from an XML file by choosing **Import** in the **File** menu. A dialog box prompts you to specify the name and location of the file. Enter the requested information and click **Open**. The type of an imported scope defaults to public. The owner defaults to the current user.

Note: If a scope or relationship being imported conflicts with an existing scope or relationship, you are notified and the conflicting item is ignored.

Exporting Scopes Export a scope to an XML file by selecting the scope in the Scope Editor and choosing **Export Selected** in the **File** menu. A dialog box prompts you to specify the name and location of the file. Enter the requested information and click **Save**.

To export all the scopes in the repository to an XML file, choose **Export All** in the **File** menu. A dialog box prompts you to specify the name and location of the file. Enter the requested information and click **Save**. To export selected scopes, use Shift-click or Ctrl-click to select the scopes, then choose **Export Selected** in the **File** menu.

Note: Relationship filters are not exported.

Hiding Columns Hide a column in the Scope Editor by deselecting the column name in the **View** menu.

Sorting Columns Click a column heading in the Scope Editor to sort list entries by that column.

Sizing Columns Grab-and-drag the border of a column heading in the Scope Editor to increase or decrease the width of the column.

Creating Composite Relationships

A composite relationship defines the indirect interaction of two objects. If a job runs a program entry point that belongs to a program, for example, the relationship between the job and program is said to be composite: defined by the chain of relationships between the job and program.

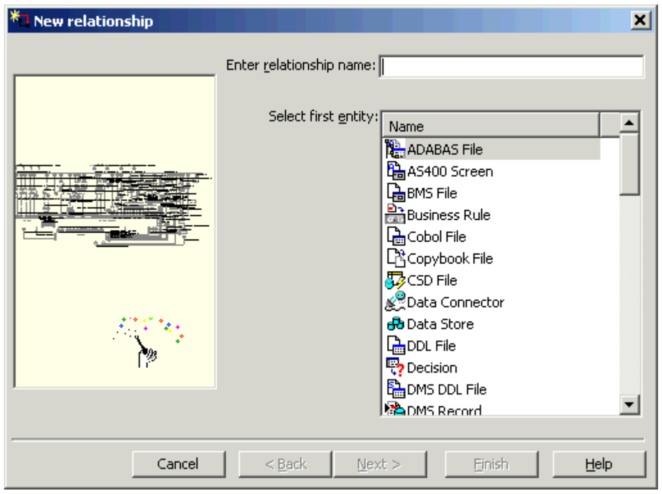
You can use the default composite relationships provided with the system, or use the Scope Editor to create custom composite relationships as described below. Custom composite relationships are available to all users of the workspace.

Note: Custom composite relationships are not copied or exported when a scope is copied or exported.

To create custom composite relationships:

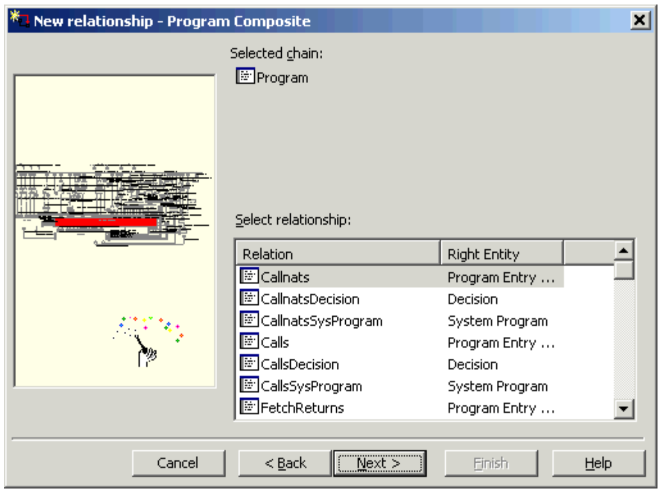
1 In the Scope Editor **File** menu, choose **New Relationship**. The New Relationship Wizard opens (Figure 2-5).

Figure 2-5 *New Relationship Wizard (Screen One)*



2 In the **Enter Relationship Name** field, enter the name of the composite relationship. In the Select First Entity pane, select the object you want to appear on the left side of the relationship chain. Click **Next**. The screen shown in Figure 2-6 appears.

Figure 2-6 *New Relationship Wizard (Screen Two)*



Relation	Right Entity
CallNats	Program Entry ...
CallnatsDecision	Decision
CallnatsSysProgram	System Program
Calls	Program Entry ...
CallsDecision	Decision
CallsSysProgram	System Program
FetchReturns	Program Entry ...

3 In the Select relationship pane, select the relationship you want to appear next in the relationship chain. Click **Next**. A screen similar to the one shown in Figure 2-6 appears.

4 Repeat step 3 for each relationship you want to include in the chain. When you are done, click **Finish**. The New Relationship Wizard closes and the new composite relationship is added to the Scope Editor window.

Defining a Relationship Filter

You can define a relationship filter by setting conditions for either side of the relationships defined in a scope you own. You might want a scope to be restricted to programs that have a cyclomatic complexity greater than 300 and that are defined in COBOL source files, for example, as shown in the steps below.

Condition definitions are based on the Repository Exchange Protocol (RXP), an XML-based API that you can use to interact with application-level information in the workspace repository. For RXP syntax, see Appendix A, “Repository Exchange Protocol Syntax.”

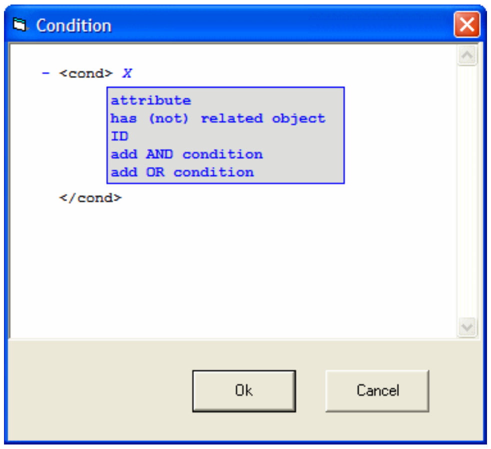
In the Scope Editor, relationships and entities for which a filter has been defined are color-coded magenta. A tool tip over the filtered entity displays the text of the condition.

To set conditions for a scope:

1. In the Scope Editor window, select the relationship you want to set conditions for and choose:
   - **Left Cond** in the right-click menu to qualify the entity on the left side of the relationship.
   - **Right Cond** in the right-click menu to qualify the entity on the right side of the relationship.

   The Condition window opens (Figure 2-7).

Figure 2-7 *Condition Window*



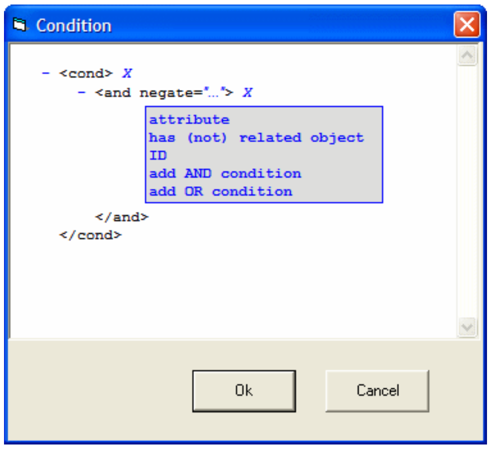
```
- <cond> X
        attribute
        has (not) related object
        ID
        add AND condition
        add OR condition
  </cond>
```

2. In the Condition window, click:
   - **attribute** to qualify the entity according to the value of a given attribute. Programs that have a cyclomatic complexity greater than 300, for example.

- **has (not) related object** to qualify the entity according to a given relationship type. Programs defined in COBOL source files, for example.
- **add AND condition** to qualify the entity according to inclusive criteria using an AND operator. Programs that have a cyclomatic complexity greater than 300 and that are defined in COBOL source files, for example.
- **add OR condition** to qualify the entity according to exclusive criteria using an OR operator. Programs that have a cyclomatic complexity greater than 300 or that are defined in COBOL source files, for example.

The Condition window displays the shell definition for the selected condition (Figure 2-8). The following steps describe how to qualify the program entity using an AND operator. The procedure for other entities and conditions is similar.

Figure 2-8 *Condition Window with Shell Definition for AND Condition*



```
- <cond> X
    - <and negate="..."> X
            attribute
            has (not) related object
            ID
            add AND condition
            add OR condition
      </and>
  </cond>
```

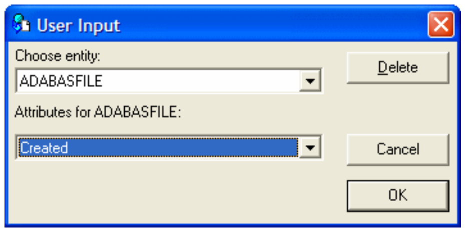
3 In the definition for the AND condition, click **attribute**. The definition for the attribute condition is displayed:

```
<attr name="..." op="..." arg="..." negate="..."/>
```

Note: Click the *X* in a definition to delete the condition.

4 Click the ellipsis (...) in `name="…"`. The User Input dialog opens (Figure 2-9).

Figure 2-9 *User Input Dialog*



5 In the User Input dialog, select the Program entity in the **Choose entity** drop-down and the Cyclomatic Complexity attribute in the **Attributes for Program** drop-down, then click **OK**. The attribute is added to the condition definition.

Note: Click **Delete** in the User Input dialog to delete the criterion defined in the dialog.

6 Click the ellipsis (...) in `op="…"`. A User Input dialog similar to the one shown in Figure 2-9 opens. In the **Choose new value** drop-down, choose the greater than (>) symbol, then click **OK**. The greater than symbol is added to the condition definition.

7 Click the ellipsis (...) in `arg="…"`. A User Input dialog similar to the one shown in Figure 2-9 opens. In the **Enter new value** drop-down, enter 300, then click **OK**. The new value is added to the condition definition:

```
<attr name="Cyclomatic Complexity" op=">" arg="300"
 negate="..."/>
```

Note: In a condition definition, `negate` means not the specified criterion. Programs that do not have a cyclomatic complexity greater than 300, for example. Click the ellipsis (…) in `negate="…"` to set its value to true. Ignore the field otherwise.

8 In the definition for the AND condition, click **has (not) related object**. The definition for the relationship type condition is displayed:

```
<hasrelated negate="...">
```

9 In the choices for the relationship type condition, click **define relationship type**. The choices for the relationship type are displayed.

10 In the choices for the relationship type, click **define relationship**. The definition for the relationship type is displayed:

```
<rel name="..." negate="..."/>
```

11 Click the ellipsis (…) in `name="…"`. A User Input dialog similar to the one shown in Figure 2-9 opens. In the **Choose entity** drop-down, select the Program entity. In the **Relations for Program** drop-down, select the IsDefinedInCobol relationship, then click **OK**. The relationship is added to the condition definition, which looks like this:

```
- <hasrelated negate="...">
    - <reltype>
         <rel name="IsDefinedInCobol" negate="..."/>
      </reltype>
  </hasrelated>
```

12 The AND condition is now complete. The diagram scope will be restricted to programs that have a cyclomatic complexity greater than 300 and that are defined in COBOL source files. The full condition looks like this:

```
- <cond>
        <and negate="...">
            <attr name="Cyclomatic Complexity" op=">"
             arg="300" negate="..."/>
          - <hasrelated negate="...">
              - <reltype>
                    <rel name="IsDefinedInCobol"
                     negate="..."/>
                </reltype>
            </hasrelated>
            </and>
        </cond>
```

Pruning a Scope

You can *prune* relationships from a scope you own directly in the Diagrammer window. When you prune a scope, keep in mind that:

- You are deleting relationships from the current scope exactly as if you were deleting them in the Scope Editor window (Figure 2-4). For that reason, you might want to save the original scope with a different name and use the renamed scope as the basis for the pruned diagram.
- *All* the relationships of the selected type are deleted for the selected object, not just the single relationship you selected in the diagram.

To prune a scope:

1. Select the relationship you want to prune in the diagram and choose:
   - **Prune type for right object** in the right-click menu to delete from the current scope all relationships of the selected type for the right object in the relationship.
   - **Prune type for left object** in the right-click menu to delete from the current scope all relationships of the selected type for the left object in the relationship.

   Redraw the Diagram as described in "Generating a Diagram for the Selected Project" on page 2-4 or "Generating a Diagram for Objects Copied and Pasted onto the Canvas" on page 2-6. Diagrammer deletes the relationships from the redrawn diagram.

Mirroring a Scope

By default, Diagrammer shows the flow of relationships *from* a diagrammed object rather than *to* a diagrammed object. Choose **Mirror Scope** in the **Scope** menu to show the flow of relationships to the object. Choose **Mirror Scope** again to return to the original view.

Black-Boxing Objects in Diagrams

The deeper your understanding of your application, the more confidently you can abstract from its lower-level details to a "bigger picture" view, one that organizes related programs in functional, structural, or other types of groupings: a Customer Maintenance subsystem, for example, in an Order Acceptance application. This is the kind of view a subject matter expert needs to evaluate whether an application does everything it is supposed to do, in the appropriate order.

The Diagrammer *black-box* feature lets you assign lower-level objects to higher-level groupings that make it easy to visualize their roles in your application. (Literally speaking, "black box" is a misnomer. The boxes as drawn are red.) Your diagram might have one black box for the Customer Maintenance subsystem, another for the Order Entry subsystem, and so forth (Figure 2-10). Because the details of these relationships are hidden in the black box until you need to view them, the subject matter expert can hone in quickly on the higher-order functions you have abstracted from them.

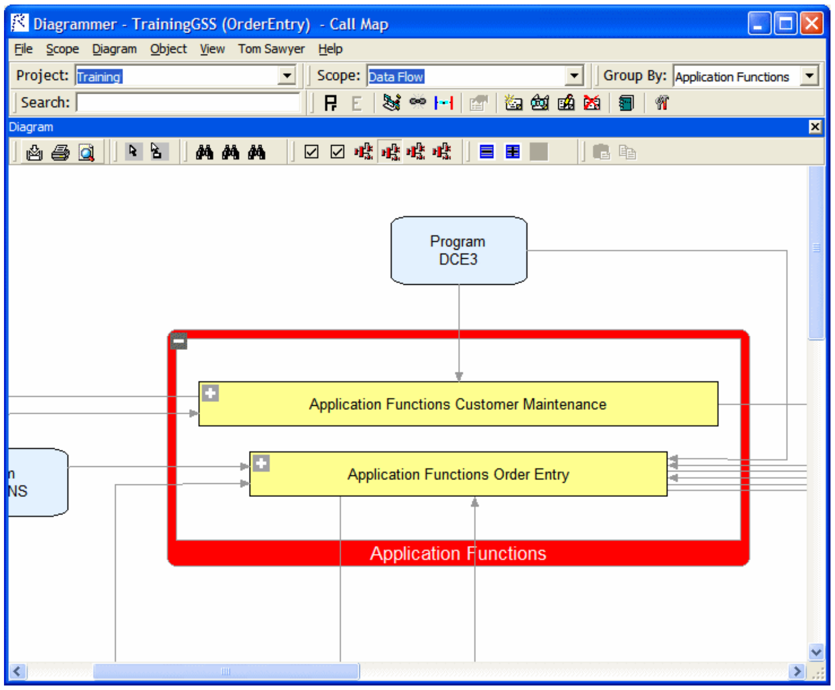
You use a simple tagging language to identify the items in each higher-level grouping, as described in "Assigning Tags to Objects" on page 2-27. Each grouping can, in turn, reference a more inclusive grouping.

If you assign the Customer Maintenance tag to one set of programs, for example, and the Order Entry tag to another, and both tags reference the Application Functions tag, then when you choose Application Functions in the **Group By** drop-down, the Diagrammer puts the programs in black boxes named Application Functions Customer Maintenance and Application Functions Order Entry (Figure 2-10).

After the diagram is drawn, you can:

- Expand or collapse the black box, as described in "Expanding and Collapsing Black Boxes" on page 2-31.
- Print a report showing the relationship of black boxes in the diagram to each other and to other diagram objects, as described in "Saving Diagram-Based Reports" on page 2-9.

Figure 2-10 *Call Map Diagram with Black Boxes*


Diagrammer - TrainingGSS (OrderEntry) - Call Map
File Scope Diagram Object View Tom Sawyer Help
Project: Training
Scope: Data Flow
Group By: Application Functions
Search:
Diagram
Program DCE3
Application Functions Customer Maintenance
Application Functions Order Entry
n NS
Application Functions

Assigning Tags to Objects

You use a simple tagging language to identify workspace objects as members of higher-level functional, structural, or other types of groupings. After you set up these groupings, you can chart them in Enterprise View Express or “black box” them in the Diagrammer.

Each member of the higher-level grouping is identified by a *tag:* Customer Maintenance, for example. Each grouping can, in turn, reference a more inclusive grouping: Application Functions, for example. Later, when you generate charts or diagrams, the tags determine what the chart measures or the diagram “black boxes.”

If you assign the Customer Maintenance tag to one set of programs, for example, and the Order Entry tag to another, and both tags reference the Application Functions tag, then when you group diagram objects by Application Functions, the Diagrammer puts the programs in black boxes named Application Functions/Customer Maintenance and Application Functions/Order Entry (Figure 2-10).

Note: You can assign different tags to the same object. A tag can reference any tag other than itself. The default tag structure is provided for Enterprise View Express use, as described in *Managing Application Portfolios* in the workbench documentation set.

To manage tags:

1A In the Browser or Search view of the Repository pane, select the objects you want to assign a tag. Right-click and choose **Assign Tags** in the pop-up menu.

Tip: Select a project or folder to assign tags to every first-level object in the project or folder.

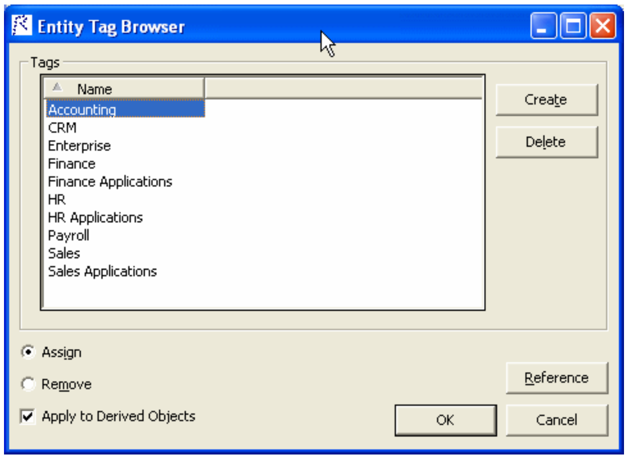
1B In the Diagram pane, select the objects you want to assign a tag. Right-click inside one of the selected objects and choose **Assign Tags** in the pop-up menu.

Tip: Use Shift-click to select multiple objects. In the Diagrammer, hold down the left mouse button and drag the mouse to “lasso” objects.

The Entity Tag Browser window opens (Figure 2-11).

Note: Choose **Entity Tag Browser** in the **File** menu to view the Entity Tag Browser window without having a diagram drawn on the canvas.

Figure 2-11 *Entity Tag Browser Window*



Accounting
CRM
Enterprise
Finance
Finance Applications
HR
HR Applications
Payroll
Sales
Sales Applications

[✓ Assign] Remove

☑ Apply to Derived Objects

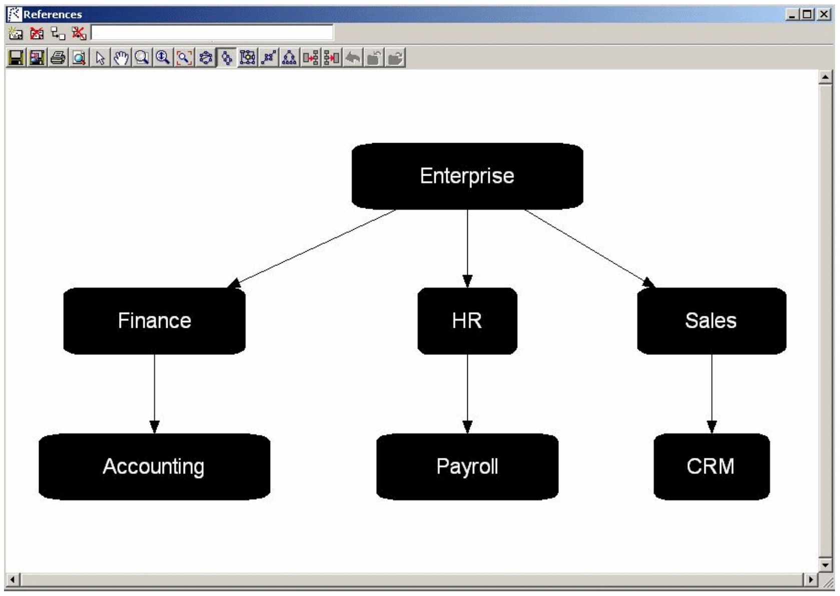
2 In the Tags pane, select the tag you want to assign and click **Assign**. To unassign a tag, select the tag and click **Remove**. If you are assigning a tag to a legacy source file, check **Assign to Derived Objects** to assign the selected tag(s) to all objects derived from this file by the workbench. Click **OK** to assign or unassign a tag.

Tip: You can assign multiple tags to an object. Use Shift-click or Ctrl-click to select multiple tags.

3 To create a tag, click **Create**. A dialog box prompts you to enter the name of the new tag. Enter the name and click **OK**. To edit a tag, select it in the Tags pane, then click inside the text box for its name and enter the new text. To delete a tag, select it and click **Delete**.

4 To create a reference from one tag to another, click **Reference**. The References window opens (Figure 2-12).

Figure 2-12 *References Window*



In the References window, select a tag and click the button on the toolbar. A Select window prompts you to select the tag to reference. Select the tag and click **OK**. The tool draws a line between the tags. To delete a reference, select the relationship line and click the button on the toolbar.

To create a tag in the References window, click the button on the toolbar. A dialog box prompts you to enter the name of the new tag. Enter the name and click **OK**. To delete a tag in the References window, select the tag and click the button on the toolbar.

To redraw the diagram in the References window, click the button on the toolbar.

To view the tags assigned to an object, right-click the object in the Repository or Diagram pane and choose **Properties** in the pop-up menu. Tags appear on the Object Tags tab.

Expanding and Collapsing Black Boxes

Expand a black box to view its contents. Collapse a black box to hide its contents.

- If a black box is collapsed (including a black box in a more inclusive black box), double-click it to expand it. Click the button on the Diagram pane toolbar to expand all the black boxes in an inclusive black box.
- If a black box is expanded (including a black box in a more inclusive black box), double-click it to collapse it. Click the button on the toolbar to collapse all the black boxes in an inclusive black box.

To view a diagram of objects in the black box only, select the black box and choose **Navigate to Child Diagram** in the right-click menu. To restore the full diagram, click the button on the tool bar.

What's Next?

Now that you have performed high-level analysis of relationship flows in the Diagrammer, you are ready to drill down deeper into your application. The next chapter shows you how to analyze program data flows.

Analyzing Global Data Flows

3

The Global Data Flow tool lets you analyze incoming and outgoing data flows in a program. You can view the memory allocation and offset for a variable to determine how changes to the variable may affect other variables, and trace assignments to and from the variable across programs.

Note: Projects must have been verified with the **Enable Data Element Flow** option set in the project verification options.

How the Global Data Flow Tool Is Implemented

The Global Data Flow tool is implemented as both a standalone and HyperView-based tool. This chapter describes the HyperView version. Usage is identical for the standalone tool. For Context and Clipper pane usage, and for HyperView usage in general, see *Analyzing Programs* in the workbench documentation set.

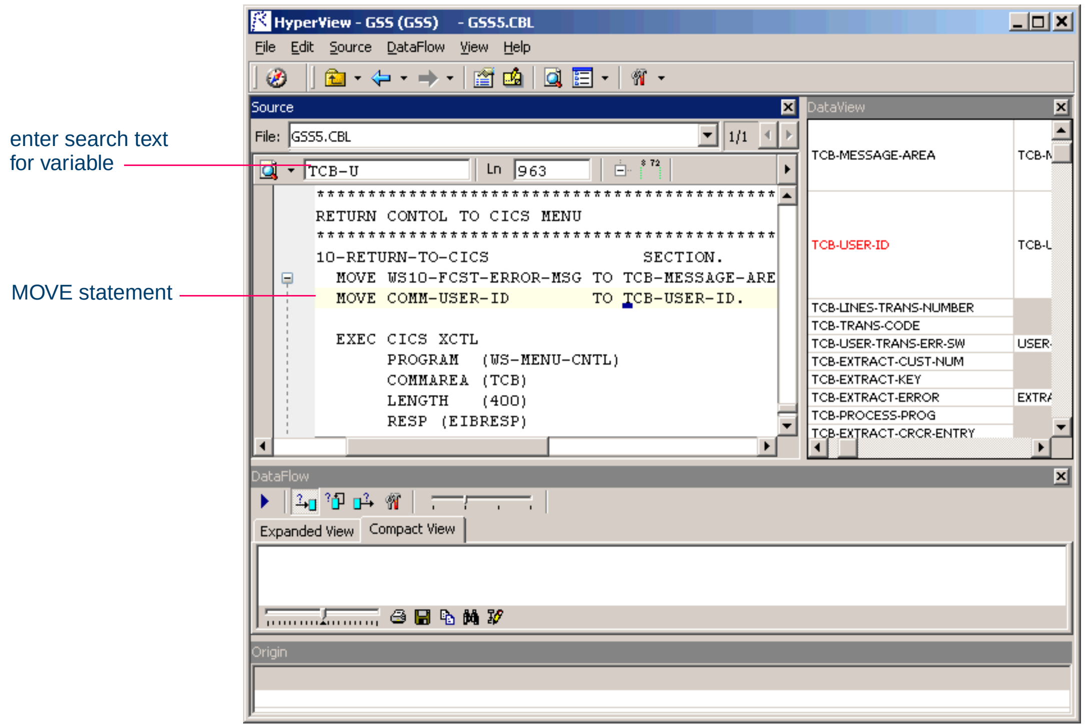
Understanding Global Data Flows

Suppose your task is to learn how data flows to and from the variable TCB-USER-ID in the GSS5 program in the GSS project. The Global Data Flow tool offers four views of a project that together let you perform low-level analysis of application data flows. Let's look at how you use these views to analyze the TCB-USER-ID variable.

Searching for the Variable

When you open the Global Data Flow tool in HyperView, the Source pane displays the source code for the selected program. In the text field next to the button on the tool bar, enter the name of the variable in our example, TCB-USER-ID, to navigate to the variable in the source code (Figure 3-1). We can see that a MOVE statement assigns the value of the COMM-USER-ID variable to TCB-USER-ID, and that the Data View pane highlights the variable in red.

Figure 3-1 *Global Data Flow Tool: Searching for a Variable*


enter search text for variable
MOVE statement
HyperView - GSS (GSS) - GSS5.CBL
File Edit Source DataFlow View Help
Source
File: GSS5.CBL
1/1
TCB-U
Ln 963
DataView
DataFlow
Expanded View
Compact View
Origin

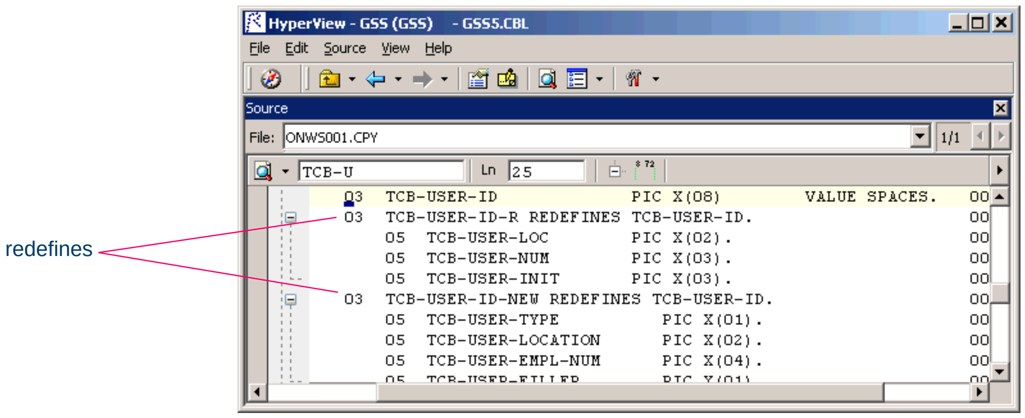
```
*********************************************
RETURN CONTOL TO CICS MENU
*********************************************
10-RETURN-TO-CICS                 SECTION.
  MOVE WS10-FCST-ERROR-MSG TO TCB-MESSAGE-ARE
  MOVE COMM-USER-ID        TO TCB-USER-ID.

  EXEC CICS XCTL
       PROGRAM  (WS-MENU-CNTL)
       COMMAREA (TCB)
       LENGTH   (400)
       RESP (EIBRESP)
```

TCB-MESSAGE-AREA	TCB-M
TCB-USER-ID	TCB-U
TCB-LINES-TRANS-NUMBER	
TCB-TRANS-CODE	
TCB-USER-TRANS-ERR-SW	USER-
TCB-EXTRACT-CUST-NUM	
TCB-EXTRACT-KEY	
TCB-EXTRACT-ERROR	EXTRA
TCB-PROCESS-PROG	
TCB-EXTRACT-CRCR-ENTRY	

Viewing Memory Usage for the Variable

Although we have located a reference to TCB-USER-ID in the GSS5 source code, we still don’t know where the variable is declared. Click TCB-USER-ID in the Data View pane to display the copybook source for the declaration (Figure 3-2).

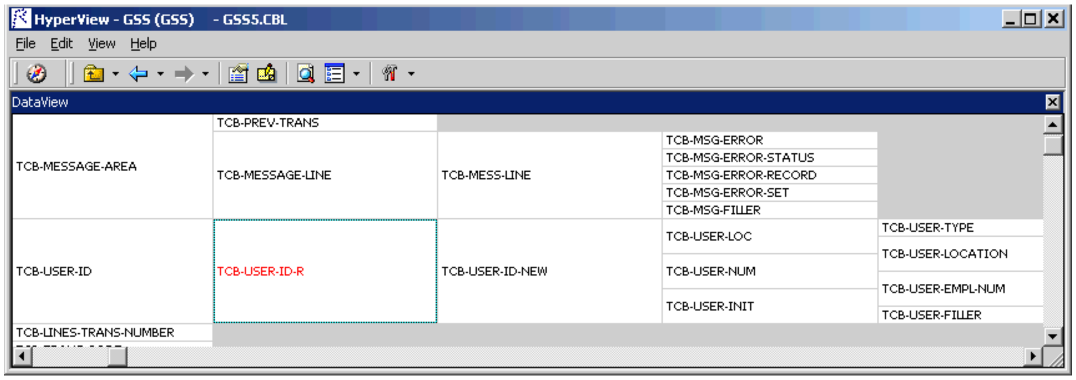
Figure 3-2 *Global Data Flow Tool: Viewing Copybook Source*



```
03  TCB-USER-ID                 PIC X(08)        VALUE SPACES.    00
03  TCB-USER-ID-R REDEFINES TCB-USER-ID.                          00
    05  TCB-USER-LOC            PIC X(02).                        00
    05  TCB-USER-NUM            PIC X(03).                        00
    05  TCB-USER-INIT           PIC X(03).                        00
03  TCB-USER-ID-NEW REDEFINES TCB-USER-ID.                        00
    05  TCB-USER-TYPE               PIC X(01).                    00
    05  TCB-USER-LOCATION           PIC X(02).                    00
    05  TCB-USER-EMPL-NUM           PIC X(04).                    00
    05  TCB-USER-FILLER             PIC X(01)                     00
```

We can see from the source that TCB-USER-ID is a structure that is redefined by the structures TCB-USER-ID-R and TCB-USER-ID-NEW. Let’s take a closer look at these structures in the Data View pane (Figure 3-3).

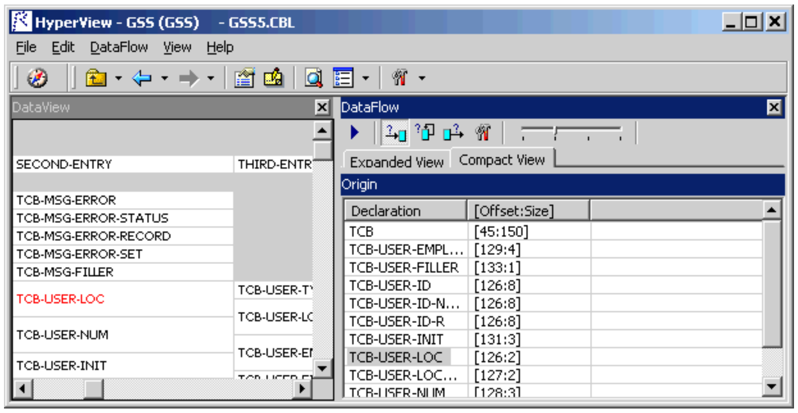
Figure 3-3 *Global Data Flow Tool: Viewing Memory Usage*



The Data View pane provides a visual representation of memory usage by the redefining structures. We can see which fields in TCB-USER-ID-NEW (in the fifth column) share memory locations with fields in TCB-USER-ID-R (in the fourth column).

Knowing this, we can be alert to the possibility that changing one of these variables might affect another variable at the same location. If we double-click TCB-USER-ID in the Data View pane, we can pinpoint the memory locations in the Origin pane, which shows the offset and size of each variable in both structures (Figure 3-4).

Figure 3-4 *Global Data Flow Tool: Viewing Memory Locations*



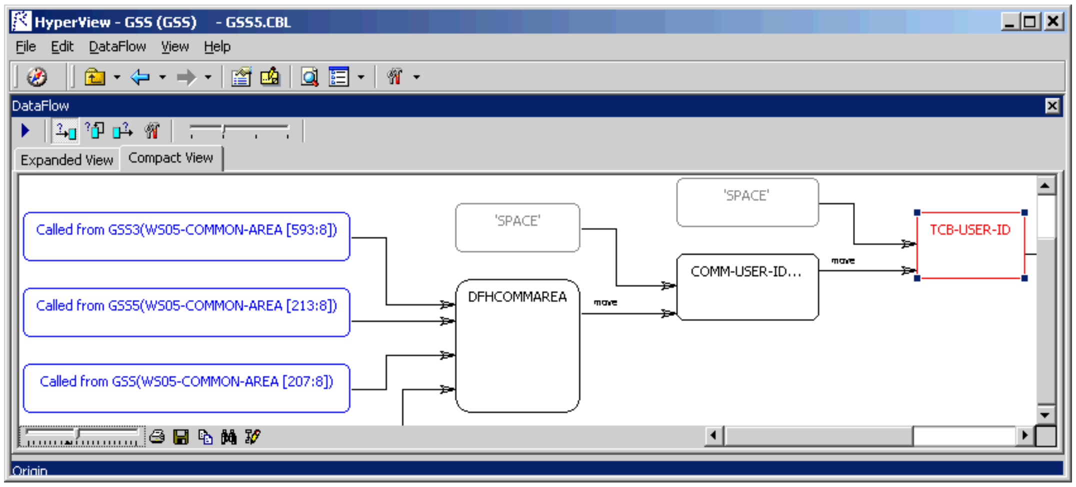
Declaration	[Offset:Size]
TCB	[45:150]
TCB-USER-EMPL...	[129:4]
TCB-USER-FILLER	[133:1]
TCB-USER-ID	[126:8]
TCB-USER-ID-N...	[126:8]
TCB-USER-ID-R	[126:8]
TCB-USER-INIT	[131:3]
TCB-USER-LOC	[126:2]
TCB-USER-LOC...	[127:2]
TCB-USER-NUM	[128:3]

Viewing the Data Flow for the Variable

Double-click TCB-USER-ID in the Data View pane to display a diagram that shows how data flows to and from the variable in the GSS5 program (Figure 3-5). The diagram traces the incoming and outgoing data flows in the program up to a *dataport,* an I/O statement or a call to or from another program.

At the dataport for the call from GSS3, for example, we can trace the COMM-USER-ID field to an 8-byte variable at memory offset 593 in the WS-05-COMMON-AREA structure. When we select the dataport and choose **Reconstruct** in the **Diagram** menu, the Global Data Flow tool displays the data flow diagram for COMM-USER-ID in the GSS3 program.

Figure 3-5 *Global Data Flow Tool: Viewing Program Data Flow*


HyperView - GSS (GSS) - GSS5.CBL
File Edit DataFlow View Help
DataFlow
Expanded View
Compact View
Called from GSS3(WS05-COMMON-AREA [593:8])
Called from GSS5(WS05-COMMON-AREA [213:8])
Called from GSS(WS05-COMMON-AREA [207:8])
'SPACE'
'SPACE'
DFHCOMMAREA
move
COMM-USER-ID...
move
TCB-USER-ID
Origin

Using the Global Data Flow Tool

You can start the Global Data Flow tool as a standalone or HyperView-based tool. Start the standalone tool by selecting a project in the Repository Browser and choosing **Data Flow** in the **Analyze** menu. Start the HyperView-based tool as described below.

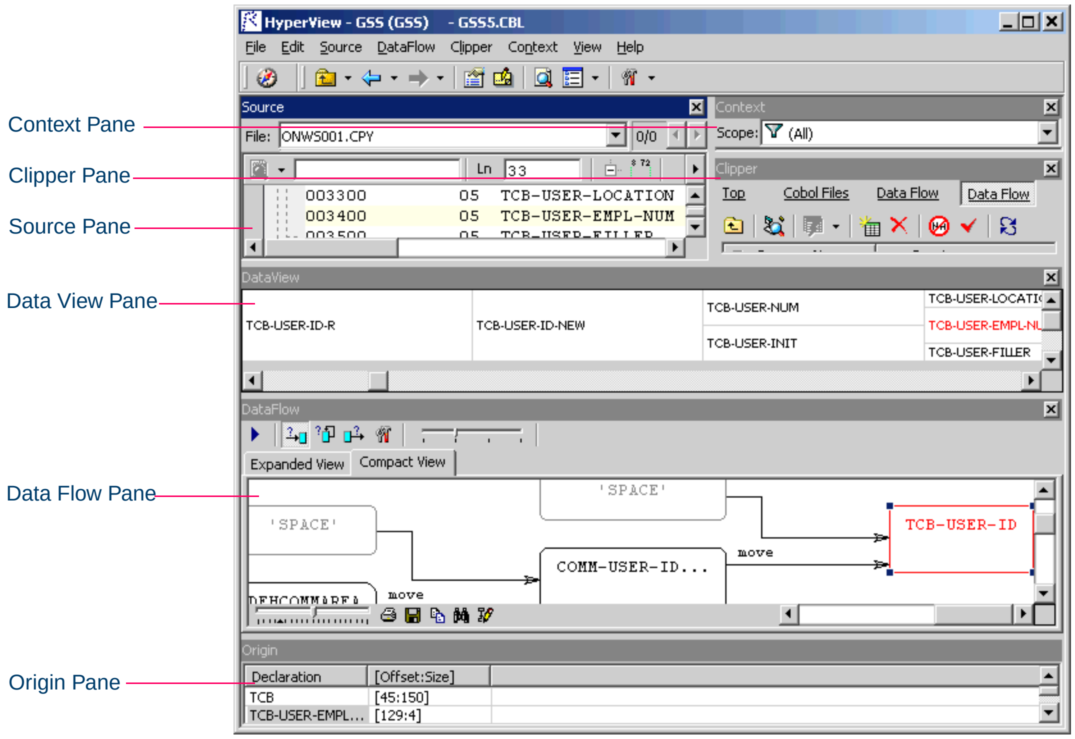
Generating Global Data Flow Information

Follow the steps below to start the Global Data Flow tool in HyperView.

To generate global data flow information in HyperView:

1 In the Repository Browser, select the program you want to analyze and choose **Interactive Analysis** in the **Analyze** menu. The HyperView window opens. In the **View** menu, choose **Data Flow**. The Global Data Flow tool opens (Figure 3-6).

Figure 3-6 *Global Data Flow Tool Window*



```
003300          05  TCB-USER-LOCATION
003400          05  TCB-USER-EMPL-NUM
003500          05  TCB-USER-FILLER
```

Declaration	[Offset:Size]
TCB	[45:150]
TCB-USER-EMPL...	[129:4]

2 In the Source pane, use the Quick Search field next to the button on the tool bar to navigate to the variable you want to analyze. The Global Data Flow tool highlights the variable name in the Source pane and displays the name in red in the Data View pane. Click the variable name in the Data View pane to display the source code for the variable declaration in the Source pane.

3 The Data View pane shows variable structures, substructures, and fields in hierarchical order from left to right. Double-click a variable name to generate a data flow diagram for the variable in the Data Flow pane, and a list of offsets and memory allocations for the variable and any related variables in the Origin pane.

4 Select a dataport in the Data Flow pane and choose **Reconstruct** in the **Data Flow** menu to display the data flow diagram for the vari-

able identified at the dataport. For more information on dataports, see “Viewing the Data Flow for the Variable” on page 3-4.

Working with Global Data Flow Information

The window for the Global Data Flow tool consists of a Source pane, Data View pane, Data Flow pane (which includes the Origin pane), Context pane, Clipper pane, and Activity Log. You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Note: For Context and Clipper pane usage, see *Analyzing Programs* in the workbench documentation set.

Source Pane

The Source pane lets you browse the source code for the program selected in the Global Diagram pane or the variable selected in the Data View or Data Flow panes. For usage information, see *Analyzing Programs* in the workbench documentation set.

Data View Pane

For the program selected in the Source pane, the Data View pane shows variable structures, substructures, and fields in hierarchical order from left to right. The selected variable is displayed in red.

Double-click a variable name to generate a data flow diagram for the variable in the Data Flow pane, and a list of offsets and memory allocations for the variable and any related variables in the Origin pane.

Origin Pane

For the variable selected in the Data View or Data Flow panes, the Origin pane displays a list of offsets and memory allocations for the variable and any related variables. For the relationship selected in the Data Flow pane, it lists all the statements that determine the data flow between the related variables. For the dataport selected in the Data Flow pane, it displays related variables.

Data Flow Pane

The Data Flow pane displays a data flow diagram for the variable selected in the Data View pane. The selected variable is displayed in red, constants in gray, and dataports in blue. Place your cursor over a variable for a moment to display a tool tip that identifies the memory offset and allocation for the variable.

Note: For tool bar usage, see Appendix B, “Common Diagramming Features.”

Select a dataport in the Data Flow pane and choose **Reconstruct** in the **Data Flow** menu to display the data flow diagram for the variable identified at the dataport. For more information on dataports, see “Viewing the Data Flow for the Variable” on page 3-4.

Tip: If the Data View pane is closed, you can generate the data flow diagram by selecting the variable name in the Source pane and choosing **Reconstruct** in the **Data Flow** menu.

Select a variable in the Data Flow pane to view its memory usage in the Data View and Origin panes and its source code in the Source pane. Select a relationship line in the Diagram pane to display all the statements that determine the data flow between variables in the Origin pane.

Note: For descriptions of the relationships between data items, see Table 5-1 in *Analyzing Programs* in the workbench documentation set.

In the **Diagram** menu, choose any combination of

- **Show Causes** to show data flows into the selected variable.
- **Show Consequences** to show data flows from the selected variable.
- **Show Self-Dependencies** to show recursive data flows for the selected variable.

Setting Global Data Flow Options

Use the Global Data Flow User Preferences window to edit the display in the Data View pane. Use the Global Data Flow Project Options window to edit the display in the Data Flow pane.

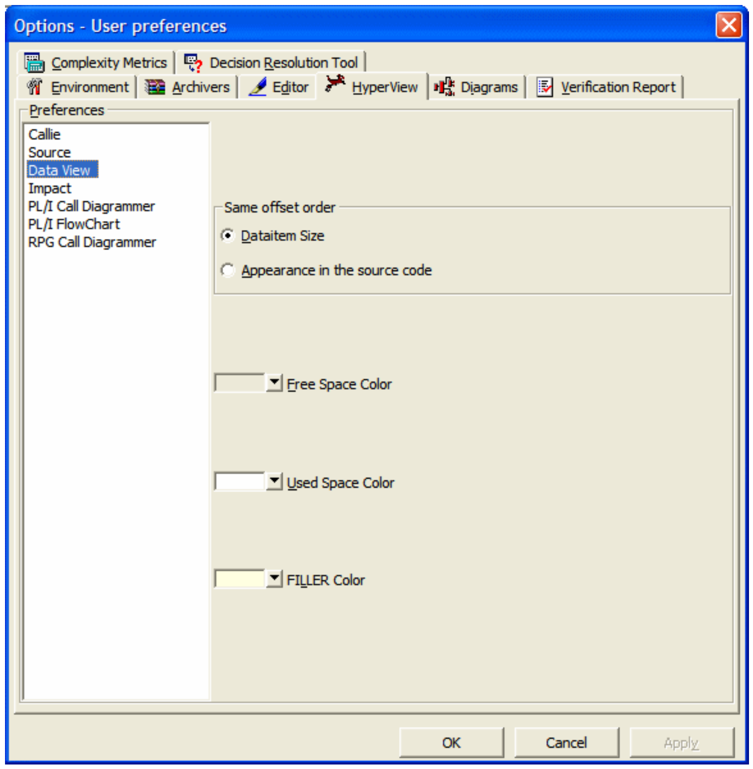
Setting Global Data Flow User Preferences

Global Data Flow User Preferences determine the order of display of variables with the same offset and the colors used in the Data View pane.

To set Global Data Flow User Preferences:

1 In the **View** menu, choose **User Preferences**. The User Preferences window opens. Click the **HyperView** tab. In the Preferences pane, click **Data View**. The Global Data Flow User Preferences window opens (Figure 3-8).

Figure 3-7 *Global Data Flow User Preferences Window*



Same offset order
[✓ Dataitem Size]
Appearance in the source code

Free Space Color
Used Space Color
FILLER Color

2 In the Same offset order group box, specify the order you want variables with the same offset to be displayed in the Data View pane. Choose:

- **Data item size** if you want variables with the same offset to be displayed in size order, largest to smallest.
- **Appearance in the source code** if you want variables with the same offset to be displayed in the order they appear in the source code.

3 The current default background color of the box representing free space in memory is displayed in the **Free Space Color** drop-down. The current background color of the box representing used space in memory is displayed in the **Used Space Color** drop-down. The current background color of the box representing filler is displayed in the **FILLER Color** drop-down.

Click the adjacent ▾ buttons to edit the color of the paragraphs. A standard Windows color control is displayed. Use the **Palette** tab to select the color from the Windows palette. Use the **System** tab to match the color with the color of standard Windows elements.

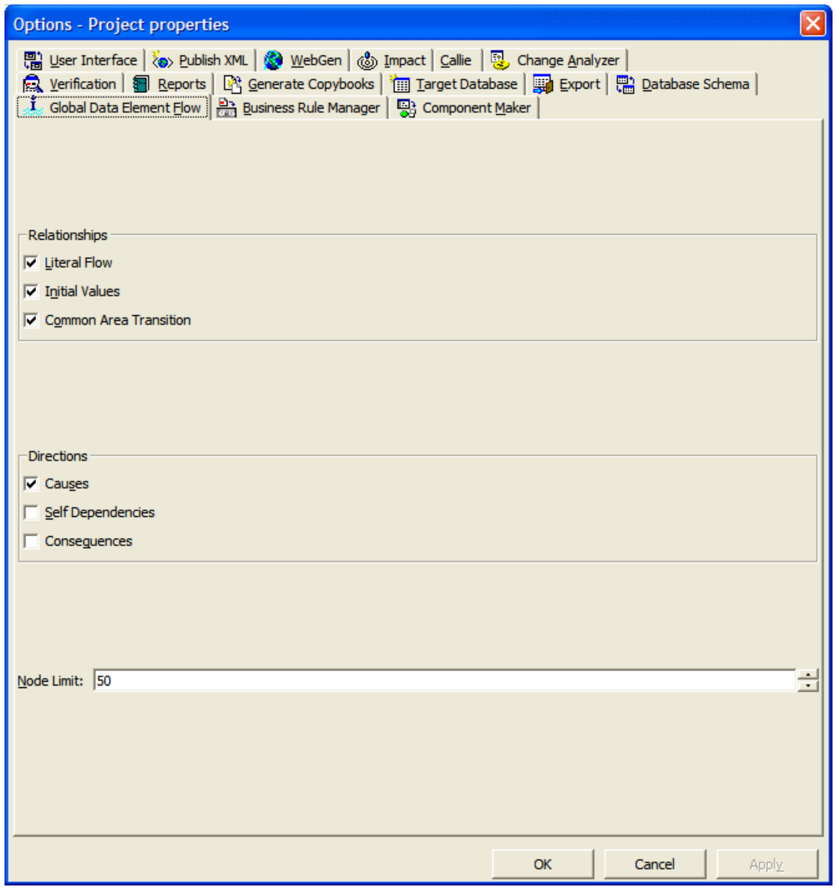
Setting Global Data Flow Project Options

Global Data Flow Project Options determine whether data flow diagrams include literals, variable values set at initialization, and Unisys Cobol common storage variables; whether they show incoming, outgoing, and recursive data flows; and the number of nodes they display.

To set Global Data Flow Project Options:

1 In the **View** menu, choose **Project Options**. The Project Options window opens. Click the **Global Data Flow** tab. The Global Data Flow Options window opens (Figure 3-8).

Figure 3-8 *Global Data Flow Options Window*



Relationships
- [✓] Literal Flow
- [✓] Initial Values
- [✓] Common Area Transition

Directions
- [✓] Causes
- [ ] Self Dependencies
- [ ] Consequences

Node Limit: 50

2 In the Relationships pane, choose any combination of:

- **Literal Flow** to include literals in the data flow diagram for the selected variable.
- **Initial Values** to include variable values set at initialization in the data flow diagram for the selected variable.
- **Common Area Transition** to include Unisys Cobol common storage variables in the data flow diagram for the selected vari-

able. Common storage variables are not explicitly declared in CALL statements.

Note: To include Unisys Cobol common storage variables, you must have verified the project with the **Perform Unisys Common-Storage Analysis** option set in the project verification options. For more information, see *Preparing Projects* in the workbench documentation set.

3 In the Directions pane, choose any combination of:

- **Causes** to generate data flows into the selected variable.
- **Consequences** to generate data flows from the selected variable.
- **Self-Dependencies** to show recursive data flows for the selected variable.

4 In the **Node Limit** combo box, enter the maximum number of diagram nodes you want to display. You might restrict the number of nodes to improve performance or make the diagram easier to read. You can also use the slider on the Node Limit tool bar to adjust the number of displayed nodes.

Note: All children of a parent node are displayed even if some of them exceed the specified maximum.

Copying Global Data Flow Diagrams to the Clipboard

Choose **Copy:Diagram** in the **Data Flow** menu to copy a global data flow diagram to the clipboard in EMF format. You can paste the diagram from the clipboard to a document in a third-party tool such as Word.

Printing Global Data Flow Diagrams

Choose **Print:Diagram** in the **Data Flow** menu to print a global data flow diagram. The Print Preview window opens, where you can change the zoom factor for the diagram to control the number of pages in the print job. Click **OK**.

Exporting Global Data Flow Diagrams

You can export global data flow diagrams to a variety of standard formats. Choose **Save Diagram** in the **Data Flow** menu to export a diagram to BED, bitmap, JPEG, Visio, Visio XML, DOT, or EMF.

Note: Visio 2002 must be installed to save a diagram to Visio. Visio 2002 is not required to save to Visio XML.

What's Next?

That's all you need to know to analyze program data flows in the Global Data Flow tool. Now let's look at how you perform low-level analysis of batch programs in the Batch Application Viewer. That's the subject of the next chapter.

Analyzing Batch Applications

4

The Batch Application Viewer lets you perform low-level analysis of batch processes. Use the Batch Application Viewer's rich set of tables and diagrams to determine if batch jobs are dependent on one another, the programs that use a data store, and the flow of data into or out of a data store.

Note: Batch Application Viewer analyzes JCL and ECL jobs. Usage is identical except for language-specific differences in terminology.

Using the Batch Application Viewer

You generate batch application information for the current project. The information is presented in linked table and diagram views. The tables cross-reference jobs, steps, procedures and programs, and data stores. The diagrams show the relationships between jobs, procedures and programs, and data stores.

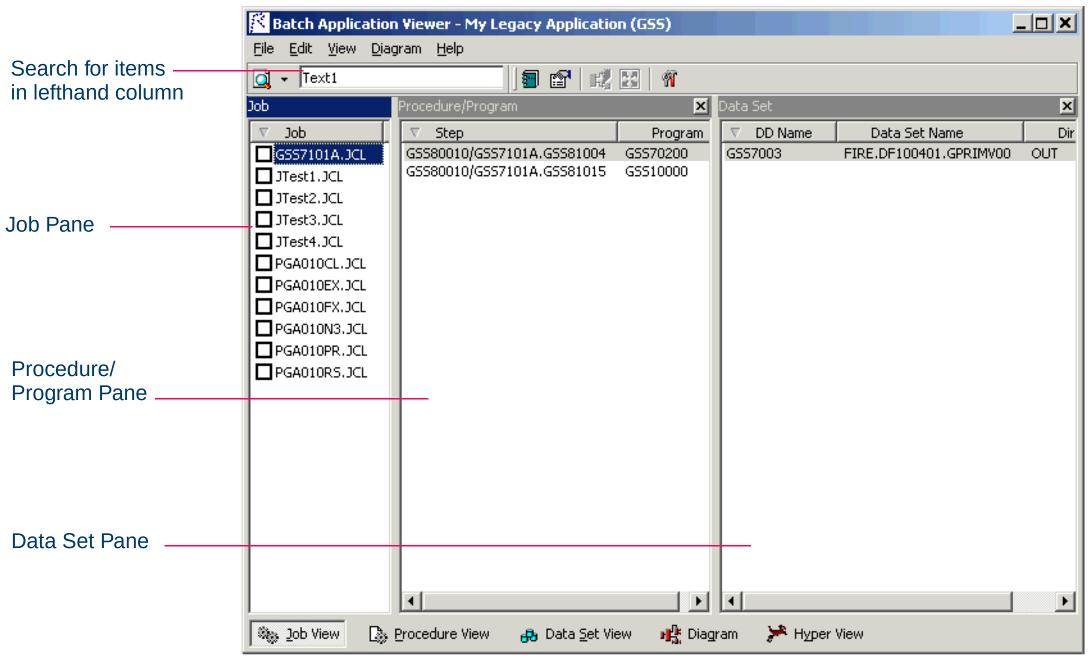
Generating Batch Application Information

Follow the steps below to generate batch application information.

To generate batch application information:

1 In the Repository Browser, select the project you want to analyze and choose **Batch Application** in the **Analyze** menu. The Batch Application Viewer window opens (Figure 4-1).

Figure 4-1 *Batch Application Viewer Window (Job View)*



Job
GSS7101A.JCL
JTest1.JCL
JTest2.JCL
JTest3.JCL
JTest4.JCL
PGA010CL.JCL
PGA010EX.JCL
PGA010FX.JCL
PGA010N3.JCL
PGA010PR.JCL
PGA010RS.JCL

Step	Program
GSS80010/GSS7101A.GSS81004	GSS70200
GSS80010/GSS7101A.GSS81015	GSS10000

DD Name	Data Set Name	Dir
GSS7003	FIRE.DF100401.GPRIMV00	OUT

2 In the Job pane, select a job to view the steps the job uses, the procedure or program each step executes, and the data stores each procedure or program reads from or writes to.

Tip: Use the search facility described in "Searching Batch Application Table Views" on page 4-7 to locate jobs.

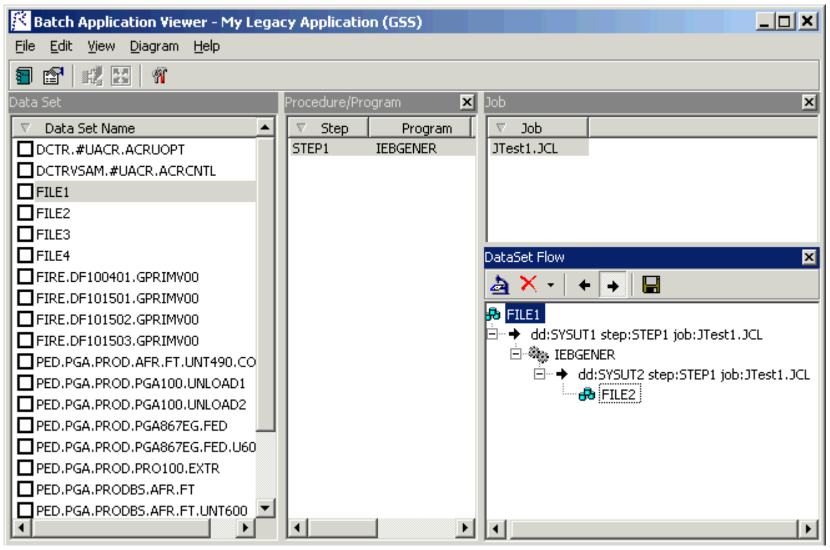
3 Click **Procedure View** at the bottom of the window to open the procedure view. In the Procedure/Program pane, select a procedure or program to view the jobs and steps that execute it, and the data stores programs read from or write to. For programs executed by procedures, Batch Application Viewer displays data stores referenced by DD statements added after the substitution of the procedure.

4 Click **Dataset View** at the bottom of the window to open the dataset view. In the Dataset pane, select a dataset to view the procedure or program that reads or writes to it, the step that executes the procedure or program, and the job that contains the step.

5 When you select the dataset view, Batch Application Viewer displays the Data Set Flow pane in the bottom right part of the window. Select a dataset in the Data Set pane and a step in the Procedure/Program pane, then click the button in the Data Set Flow pane to view the flow of data into or out of the selected dataset (Figure 4-2).

Click the ← button in the Data Set Flow pane to view the flow of data into the dataset, click the → button to view the flow of data out of the dataset.

Figure 4-2 *Batch Application Viewer Window (Data Set Flow)*


Batch Application Viewer - My Legacy Application (GSS)
File Edit View Diagram Help
Data Set
Procedure/Program
Job
DataSet Flow

Data Set Name
DCTR.#UACR.ACRUOPT
DCTRVSAM.#UACR.ACRCNTL
FILE1
FILE2
FILE3
FILE4
FIRE.DF100401.GPRIMV00
FIRE.DF101501.GPRIMV00
FIRE.DF101502.GPRIMV00
FIRE.DF101503.GPRIMV00
PED.PGA.PROD.AFR.FT.UNT490.CO
PED.PGA.PROD.PGA100.UNLOAD1
PED.PGA.PROD.PGA100.UNLOAD2
PED.PGA.PROD.PGA867EG.FED
PED.PGA.PROD.PGA867EG.FED.U60
PED.PGA.PROD.PRO100.EXTR
PED.PGA.PRODBS.AFR.FT
PED.PGA.PRODBS.AFR.FT.UNT600

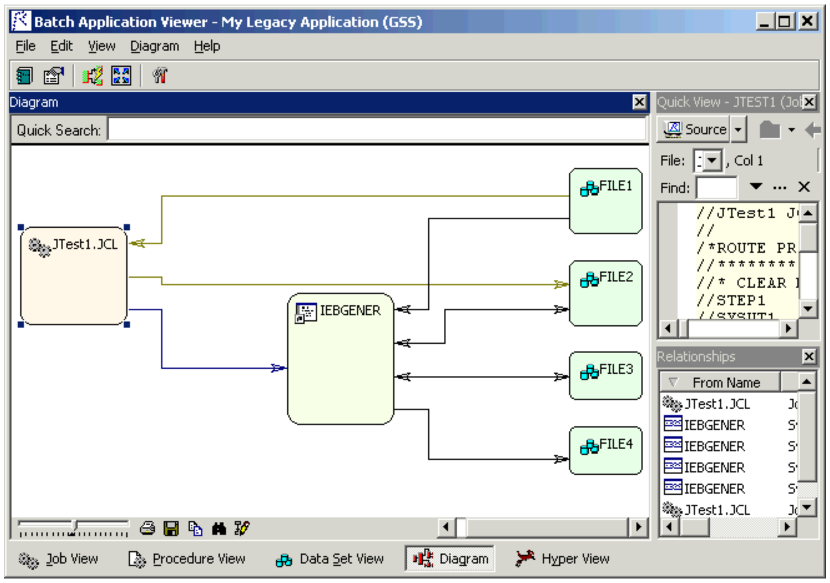
Step	Program
STEP1	IEBGENER

Job
JTest1.JCL

```
FILE1
⊟→ dd:SYSUT1 step:STEP1 job:JTest1.JCL
   ⊟ IEBGENER
      ⊟→ dd:SYSUT2 step:STEP1 job:JTest1.JCL
            FILE2
```

6 In any view, place a check mark next to one or more objects and click **Diagram** at the bottom of the window to display a diagram of batch application objects related to the checked objects. The diagram shows the relationships between jobs, programs or procedures, and data stores (Figure 4-3). For tool bar usage, see Appendix B, “Common Diagramming Features.”

Figure 4-3 *Batch Application Viewer Window (Diagram View)*



```
//JTest1 J
//
/*ROUTE PR
//********
//* CLEAR I
//STEP1
```

From Name	
JTest1.JCL	J
IEBGENER	S
IEBGENER	S
IEBGENER	S
IEBGENER	S
JTest1.JCL	J

7 In any table view or diagram, select an object and click **HyperView** at the bottom of the window to view the object in HyperView. For procedures or data stores referenced by procedures, HyperView navigates to the object in the source code for the procedure. Otherwise, it navigates to the object in the JCL file. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

Working with Batch Application Table Views

Batch Application Viewer offers three table views of a batch application:

- The *Job View* cross-references the selected job with the steps the job contains, the procedure or program each step executes, and the DD and dataset names of the files each procedure or program reads from or writes to. Click **Job View** at the bottom of the Batch Application Viewer window to open the job view.
- The *Procedure View* cross-references the selected procedure or program with the jobs and steps that execute it, and the selected program with the DD and dataset names of the files the program reads from or writes to. Click **Procedure View** at the bottom of the Batch Application Viewer window to open the procedure view.

Note: For programs executed by procedures, Batch Application Viewer displays the DD and dataset names of files referenced by DD statements added after the substitution of the procedure.

- The *Dataset View* cross-references the selected dataset with the procedure or program that reads or writes to it, the step that executes the procedure or program, and the job that contains the step. Click **Dataset View** at the bottom of the Batch Application Viewer window to open the dataset view.

Note: All batch application table views consist of a Job pane, Program/Procedure pane, and Data Set pane. For the data set view, Batch Application Viewer also displays a Data Set Flow pane, where you can investigate the flow of data into or out of a data store.

You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Sorting Entries Click a column heading in a table view pane to sort the entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Selecting Objects Click an object in a table view pane to select it. To select a range of objects, hold down the Shift key, click the first object in the range, then click the last object in the range. To select objects that are not in a range, hold down the Control key, then click each object you want to select.

Selecting Entries to Diagram Place a check mark next to an object and click **Diagram** at the bottom of the Batch Application Viewer window to display a diagram of batch application objects related to the object.

To place a check mark next to all the objects in a table view pane, choose **Check All** in the **Edit** menu. To uncheck all the selected objects, choose **Uncheck All** in the **Edit** menu.

To place a check mark next to selected objects, select the objects and choose **Check Selected** in the **Edit** menu. To uncheck all the objects, choose **Uncheck Selected** in the **Edit** menu.

Viewing the Flow of Data into or out of a Dataset Select a dataset in the Data Set pane and a step in the Procedure/Program pane, then click the button in the Data Set Flow pane to view the flow of data into or out of the selected dataset (Figure 4-2). Click the left arrow in the Data Set Flow pane to view the flow of data into the dataset, click the right arrow to view the flow of data out of the dataset.

Viewing Properties Select an object in a table view pane and choose **Properties** in the **View** menu to display a set of tabs with object properties. For usage information, see *Getting Started* in the Modernization Workbench document set.

Exporting a Table View Report Open a table view and choose **Report** in the **File** menu to export a printed report on the view to HTML, Excel, RTF, Word, or formatted text.

Searching Batch Application Table Views

Use the Batch Application Viewer search facility to locate objects in the lefthand column of table views. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

For simple searches, enter the text for the search in the field next to the button on the tool bar. Batch Application Viewer locates text matches as you type. Click the button or choose **Search** in the drop-down menu under the adjacent ▾ button to find the next matching object. Choose **Find All** in the drop-down menu to find all matching objects.

For wildcard searches, enter the text for the search in the field next to the button and choose **Wild Search** in the drop-down menu. Choose **Wild Search** again to navigate to the next matching object. Choose **Wild Find All** in the drop-down menu to find all matching objects.

Working with Batch Application Diagrams

Batch application diagrams show the relationships between jobs, procedures and programs, and data stores (Figure 4-3). For tool bar usage, see Appendix B, "Common Diagramming Features."

Generating a Batch Application Diagram

Follow the steps below to generate a batch application diagram.

To generate a batch application diagram:

1 In the appropriate table view pane, place a check mark next to the objects you want to diagram, as described in "Selecting Entries to Diagram" on page 4-6.

2 In the **Diagram** menu, choose the relationships you want to view in the diagram:

- **Job Dependencies** show dependencies between jobs. Jobs are regarded as dependent if one writes to a dataset and the other reads from the same dataset.
- **Data Set Usage** shows interactions with datasets.
- **JCL Procedures Usage** shows interactions with procedures.
- **Program Usage** shows interactions with programs.

Tip: Use these choices to control the level of detail in the diagram. The choice is grayed out with a check mark beside it if the relationship must be displayed in the diagram. You can also set these choices after you display the diagram.

3. Edit the color scheme for the relationships if necessary, as described in "Editing the Color Scheme for Batch Application Diagrams" on page 4-8.
4. Click **Diagram** at the bottom of the Batch Application Viewer window. The diagram is displayed in the Diagram pane.

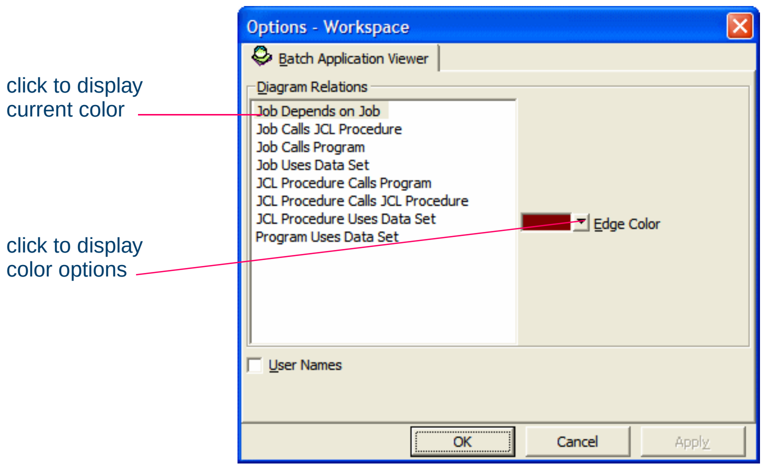
Editing the Color Scheme for Batch Application Diagrams

Diagram objects are color-coded based on the color scheme set in the Diagrammer tool described in Chapter 2, "Analyzing Relationship Flows." Relationship lines are color-coded based on your settings in the Batch Application View Options window.

To edit the color scheme for relationships in batch diagrams:

1. In the **View** menu, choose **Options**. The Options window opens (Figure 4-4).
2. In the Diagram Relations pane, click the relationship type whose color you want to edit. The current color of the type is displayed in the **Edge Color** drop-down.
3. Click the arrow beside the drop-down to display a standard Windows color control. Use the Palette tab to select the color of the relationship type from the Windows palette. Use the System tab to match the color of the relationship type with the color of standard Windows elements.

Figure 4-4 *Options Window*



Job Depends on Job
Job Calls JCL Procedure
Job Calls Program
Job Uses Data Set
JCL Procedure Calls Program
JCL Procedure Calls JCL Procedure
JCL Procedure Uses Data Set
Program Uses Data Set

Creating Job Dependencies in Batch Application Diagrams

Batch Application Viewer treats jobs as dependent if one writes to a dataset and the other reads from the same dataset. Occasionally, you may want to define dependencies between jobs based on other criteria: administrative needs such as scheduling, for example. Batch Application Viewer modifies the diagram only, not the repository.

Defining Job Dependencies To define dependencies between jobs in a batch application diagram, hold down the Alt key, select either job, and drag-and-drop the relationship to the other job. The relationship is displayed in red with three plus (+) signs at either end of the relationship line.

Exporting and Importing Job Dependencies To export job dependencies to a file, for import to diagrams for other projects, choose **Export Dependency Changes** in the **File** menu. A Save As dialog opens, where you can specify the name and folder for the export file.

To import job dependencies from a file, choose **Import Dependency Changes** in the **File** menu. An Open dialog appears, where you can select the file you want to import.

Removing Job Dependencies To remove all user-defined dependencies from a batch application diagram, choose **Clean Up Dependencies** in the **File** menu.

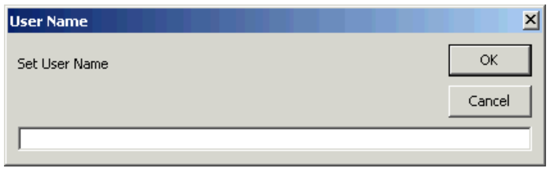
Tip: In certain circumstances (jobs that use cataloged file generation, for example), Batch Application Viewer may create *false dependencies*. You must remove these relationships manually by right-clicking the dependency and choosing **Remove** from the pop-up menu.

Creating User Names for Objects in Batch Application Diagrams

You can give diagram objects friendlier names, called *user names.* Batch Application Viewer modifies the diagram only, not the repository.

Creating and Displaying User Names To create a user name for an object in a batch application diagram, select the object and choose **User Name** in the **View** menu. The User Name dialog opens (Figure 4-5). Enter the user name in the text field and click **OK**. To display user names in a diagram, select **User Names** in the Batch Application Viewer Options window (Figure 4-4).

Figure 4-5 *User Name Dialog*



Exporting and Importing User Names To export user names to a file, for import to diagrams for other projects, choose **Export User Names** in the **File** menu. A Save As dialog opens, where you can specify the name and folder for the export file.

To import user names from a file, choose **Import User Names** in the **File** menu. An Open dialog appears, where you can select the file you want to import.

Removing User Names To remove all user names from a batch application diagram, choose **Clean Up User Names** in the **File** menu.

Copying Batch Application Diagrams to the Clipboard

Choose **Copy to Clipboard** in the **Diagram** menu to copy a batch application diagram to the clipboard in EMF format. You can paste the diagram from the clipboard to a document in a third-party tool such as Word.

Printing Diagrams

Choose **Print** in the **Diagram** menu to print a diagram. The Print Preview window opens, where you can change the zoom factor for the diagram to control the number of pages in the print job. Click **OK**.

Exporting Diagrams

You can export diagrams to a variety of standard formats. Choose **Save** in the **File** menu to export a diagram to BED, bitmap, JPEG, Visio, Visio XML, DOT, or EMF. If you choose **Save Multiple Pages** for a bitmap or JPEG file, a dialog opens where you can specify the size and zoom factor for the diagram.

Note: Visio 2002 must be installed to save a diagram to Visio. Visio 2002 is not required to save to Visio XML.

Working with HyperView in Batch Application Viewer

Select an object in a table view or diagram and click **HyperView** at the bottom of the Batch Application Viewer window to view the object in HyperView. For procedures or data stores referenced by procedures, HyperView navigates to the object in the source code for the procedure. Otherwise, it navigates to the object in the JCL file. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

What’s Next?

Now that you know how to analyze batch applications in the Batch Application Viewer, let’s look at how you analyze data operations in the workbench CRUD Report and the IMS Port Analysis feature. That’s the subject of the next chapter.

Analyzing Data Operations 5

The workbench parser generates relationships that show the data operations a program performs (Insert, Read, Update, or Delete) and the data objects on which the program operates (files, segments, tables, etc.). This chapter looks at a report that lets you view these relationships in a convenient format, and a mechanism that lets you trace PSB usage through an entire IMS application call sequence.

Viewing CRUD Reports

The CRUD Report for a project shows the data operations each program in the project performs, and the data objects on which the programs operate.

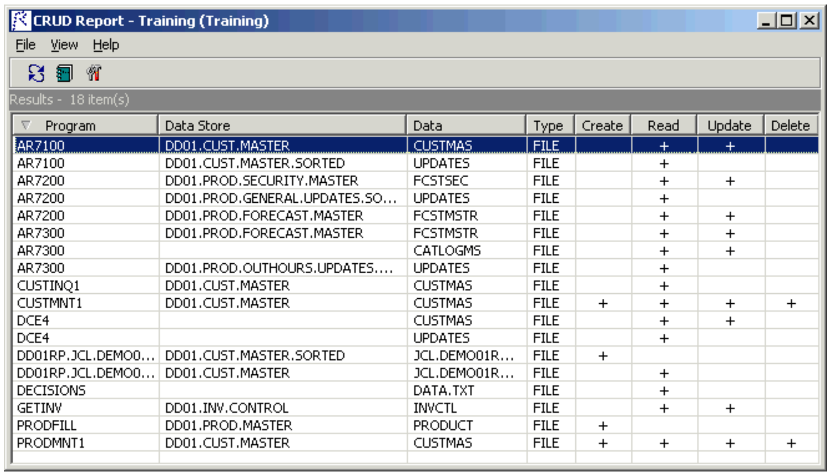
Note: The IMS data column of the CRUD report behaves differently from the columns for other data types. What appears in the IMS data column cells depends on what can be determined. If the segment can be determined, the cell is populated with the

PSB name and segment name. Otherwise, the segment name is left blank. The format is xxxxxx.yyyyyyy, where xxxxxx is the PSB name and yyyyyyy is the segment name or blank if the segment cannot be determined.

To generate a CRUD report:

1 In the Repository Browser, select a project and choose **CRUD Report** in the **Prepare** menu. The CRUD Report window opens (Figure 5-1).

Figure 5-1 *CRUD Report*



Program	Data Store	Data	Type	Create	Read	Update	Delete
AR7100	DD01.CUST.MASTER	CUSTMAS	FILE		+	+	
AR7100	DD01.CUST.MASTER.SORTED	UPDATES	FILE		+		
AR7200	DD01.PROD.SECURITY.MASTER	FCSTSEC	FILE		+	+	
AR7200	DD01.PROD.GENERAL.UPDATES.SO...	UPDATES	FILE		+		
AR7200	DD01.PROD.FORECAST.MASTER	FCSTMSTR	FILE		+	+	
AR7300	DD01.PROD.FORECAST.MASTER	FCSTMSTR	FILE		+	+	
AR7300		CATLOGMS	FILE		+	+	
AR7300	DD01.PROD.OUTHOURS.UPDATES....	UPDATES	FILE		+		
CUSTINQ1	DD01.CUST.MASTER	CUSTMAS	FILE		+		
CUSTMNT1	DD01.CUST.MASTER	CUSTMAS	FILE	+	+	+	+
DCE4		CUSTMAS	FILE		+	+	
DCE4		UPDATES	FILE		+		
DD01RP.JCL.DEMO0...	DD01.CUST.MASTER.SORTED	JCL.DEMO01R...	FILE	+			
DD01RP.JCL.DEMO0...	DD01.CUST.MASTER	JCL.DEMO01R...	FILE		+		
DECISIONS		DATA.TXT	FILE		+		
GETINV	DD01.INV.CONTROL	INVCTL	FILE		+	+	
PRODFILL	DD01.PROD.MASTER	PRODUCT	FILE	+			
PRODMNT1	DD01.CUST.MASTER	CUSTMAS	FILE	+	+	+	+

Tip: Click a column heading in the Results pane to sort the entries by that column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

2 To select the types of program-to-data object relationships to display, and the data operations to show for each type, choose **Options** in the **View** menu. The CRUD Report Options window opens (Figure 5-2).

3 In the CRUD Report Options window:

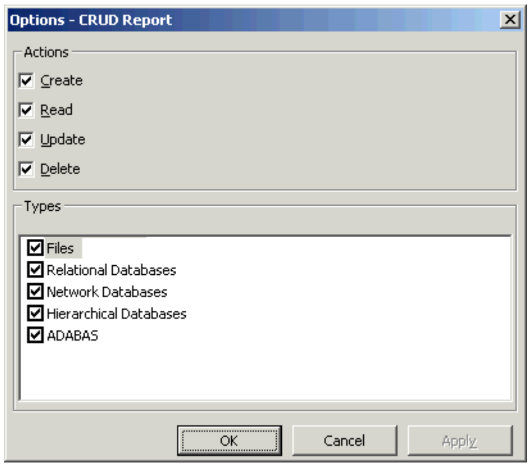
- Place a check mark next to each type of program-to-data object relationship you want to display.
- Place a check mark next to each data operation you want to display.

Click **Apply** if you want to save your settings without dismissing the CRUD Report Options window. Click **OK** if you want to save your settings and dismiss the CRUD Report Options window.

4 Choose **Refresh** in the File menu to refresh the report.

5 Choose **Report** in the **File** menu to export the report to HTML, Excel, RTF, Word, or formatted text.

Figure 5-2 *CRUD Report Options Window*


Options - CRUD Report

Actions
- [✓] Create
- [✓] Read
- [✓] Update
- [✓] Delete

Types
- [✓] Files
- [✓] Relational Databases
- [✓] Network Databases
- [✓] Hierarchical Databases
- [✓] ADABAS

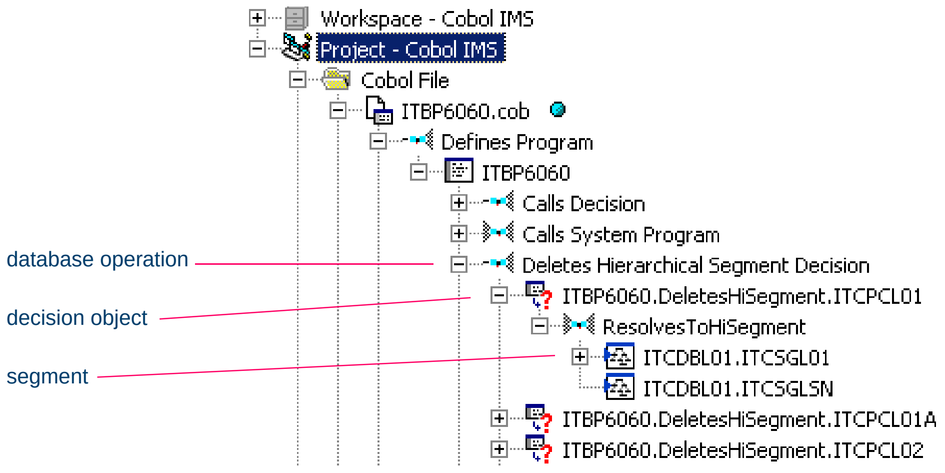
Enabling IMS Port Analysis

It is virtually impossible to determine from program code the database segments or screens an IMS program operates on. Only an application-wide analysis can trace PSB usage through the entire application call sequence.

To determine the types of database operation (insert, read, update, or delete) IMS programs perform, and to list in the browser each of the database segments or screens the operations are performed on, select the project or the individual source files for the programs and choose **IMS Analysis** in the **Prepare** menu.

Figure 5-3 shows typical results. The objects marked with the icon are abstract *decision objects,* indicating that the database operation, in this case, Deletes, has been resolved to multiple segments.

Figure 5-3 *IMS Port Analysis Results*



Mapping Root Programs to PSBs in JCL or System Definition Files

You must identify IMS “root programs” and corresponding PSBs in a JCL file for a batch application or a System Definition file for an online application. A root program is directly invoked by IMS with a list of PCBs as parameters. It can pass these PCBs as parameters in calls to other programs.

If you do not have actual JCL or System Definition files, you must create dummy ones. Analyzing the application without these files does nothing. Sample JCL and System Definition files follow:

```
Sample JCL file:
   //imsbatch JOB
   //S1    EXEC PGM=DFSRRC00,REGION=2048K,
   //
   PARM=(DLI,progname,psbname,7,0000,,0,,N,0,T,0,,N,,,N
   )
   //
Sample System Definition file:
   APPLCTN PSB=progname
   TRANSACT CODE=trnname
```

Verification Order for IMS Applications

If you verify an entire project for an IMS application, the workbench parses the source files in appropriate order, taking account of the dependencies between file types. Otherwise, verify source files in the following order:

- DBD files (for GSAM databases)
- MFS files
- PSB files
- Cobol or PL/I files

Note: For Cobol files, set the **Perform Program Analysis** and **Enable Data Element Flow** project verification options. For PL/I files, set the **Enable Data Element Flow** project verification option.

- JCL or System Definition files.

Reverifying Files in IMS Applications

If you reverify a root program, the JCL or System Definition file that maps the program to a PSB will be invalidated. If you reverify non-root programs, all call chains leading to them will be analyzed and any JCLs or System Definition files that start corresponding root programs will be invalidated. Make sure to reverify invalidated files.

Conversely, if you change a program-to-PSB mapping inside a JCL or System Definition file, or change the PSB file itself, make sure to reverify the mapped program before reverifying the JCL or System Definition file.

When you rerun IMS Analysis, it will process all complete call chains, starting from all reverified JCLs and System Definition files. You can limit the number of root programs that are re-analyzed in subsequent runs of IMS calls analysis by setting up System Definition files so that they reference one transaction only.

Note: If you rerun IMS Analysis without changing anything in the project, it will end with the warning “No information to perform IMS Analysis.” If you receive this message on the first run of IMS Analysis, make sure that all JCLs, System Definition files, and corresponding root programs have been verified, and that you have a call chain from root to every IMS-relevant program in the project (check for the strings “+IMSC” or “+IMSE” in the Environment attribute on the System tab of the properties for the program).

What’s Next?

That’s all you need to know to use the workbench CRUD report and IMS port analysis feature to analyze project data operations. Now let’s look at how you use the legacy estimation tools to estimate project complexity and effort.

Estimating Complexity and Effort

6

Suppose you are planning to implement a change request for a program and want to know how long it will take to complete the change. Modernization Workbench Legacy Estimation tools let you compare programs based on weighted values for selected complexity metrics. Based on the comparison, you can develop a credible estimate of the time required to make the requested change.

The complexity metrics used in the calculation are a combination of industry standard and Modernization Workbench-generated statistics (Table 6-1). If your own analysis shows that a given program is more or less complex than the weighted calculation would suggest, you can set a *change magnitude* to override the calculated value for the program.

The program might have thousands of source lines, for example, increasing its calculated complexity, while actually being very easy to modify. When you use a change magnitude, your “subjective” estimate of the effort involved, Small, Medium, Large, Extra Large, becomes an input to the effort calculation, along with the weighted values.

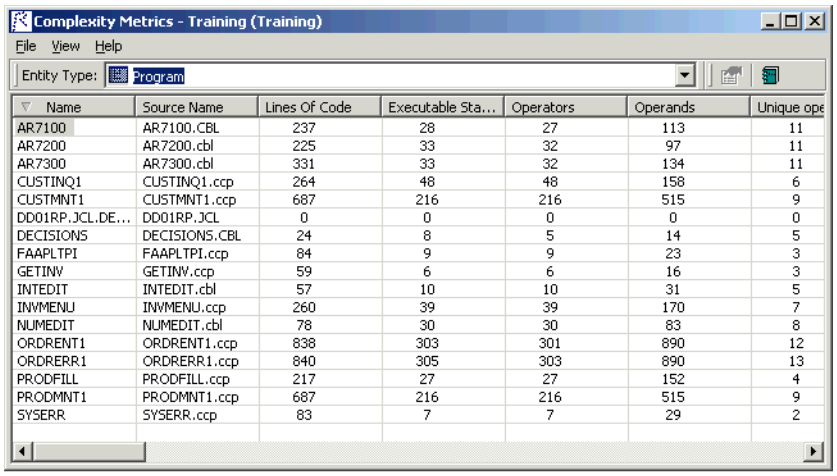
Using the Complexity Metrics Tool

Use the Complexity Metrics tool to compare raw complexity values for the objects in your project.

To generate complexity metrics:

1. In the Repository Browser, select the project whose complexity metrics you want to calculate and choose **Complexity** in the **Analyze** menu. A blank Complexity Metrics window opens.
2. In the **Entity Type** drop-down, choose the type of object you want to calculate complexity metrics for. Figure 6-1 shows the results for the program object. The metrics are described in Table 6-1.

Figure 6-1 *Complexity Metrics Window*



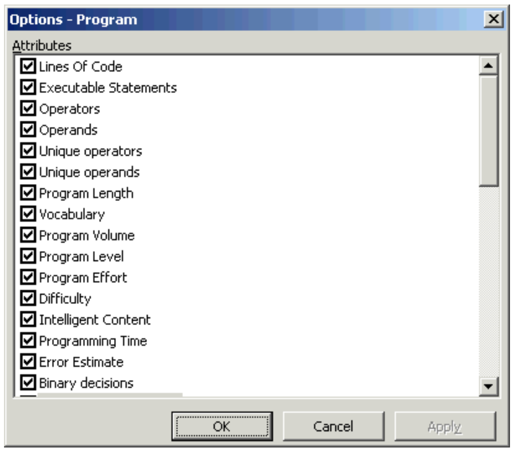
Entity Type: Program

Name	Source Name	Lines Of Code	Executable Sta...	Operators	Operands	Unique ope
AR7100	AR7100.CBL	237	28	27	113	11
AR7200	AR7200.cbl	225	33	32	97	11
AR7300	AR7300.cbl	331	33	32	134	11
CUSTINQ1	CUSTINQ1.ccp	264	48	48	158	6
CUSTMNT1	CUSTMNT1.ccp	687	216	216	515	9
DD01RP.JCL.DE...	DD01RP.JCL	0	0	0	0	0
DECISIONS	DECISIONS.CBL	24	8	5	14	5
FAAPLTPI	FAAPLTPI.ccp	84	9	9	23	3
GETINV	GETINV.ccp	59	6	6	16	3
INTEDIT	INTEDIT.cbl	57	10	10	31	5
INVMENU	INVMENU.ccp	260	39	39	170	7
NUMEDIT	NUMEDIT.cbl	78	30	30	83	8
ORDRENT1	ORDRENT1.ccp	838	303	301	890	12
ORDRERR1	ORDRERR1.ccp	840	305	303	890	13
PRODFILL	PRODFILL.ccp	217	27	27	152	4
PRODMNT1	PRODMNT1.ccp	687	216	216	515	9
SYSERR	SYSERR.ccp	83	7	7	29	2

Tip: Click a column heading to sort the entries by that column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

3. To select the metrics to be included in the calculation, choose **Columns** in the **View** menu. The Attributes window opens (Figure 6-2).

Figure 6-2 *Complexity Metrics (Attributes Window)*


Options - Program

Attributes

- ☑ Lines Of Code
- ☑ Executable Statements
- ☑ Operators
- ☑ Operands
- ☑ Unique operators
- ☑ Unique operands
- ☑ Program Length
- ☑ Vocabulary
- ☑ Program Volume
- ☑ Program Level
- ☑ Program Effort
- ☑ Difficulty
- ☑ Intelligent Content
- ☑ Programming Time
- ☑ Error Estimate
- ☑ Binary decisions

4 In the Attributes window, place a check mark next to each complexity metric you want to calculate. Click **Apply** if you want to save your settings without dismissing the Attributes window open. Click **OK** if you want to save your settings and dismiss the Attributes window.

5 Select an object and choose **Properties** in the **View** menu to display a set of tabs with object properties. For usage information, see *Getting Started* in the workbench documentation set.

Using the Effort Estimation Tool

Use the Effort Estimation tool to compare source files based on weighted values for selected complexity metrics.

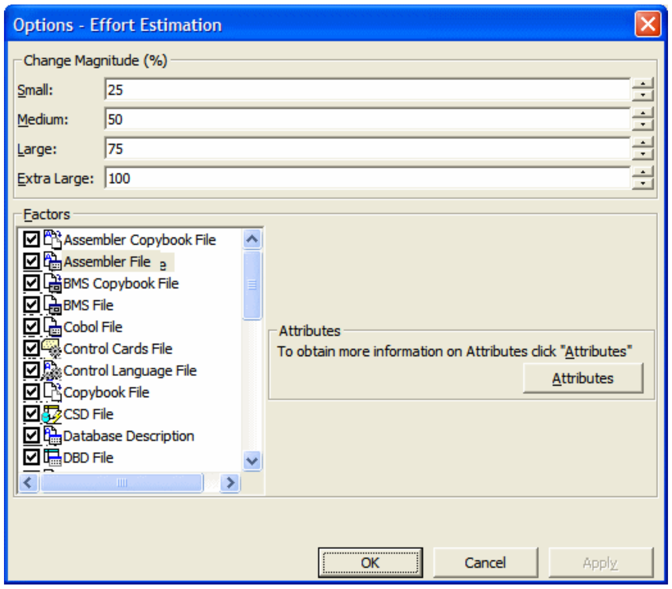
To generate effort estimation statistics:

1 In the Repository Browser, select the project whose effort estimation you want to calculate and choose **Effort** in the **Analyze** menu. A blank Effort Estimation window opens.

2 To select the file types to be included in the calculation, and the complexity metrics for each type, and to set the percentage factor for the

change magnitudes used in the calculation, choose **Options** in the **View** menu. The Effort Estimation Options window opens (Figure 6-3).

Figure 6-3 *Effort Estimation Options Window*


Options - Effort Estimation

Change Magnitude (%)

Small: 25
Medium: 50
Large: 75
Extra Large: 100

Factors

- ☑ Assembler Copybook File
- ☑ Assembler File
- ☑ BMS Copybook File
- ☑ BMS File
- ☑ Cobol File
- ☑ Control Cards File
- ☑ Control Language File
- ☑ Copybook File
- ☑ CSD File
- ☑ Database Description
- ☑ DBD File

Attributes

To obtain more information on Attributes click "Attributes"

3 In the Effort Estimation Options window:

- Place a check mark next to each source file type you want to include in the calculation.
- For each source file type to be included in the calculation, click **Attributes** to edit the complexity metrics used in the calculation. In the Attributes window, select each complexity metric you want to use, then enter its weight in the **Weighting Factor** field. If you want Cyclomatic Complexity to have twice the weight of Conditional Complexity, for example, set the weighting factor for Cyclomatic Complexity to 2 and the weighting factor for Conditional Complexity to 1. The metrics are described in Table 6-1.
- To set the percentage factor for the change magnitudes used in the calculation, enter the percentage you want to be applied in the

combo box for each change magnitude value: Small, Medium, Large, and Extra Large. The values shown in Figure 6-3 are representative. For background on change magnitudes, see "Specifying Change Magnitudes" on page 6-6.

Click **Apply** if you want to save your settings without dismissing the Effort Estimation Options window open. Click **OK** if you want to save your settings and dismiss the Effort Estimation Options window.

4 Click the ▶ button on the tool bar to generate effort estimation statistics. The statistics for the selected source file type are displayed. Figure 6-4 shows the results for Cobol source files.

Tip: Click a column heading to sort the entries by that column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

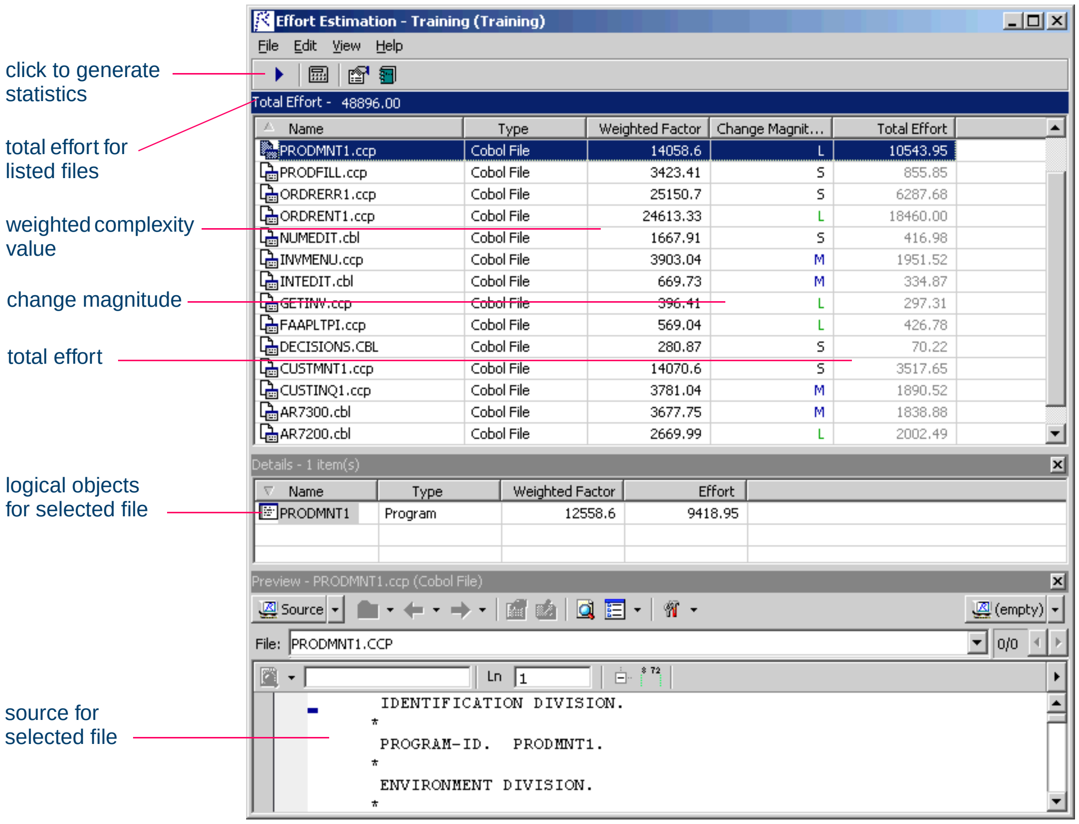
5 Select a source file in the Effort pane to show the effort statistics for each logical object for the file in the Details pane.

6 Select a source file in the Effort pane to view HyperCode for the file in the Preview pane. The information available depends on the type of object selected. For example, you see only source code for a copybook, but full HyperCode for a program.

Note: "HyperCode" is shorthand for the information displayed in HyperView. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

7 Select a file and choose **Properties** in the **View** menu to display a set of tabs with object properties. For usage information, see *Getting Started* in the workbenchdocumentation set.

Figure 6-4 *Effort Estimation Window*


click to generate statistics
total effort for listed files
weighted complexity value
change magnitude
total effort
logical objects for selected file
source for selected file
Effort Estimation - Training (Training)
File Edit View Help

Total Effort - 48896.00

Name	Type	Weighted Factor	Change Magnit...	Total Effort
PRODMNT1.ccp	Cobol File	14058.6	L	10543.95
PRODFILL.ccp	Cobol File	3423.41	S	855.85
ORDRERR1.ccp	Cobol File	25150.7	S	6287.68
ORDRENT1.ccp	Cobol File	24613.33	L	18460.00
NUMEDIT.cbl	Cobol File	1667.91	S	416.98
INVMENU.ccp	Cobol File	3903.04	M	1951.52
INTEDIT.cbl	Cobol File	669.73	M	334.87
GETINV.ccp	Cobol File	396.41	L	297.31
FAAPLTPI.ccp	Cobol File	569.04	L	426.78
DECISIONS.CBL	Cobol File	280.87	S	70.22
CUSTMNT1.ccp	Cobol File	14070.6	S	3517.65
CUSTINQ1.ccp	Cobol File	3781.04	M	1890.52
AR7300.cbl	Cobol File	3677.75	M	1838.88
AR7200.cbl	Cobol File	2669.99	L	2002.49

Details - 1 item(s)

Name	Type	Weighted Factor	Effort
PRODMNT1	Program	12558.6	9418.95

Preview - PRODMNT1.ccp (Cobol File)

```
IDENTIFICATION DIVISION.
*
PROGRAM-ID.  PRODMNT1.
*
ENVIRONMENT DIVISION.
*
```

Specifying Change Magnitudes

What if your own analysis of a project shows that a given program will actually take much less time to change than the weighted effort estimation would suggest. The program might have thousands of source lines, for example, increasing its calculated complexity, while actually being very easy to modify.

A *change magnitude* is a way of overriding the calculated value for a source file. Your "subjective" estimate of the effort involved, Small, Medium, Large, Extra Large, becomes an input to the effort calculation, along with the weighted values.

You typically set the change magnitude for a file in the HyperView Context pane, but you can also set it (or change it) in the Effort Estimation tool. For Hyperview usage, see *Analyzing Programs* in the Modernization Workbench document set.

In the Effort Estimation window (Figure 6-4), select the files whose change magnitudes you want to set and choose **Set Change Magnitude:magnitude** in the **Edit** menu, where *magnitude* is S for Small, M for Medium, L for Large, or XL for Extra Large. The effort calculation for the selected files is automatically updated.

Tip: To select a range of files, hold down the Shift key, click the first item in the range, then click the last item in the range. To select files that are not in a range, hold down the Control key, then click each file you want to add.

Exporting Complexity and Effort Reports

Choose **Report** in the **File** menu to display a printable complexity or effort report. In the printable report, click **Print** to print the report. Click **Save** to export the report to HTML, Excel, RTF, Word, or formatted text.

Supported Complexity Metrics

The following table describes the industry standard and Modernization Workbench-generated complexity metrics supported by the legacy estimation tools. In the table, “program” refers to Cobol, PL/I, Natural, RPG, and Assembler program entities. For details on language-specific metrics, see the *Parser Reference Manual* in the workbench documentation set.

Table 6-1 *Complexity Metrics*

Metric	Object Types	Description
Absolute Complexity	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Binary Decisions divided by the number of statements.
Abstract Classes	Java Package, Java File	Number of abstract classes and interfaces.
Abstract Methods	Java Package, Java File	Number of abstract methods.
Abstractness	Java Package	Ratio of Number of Abstract Classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package.
Afferent Coupling	Java Package	Number of other packages that depend on classes in the package. An indicator of the package's responsibility.
Areas	DMS DDL File, IDMS Schema File, IDMS Subschema	Number of areas.
Asynchronous Calls	Program	Number of asynchronous calls, such as Cobol INITIATE statements.
Average Absolute Complexity	Java Class, Java Interface, Java Package, Java File	Average Absolute Complexity of methods.
Average Binary Decisions	Java Class, Java Interface, Java Package, Java File	Average Binary Decisions of methods.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Average Catch Clauses	Java Class, Java Interface, Java Package, Java File	Average Catch Clauses in methods.
Average Computational Statements	Java Class, Java Interface, Java Package, Java File	Average Computational Statements of methods.
Average Conditional Complexity	Java Class, Java Interface, Java Package, Java File	Average Conditional Complexity of methods.
Average Cyclomatic Complexity	Java Class, Java Interface, Java Package, Java File, PB Object	Average Cyclomatic Complexity of methods.
Average Depth of Inheritance Hierarchy	Java Package, Java File	Average Depth of Inheritance Hierarchy of classes and interfaces.
Average Essential Complexity	Java Class, Java Interface, Java Package, Java File	Average Essential Complexity of methods.
Average Extended Cyclomatic Complexity	Java Class, Java Interface, Java Package, Java File	Average Extended Cyclomatic Complexity of methods.
Average IF Statements	Java Class, Java Interface, Java Package, Java File	Average IF Statements in methods.
Average Lack of Cohesion Per Type	Java Package, Java File	Average Lack of Cohesion of classes and interfaces.
Average Loop Statements	Java Class, Java Interface, Java Package, Java File	Average Loop Statements in methods.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Average Maximum Switch Cases	Java Class, Java Interface, Java Package, Java File	Average Maximum Switch Cases in methods.
Average Nested Block Depth	Java Class, Java Interface, Java Package, Java File, PB Object	Average Nested Block Depth of methods.
Average Parameters	Java Class, Java Interface, Java Package, Java File, PB Object	Average Parameters in methods.
Average Specialization Index Per Type	Java Package, Java File	Average Specialization Index of classes and interfaces.
Average Subtypes	Java Package	Average Subtypes in methods.
Average Switch Cases	Java Class, Java Interface, Java Package, Java File	Average Switch Cases in methods.
Average Switch Statements	Java Class, Java Interface, Java Package, Java File	Average Switch Statements in methods.
Average Unique Method Calls	Java Class, Java Interface, Java Package, Java File	Average Unique Method Calls in methods.
Binary Decisions	Java Method, PB Method, Program, VB Function, VB Method, C/C++ Function, C++ Member Function	Number of branching conditions in the flow graph with two possible outcomes. Includes statements with implicit condition evaluation (loops, AT END, and so on).
Blank Lines	All source files	Number of blank lines of source (sequence number area content is ignored).
Catch Clauses	Java Method, PB Method	Number of catch clauses.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Classes	VB Library	Number of classes.
Columns	DDL File	Number of columns.
Comment Lines	PL/I File, PL/I Include File	Number of lines of source containing comments only and no code.
Comments Ratio	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, PB Method, PB Object	Number of Comments divided by Lines of Code.
Computational Statements	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Number of statements performing arithmetic calculations.
Conditional Complexity	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Binary Decisions plus Unique Operands in Conditions.
Conditional Statements	Program	Number of branching statements with nested statements executed under certain conditions, not including conditional GOTOs.
Constructors	C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, PB Method, PB Object	Number of constructors.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Control Cards Usages	JCL File, JCL Procedure	Number of DD statements referencing control cards identified in Legacy.xml to generate program to control card relationships.
Cyclomatic Complexity	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	**v(G) = e - n + 2**, where v(G) is the cyclomatic complexity of the flow graph (G) for the program in question, e is the number of edges in G, and n is the number of nodes. Quantity of decision logic. The number of linearly independent paths (minimum number of paths to be tested). **v(G) = DE + 1**, where DE is the number of binary decisions made in the program.
Data Elements	Copybook File, Program	Number of declared data items (elementary structures and their fields). For PL/I, the implicit variable DFHEIBLK for CICS statements is counted as a data element.
Data Members	C++ Class	Number of data members.
Data Sets	DMSII Database	Number of data sets.
Dead Data Elements	Program	Number of dead data elements in programs and used include files. Dead data elements are unused structures at any data level, all of whose parents and children are unused. For PL/I, number of declared data items in dead internal procedures.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Dead Data Elements from Includes	Program	Number of dead data elements in include files. Dead data elements are unused structures at any data level, all of whose parents and children are unused. For PL/I, N/A.
Dead Lines	Program	Number of dead lines in programs and used include files. Dead lines are source lines containing Dead Data Elements or Dead Statements. For Cobol, also source lines containing "dead constructs." For more information on dead constructs, see the *Parser Reference Manual*.
Dead Lines from Includes	Program	Number of dead lines in include files. Dead lines are source lines containing Dead Data Elements from Includes or Dead Statements from Includes. For Cobol, also source lines containing "dead constructs." For more information on dead constructs, see the *Parser Reference Manual*.
Dead Statements	Program	Number of dead statements in programs and used include files. A dead statement is a procedural statement that can never be reached during program execution. For PL/I, dead DECLARE statements are not calculated as dead statements.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Dead Statements from Includes	Program	Number of dead statements in include files. A dead statement is a procedural statement that can never be reached during program execution.
Depth of Inheritance Hierarchy	C++ Class, Java Class, Java Interface, PB Object	Maximum nesting of the inheritance hierarchy.
Destructors	C++ Class, VB Class	Number of destructors.
Difficulty	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	**D = (n1 / 2) * (N2 / n2)**, where n1 is Unique Operators, N2 is Operands, and n2 is Unique Operands.
Disjoint Data Sets	DMSII Database	Number of disjoint data sets.
Efferent Coupling	Java Package	Number of other packages that the classes in the package depend on. An indicator of the package's independence.
Entry Points	Program	Number of program entry points.
Error Estimate	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**B = E**(2/3) / 3000**, where E is Programming Effort.
Essential Complexity	C/C++ Function, C++ Member Function, Java Method, PB Method, Program	Quantity of unstructured logic (a loop with an exiting GOTO statement, for example). v(G) for reduced graph without D-structured primes.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Exception Handling Statements	C/C++ Function, C++ Member Function, VB Function, VB Method	Number of exception handling statements.
EXEC Cataloged Procedure Steps	JCL File, JCL Procedure	Number of EXEC statements invoking cataloged procedures.
EXEC In-stream Procedure Steps	JCL File, JCL Procedure	Number of EXEC statements invoking instream procedures.
Executable Statements	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	All Cobol Procedure Division statements, plus CONTINUE and NEXT STATEMENT. All PL/I statements, except BEGIN, DECLARE, DO (block), END, ENTRY, PACKAGE, and PROCEDURE. All Natural statements, except DEFINE DATA and IGNORE. For object-oriented languages, all assignments, function calls (alone on a line), calls, returns, IF, DO, FOR, CHOOSE, EXIT, CONTINUE, and GOTO statements.
Extended Cyclomatic Complexity	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Cyclomatic Complexity plus Logical Operators in Conditions. Number of all paths in the program.
Fields	AS/400 Database File, Java Class, Java Interface, Java Package, Java File, PB Object, VB Class	Number of fields.
Fields of Adabas File Number	Natural DDM File	Number of fields of Adabas file number.

Table 6-1 *Complexity Metrics* (continued)

Metric	**Object Types**	**Description**
Foreign Keys	DDL File	Number of foreign keys.
Function Calls	C/C++ Function, C++ Member Function, VB Function, VB Method	Number of function calls.
Function Points	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, VB Class, VB Function, VB Method	Lines of Code divided by K, where K depends on the language: Cobol=77, Natural=52, PL/I=67, PowerBuilder=24. For RPG, Lines of Code/K + 0.5, where K=61. Estimate of the number of end-user business functions implemented by the program.
Functions	VB Library	Number of functions.
Global Data Items	DMSII Database	Number of global data items.
GoTo Statements	C/C++ Function, C++ Member Function, Program, VB Function, VB Method	Number of GOTO statements, including conditional GOTOs.
Hidden Fields	Map	Number of hidden fields.
IF Statements	C/C++ Function, C++ Member Function, Java Method, PB Method, VB Function, VB Method	Number of IF statements.
Import Statements	Java Package, Java File	Number of import statements.
Include Copybook Statements	APS File, APS Include File	Number of include copybook statements.
Include Data Structure	APS Application	Number of included data structures.
Include DDI	APS Application	Number of included DDIs.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Include DDL/PSB	APS Application	Number of included data definitions.
Include Global Data Area Statements	Natural File, Natural Subroutine File	Number of included global data area statements.
Include Local Data Area Statements	Natural File, Natural Subroutine File	Number of included local data area statements.
Include Macro Statements	APS File, APS Include File	Number of included macro statements.
Include Macro/Copybook	APS Application	Number of included macros/copybooks.
Include Parameter Data Area Statements	Natural File, Natural Subroutine File	Number of included parameter data area statements.
Include Program	APS Application	Number of included programs.
Include Report	APS Application	Number of included reports.
Include Resource Statement	Copybook File, Cobol File (ACUCOBOL only)	Number of included resource statements.
Include Screen	APS Application	Number of included screens.
Include Statements	All source files, except DDL File, Java Interface, Java File, PB Object	Number of language-specific include statements.
Inherited Methods	C++ Class, VB Class	Number of inherited methods.
Inner Call Statements	Program	Number of statements that invoke Inner Procedures.
Inner Procedures	Program	Number of structured pieces of code that cannot be invoked from external programs (Cobol paragraphs, inner PL/I procedures, Natural inline subroutines).

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Input Fields	Map	Number of input fields.
Input/Output Fields	Map	Number of input/output fields.
Instability	Java Package	Ratio of Efferent Coupling (Ce) to total coupling (Ce + Ca), such that **I = Ce / (Ce + Ca)**. An indicator of the package's resilience to change.
Intelligent Content	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**I = L * V**, where L is Program Level and V is Program Volume. Complexity of a given algorithm independent of the language used to express the algorithm.
Interfaces	Java Class, Java Interface, Java Package, Java File	Number of interfaces.
IO Statements	Program	Number of statements performing input/output operations.
Key Fields	AS/400 Database File	Number of key fields.
Labels	C/C++ Function, C++ Member Function, VB Function, VB Method	Number of labels.
Lack of Cohesion	Java Class, Java Interface	**LOCM = (m - sum(mA)/a1+a2) / (m-1)**, where mA is the number of methods that access class attributes.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Lines of Code	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of lines of code, including copybooks (if applicable), but not including comments and blank lines.
Lines with Both Comments and Code	PL/I File, PL/I Include File	Number of lines of source containing both comments and code.
Lines with Comments	All source files, except PL/I, PL/I Include, and Java Files	Number of lines of source containing comments, including inline comments placed on lines with statements.
Local Data Items	C/C++ Function, C++ Member Function, VB Function, VB Method	Number of local data items.
Logical Operators in Conditions	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, VB Class, VB Function, VB Method	Number of binary logical operators used in conditions.
Logical Records	IDMS Subschema File	Number of logical records.
Loop Statements	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Number of repetitively executing statements.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Macro Assignments	PL/I File, PL/I Include File	Number of assignments to global macro variables outside any macro procedure.
Macro Declarations	PL/I File, PL/I Include File	Number of global macro variables. Macro variables within macro procedures are not counted.
Macro Lines	PL/I File, PL/I Include File	Number of lines for include statements, macro invocations, declarations, macro procedures, and %-statements. Blank lines and comments lines are ignored.
Macro Procedures	PL/I File, PL/I Include File	Number of macro procedures declared in the file.
Macro Statements	PL/I File, PL/I Include File	Number of macro statements.
Maintainability Index	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**MI = 171 - 5.2 * ln (PgmVolume) - 0.23 * ExtCycComp - 16.2 * ln (LOC) + 50 * sin (sqrt (2.46 * CommentLines/SourceLines))**, where PgmVolume is Program Volume, ExtCycComp is Extended Cyclomatic Complexity, LOC is Lines of Code, CommentLines is Comment Lines, and SourceLines is Source Lines.
Maps	Natural Map	Number of maps.
Maximum Depth of Logic Nesting	C/C++ Function, C++ Member Function	Maximum nesting of logic.
Maximum Switch Cases	C/C++ Function, C++ Member Function, Java Method, PB Method	Maximum Switch Cases.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Member Function Calls	C/C++ Function, C++ Member Function	Number of member function calls.
Member Functions	C++ Class	Number of member functions.
Method Calls	VB Function, VB Method	Number of method calls.
Methods	Java Class, Java Interface, Java Package, Java File, PB Object, VB Class	Number of methods.
Nested Block Depth	Java Method, PB Method	Maximum nesting of IF, CHOOSE, TRY, DO, SWITCH, or FOR constructs. 1 is added to the depth in the IF and ELSE parts of IF statements, the bodies of loops, and the code in each case of a Choose statement.
Nesting Level	Program, VB Function, VB Method	Maximum nesting of conditional statements within conditional statements (0 if no conditional statements, 1 if no nesting).
Non-returning Calls	Program	Number of non-returning calls, such as Cobol XCTL statements.
Normalized Distance	Java Package	Perpendicular distance of the package from the idealized line A + I = 1. An indicator of the package's balance between abstractness and stability.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Number of Comments	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, PB Method, PB Object	For C/C++, number of items containing text preceded by // or enclosed between /* and */. For Java, number of lines of source containing comments, including inline comments placed on lines with statements. For PowerBuilder, number of lines of source containing comments only and no code.
Number of Controls	PB DataWindow	Number of controls.
Number of Entries	CSD File, FCT File, PCT File, Valtab File	Number of entries.
Number of Executes	ECL File	Number of executes.
Number of Lines	PB DataWindow, PB Query, PB Pipeline	Number of lines.
Number of PCBs	PSB File	Number of PCBs.
Number of Processor Calls	ECL File	Number of processor calls.
Omit Fields	AS/400 Database File	Number of Omit fields.
Operands	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of operand occurrences (N2). Operands are variables and literals used in operators. Compare Unique Operands.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Operators	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of operator occurrences (N1). Operators are executable statements and unary and binary operations. Compare Unique Operators.
Output Fields	Map	Number of output fields.
Overridden Methods	Java Class, Java Interface, Java Package, Java File	Number of overridden methods.
Parameters	C/C++ Function, C++ Member Function, Java Method, PB Method, Program, VB Function, VB Method	Number of Cobol Procedure Division USING...RETURNING parameters. Number of PL/I PROCEDURE parameters. Number of Natural PARAMETER and PARAMETER USING parameters. Otherwise, number of parameters in method calls.
Path Groups	IDMS Subschema File	Number of path groups.
Pointers	C/C++ Function, C++ Member Function, C++ Class, Program, VB Function, VB Method	Number of data elements declared as pointers.
Primary Keys	DDL File	Number of primary keys.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Program Length	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**N = N1 + N2**, where N1 is Operators and N2 is Operands.
Program Level	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**L = 1 / D**, where D is Difficulty.
Program Volume	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**V = N * log2(n)**, where N is Program Length and n is Vocabulary. Minimum number of bits required to code the program.
Programming Effort	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**E = V / L**, where V is Program Volume and L is Program Level. Estimated mental effort required to develop the program.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Programming Time	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**T = E / 18**, where E is the Programming Effort and 18 is Stroud's Number. Estimated amount of time required to implement the algorithm, in seconds.
Public Data Members	C++ Class	Number of public data members.
Public Fields	VB Class	Number of public fields.
Public Member Functions	C++ Class	Number of public member functions.
Public Methods	VB Class	Number of public methods.
Records	ADL File, AS/400 Database File, DMS DDL File, IDMS Schema File, IDMS Subschema File	Number of records.
Remapped Data Sets	DMSII Database	Number of remapped data sets.
Response for Class	C++ Class, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object	Response for class.
Returning Calls	Program	Number of returning calls, such as Cobol CALL or LINK statements.
Screens	AS/400 Device Description, BMS File, DPS File, ICL Form, MFS File	Number of screens.
Segments	DBD File	Number of segments.
SELECT (SWITCH) Statements	VB Method	Number of SWITCH statements.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Select Fields	AS/400 Database File	Number of select fields.
Sets	ADL File, DMS DDL File, DMSII Database, IDMS Schema File, IDMS Subschema File	Number of sets.
Sliceable Dead Lines	PL/I Program	Number of Dead Lines that can be sliced using the Application Architect Dead Code Elimination method. Dead procedures containing either preprocessor statements or statements subject to macro preprocessor replacement are not counted as sliceable dead lines, since it is not always possible to determine whether they can be safely removed. Lines from include files are not counted.
Source Lines	All source files	Number of lines of source.
Specialization Index	Java Class, Java Interface	Number of subclasses divided by number of superclasses.
Static Data Items	C/C++ Function, C++ Member Function, VB Function, VB Method	Number of static data items.
Static Data Members	C++ Class	Number of static data members.
Static Fields	Java Class, Java Interface, Java Package, Java File, PB Object, VB Class	Number of static fields.
Static Functions	C++ Class	Number of static functions.
Static Member Functions	C++ Class	Number of static member functions.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Static Methods	Java Class, Java Interface, Java Package, Java File, VB Class	Number of static methods.
Steps	JCL File, JCL Procedure, Job	Number of steps.
Subsets	DMSII Database	Number of subsets.
Subtypes	Java Class, Java Interface, Java File, PB Object	Number of subtypes.
Switch Cases	C/C++ Function, C++ Member Function, Java Method, PB Method	Number of SWITCH cases.
Switch/Choose Statements	Java Method, PB Method	Number of SWITCH in Java, CHOOSE statements. in PowerBuilders.
SWITCH/SELECT Statements	C/C++ Function, C++ Member Function, VB Function	Number of SWITCH statements.
Tables	DDL File	Number of tables.
Total EXEC Cataloged Procedure Steps	JCL File	Number of EXEC statements invoking cataloged procedures in the job and invoked procedures. If a procedure is invoked multiple times, it is counted each time.
Total Include Copybook Statements	APS File, APS Include	Total number of include copybook statements.

Table 6-1 *Complexity Metrics* (continued)

Metric	**Object Types**	**Description**
Total Include Statements	Assembler File, BMS File, Cobol File, Copybook File, DMS II DASDL File, ECL File, JCL File, MFS File, Natural File, Natural Subroutine File, PL/I File, PSB File, RPG File, VB Project File, WFL File	Total Include Statements. For PL/I, all the include statements in the file and the used include files.
Unique Function Calls	C/C++ Function, C++ Member Function	Number of unique function calls.
Unique Member Function Calls	C/C++ Function, C++ Member Function	Number of unique member function calls.
Unique Method Calls	Java Method, PB Method, PB Object	Number of distinct method calls.
Unique Operands	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of distinct operands (n2). Operands are variables and literals used in operators. Uniqueness of literals is determined by their notation. Compare Operands.
Unique Operands in Conditions	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of distinct operands used in conditions.

Table 6-1 *Complexity Metrics* (continued)

Metric	Object Types	Description
Unique Operators	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	Number of distinct operators (n1). Operators are statements and unary and binary operations. Compare Operators.
Unique System Macro Instructions	Assembler File	Number of unique system macro instructions.
Vocabulary	C/C++ Function, C++ Member Function, C++ Class, Java Method, Java Class, Java Interface, Java Package, Java File, Program, PB Method, PB Object, VB Class, VB Function, VB Method	**n = n1 + n2**, where n1 is the number of Unique Operators and n2 is the number of Unique Operands.
Weighted Methods	Java Class, Java Interface, Java Package, Java File, PB Object	Number of weighted methods.

What's Next?

Now that you know how to estimate project complexity and effort, let's look at one final project analysis task: using Change Analyzer to understand how a change in the definition or usage of a data field might impact the rest of the application.

Identifying Classes of Data Items

7

The Change Analyzer identifies the *class* of data items used to perform a business function in a legacy application. Among other uses, it lets you answer the kinds of “What if?” questions posed in the recent past by the industry-wide changes for Y2K, Zip+4, and the Euro dollar: “What if I change the type of this variable, or the length of this field? What *other* fields will I also have to change?”

Use change analysis results to prepare project plans and technical specifications. You can generate reports showing source entities that may require modification, lines of code affected, and the like.

Understanding Data Item Classification

Suppose your organization is considering adding support for a new currency and that you are going to have to expand the existing exchange rate data field from 9(5)V9(3) to 9(5)V9(6) to accommodate the currency. You will need to know the data fields that are affected in the database, intermediate fields that may contain or use the exchange rate field in calculations, and so forth.

Seed Fields

Use Change Analyzer to search for the exchange rate field. The object of the search is called a *seed field:*

- If the application uses field names like EX-RATE, EXCH-RATE, EXCHANGE-RATE, and RATE-OF-EXCHG, you would search for data names that contain *EXCH* or *RATE*.
- If you know that some fields already contain the required number of decimal positions and are interested only in those that don’t, you might further limit the search by filtering on the PICTURE clause format of the variable, selecting only data items that have a format of 9(5)V9(3).
- You might limit the search even further by choosing fields that have a given initial value.
- If you know there are data fields that will meet the search criteria for name, but are not what you are looking for, you might set up a list of names to exclude, such as INTEREST-RATE and *PRORATE*.

Synonyms

When you execute the search, Change Analyzer returns not only the fields that match the search criteria but any *synonyms* for the fields. A synonym is a data field whose value is related to the value of the matched field (a field whose value is assigned by a MOVE or REDEFINE statement, for example). In these cases, if you increase the size of the matched field, you probably will have to increase the size of the synonym as well.

Seed Lists

As you examine the source code for seed fields and synonyms in Change Analyzer and determine whether a field is affected by a proposed change, you move the fields between *seed lists:*

- The *Working* list contains seed fields or synonyms that you are currently investigating.
- The *Affected* list contains seed fields or synonyms that you have determined would be affected by the proposed change.

- The *Clean* list contains seed fields or synonyms that you have determined would not be affected by the proposed change.

You can generate reports based on these lists, as well as reports showing the affected source code in context.

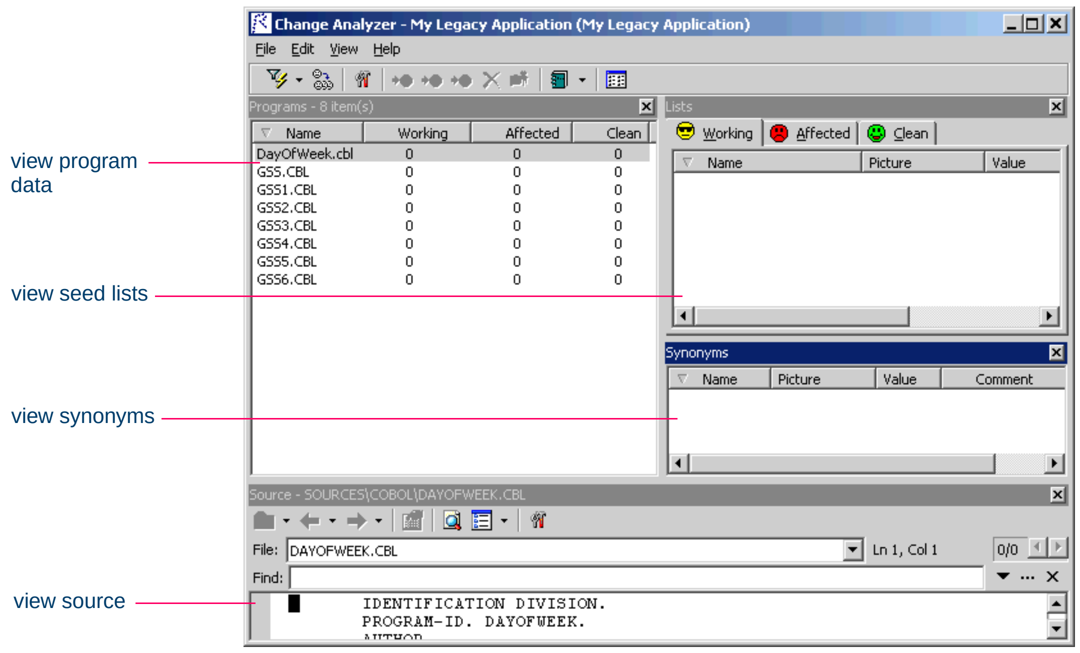
Getting Started in Change Analyzer

This section describes a sample use of Change Analyzer that should help you get oriented in it. Assume that you want to evaluate the impact of changing the YEAR field in your application.

To evaluate change impacts:

1 In the Repository Browser, select the project you want to analyze and choose **Field Change** in the **Analyze** menu. The Change Analyzer window opens (Figure 7-2).

Figure 7-1 *Change Analyzer Window*

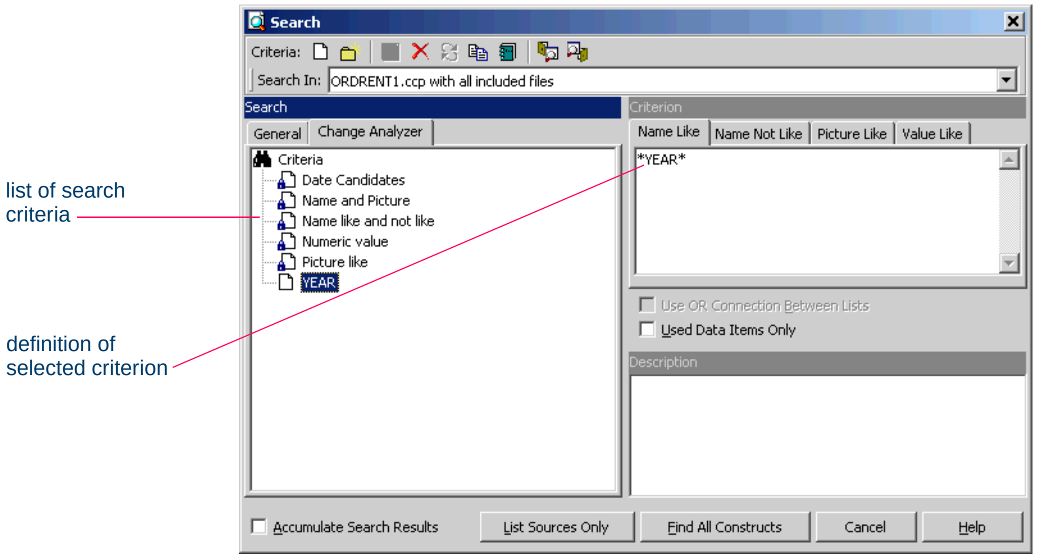


Name	Working	Affected	Clean
DayOfWeek.cbl	0	0	0
GSS.CBL	0	0	0
GSS1.CBL	0	0	0
GSS2.CBL	0	0	0
GSS3.CBL	0	0	0
GSS4.CBL	0	0	0
GSS5.CBL	0	0	0
GSS6.CBL	0	0	0

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. DAYOFWEEK.
```

2 In the **File** menu, choose **Apply Filter:To Working**. The Search window opens (Figure 7-2).

Figure 7-2 *Search Window (After Step 4)*



Search In: ORDRENT1.ccp with all included files

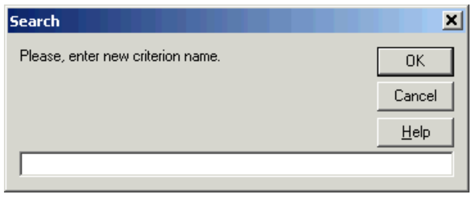
- Criteria
  - Date Candidates
  - Name and Picture
  - Name like and not like
  - Numeric value
  - Picture like
  - YEAR

YEAR

- [ ] Use OR Connection Between Lists
- [ ] Used Data Items Only

3 In the Search window, click the Change Analyzer tab to view a list of recognized search criteria. Select a search criterion to view its definition in the tabs in the righthand portion of the window. Click the button on the tool bar to define a new criterion. The New Search Criterion dialog opens (Figure 7-3).

Figure 7-3 *New Search Criterion Dialog*



Please, enter new criterion name.

4 In the text field, enter YEAR and click **OK**. The YEAR criterion is displayed in the list of recognized criteria in the Change Analyzer tab in the Search window.

5 In the Change Analyzer tab, select the YEAR criterion, enter *YEAR* in the Name Like tab, and click **Find All Constructs**. Change Analyzer returns the seed fields for the criterion in the Working list and their synonyms in the Synonyms pane. Select a program in the Programs pane to view its seed fields and synonyms.

6 In the Working tab, select a seed field to navigate to its source in the Source pane. Examine the source to determine whether the field will be affected by the proposed change:

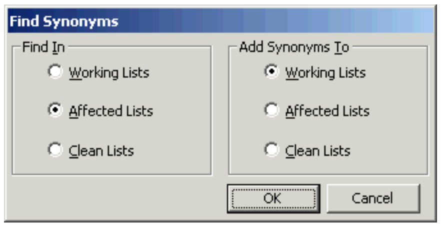
- If the field will be affected by the change, choose **Move to Affected** in the **Edit** menu. Change Analyzer moves the field to the Affected tab.
- If the field will not be affected by the change, choose **Move to Clean** in the **Edit** menu. Change Analyzer moves the field to the Clean tab.

7 Repeat step 6 for each seed field in the Working list.

8 Repeat step 6 and step 7 for each program in the Programs pane.

9 In the **File** menu, choose **Find Synonyms**. The Find Synonyms dialog opens (Figure 7-4).

Figure 7-4 *Find Synonyms Dialog*


Find Synonyms

Find In: Working Lists [✓ Affected Lists] Clean Lists

Add Synonyms To: [✓ Working Lists] Affected Lists Clean Lists

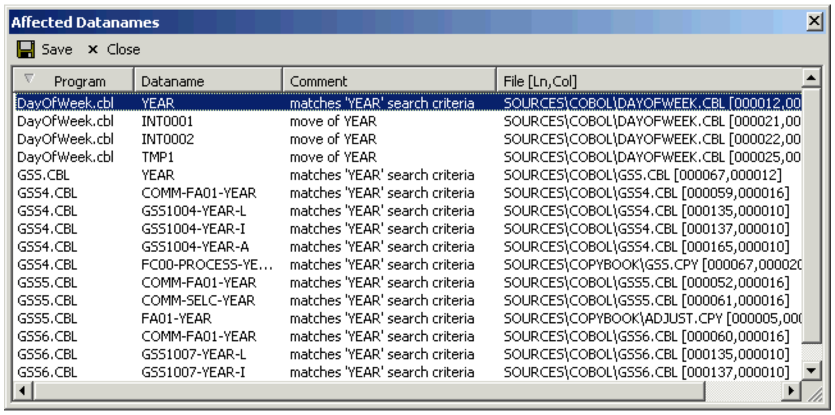
10 In the Find In pane, select **Affected Lists**. In the **Add Synonyms To** pane, select **Working Lists**. Click **OK**. Change Analyzer returns the synonyms for the affected fields in the Working list.

11 Repeat step 6 for each synonym in the Working list.

12 In the **File** menu, choose **Remove Unused:From Affected Lists** to remove fields that are declared but not used from the Affected list.

13 In the **File** menu, choose **Report:Affected Datanames**. Change Analyzer generates the Affected Datanames report (Figure 7-5).

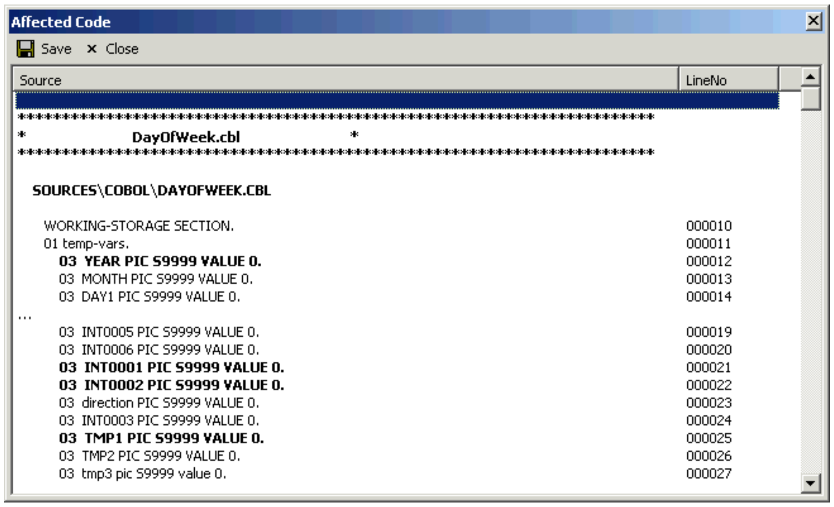
Figure 7-5 *Affected Datanames Report*



Program	Dataname	Comment	File [Ln,Col]
DayOfWeek.cbl	YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\DAYOFWEEK.CBL [000012,00
DayOfWeek.cbl	INT0001	move of YEAR	SOURCES\COBOL\DAYOFWEEK.CBL [000021,00
DayOfWeek.cbl	INT0002	move of YEAR	SOURCES\COBOL\DAYOFWEEK.CBL [000022,00
DayOfWeek.cbl	TMP1	move of YEAR	SOURCES\COBOL\DAYOFWEEK.CBL [000025,00
GSS.CBL	YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\GSS.CBL [000067,000012]
GSS4.CBL	COMM-FA01-YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\GSS4.CBL [000059,000016]
GSS4.CBL	GSS1004-YEAR-L	matches 'YEAR' search criteria	SOURCES\COBOL\GSS4.CBL [000135,000010]
GSS4.CBL	GSS1004-YEAR-I	matches 'YEAR' search criteria	SOURCES\COBOL\GSS4.CBL [000137,000010]
GSS4.CBL	GSS1004-YEAR-A	matches 'YEAR' search criteria	SOURCES\COBOL\GSS4.CBL [000165,000010]
GSS4.CBL	FC00-PROCESS-YE...	matches 'YEAR' search criteria	SOURCES\COPYBOOK\GSS.CPY [000067,00002
GSS5.CBL	COMM-FA01-YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\GSS5.CBL [000052,000016]
GSS5.CBL	COMM-SELC-YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\GSS5.CBL [000061,000016]
GSS5.CBL	FA01-YEAR	matches 'YEAR' search criteria	SOURCES\COPYBOOK\ADJUST.CPY [000005,00
GSS6.CBL	COMM-FA01-YEAR	matches 'YEAR' search criteria	SOURCES\COBOL\GSS6.CBL [000060,000016]
GSS6.CBL	GSS1007-YEAR-L	matches 'YEAR' search criteria	SOURCES\COBOL\GSS6.CBL [000135,000010]
GSS6.CBL	GSS1007-YEAR-I	matches 'YEAR' search criteria	SOURCES\COBOL\GSS6.CBL [000137,000010]

14 In the **File** menu, choose **Report:Affected Code**. Change Analyzer generates the Affected Code report (Figure 7-6).

Figure 7-6 *Affected Code Report*



```
***********************************************************************************
*                   DayOfWeek.cbl                  *
***********************************************************************************

  SOURCES\COBOL\DAYOFWEEK.CBL

    WORKING-STORAGE SECTION.                                          000010
    01 temp-vars.                                                     000011
       03 YEAR PIC S9999 VALUE 0.                                     000012
       03 MONTH PIC S9999 VALUE 0.                                    000013
       03 DAY1 PIC S9999 VALUE 0.                                     000014
...
       03 INT0005 PIC S9999 VALUE 0.                                  000019
       03 INT0006 PIC S9999 VALUE 0.                                  000020
       03 INT0001 PIC S9999 VALUE 0.                                  000021
       03 INT0002 PIC S9999 VALUE 0.                                  000022
       03 direction PIC S9999 VALUE 0.                                000023
       03 INT0003 PIC S9999 VALUE 0.                                  000024
       03 TMP1 PIC S9999 VALUE 0.                                     000025
       03 TMP2 PIC S9999 VALUE 0.                                     000026
       03 tmp3 pic S9999 value 0.                                     000027
```

Using Change Analyzer

You generate change impact information for the current project. The project must have been verified with the **Enable Data Element Flow** option set in the project verification options.

Note: For more information on verification, see *Preparing Projects* in the workbench documentation set.

Working with Change Impact Information

The window for the Change Analyzer consists of a Programs pane, Lists pane, Synonyms pane, Source pane, and Activity Log. You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Sorting Entries Click a column heading in a pane to sort the entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Programs Pane

The Programs pane displays the programs in the selected project and the distribution of their seed fields in the seed lists. Select a program to view its seed fields in the List pane and their synonyms in the Synonyms pane. You also select a program to view its source in the Source pane.

Lists Pane

The Lists pane displays the Working, Affected, and Clean lists for the program selected in the Programs pane. Click the appropriate tab to view a list. Create projects from a list as described in "Creating Projects in Change Analyzer" on page 7-15. Generate reports as described in "Generating and Exporting Change Analyzer Reports" on page 7-16.

Each tab and its corresponding report contains the following columns:

Tip: Use the Report Control option to restrict the columns displayed in the tab.

- The Name column lists the seed fields or synonyms on the list.
- The Length column displays the length of the seed fields or synonyms on the list.
- The Normalized Picture column displays the normalized picture of the seed fields or synonyms on the list.
- The Picture column displays the format of the seed fields or synonyms on the list.
- The Value column displays the initial value of the seed fields or synonyms on the list.
- The Comment field describes the basis for including the seed field or synonym on the list. Select an entry in the Lists or Synonyms pane and choose **Comment** in the **Edit** menu to open a dialog where you can modify the comment.

Applying a Search Filter to a List To apply a search filter to a list, choose **Apply Filter:To List** in the **File** menu, then follow the instructions for performing searches in "Searching for Seed Fields in Change Analyzer" on page 7-13.

Selecting Fields Click a field on a list to select it. To select all the fields on a list, click the tab for the list and choose **Select All** in the **Edit** menu.

To select fields based on search patterns, click the tab for the list and choose **Select** in the **Edit** menu. A Select dialog opens, where you can enter the search patterns. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

To select a range of fields, hold down the Shift key, click the first field in the range, then click the last field in the range. To select fields that are not in a range, hold down the Control key, then click each field you want to select.

Viewing Source Select a field to view its source in the Source pane.

Moving Fields between Lists To move fields between lists, select the fields, then choose **Move to List** in the **Edit** menu. ***Removing Unused Fields from Lists*** To remove fields that are declared but not used from a list, choose **Remove Unused:From List** in the **File** menu.

Deleting Seed Fields from a List To delete a seed field from a list, select the field and choose **Delete** in the **Edit** menu. You are prompted to confirm the deletion. Click **Yes**.

Clearing Lists Search results are added to any previous results in a list. To clear the previous results from the list, choose **Clear:List** in the **File** menu. To clear all the lists, choose **Clear:All Lists** in the **File** menu.

Synonyms Pane

The Synonyms pane displays the synonyms for the field selected in the Lists pane. Select synonyms the same way you select list fields, as described in "Selecting Fields" on page 7-9.

Viewing Source Select a synonym to view its source in the Source pane.

Moving Synonyms to Lists To move synonyms to a list, choose **Find Synonyms** in the **File** menu. The Find Synonyms dialog opens, where you can select the list of seed fields to find synonyms for and the list to move the synonyms to.

Source Pane

The Source pane lets you browse the source code for the program selected in the Programs pane and the field or synonym selected in the Lists or Synonyms pane. Usage is similar to that for the Source pane in HyperView. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

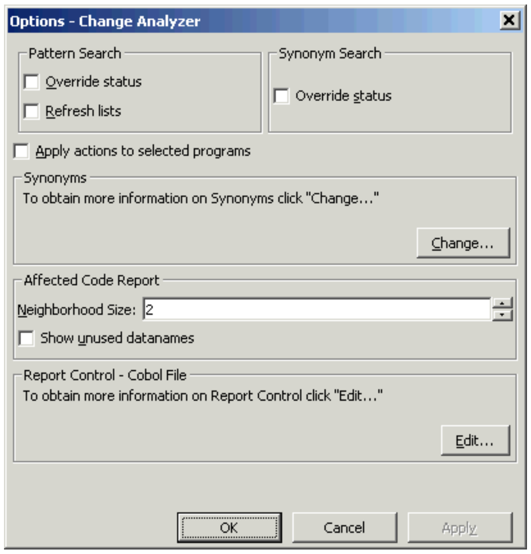
Setting Change Analyzer Options

Change Analyzer options control seed field and synonym pattern searches and the amount of information displayed in the Affected Code report.

To set Change Analyzer options:

1 In the **View** menu, choose **Options**. The Change Analyzer options window opens (Figure 7-7).

Figure 7-7 *Change Analyzer Options Window*



To obtain more information on Synonyms click "Change..."

Neighborhood Size: 2

To obtain more information on Report Control click "Edit..."

2 In the Pattern Search group box, choose any combination of:

- **Override status** if you want the search results to override the current list assignment of a field. If you do not select this option, Change Analyzer applies the search results only to fields not yet assigned to a seed list.
- **Refresh lists** if you want the search results to replace fields in the target seed list. If you do not select this option, Change Analyzer adds the search results to the fields already assigned to the target list.

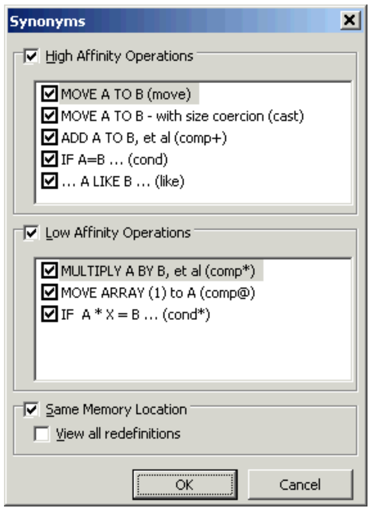
3 In the Synonym Search group box, choose **Override status** if you want the search results to override the current list assignment of a synonym. If you do not select this option, Change Analyzer applies the search results only to synonyms not yet assigned to a seed list.

4 Choose **Apply actions to selected programs** if you want Change Analyzer to act only on the programs selected in the Programs pane.

5 In the Affected Code Report pane, specify in the **Neighborhood Size** combo box the number of lines of unaffected code you want the report to display above and below the line of affected code. (The line of affected code is displayed in **bold** in the report.) Then choose **Show unused datanames** if you want the report to include unused data fields.

6 In the Synonyms group box, click **Change**. The Synonyms Options window opens (Figure 7-8).

Figure 7-8 *Synonym Options Window*


Synonyms
☑ High Affinity Operations
☑ MOVE A TO B (move)
☑ MOVE A TO B - with size coercion (cast)
☑ ADD A TO B, et al (comp+)
☑ IF A=B ... (cond)
☑ ... A LIKE B ... (like)
☑ Low Affinity Operations
☑ MULTIPLY A BY B, et al (comp*)
☑ MOVE ARRAY (1) to A (comp@)
☑ IF A * X = B ... (cond*)
☑ Same Memory Location
☐ View all redefinitions
OK
Cancel

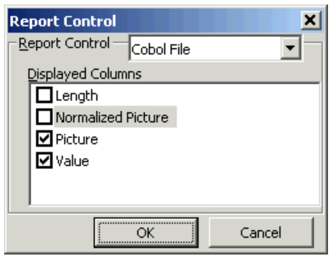
7 Choose any combination of:

- **High-Affinity Operations** if you want Change Analyzer to identify as synonyms data fields that have a high affinity to seed fields, such as those used in assignment and addition operations. Then choose the types of high-affinity operations in the list box.
- **Low-Affinity Operations** if you want Change Analyzer to identify as synonyms data fields that have a low affinity to seed fields, such as those used in multiplication and division operations. Then choose the types of low-affinity operations in the list box.
- **Same Memory Location** if you want Change Analyzer to identify as synonyms data fields that share a memory location with seed fields. Then choose **View all redefinitions** to identify as synonyms all data fields that share a memory location with seed fields. If you do not select this option, Change Analyzer identifies as synonyms only data fields that are exact redefinitions of a seed field, such as one declared with the same size and offset.

Click **OK** to dismiss the Synonyms Options window and return to the Change Analyzer Options window.

8 In the Report Control group box, click **Edit**. The Report Control Options window opens (Figure 7-9).

Figure 7-9 *Report Control Options Window*



9 In the Report Control group box, select the type of file you are analyzing in the drop-down, then choose the columns you want to be displayed in Change Analyzer lists and their corresponding reports. Click **OK** to dismiss the Report Control Options window and return to the Change Analyzer Options window.

Searching for Seed Fields in Change Analyzer

The Change Analyzer search facility (Figure 7-2) contains two tabs:

- The General tab opens the HyperView advanced search facility.
- The Change Analyzer tab opens a scoped version of the advanced search facility for Change Analyzer.

Ordinarily, the scoped version of the tool should be sufficient for most searches. If you are already familiar with the advanced search facility, however, you may want to use it instead of the scoped tool. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

Note: Change Analyzer returns only constants and literals found in the Procedure section of Cobol programs.

To search for seed fields:

1. In the **File** menu, choose **Apply Filter: To List**. The Change Analyzer Search window opens (Figure 7-2).
2. In the Search window, click the Change Analyzer tab. The Change Analyzer tab displays a list of recognized search criteria. Define a new criterion as described in step 3 on page 7-4.
3. Select a criterion to edit its definition in the tabs in the righthand portion of the window. Each tab specifies a condition in the definition. The definition can consist of any combination of conditions.

   For each condition, enter a list of patterns you want to match, one pattern per line. Select:

   - Name Like to specify patterns that are like the name of the field you are searching for.
   - Name Not Like to specify patterns that are unlike the name of the field you are searching for.
   - Picture Like to specify patterns that are like the format of the field you are searching for.
   - Value Like to specify patterns that are like the initial value of the field you are searching for.

   You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).
4. Select **Use OR Connection Inside List** if you want the patterns listed in the tabs to be ORed. If you do not select this option, the patterns are ANDed.
5. Select **Use OR Connection Between Lists** if you want the conditions defined in the tabs to be ORed. If you do not select this option, the conditions are ANDed.
6. Select **Used Data Items Only** if you want the search results to consist only of fields that are used in the selected programs. If you do not select this option, search results include fields that are declared but not used.

7 In the Search in pane, click **Name Like** to specify a pattern that is like the name of the program or programs you want to limit the search to. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

8 Click **Find All Constructs** to execute the search. Change Analyzer returns the seed fields for the selected criterion in the Working list and their synonyms in the Synonyms pane. The results are added to any previous results.

Note: Tool bar usage in the Search window is identical to that for the HyperView advanced search facility. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set.

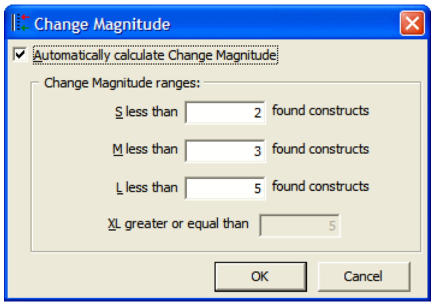
Creating Projects in Change Analyzer

You can create a project directly in Change Analyzer from the results of your analysis. The project contains only source files with fields in a Working, Affected, or Clean list.

To create a project from a list:

1 In the **File** menu, choose **Create Project:From List**. The New Project dialog opens. Enter the name of the new project in the text field and click **OK**. The Change Magnitude window opens (Figure 7-10).

Figure 7-10 *Change Magnitude Window*


Change Magnitude

[✓] Automatically calculate Change Magnitude

Change Magnitude ranges:

S less than	2	found constructs
M less than	3	found constructs
L less than	5	found constructs
XL greater or equal than	5	

2 In the Change Magnitude window, select the **Automatically calculate change magnitude** check box if you want Change Analyzer to set change magnitudes for the listed source files based on the ranges specified in the fields below the check box. The Change Analyzer settings will override any existing change magnitudes for the files.

The fields below the check box contain default ranges for the available values: Small, Medium, Large, and Extra Large. Compare the number of fields in each source file listed in the Programs pane with the default ranges, then modify the ranges as necessary.

If you want Cobol source files with 6 to 10 fields in a list to have a Large change magnitude, for example, set the range for Medium to less than 6 and the range for Large to less than 11. When you are satisfied with your choices, click **O**K. For more information on change magnitudes, see "Specifying the Change Magnitude for a Source File" on page 2-5.

Generating and Exporting Change Analyzer Reports

In the **File** menu, choose **Report:*Type*** to display printable Change Analyzer reports:

- The Affected Datanames Report (Figure 7-5) lists the datanames in the Affected list. In addition to the information displayed in the Affected list, the report displays the program that uses the dataname, a comment that identifies the search criterion the dataname matched, the source file in which the dataname is declared, and the line and column numbers of the declaration.
- The Possible Datanames Report lists every dataname in the Working, Affected, and Clean lists. In addition to the information displayed in the Affected Datanames report, the Possible Datanames report displays the disposition of datanames in seed lists.
- The Clean Datanames Report lists the datanames in the Clean list. The report displays the same information as the Affected Datanames report.
- The Affected Code Report (Figure 7-6) displays the source code for the datanames in the Affected list in both the program that uses the

field and the source file in which the field is declared. The line of affected code is displayed in **bold**, surrounded by the number of lines of neighboring code you specified in step 5 on page 7-11.

- The Metrics Report displays, for each program in the selected project, the number of declarations that contain possible, affected, or clean datanames, the percentage of declarations that contain affected datanames, and the percentage of uses of affected datanames.

In the printable report, click **Print** to print the report. Click **Save** to export the report to HTML, Excel, RTF, Word, or formatted text.

What's Next?

That's all you need to know to use the Change Analyzer tool to identify data item classes. Now let's look at how you use SOA Analyzer to assess legacy programs for their affinity with modern service-oriented architectures (SOA).

Performing SOA Analysis

8

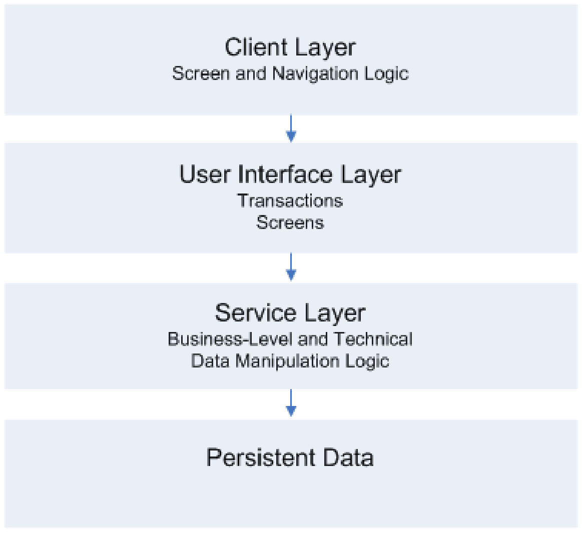
The Modernization Workbench SOA Analyzer lets you assess legacy programs for their affinity with modern service-oriented architectures (SOA): how modular the programs are, the extent to which they encapsulate data operations, whether they isolate business logic, and similar measures. Once you have identified SOA-conforming programs, you can use the tool to create wrappers that expose the programs to other applications as Web services.

Understanding SOA Analysis

Most legacy applications can be abstracted to an “ideal architecture” similar to the one shown in Figure 8-1. In this architecture, UI programs, transitional programs with navigation logic, data abstraction programs, and data access activities all are implemented in self-contained modules deployed in sharply demarcated layers.

The reality, of course, is usually very different. Abstract knowledge of the data, business logic, data access, and UI may be enmeshed in the same programs, making it extremely difficult to isolate candidates for reuse.

Figure 8-1 *Abstraction of Legacy Application*



SOA Analyzer evaluates legacy applications against an abstract "ideal architecture."

That's where SOA Analyzer comes in. It categorizes programs according to the functions they perform in an abstract legacy architecture: UI, data abstraction, data access, and so forth. On the basis of the layering analysis it performs and the wealth of analysis tools it provides, users can identify legacy programs that now or with some rewriting can be exposed as Web services. It also provides facilities that make it easy to generate a user interface and navigation scheme on another platform.

Understanding Client-Layer Programs

Client-layer programs contain the logic for capturing user requests, validating input data, invoking appropriate service programs from lower layers, and formatting the output that responds to user requests.

In this layer, execution moves to lower layers via CALL or LINK operations and to other programs of the same layer via exclusive transfer of control; in CICS, for example, by XCTL or RETURN TRANID statements.

SOA Analyzer identifies two types of program in this layer:

- *Client* programs receive and send screens.
- *Transitional* programs make navigation decisions.

The benefit of transitional programs is that they encapsulate navigation logic so that client programs can be confined to screen logic. A program CPROG1, for example, may receive SCREEN1 and pass exclusive control to a "traffic dispatcher" TPROG that analyzes the user request to decide which is the next screen to be presented to the user.

It may be that TPROG passes exclusive control back to CPROG1, which presents SCREEN1 again, with additional data that satisfies the request. Or it may be that TPROG decides to transfer control to CPROG2, which presents SCREEN2. In any case, navigation logic is confined to a single program.

Understanding Service-Layer Programs

Service-layer programs consist of all programs that satisfy a request without regard to screen communication or screen logic. Most programs in this layer interact in one way or another with the application's persistent data, but they may also perform certain pure logic related to computations or formatting. A service program may retrieve invoice information based on an invoice number, for example, or it may simply compute the number of business days between two dates.

SOA Analyzer identifies two types of program in this layer:

- *Frontier* programs with business level data manipulation logic.
- *Subfrontier* programs with both business level and technical data manipulation logic.

Frontier programs are called directly by client layer programs. Subfrontier programs consist of:

- *Indirect data access* programs, which access data through some other called service program, or perform computations or formatting.
- *Direct data access* programs, which access persistent data in VSAM files or databases.

Understanding Anomalous Programs

SOA Analyzer identifies four types of non-conforming, or *anomalous,* programs. These programs typically do not contain errors and may function satisfactorily. But they contain exceptions to an SOA approach that probably will cause problems in migration.

Client Programs with Data All programs that belong to the client layer should deal only with screen logic and navigation. Data access should be accomplished through programs belonging to the frontier and subfrontier layers, which in turn are called by these client programs.

This is always the case in a well-layered application, but in real life programmers may take short cuts and combine screen access with data access. A program that displays a customer inquiry screen, for example, may also read the customer's date of birth from a master file and calculate the customer's age, which is then displayed on the screen.

Client Programs Called It is natural to assume that all called programs are potential services, because a call is a request to the called program to perform a task. But if a called program also converses a screen, it cannot be made into a service.

Perhaps the most common scenario in which this anomaly arises is one in which a program written as a service encounters some kind of exception. The programmer finds it convenient to have the program immediately report the exception to the operator, by sending a screen with a message. By writing the program in this style, the programmer simplifies the code, but renders it unfit for a service, which by definition cannot converse a user interface.

Transitional Programs Called Transitional programs are dedicated to directing screen navigation, not to the fulfillment of a task. Although it may look like a potential service, a called traffic-directing program is not a good candidate for wrapping.

Called Program on a Path to a Screen Conversation This anomaly occurs when a client program calls a "service" program that in turn calls another client program. Although the program in between seems to be a service, because it is called and does not converse a screen, it is in fact impossible to turn it into a service, because one of its called programs converses a screen.

Starting SOA Analyzer

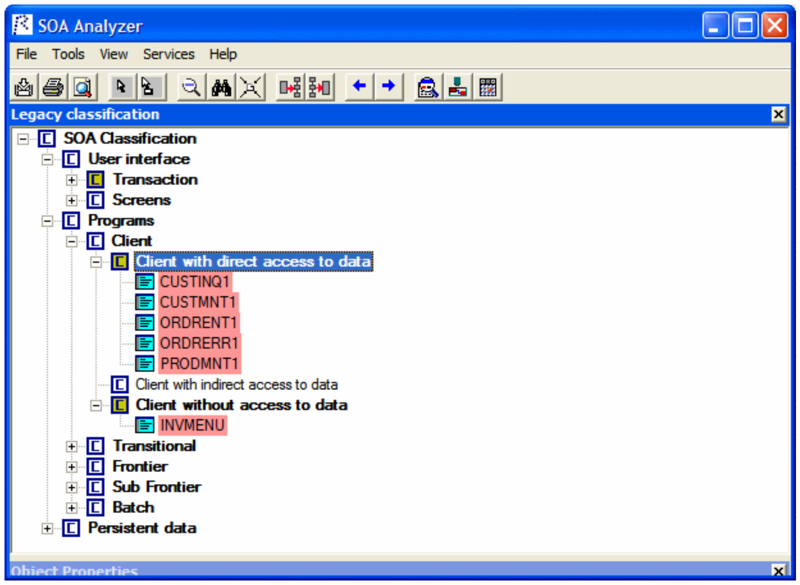
SOA Analyzer is certified for COBOL CICS with VSAM or DB/2 online applications. Batch applications or the batch portions of applications are captured and categorized, but cannot be enabled as services without first being converted to CICS.

To start SOA Analyzer:

1 In the Repository Browser, select the project you want to analyze and choose **SOA Analysis** in the **Extract** menu. The SOA Analyzer window opens. SOA Analyzer displays the analysis results for the workspace (Figure 8-2).

Tip: The panes in the SOA Analyzer window open automatically as needed. You can show or hide panes by selecting the appropriate choice in the **View** menu.

Figure 8-2 *SOA Analyzer Window with Analysis Results*



SOA Analyzer Basics

This section describes basic tasks you perform with objects in SOA Analyzer: viewing classifications, viewing source code and properties, assigning business names and tags, and calculating reuse and other metrics.

Viewing Object Classifications

The Legacy classification pane categorizes legacy objects in the classifications listed in Table 8-1. Anomalous programs are displayed with a red background, and classified as described in "Understanding Anomalous Programs" on page 8-4.

Table 8-1 *Program Classifications*

Category	Description
Batch with direct access to data	A program that: • Is not on an execution path that contains a screen. • Accesses a data store directly.
Batch with indirect access to data	A program that: • Is not on an execution path that contains a screen. • Calls a program that accesses a data store directly.
Batch without access to data	A program not on an execution path that contains a screen.
Client with direct access to data	A program that: • Sends or receives a screen. • Accesses a data store directly.
Client with indirect access to data	A program that:. • Sends or receives a screen • Calls a program that accesses a data store directly.
Client without access to data	A program that sends or receives a screen.

Table 8-1 *Program Classifications* (continued)

Category	Description
Frontier with direct access to data	A program that: • Is called (via a returning call like CALL or LINK) or XCTLed to by a client or transitional program. • Accesses a data store directly.
Frontier without access to data	A program that is called (via a returning call like CALL or LINK) or XCTLed to by a client or transitional program.
Subfrontier with direct access to data	A program that: • Is called (via a returning call like CALL or LINK) or XCTLed to by a frontier or subfrontier program. • Accesses a data store directly.
Subfrontier with indirect access to data	A program that: • Is called (via a returning call like CALL or LINK) or XCTLed to by a frontier or subfrontier program. • Calls a program that accesses a data store directly.
Subfrontier	A program that is called (via a returning call like CALL or LINK) or XCTLed to by a frontier or subfrontier program.
Transitional with direct access to data	A program that: • Is XCTLed to by a client or transitional program. • XCTLs to a client or transitional program. • Accesses a data store directly.

Table 8-1 *Program Classifications* (continued)

Category	Description
Transitional without access to data	A program that: • Is XCTLed to by a client or transitional program. • XCTLs to a client or transitional program.

Viewing Source

To view the source for an object, select the object in the Legacy classification, Diagram, or Services panes and choose **Show Source** in the right-click menu. The source is displayed in the Source pane.

The information available depends on the type of object you selected. You see only source code for a copybook, for example, but full HyperView information for a program. You can also view the properties of the selected object.

Choose the information you want to view for the object from the **Source** drop-down. For HyperView usage information, see *Analyzing Programs* in the workbench documentation set. For Properties pane usage information, see *Getting Started* in the workbench documentation set.

Viewing Object Properties

To view the properties of an object, select the object in the Legacy classification pane. The properties of the object are displayed in the Object Properties pane. The properties are described in Table 8-2.

Table 8-2 *Object Properties*

Property	Description
Name	The name of the object.
Type	The type of the object.
Layer	The layer in which the object is classified.

Table 8-2 *Object Properties*

Property	Description
Access type	The type of data access by the object.
Access mode	The mode of data access by the object, direct or indirect.
Anomaly	The type of anomaly, if any, that characterizes the object.

Assigning a Business Name and Description to an Object

To assign a business name and description to an object, select the object in the Legacy classification or Diagram panes and choose **Business Name** in the right-click menu. A dialog box opens, where you can enter the business name and business description. For more information on business names, see *Analyzing Programs* in the workbench documentation set.

Assigning a Tag to an Object

To assign a tag to an object, select the object in the Legacy classification or Diagram panes and choose **Assign Tag** in the right-click menu. The Entity Tag Browser window opens. Follow the instructions for assigning tags in "Assigning Tags to Objects" on page 2-27.

Saving Classifications as Tags

To create tags based on the names of the basic program classifications, choose **Save classification as tags** in the **Tools** menu.

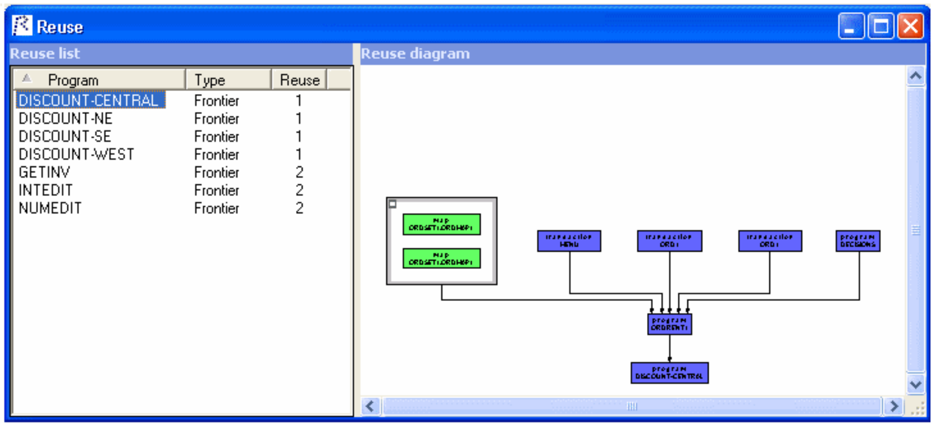
Determining Program Reuse

Generally speaking, the more often a program is called, the better a candidate it is to be made into a Web service. Use SOA Analyzer to determine the number of times a program is called.

To determine program reuse:

1. In the **Tools** menu, choose **Find reuse**. The Reuse window opens (Figure 8-3).

Figure 8-3 *Reuse Window*



Program	Type	Reuse
DISCOUNT-CENTRAL	Frontier	1
DISCOUNT-NE	Frontier	1
DISCOUNT-SE	Frontier	1
DISCOUNT-WEST	Frontier	1
GETINV	Frontier	2
INTEDIT	Frontier	2
NUMEDIT	Frontier	2

2 The Reuse column shows the number of times each called program is called. Select a program to view its reuse diagram.

Viewing SOA Measurements

The SOA measurements report displays a summary of SOA analysis results in HTML format. To display the report, choose **Compute measurements** in the **Tools** menu. The report opens in a Web browser window.

Note: The reuse measurement is calculated as follows: 100 * (number of calls - number of programs called) / number of calls

Working with Diagrams

This section describes basic tasks you perform with diagrams in SOA Analyzer. For guidance on how to use the diagrams to identify potential problems, see “Identifying Problems” on page 8-20. For complete information on diagrammer functions, see “Generating Diagrams” on page 2-4. Table 8-3 describes the notations used in the diagrams.

Tip: Click the plus sign (+) next to a node to expand diagrams that exceed the number of edges specified in the diagram options. To change the number of edges, see “Setting Diagram Options” on page 8-18.

Table 8-3 *Diagram Legend*

Notation	Description
Calls	Program calls another program.
D	Program deletes from a data store.
I	Program inserts into a data store.
Initiates	Transaction initiates a program.
R	Program reads a data store.
Receives	Program receives a screen.
Sends	Program sends a screen.
Starts	Program starts a transaction.
Trans	Program transfers control to a program.
U	Program updates a data store.

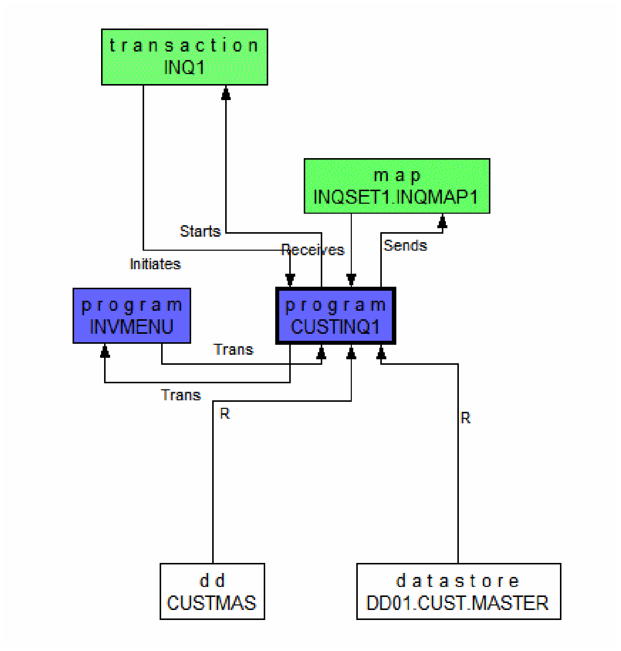
Drawing a Vertical Slice

A *vertical slice* shows the context in which an artifact is used, and how it uses other artifacts. The vertical slice for the client program CUSTINQ in Figure 8-4 shows that the program accesses data directly, in addition to sending and receiving screens.

To draw a vertical slice:

1 In the Legacy classification pane, select the artifact whose vertical slice you want to show and choose **Show vertical slice** in the right-click menu. The diagram of the vertical slice opens in the Diagrammer pane (Figure 8-4).

Figure 8-4 *Vertical Slice for CUSTINQ*



Drawing a Horizontal Slice

A *horizontal slice* shows how control passes between artifacts, usually through exclusive change of control or through screen navigation. Figure 8-5 shows the horizontal slice for the ORDSET1.ORDMAP1 screen.

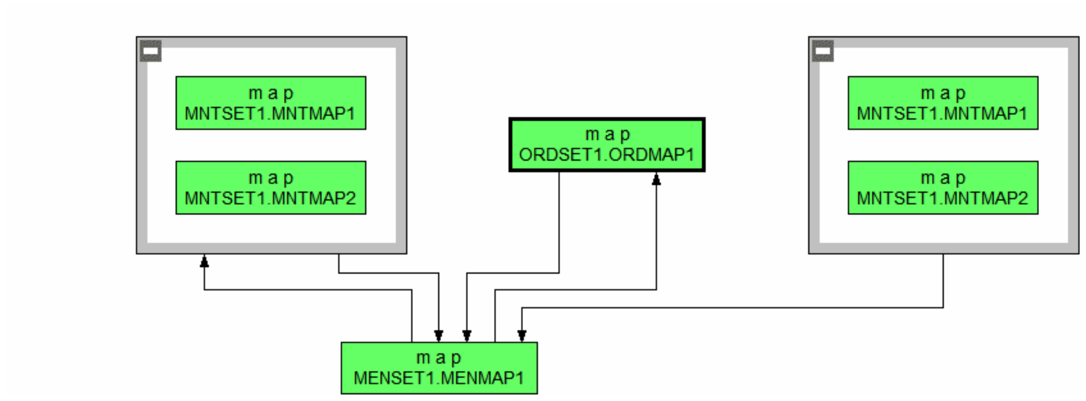
The gray block on the left shows the screens from which the operator could potentially move to ORDSET1.ORDMAP1, while the block on the right shows the screens to which the operator could potentially move from ORDSET1.ORDMAP1. "Potentially," for the simple reason that static analysis cannot discover the exact transitions that will occur at run time, because these may be controlled deep in a program or even in a file or table.

"Mining the User Interface and Navigation" on page 8-31 shows how you can specify the events that lead to transitions between screens at run time; view a complete screen navigation diagram that reflects the event specification; and export the screen layout and event specification to an XML file.

To draw a horizontal slice:

1 In the Legacy classification pane, select the artifact whose horizontal slice you want to show and choose **Show horizontal slice** in the right-click menu. The diagram of the horizontal slice opens in the Diagrammer pane (Figure 8-5).

Figure 8-5 *Horizontal Slice for ORDSET1.ORDMAP1 (Cropped)*



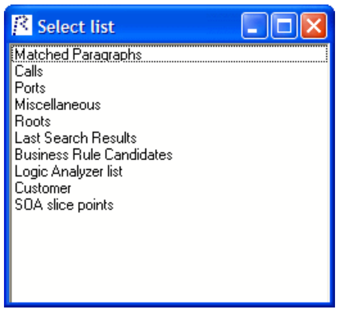
Locating Items of Interest with a Clipper List

Clipper lists record the results of program searches and analyses. When you *intersect* an SOA Analyzer diagram with a Clipper list, you make references to the search or analysis results available in the diagram. From the intersection, you can navigate directly to the location of constructs of interest in the Source pane. For more information on Clipper lists, see *Analyzing Programs* in the workbench documentation set.

To intersect a diagram with a Clipper list:

1 With a diagram open, choose **Intersect list** in the **Tools** menu. The Select list dialog opens (Figure 8-6).

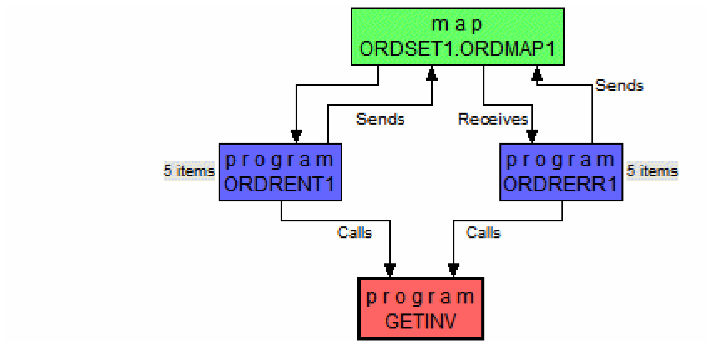
Figure 8-6 *Select List Dialog*



Matched Paragraphs
Calls
Ports
Miscellaneous
Roots
Last Search Results
Business Rule Candidates
Logic Analyzer list
Customer
SOA slice points

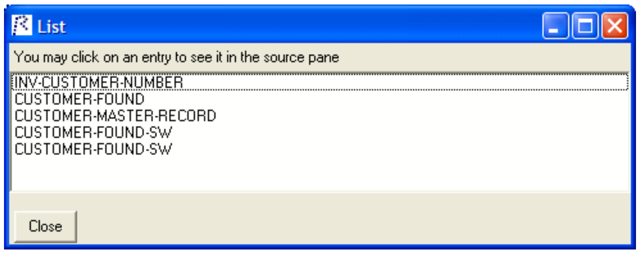
2 The Select List dialog shows every list available in Clipper. Click the list you want to intersect. For each diagram object that contains a list occurrence, SOA Analyzer inserts a label with the number of occurrences (Figure 8-7).

Figure 8-7 *Vertical Slice for GETINV with List Intersections (Cropped)*



3 In the diagram, double-click the label for the intersection you want to investigate. The List dialog opens (Figure 8-8).

Figure 8-8 *List Dialog*



You may click on an entry to see it in the source pane

```
INV-CUSTOMER-NUMBER
CUSTOMER-FOUND
CUSTOMER-MASTER-RECORD
CUSTOMER-FOUND-SW
CUSTOMER-FOUND-SW
```

4 The List dialog displays every list occurrence. Click an occurrence to navigate to it in the Source pane.

Showing Program Inner Structure

Program *inner structure* shows the flow of paragraphs in a program. The paragraphs are color-coded to indicate their function: data access, screen access, and the like (Figure 8-9).

That means you can quickly identify portions of anomalous programs to "slice out" as services in Application Architect. Once you have decided on the paragraphs to slice, you can assign them to a Clipper list for use in Architect. For more information on Application Architect, see *Creating Components* in the workbench documentation set.

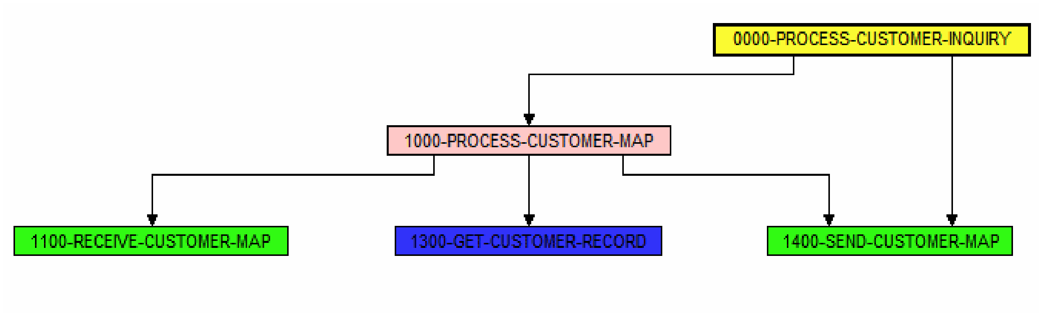
To show inner structure:

1 In the Legacy classification pane, select the program whose inner structure you want to show and choose **Show inner structure** in the right-click menu. The diagram of the inner structure opens in the Diagram pane (Figure 8-9).

Note: Only paragraphs relevant to the task of slicing out service code are shown in the diagram.

2 Select each paragraph you want to slice and choose **Create slice item** in the right-click menu. The paragraphs are included in a Clipper list called "SOA slice points" in the Architect category.

Figure 8-9 *Inner Structure of CUSTINQ*


0000-PROCESS-CUSTOMER-INQUIRY
1000-PROCESS-CUSTOMER-MAP
1100-RECEIVE-CUSTOMER-MAP
1300-GET-CUSTOMER-RECORD
1400-SEND-CUSTOMER-MAP

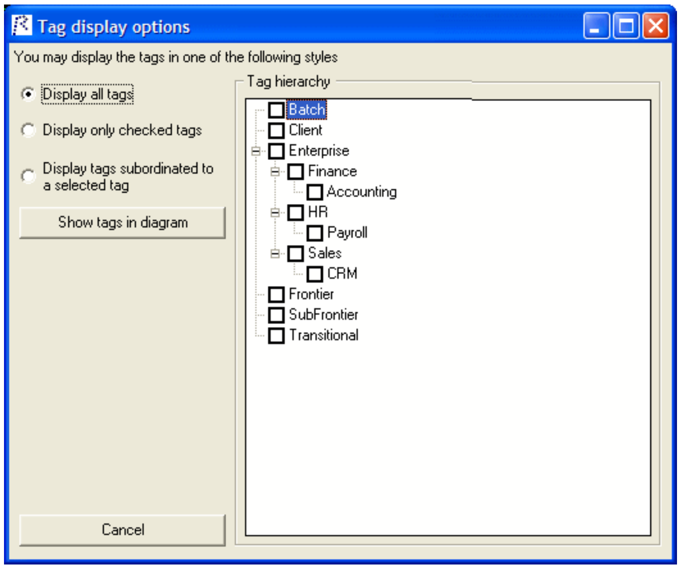
Showing Tags in a Diagram

Showing the tags assigned to objects in a diagram is a good way to understand their roles in the application. Figure 8-11 shows the vertical slice for the program CUSTINQ with tags displayed.

To show tags:

1 With a diagram open, choose **Display tags** in the **Tools** menu. The Tag display options dialog opens (Figure 8-10).

Figure 8-10 *Tag Display Options Dialog*

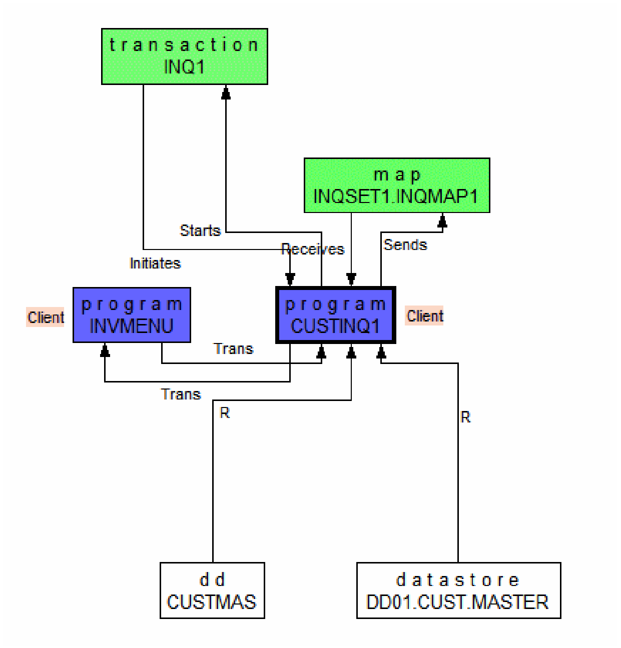


2 In the Tag display options dialog, click the plus sign (+) next to a tag to expand the tag hierarchy. Select:

- **Display all tags**, if you want to display all tags assigned to objects in the diagram.
- **Display only checked tags**, if you want to display only tags you place a check mark next to.
- **Display tags subordinated to a selected tag**, if you want to display only tags below the selected tag in the hierarchy (direct children only). Select the tag by clicking it (not by checking it).

When you are satisfied with your choices, click **Show tags in diagram**. For each object assigned a tag, SOA Analyzer inserts a label with the name of the tag (Figure 8-11).

Figure 8-11 *Vertical Slice for CUSTINQ with Tags*



Setting Diagram Options

Diagram options determine the colors of objects in diagrams, whether the object type is included in the object caption, and the maximum number of edges.

To set diagram options:

1 In the **View** menu, choose **Legend**. The Legend for the diagram color scheme is displayed. On the Legend pane, click **Options**. The Options dialog opens above the Legend pane (Figure 8-12).

2 In the Diagram Colors list box, select the object type whose color you want to change. The current background color is displayed in the **Edge Color** drop-down. Click the adjacent ▾ button to edit the color of the object type. A standard Windows color control is displayed. Use the **Palette** tab to select the color from the Windows palette. Use

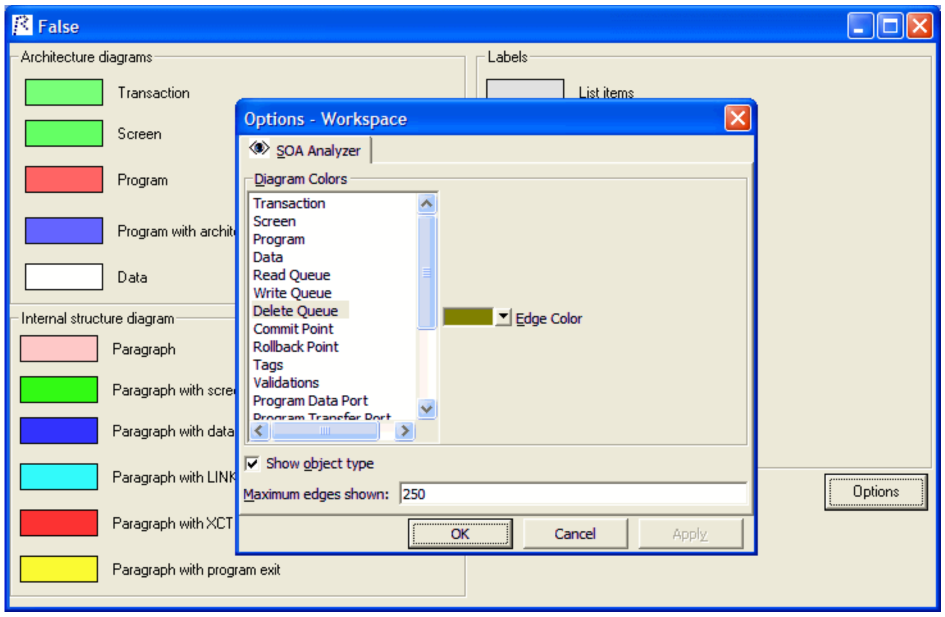
the **System** tab to match the color with the color of standard Windows elements.

3. To exclude the object type from the object caption, select the object type in the Diagram Colors list box and deselect **Show object type**.

4. To change the maximum number of edges shown in the diagram, enter the new value in the **Maximum edges shown** field.

Note: Restricting the number of edges ensures against very long load times for large diagrams. Even when the number of edges exceeds the limit specified in the diagram options, you can still expand the diagram by clicking the plus sign (+) next to the nodes at the diagram boundary.

Figure 8-12 *Options Dialog Above Legend Pane*



☑ Show object type

Maximum edges shown: 250

Identifying Problems

This section describes how to use SOA Analyzer diagrams to identify problems in legacy programs that may need to be corrected before the programs can be exposed as Web services. For information on how to perform basic tasks in SOA Analyzer diagrams, see "Working with Diagrams" on page 8-10.

Identifying Missing Validation Rules

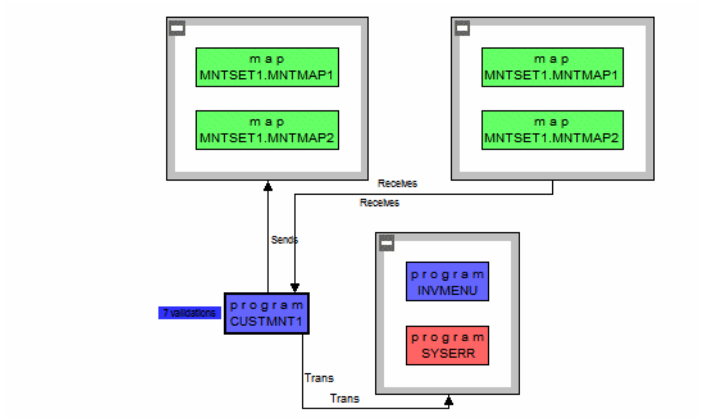
Before you can expose a called program as a Web service, you must ensure that its input data is validated either before invocation or inside the program. Failure to validate data may result in service failure.

When you intersect an SOA Analyzer diagram with a list of validation rules generated in Business Rule Manager, you make references to the rules available in the diagram. From the intersection, you can investigate the rules in the Source pane. For more information on validation rules, see *Analyzing Programs* in the workbench documentation set.

To intersect a diagram with validation rules:

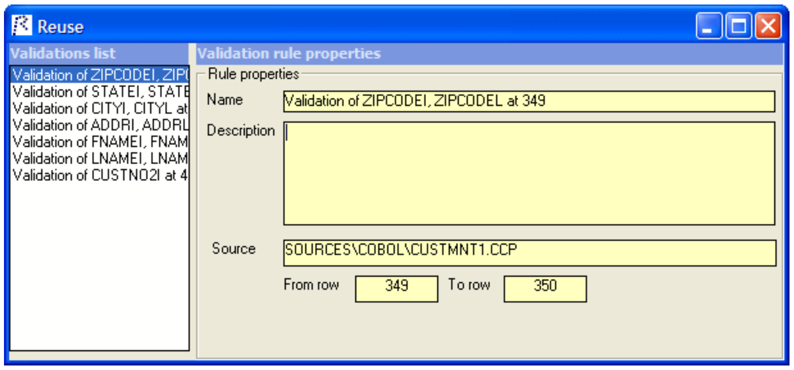
1 With a diagram open, choose **Intersect validations** in the **Tools** menu. For each program that contains a validation rule, SOA Analyzer inserts a label with the number of rules (Figure 8-13).

Figure 8-13 *Vertical Slice for CUSTMNT1 with Validation Rule Intersection (Cropped)*



2 In the diagram, double-click the label for the intersection you want to investigate. The Validation Rule dialog opens (Figure 8-14).

Figure 8-14 *Validation Rule Dialog*



Validation of ZIPCODEI, ZIPC
Validation of STATEI, STATE
Validation of CITYI, CITYL at
Validation of ADDRI, ADDRL
Validation of FNAMEI, FNAM
Validation of LNAMEI, LNAM
Validation of CUSTNO2I at 4

Name: Validation of ZIPCODEI, ZIPCODEL at 349
Description:
Source: SOURCES\COBOL\CUSTMNT1.CCP
From row: 349 To row: 350

3 In the Validation Rule dialog, select a rule to display its name, description, and location. Use the location information to navigate to the code for the rule in the Source pane.

Identifying Queue Access

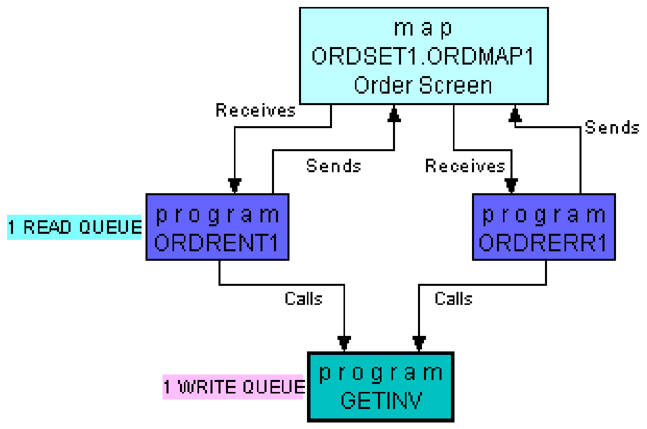
Called and caller programs in a CICS environment ordinarily communicate through the CICS COMMAREA. But it is not uncommon, especially when dealing with large amounts of data, for called programs to write the data into a temporary storage queue, which the caller reads from. Because most CICS adapters are equipped only to deal with data passing through the COMMAREA, it's important to identify queue usage and replace it with another storage method before exposing a called program as a Web service.

When you intersect an SOA Analyzer diagram with reads or writes to a storage queue, you make references to the operations available in the diagram. From the intersection, you can investigate the operations in the Source pane.

To intersect a diagram with reads or writes to a queue:

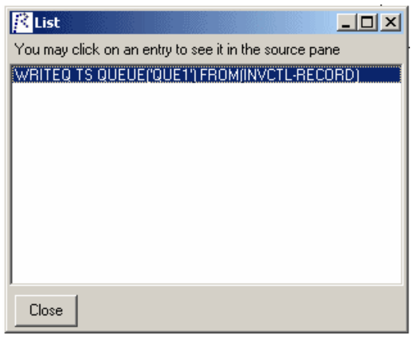
1 With a diagram open, choose **Intersect queues** in the **Tools** menu. For each program that reads or writes to a queue, SOA Analyzer inserts a label with the number of reads or writes (Figure 8-15).

Figure 8-15 *Vertical Slice for GETINV with Queue Intersection (Cropped)*



2 In the diagram, double-click the label for the intersection you want to investigate. The List dialog opens (Figure 8-16).

Figure 8-16 *Queue List Dialog*



You may click on an entry to see it in the source pane

WRITEQ TS QUEUE('QUE1') FROM(INVCTL-RECORD)

3 In the List dialog, click a read or write statement to navigate to its code in the Source pane.

Identifying Transaction Sync Points

When a CICS application involves a number of related updates to tables or files, it ensures data integrity by committing the changes only after every update has been performed successfully, and rolling them back if any of the updates fails.

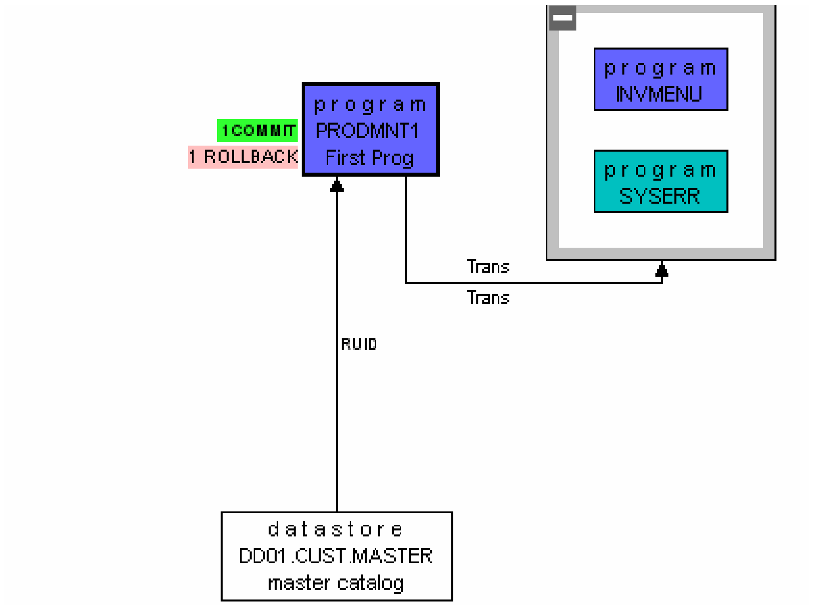
In some legacy systems, these updates may be performed by different programs. An item marked for shipping by one program may be taken out of inventory by another, for example. Wrapping these programs as Web services without understanding how the operations they perform are synchronized will cause the data to be compromised if one of the programs fails.

To prevent this, you need to understand the boundaries of the transaction, by analyzing the sync points at which commits or rollbacks are performed. When you intersect an SOA Analyzer diagram with transaction sync points, you make references to the sync points available in the diagram. From the intersection, you can investigate the sync points in the Source pane.

To intersect a diagram with transaction sync points:

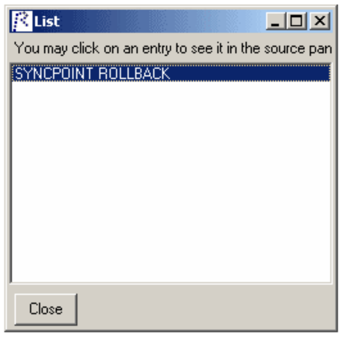
1. With a diagram open, choose **Intersect sync points** in the **Tools** menu. For each program that performs a commit or rollback, SOA Analyzer inserts a label with the number of commits or rollbacks (Figure 8-17).

Figure 8-17 *Vertical Slice for PRODMNT1 with Sync Point Intersection (Cropped)*



2 In the diagram, double-click the label for the intersection you want to investigate. The Sync Points List dialog opens (Figure 8-18).

Figure 8-18 *Sync Points List Dialog*



You may click on an entry to see it in the source pan

SYNCPOINT ROLLBACK

3 In the Sync Points List dialog, click a sync point to navigate to the code for the commit or rollback in the Source pane.

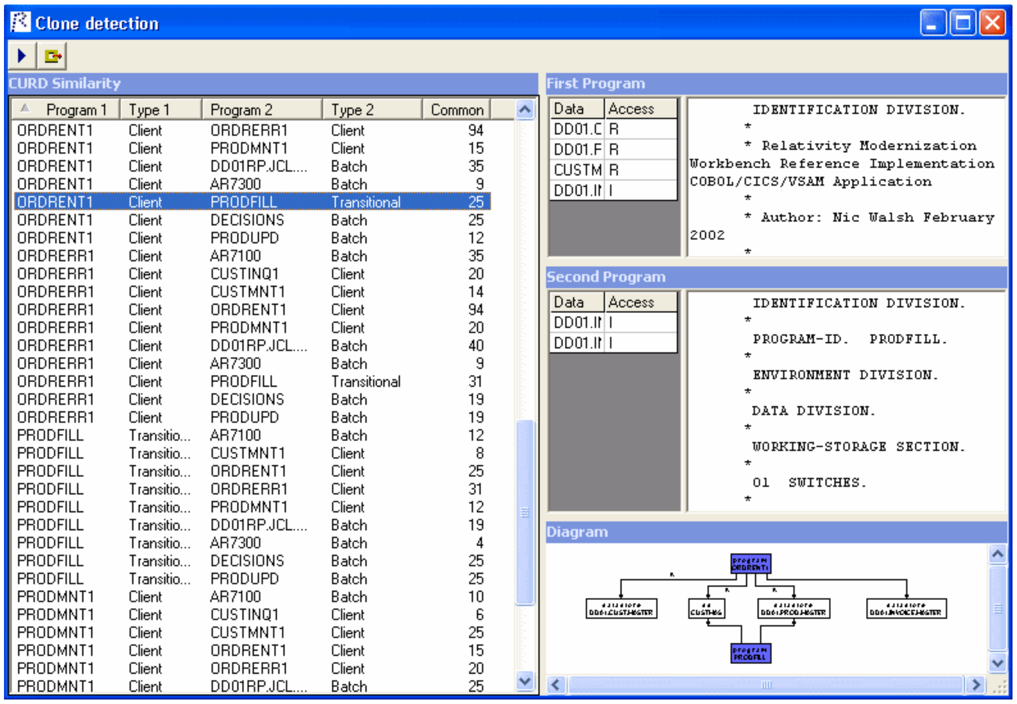
Identifying Duplicated CRUD Operations

In many legacy applications, more than one program performs essentially the same operation against a data store. A first cut at SOA Analysis might suggest that one of these programs is a good Web service candidate, when a deeper analysis would show that another program is better-suited. Use the SOA Analyzer *clone detection* utility to ensure that you know all the programs that perform the same operation against a data store before deciding which to promote as a Web service.

To detect clones:

1. In the **Tools** menu, choose **Detect clones**. The Clone detection window opens. Click the ▶ button to start clone detection. SOA Analyzer populates the Clone detection window with the results (Figure 8-19).

Figure 8-19 *Clone Detection Window*



Program 1	Type 1	Program 2	Type 2	Common
ORDRENT1	Client	ORDRERR1	Client	94
ORDRENT1	Client	PRODMNT1	Client	15
ORDRENT1	Client	DD01RP.JCL....	Batch	35
ORDRENT1	Client	AR7300	Batch	9
ORDRENT1	Client	PRODFILL	Transitional	25
ORDRENT1	Client	DECISIONS	Batch	25
ORDRENT1	Client	PRODUPD	Batch	12
ORDRERR1	Client	AR7100	Batch	35
ORDRERR1	Client	CUSTINQ1	Client	20
ORDRERR1	Client	CUSTMNT1	Client	14
ORDRERR1	Client	ORDRENT1	Client	94
ORDRERR1	Client	PRODMNT1	Client	20
ORDRERR1	Client	DD01RP.JCL....	Batch	40
ORDRERR1	Client	AR7300	Batch	9
ORDRERR1	Client	PRODFILL	Transitional	31
ORDRERR1	Client	DECISIONS	Batch	19
ORDRERR1	Client	PRODUPD	Batch	19
PRODFILL	Transitio...	AR7100	Batch	12
PRODFILL	Transitio...	CUSTMNT1	Client	8
PRODFILL	Transitio...	ORDRENT1	Client	25
PRODFILL	Transitio...	ORDRERR1	Client	31
PRODFILL	Transitio...	PRODMNT1	Client	12
PRODFILL	Transitio...	DD01RP.JCL....	Batch	19
PRODFILL	Transitio...	AR7300	Batch	4
PRODFILL	Transitio...	DECISIONS	Batch	25
PRODFILL	Transitio...	PRODUPD	Batch	25
PRODMNT1	Client	AR7100	Batch	10
PRODMNT1	Client	CUSTINQ1	Client	6
PRODMNT1	Client	CUSTMNT1	Client	25
PRODMNT1	Client	ORDRENT1	Client	15
PRODMNT1	Client	ORDRERR1	Client	20
PRODMNT1	Client	DD01RP.JCL....	Batch	25

First Program

Data	Access
DD01.C	R
DD01.F	R
CUSTM	R
DD01.I	I

```
       IDENTIFICATION DIVISION.
      *
      * Relativity Modernization
Workbench Reference Implementation
COBOL/CICS/VSAM Application
      *
      * Author: Nic Walsh February
2002
      *
```

Second Program

Data	Access
DD01.I	I
DD01.I	I

```
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID.   PRODFILL.
      *
       ENVIRONMENT DIVISION.
      *
       DATA DIVISION.
      *
       WORKING-STORAGE SECTION.
      *
       01  SWITCHES.
      *
```

2 Each row in the CRUD Similarity pane compares one program in the project with another to determine the extent to which they perform similar CRUD operations. The Common value is the percentage of CRUD operations they perform that are similar.

Note: CRUD similarity is calculated from the data operations perspective only. The logic used in the operations may differ.

3 Select a row to view the CRUD operations performed by each program in the righthand pane. You can also view the source for each program and a diagram describing their interaction.

Creating Web Services

This section looks at some guidelines for choosing Web service candidates, then shows you how to create and export candidates as Web services.

Identifying Potential Web Services

Both functional and technical considerations will affect your choice of Web service candidates. To publish an order entry process as a Web service, for example, you need to decide whether to wrap the program handling the client order entry screen, or some other program that actually services the order entry process.

The classifications provided by SOA Analyzer (Table 8-1) can help you make this determination. Here are some guidelines worth considering:

- A client program is not typically the most desirable choice for SOA publishing, since it contains mostly screen handling and validation logic.
- A transitional program does not typically include business logic. It mainly serves as a connector between client programs.
- A frontier program is usually the best candidate for coarse-grained SOA publishing, since it resides at the top layer of server-level programs and is isolated from client-level validations and pop-up screens. It is therefore most likely to contain, or point to, meaningful

business logic without the distraction of user interface and screen validation logic.

- A subfrontier program may also be safely published, although if it resides very close to the data, it may not contain services as meaningful as desired.
- Anomalous programs may not require much effort to re-architect. A client data program, for example, may actually be a business logic program that contains a pop-up screen. Once you change the program to remove the pop-up logic, shifting it to the application server level in the new application, you can safely publish it as a Web service.

Creating and Exporting Web Services

Once you have identified appropriate candidates for reuse, you can use SOA Analyzer to create wrappers that expose them to other applications as Web services. You typically use SOA Analyzer to perform “data port wrapping” of programs in the data abstraction layer.

The wrapping allows external applications to call the programs in the same way they are called in the native environment. The wrapping infrastructure knows which programs to call and how to convert program input/output parameters into correctly formed XML messages that can be consumed by an SOA architecture.

Defining a Web Service

Initially a Web service is just a shell. This section describes how to name and document the Web service, and how to assign candidates, or *seeds,* to the service.

To define a Web service:

1. In the **Services** menu, choose **New Service**. A placeholder for the new service is added to the Services pane. To name the service, click in the **Name** field in the Service properties pane to open a text box where you can enter the service name. To delete the service, select it and choose **Delete** in the right-click menu.

2. To enter a description of the service, click Documentation in the service tree. In the Service properties pane, click in the **Description** field to open a text box where you can enter the service description.
3. To add a legacy program to a service, select the service in the Services pane. In the Legacy classification or Diagram pane, select the legacy program you want to add and choose **Add to Service** in the right-click menu. SOA Analyzer adds the program to the service tree as a seed. You can add as many seeds as necessary.
4. In the Services pane, select a seed to view its properties in the Service properties pane. Click in a field in the Service properties pane to open a text box where you can edit service properties related to the seed.

Importing Data Element Synonyms

The tree for a seed displays the data elements for each port in the program. In many cases, the actual parameters of the program are defined in a *synonym* for the data element. A data port that appears as DFHCOMMAREA in the linkage section, for example, may immediately be moved to another 01 structure that actually details the parameters of the program. You need to import this second structure as a synonym.

To define a synonym for a data element:

1. In the Services pane, select a data element and choose **View in source** in the right-click menu to navigate to its declaration in the Source pane. A declaration such as `DFHCOMMAREA PIC X(119)` will almost certainly have a synonym.
2. Select the declaration for which you want to define a synonym in the Source pane and choose **Instances** in the right-click menu. The Declaration Properties window opens.
3. In the Declaration Properties window, select an instance to navigate to it in the Source pane. Choose **Declaration** in the right-click menu to view its declaration.
4. When you are satisfied that a structure is an appropriate synonym, select it in the Source pane, then select the data element for the seed port in the Services pane and choose **Import synonym** in the right-click menu. You can import as many synonyms as necessary.

5 SOA Analyzer adds the structure to the seed tree, with the data items at the lowest level automatically selected for inclusion in the service. Items selected for inclusion are shown in **bold**. If you want to include a higher-level item instead of the lower-level items (a field with a complete date, for example, rather than the year, month, and day fields that constitute the date), select the higher-level item and choose **Include** in the right-click menu.

Editing Data Elements

Select a data element in the Services pane to view its properties in the Service properties pane. To edit a data element, select it and choose:

- **View glossary** in the right-click menu to assign a business name and description to the data element. A dialog box opens, where you can enter the business name and business description. For more information on business names, see *Analyzing Programs* in the workbench documentation set.
- **Make input** in the right-click menu to define the data element as an input field.
- **Make output** in the right-click menu to define the data element as an output field.
- **Make input/output** in the right-click menu to define the data element as an input/output field.

Tip: Needless to say, it makes little sense to redefine an input field as an output field, or vice versa. But you may have occasion to define an input or output field as an input/output field.

- **Include** in the right-click menu to include the data element in the service. Items selected for inclusion are shown in **bold**.
- **Include children** in the right-click menu to include the children of the data element in the service.
- **Exclude** in the right-click menu to exclude the data element from the service.
- **Remove** in the right-click menu to delete the data element from the seed tree.

Generating and Exporting the Wrapper for a Web Service

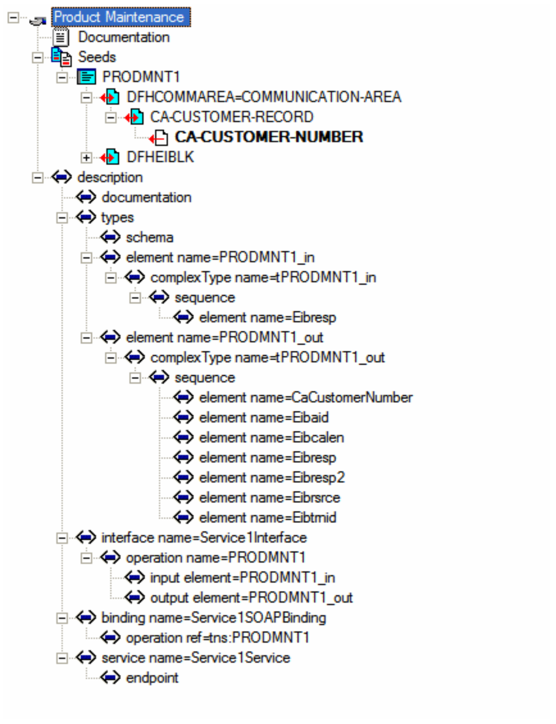
Once you have chosen the structures that will serve as the input and output messages for the Web service, you can generate and export the Web Service Description Language (WSDL) message syntax for the service.

To generate and export the wrapper for a Web service:

1 In the Services pane, select the service you want to wrap and choose **Generate WSDL** in the right-click menu. SOA Analyzer generates the WSDL for the wrapper and displays it in tree form in the Services pane (Figure 8-20).

2 To export the WSDL to a file, select the service and choose **Export WSDL** in the right-click menu. A Select Folder dialog opens, where you can specify the location of the file. The file has the same name as the service.

Figure 8-20 *Service Definition*



```
Product Maintenance
  Documentation
  Seeds
    PRODMNT1
      DFHCOMMAREA=COMMUNICATION-AREA
        CA-CUSTOMER-RECORD
          CA-CUSTOMER-NUMBER
      DFHEIBLK
  description
    documentation
    types
      schema
      element name=PRODMNT1_in
        complexType name=tPRODMNT1_in
          sequence
            element name=Eibresp
      element name=PRODMNT1_out
        complexType name=tPRODMNT1_out
          sequence
            element name=CaCustomerNumber
            element name=Eibaid
            element name=Eibcalen
            element name=Eibresp
            element name=Eibresp2
            element name=Eibrsrce
            element name=Eibtmid
    interface name=Service1Interface
      operation name=PRODMNT1
        input element=PRODMNT1_in
        output element=PRODMNT1_out
    binding name=Service1SOAPBinding
      operation ref=tns:PRODMNT1
    service name=Service1Service
      endpoint
```

Mining the User Interface and Navigation

SOA Analyzer provides facilities that make it easy to generate a user interface and navigation scheme on another platform. This section describes how to:

- Specify the events that lead to transitions between screens at run time.
- View a complete screen navigation diagram that reflects the event specification.
- Export the screen layout and event specification to an XML file.

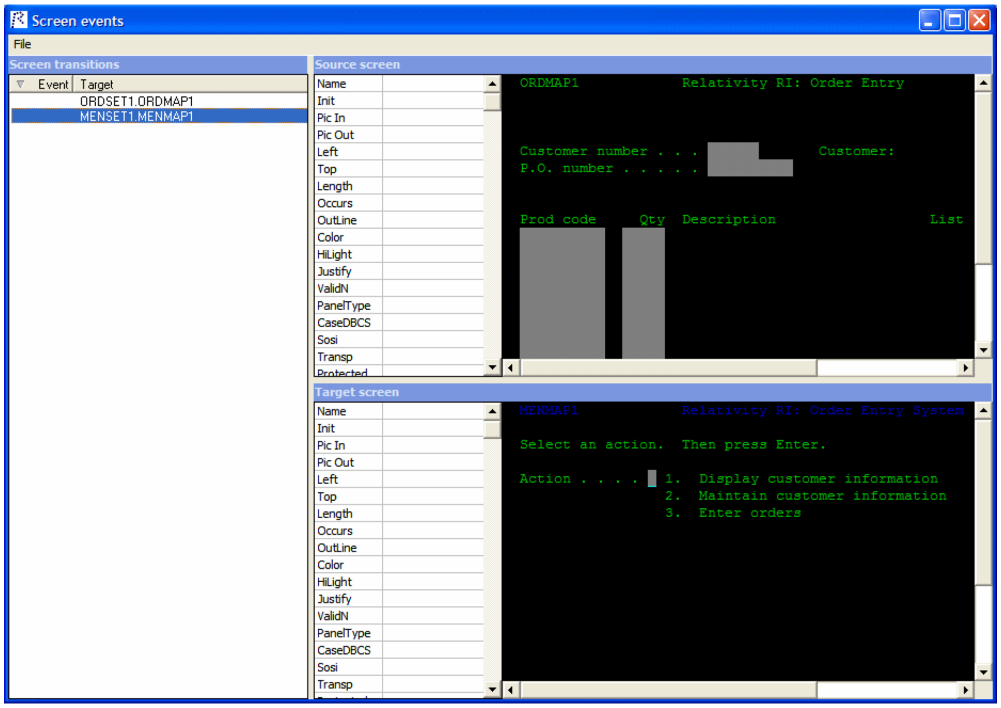
Specifying Screen Events

Static analysis cannot discover the exact transitions between screens that occur at run time, because these may be controlled deep in a program or even in a file or table. You need to view the screen source to determine the intended runtime flow, then specify the events that lead to transitions between screens at run time.

To specify screen events:

1. In the Legacy classification or Diagram pane, select the screen for which you want to specify events and choose **Specify screen events** in the right-click menu. The Screen events window opens (Figure 8-21).
2. The Screen transitions pane lists every transition between the selected screen and a target screen, including transitions to itself. Select a transition to view the source and properties for the selected screen in the Source screen pane, and the source and properties for the target screen in the Target screen pane.
3. To specify an event (“Return,” for example, to indicate that the operator returns from the source screen to the target screen), click in the Event column for the transition to open a text box where you can enter the event name.
4. Repeat these steps for the remaining screens in the application.

Figure 8-21 *Screen Events Window*



Event	Target
	ORDSET1.ORDMAP1
	MENSET1.MENMAP1

```
ORDMAP1            Relativity RI: Order Entry


Customer number . . .               Customer:
P.O. number . . . . .


Prod code     Qty  Description                  List
```

```
MENMAP1            Relativity RI: Order Entry System

Select an action.  Then press Enter.

Action . . . .   1.  Display customer information
                 2.  Maintain customer information
                 3.  Enter orders
```

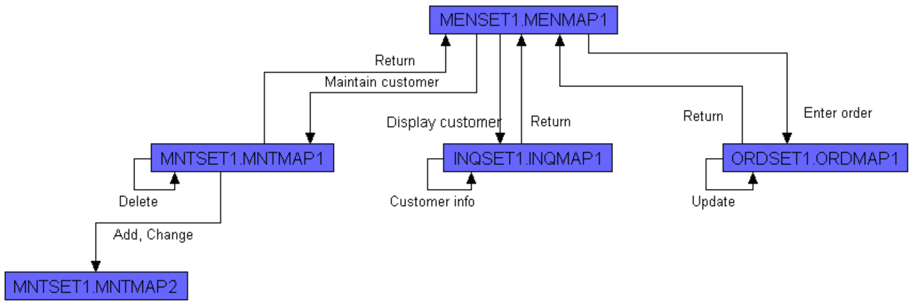
Generating a Screen Navigation Diagram

Once you have completely specified the screen events that define the transitions between screens, you can generate a screen navigation diagram that reflects the event specification.

To generate a screen navigation diagram:

1 In the **Tools** menu, choose **Show navigation**. The screen navigation diagram opens (Figure 8-22).

Figure 8-22 *Screen Navigation Diagram*



Exporting the User Interface and Navigation

When you are sure that you have understood the screen flow for the application, you can export the layout and event specification for the screens to XML files. Based on the sample XSLT provided with SOA Analyzer, you can write your own XSLT to convert the XML files to the HTML in which the screens will be rendered on a modern platform.

To export the user interface and navigation:

1 In the **File** menu, choose **Export user interfaces**. A Select Folder dialog opens, where you can specify the location of the XML files. An XML file is created for each screen in the project. The file has the same name as the screen.

Tip: You can find the sample XSLT provided with SOA Analyzer in the Modernization Workbench SOAInstall folder.

What’s Next?

That completes your tour of the workbench project analysis tools! Now you’re ready to start using the program analysis tools, collectively called HyperView, to view programs interactively and perform program analysis in stages. *Analyzing Programs* in the workbench documentation set describes HyperView.

Repository Exchange Protocol Syntax

A

The Repository Exchange Protocol (RXP) is an XML-based API that you can use to interact with application-level information in the workspace repository. This appendix describes the RXP query syntax.

Query Syntax

An RXP query consists of the tags and attributes described in this section.

Query

```
<query [name='QueryName']> { Object } </query>
```

Object

```
 <object [global='false']>
   [ <objecttype> ObjectTypeCondition </objecttype> ]
   [ <cond> Condition </cond> ]
```

```
    [ <fetchtype as='FieldName'/> ]
    [ <fetchid as='FieldName'/> ]
    [ <fetchdisplay as='FieldName'/> ]
    [ <fetchorigin as='FieldName'/> ]
    [ <fetchsource as='FieldName'/> ]
    [ <fetchsize as='FieldName'/> ]
    { <fetchconst type='FieldType' value='Constant'
      as='FieldName'/> }
    { <fetchattr attr='AttrName' as='FieldName'/> }
    [ <related [countas='FieldName'][optional='true']>
      RelatedSpec
      </related>]
</object>
```

ObjectTypeCondition

```
  <typeset flag='FlagName' [negate='true']/>
| <type name='EntityName' [negate='true']/>
| <and [negate='true']> { ObjectTypeCondition }
  </and>
| <or [negate='true']> { ObjectTypeCondition }
  </or>
```

RelatedSpec

```
[ <reltype> RelTypeCondition </reltype> ]
[ <cond> Condition </cond> ]
[ <fetchtype as='FieldName'/> ]
{ <fetchattr attr='AttrName' as='FieldName'/> }
Object
```

RelTypeCondition

```
  <relset flag='RelFlagName' [negate='true']
     [incoming='true']/>
| <rel name='RelationName' [negate='true']/>
| <and [negate='true']> { RelTypeCondition } </and>
| <or [negate='true']> { RelTypeCondition } </or>
```

Condition

```
  <attr name='AttrName' op='Operation'
     arg='Argument' [negate='true']/>
| <hasrelated [negate='true']> RelatedSpec
  </hasrelated>
| <id equals='Integer' [negate='true']/>
| <id in='Integer{,Integer}' [negate='true']/>
| <source equals='String' [negate='true']/>
| <source in='String{,String}' [negate='true']/>
| <origin equals='SID' [negate='true']/>
| <origin in='SID{,SID}' [negate='true']/>
| <and [negate='true']> { Condition } </and>
| <or [negate='true']> { Condition } </or>
```

EntityName

The name of a repository entity.

AttrName

The name of an entity attribute.

FieldName

The field name of the returned record set.

Operation

= | <> | > | >= | < | <= | like | in | between

FlagName

LEGACY | PROGRAMCODE | SYSTEM | KNOWLEDGE | GENERATED | EXTRACT | COMPOSITE

RelFlagName

REFER | USE | GENERATE | PRODUCE

RelationName

As defined in the repository metamodel.

Argument

The argument to the operation that depends both on argument type and operation.

QueryName

A string.

Example 1

This example queries the repository for the object ID and parse status of the GSS.cbl source file:

```
<query name="Select a COBOL object by name">
  <object>
    <objecttype>
      <type name="COBOL"/>
    </objecttype>
    <fetchid as="ID"/>
      <fetchattr name="ParseStatus" as="Parsed"/>
    <cond>
      <attr name="Name" op="=" arg="GSS.cbl"/>
    </cond>
  </object>
</query>
```

Example 2

This example queries the repository for copybooks used in three Cobol programs:

```
<query name="Find COPYBOOKs used in given programs">
  <object>
    <objecttype>
      <type name="COBOL"/>
    </objecttype>
    <cond>
      <attr name="Name" op="in"
       arg="'GSS1.CBL','GSS2.CBL','GSS3.CBL'"/>
    </cond>
    <related>
      <reltype>
         <relset flag="USE"/>
      </reltype>
      <object>
        <fetchid as="ID2"/>
        <fetchtype as="ObjectType2"/>
        <fetchdisplay as="ObjectName2"/>
      </object>
    </related>
  </object>
</query>
```

A-6 Repository Exchange Protocol Syntax

Example 2

Common Diagramming Features

B

This appendix describes features shared by all the workbench diagramming tools, except the Diagrammer itself. Use these features to edit diagrams, save diagrams in standard formats, print diagrams, and more.

Zooming

Use the slider on the tool bar in the lower lefthand corner of the tool window to zoom in or out on a diagram. Click the 1:1 button on the tool bar to restore the 100% zoom level.

Printing Diagrams

Click the button on the tool bar in the lower lefthand corner of the tool window to print the diagram. The Print Preview window opens, where you can change the zoom factor for the diagram to control the number of pages in the print job. Click **OK**.

Saving Diagrams

Click the button on the tool bar in the lower lefthand corner of the tool window to save a diagram to BED, bitmap, JPEG, Visio, Visio XML, DOT, or EMF.

Note: Visio 2002 must be installed to save a diagram to Visio. Visio 2002 is not required to save to Visio XML.

Copying Diagrams

Click the button on the tool bar in the lower lefthand corner of the tool window to copy the diagram to the clipboard in EMF format. You can paste the diagram from the clipboard to a document in a third-party tool such as Word.

Using the Diagram Editor

Use the built-in Diagram Editor to edit diagrams, change the appearance of diagrams, cut-and-paste from a diagram, and other tasks.

Opening a Diagram in the Diagram Editor

You can invoke the Diagram Editor from any tool that displays a diagram, or by double-clicking `BEDit.exe` in the Modernization Workbench Bin directory. If you open the Diagram Editor from within another tool, the Editor creates a copy of the diagram on view in the tool.

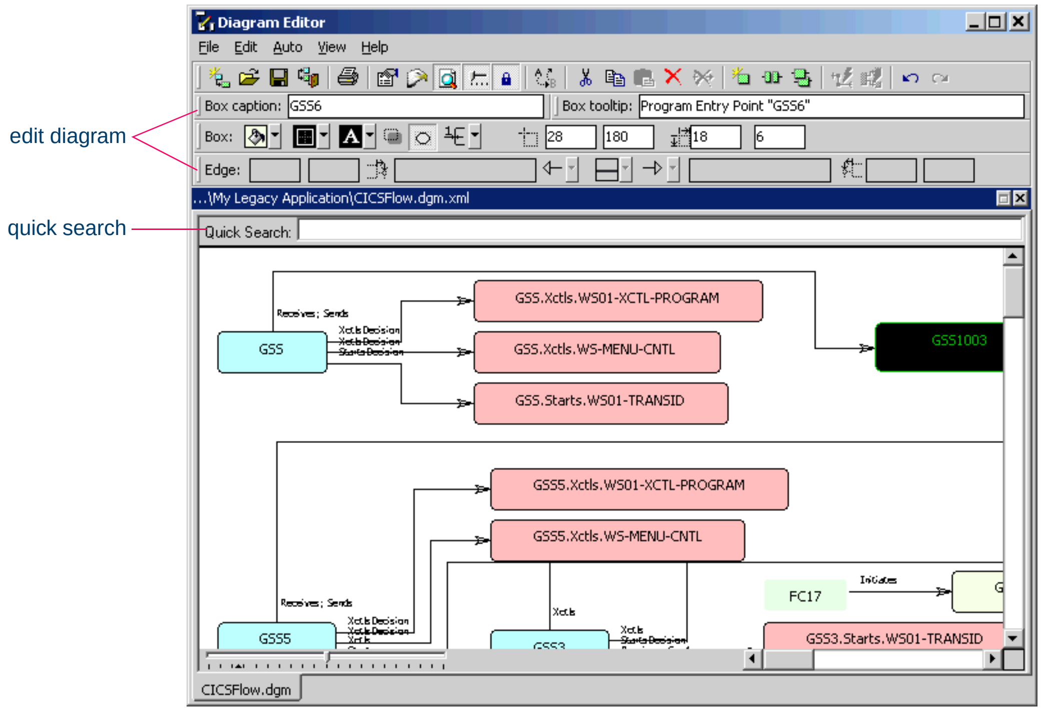
To open a diagram in the Diagram Editor:

1A If the Diagram Editor is not open, click the button on the tool bar in the lower lefthand corner of the tool window. The diagram on view in the tool opens in the Diagram Editor (Figure B-1).

1B If the Diagram Editor is already open, choose **Open** in the **File** menu. The Open diagram dialog appears. Select the diagram you want to open and click **Open**.

Tip: Choose **New Diagram** in the **File** menu to create a new blank diagram. You can paste portions of an existing diagram into the blank diagram, as described in "Copying Objects" on page B-6.

Figure B-1 *Diagram Editor Window*



Box caption: GSS6
Box tooltip: Program Entry Point "GSS6"
28 180 18 6
...\My Legacy Application\CICSFlow.dgm.xml

Searching for Objects in the Diagram Editor

The Diagram Editor offers facilities for navigating to general locations in a diagram, or to specific objects. You can use the search and replace feature to modify captions or tool tips globally.

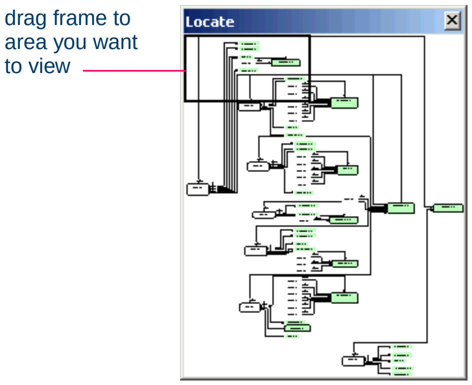
Using the Locate Facility

Use the Locate facility to navigate to general locations in a diagram. To invoke the Locate facility, click **Locate** in the **View** menu. The Locate window opens (Figure B-2).

The Locate window displays the entire diagram on view in the Diagram Editor main window. A frame surrounds the portion of the diagram visible in the main window.

Drag the frame to the area of the diagram you want to view. The Diagram Editor displays the selected area in the main window.

Figure B-2 *Locate Window*



Using the Quick Search Facility

Use the Quick Search facility to navigate to objects in a diagram. To invoke Quick Search, choose **Search pane** in the **View** menu. The Quick Search facility opens at the top of the window.

Enter the text for the search in the **Quick Search** field. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

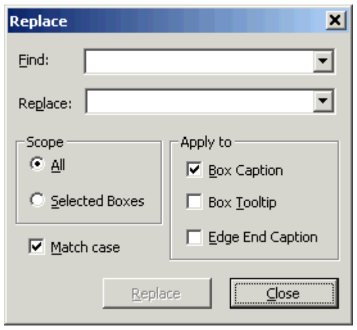
Using the Search and Replace Facility

Use the Search and Replace facility to modify captions or tool tips globally. The Diagram Editor modifies the diagram only, not the repository.

To search for and replace text in a diagram:

1 In the **Edit** menu, choose **Replace**. The Replace window opens (Figure B-3).

Figure B-3 *Replace Window*



2 Enter the text you want to replace in the **Find** field. Enter the text you want to substitute in the **Replace** field.

3 Select **Match case** if you want to match the case of the entered text.

4 You can apply the change to any combination of object captions, relationship captions, or tool tips by selecting the appropriate choices in the Apply to pane. If you apply the change to object captions, select the appropriate choice in the Scope pane to apply the change to all the objects that contain the text or only the selected objects that contain the text.

5 Click **Replace** to run the operation and dismiss the Replace window.

Selecting Objects in the Diagram Editor

Select an object or relationship in the Diagram Editor by clicking it. Deselect an object by clicking on white space or selecting another object. To select all the objects in the diagram, choose **Select All** in the **Edit** menu.

To select a range of objects, click in the diagram, hold down the right mouse button, drag the selection box over the objects you want to select, and release the mouse button. To select objects that are not in a range, hold down the Control key, then click each object you want to select.

Editing the Layout of a Diagram in the Diagram Editor

You can change the layout of a diagram, and copy, add or delete objects from the diagram. The Diagram Editor modifies the diagram only, not the repository.

Unlocking the Diagram Layout

When you open a diagram in the Diagram Editor, the diagram layout is *locked*. You cannot move or resize the objects in the diagram. To unlock the diagram layout, deselect **Lock Layout** in the **Edit** menu.

Autoplacing Objects

To rearrange the objects in a diagram automatically, choose **Auto Place** in the **Auto** menu. You are prompted to confirm that you want to autoplace the objects. Click **Yes**.

Splitting Objects

To “split” an object that has multiple relationships into an equal number of objects with a single relationship, select the object and choose **Split** in the **Edit** menu. Rearrange the objects manually or use the autoplace feature to rearrange the objects automatically.

Moving and Resizing Objects

To move an object in a diagram, drag-and-drop the object to the new location, or enter the grid coordinates of the location in the fields beside the symbol on the Box tool bar. The Diagram Editor redraws the relationships for the object.

To resize an object in a diagram, grab the border of the object with the mouse and drag it to a new location, or enter the grid coordinates for the height and width of the resized box in the fields beside the symbol on the Box tool bar.

Copying Objects

To create a copy of an object and all its relationships, select the object and choose **Clone** in the **Edit** menu.

To copy an object but not its relationships to the clipboard, select the object and choose **Copy** in the **Edit** menu. To paste the object from the clipboard, choose **Paste** in the **Edit** menu.

Tip: Choose **Clipboard** in the **View** menu to view the contents of the clipboard.

Adding Objects

To add an empty object to a diagram, click **New Box** in the **Edit** menu. To create a relationship between the new object and another object, hold down the Alt key, select either object, and drag-and-drop the relationship to the other object.

Deleting Objects

To delete an object from a diagram, select the object and click **Delete** in the **Edit** menu. To delete an object from the diagram and copy it to the clipboard, select the object and choose **Cut** from the **Edit** menu. The Diagram Editor also deletes the relationships for the object.

Autoattaching Relationships

To redraw a relationship line in a diagram automatically, select the relationship line and choose **Auto Attach** in the **Auto** menu.

Redrawing Relationships

To redraw a relationship in a diagram, grab the relationship line with the mouse and drag it to a new location, or enter the grid coordinates for the length and offset of each end of the redrawn line in the fields beside the and symbols on the Edge tool bar.

Deleting Relationships

To delete a relationship from a diagram, select the relationship and click **Delete Edge** in the **Edit** menu.

Editing Objects in the Diagram Editor

You can change the caption, color scheme, and border characteristics of an object. The Diagram Editor modifies the diagram only, not the repository.

Editing the Captions of an Object and Tool Tip

To edit the caption of an object, select the object and enter the new caption in the text field on the Box caption tool bar. To edit the caption of the tool tip for an object, select the object and enter the new caption in the text field on the Box tooltip tool bar.

Editing the Color Scheme for an Object

You can change the color of the background, border, and caption of an object:

- To change the background color of an object, select the object and click the arrow beside the symbol on the Box tool bar.
- To change the border color of an object, select the object and click the arrow beside the symbol on the Box tool bar.
- To change the color of the caption for an object, select the object and click the arrow beside the A symbol on the Box tool bar.

A standard Windows color control is displayed. Use the **Palette** tab to select the color of the background, border, or caption from the Windows palette. Use the **System** tab to match the color of the background, border, or caption with the color of standard Windows elements.

Editing the Border of an Object

You can change the thickness of the border for an object, specify that the border have rounded or square corners, and add a background shadow to the border:

- To change the thickness of the border for an object, select the object and click the arrow beside the drop-down on the Box tool bar. In the drop-down, choose the thickness you want, where 1 is the thinnest border and 4 is the thickest.

- To specify that the border of an object has rounded corners, select the object and click the button on the Box tool bar. The button is a toggle. Click the button again to specify that the border has square corners.
- To add a background shadow to the border for an object, select the object and click the button on the Box tool bar. The button is a toggle. Click the button again to specify that the border has no shadow.

Editing Relationships in the Diagram Editor

You can add a caption to, or delete a caption from, either end of a relationship, edit the captions, and edit the cardinality of the relationship. The Diagram Editor modifies the diagram only, not the repository.

Adding, Deleting, or Editing a Caption for a Relationship

There are two text fields for relationship captions on the Edge tool bar. The field on the left contains the caption for the left end of the relationship, the field on the right contains the caption for the right end of the relationship.

To add a caption to a relationship, select the relationship and enter the new caption in the appropriate text field. To delete a caption from a relationship, select the relationship and clear the caption in the appropriate text field. To edit the caption for a relationship, select the relationship and enter the new caption in the appropriate text field.

Editing the Cardinality of a Relationship

There are two drop-downs for relationship cardinalities on the Edge tool bar. The drop-down on the left contains the cardinality for the left end of the relationship, the drop-down on the right contains the cardinality for the right end of the relationship.

To edit the cardinality of a relationship, select the relationship and click the appropriate drop-down on the Edge tool bar. In the drop-down, choose the cardinality for the relationship.

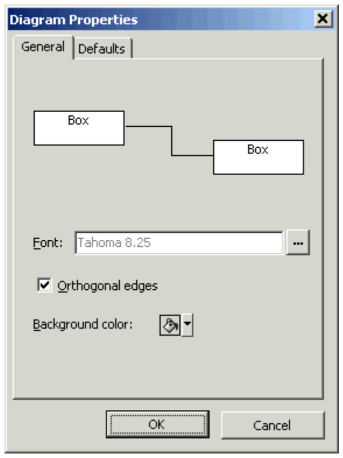
Editing General Properties of a Diagram in the Diagram Editor

Use the Diagram Properties window to specify general properties of a diagram in the Diagram Editor: the font for captions, whether relationships are displayed as orthogonal (right-angled) lines, and the background color of the Diagram Editor window.

To edit the general properties of a diagram:

1 In the **File** menu, choose **Properties**. The Diagram Properties window opens (Figure B-4).

Figure B-4 *Diagram Properties Window*



Font: Tahoma 8.25

☑ Orthogonal edges

Background color:

2 Click the button next to the **Font** drop-down. A standard Windows font control is displayed, where you can choose the font type, style, and size for diagram captions.

3 Select **Orthogonal Edges** if you want the Diagram Editor to display relationships as orthogonal (right-angled) lines.

4 Click the arrow beside the **Background color** drop-down to specify the background color for the Diagram Editor window. A standard Windows color control is displayed. Use the **Palette** tab to select the color of the text or background from the Windows palette. Use the **System** tab to match the color of the text or background with the color of standard Windows elements.

5 Click **OK** to apply your changes and dismiss the Diagram Properties window.

> ***Zooming in the Diagram Editor***
>
> Use the slider on the tool bar in the lower lefthand corner of the Diagram Editor window to zoom in or out on the diagram. To display the slider, choose **Zoom** in the **View** menu.

Saving Diagrams in the Diagram Editor

Choose **Save** in the **File** menu to save a diagram in the Diagram Editor. The Save Diagram dialog opens. In the **Save in** drop-down, choose the folder to save the new diagram in. In the **File name** field, enter the name of the new diagram. Choose a name that describes the diagram as closely as possible. Click **Save**.

Copying Diagrams with a Different Name (Saving As) in the Diagram Editor

Choose **Save as** in the **File** menu to copy a diagram with a different name in the Diagram Editor. The Save Diagram dialog opens. In the **Save in** drop-down, choose the folder to save the new diagram in. In the **File name** field, enter the name of the new diagram. Choose a name that describes the diagram as closely as possible. Click **Save**.

Printing Diagrams in the Diagram Editor

Choose **Print** in the **File** menu to print a diagram in the Diagram Editor. The Print Preview window opens, where you can change the zoom factor for the diagram to control the number of pages in the print job. Click **OK**.

Exporting Diagrams in the Diagram Editor

You can export diagrams to a variety of standard formats in the Diagram Editor. Choose **Export** in the **File** menu to export a diagram to BED, bitmap, JPEG, Visio, Visio XML, DOT, or EMF.

Note: Visio 2002 must be installed to save a diagram to Visio. Visio 2002 is not required to save to Visio XML.

Glossary

ADABAS
: ADABAS is a Software AG relational DBMS for large, mission-critical applications.

API
: API stands for application programming interface, a set of routines, protocols, and tools for building software applications.

applet
: See Java applet.

AS/400
: The AS/400 is a midrange server designed for small businesses and departments in large enterprises.

BMS
: BMS stands for Basic Mapping Support, an interface between application formats and CICS that formats input and output display data.

BSTR
: BSTR is a Microsoft format for transferring binary strings.

CDML
CDML stands for Cobol Data Manipulation Language, an extension of the Cobol programming language that enables applications programmers to code special instructions to manipulate data in a DMS database and to compile those instructions for execution.

CICS
CICS stands for Customer Information Control System, a program that allows concurrent processing of transactions from multiple terminals.

Cobol
Cobol stands for Common Business-Oriented Language, a high-level programming language used for business applications.

COM
COM stands for Component Object Model, a software architecture developed by Microsoft to build component-based applications. COM objects are discrete components, each with a unique identity, which expose interfaces that allow applications and other components to access their features.

complexity
An application's complexity is an estimate of how difficult it is to maintain, analyze, transform, and so forth.

component
A component is a self-contained program that can be reused with other programs in modular fashion.

construct
A construct is an item in the parse tree for a source file — a section, statement, condition, variable, or the like. A variable, for example, can be related in the parse tree to any of three other constructs — a declaration, a dataport, or a condition.

copybook
A copybook is a common piece of source code to be copied into many Cobol source programs. Copybooks are functionally equivalent to C and C++ include files.

CORBA

CORBA stands for Common Object Request Broker Architecture, an architecture that enables distributed objects to communicate with one another regardless of the programming language they were written in or the operating system they are running on.

CSD file

CSD stands for CICS System Definition. A CSD file is a VSAM data set containing a resource definition record for every resource defined to CICS.

database schema

A database schema is the structure of a database system, described in a formal language supported by the DBMS. In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables.

dataport

A dataport is an input/output statement or a call to or from another program.

DB/2

DB/2 stands for Database 2, an IBM system for managing relational databases.

DBCS

DBCS stands for double-byte character string, a character set that uses two-byte (16-bit) characters rather than one-byte (8-bit) characters.

DBMS

DBMS stands for database management system, a collection of programs that enable you to store, modify, and extract information from a database.

DDL

DDL stands for Data Description Language (DDL), a language that describes the structure of data in a database.

decision resolution

Decision resolution lets you identify and resolve dynamic calls and other relationships that the parser cannot resolve from static sources.

DMS

DMS stands for Data Management System, a Unisys database management software product that conforms to the CODASYL (network) data model and enables data definition, manipulation, and maintenance in mass storage database files.

DPS

DPS stands for Display Processing System, a Unisys product that enables users to define forms on a terminal.

ECL

ECL stands for Executive Control Language, the operating system language for Unisys OS 2200 systems.

effort

Effort is an estimate of the time it will take to complete a task related to an application, based on weighted values for selected complexity metrics.

EJB

EJB stands for Enterprise JavaBeans, a Java API developed by Sun Microsystems that defines a component architecture for multi-tier client/server systems.

EMF

EMF stands for Enhanced MetaFile, a Windows format for graphic images.

entity

An entity is an object in the repository model for a legacy application. The relationships between entities describe the ways in which the elements of the application interact.

FCT

FCT stands for File Control Table (FCT), a CICS table that contains processing requirements for output data streams received via a remote job entry session from a host system. Compare PCT.

HTML

HTML stands for HyperText Markup Language, the authoring language used to create documents on the World Wide Web.

IDL

IDL stands for Interface Definition Language (IDL), a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language.

IDMS

IDMS stands for Integrated Database Management System, a Computer Associates database management system for the IBM mainframe and compatible environments.

IMS

IMS stands for Information Management System, an IBM program product that provides transaction management and database management functions for large commercial application systems.

Java

Java is a high-level object-oriented programming language developed by Sun Microsystems.

Java applet

A Java applet is a program that can be sent with a Web page. Java applets perform interactive animations, immediate calculations, and other simple tasks without having to send a user request back to the server.

JavaBeans

JavaBeans is a specification developed by Sun Microsystems that defines how Java objects interact. An object that conforms to this specification is called a JavaBean.

JCL

JCL stands for Job Control Language, a language for identifying a job to OS/390 and for describing the job's requirements.

JDBC

JDBC stands for Java Database Connectivity, a standard for accessing diverse database systems using the Java programming language.

job

A job is the unit of work that a computer operator or a program called a *job scheduler* gives to the operating system. In IBM mainframe operating systems, a job is described with job control language (JCL).

logical component
A logical component is an abstract repository object that gives you access to the source files that comprise a component.

MFS
MFS stands for Message Format Service, a method of processing IMS input and output messages.

Natural
Natural is a programming language developed and marketed by Software AG for the enterprise environment.

object model
An object model is a representation of an application and its encapsulated data.

object-oriented programming
Object-oriented programming organizes programs in terms of objects rather than actions, and data rather than logic.

ODBC
ODBC stands for Open Database Connectivity, a standard for accessing diverse database systems.

orphan
An orphan is an object that does not exist in the reference tree for any startup object. Orphans can be removed from a system without altering its behavior.

parser
The parser defines the object model and parse tree for a legacy application.

parse tree
A parse tree defines the relationships between the constructs that comprise a source file — its sections, paragraphs, statements, conditions, variables, and so forth.

PCT
PCT stands for Program Control Table, a CICS table that defines the transactions that the CICS system can process. Compare FCT.

PL/I

PL/I stands for Programming Language One, a third-generation programming language developed in the early 1960s as an alternative to assembler language, Cobol, and FORTRAN.

profile

Profiles are HTML views into a repository that show all of the analysis you have done on an application. Profiles are convenient ways to share information about legacy applications across your organization.

QSAM

QSAM stands for Queued Sequential Access Method, a type of processing that uses a queue of data records—either input records awaiting processing or output records that have been processed and are ready for transfer to storage or an output device.

relationship

The relationships between entities in the repository model for a legacy application describe the ways in which the elements of the application interact.

relaxed parsing

Relaxed parsing lets you verify a source file despite errors. Ordinarily, the parser stops at a statement when it encounters an error. Relaxed parsing tells the parser to continue to the next statement.

repository

A repository is a database of program objects that comprise the model for an application.

schema

See database schema.

SQL

SQL stands for Structured Query Language, a standard language for relational database operations

system program

A system program is a generic program — a mainframe sort utility, for example — provided by the underlying system and used in unmodified form in the legacy application.

TIP
TIP stands for Transaction Processing, the Unisys real-time system for processing transactions under Exec control.

transaction
A transaction is a sequence of information exchange and related work (such as database updating) that is treated as a unit for the purposes of satisfying a request and for ensuring database integrity.

VALTAB
VALTAB stands for Validation Table, which contains the information the system needs to locate, load, and execute transaction programs. See also TIP.

VSAM
VSAM stands for Virtual Storage Access Method, an IBM program that controls communication and the flow of data in a Systems Network Architecture network.

XML
XML stands for Extensible Markup Language, a specification for creating common information formats.

Index